7.Parsing large files

October 11, 2018

0.0.1 How to transform large text files into Parquet files

This file needs to be run on AWS, and we need to make sure that the dbutils code works.

The weather data, as text, is about 7GB, after some organization, stored in 94 files, each of size about 85MB.

Initially, I used the following code to read in the data. Even though I used a reasonably large cluster, with about 100GB of memory, 5 worker nodes and 20 cores.

It took about 106 minutes (one hour and 46 minutes) to complete this job! Why?

For this first exercise, we will use the reduced dataset (10 percent) provided for the KDD Cup 1999, containing nearly half million **nework interactions**. First, download and read the gzip file:

```
In [3]: #start the SparkContext
        #import findspark
        #findspark.init()

        from pyspark import SparkContext
        sc = SparkContext(master="local[3]")
```

```
In [4]: import urllib.request, urllib.parse, urllib.error
        f = urllib.request.urlretrieve ("http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data
        data_file = "./kddcup.data_10_percent.gz"
        raw_data = sc.textFile(data_file)
        print(raw_data.count())
        print(raw_data.take(1))
```

```
494021
['0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.0
```

```
from pyspark.sql import Row
import numpy as np
# 1. Split each line using comma,
# 2. remove first line (which is the names of each column)
# 3. sort the data based on the "year" attribute

path = '/mnt/NCDC-weather/WeatherUncompressed/'
file_list=dbutils.fs.ls(path)
dataRows=range(len(file_list))
```

```
dataFrames=range(len(file_list))

for i in range(len(file_list)):
    filename=file_list[i].path
    data = sc.textFile(filename)
    dataRows[i] = data.map(lambda s: s.split(',')) \
                .filter(lambda d: d[0] != 'station') \
                .filter(lambda d:len(d)==368)\
                .map(lambda d: tuple(d[0:2]) + tuple([convert(x) for x in d[2:]])) \
                .sortBy(lambda d: d[2])
    dataFrames[i] = sqlContext.createDataFrame(dataRows[i], index)
    if i==0:
      combinedDataFrame=dataFrames[i]
    else:
      combinedDataFrame=combinedDataFrame.unionAll(dataFrames[i])
    print filename
    print combinedDataFrame.count()
```

The reason that this code is so slow is that the for loop which iterates over the files, while quite sensible for a single computer, is a very bad idea when using a cluster with 20 cores. It forces the cluster to read one file at a time, which means that at each point of time only one file is being read.

A simpler code, shown below, finished the same task in 8 minutes!

```
from pyspark.sql import Row
import numpy as np
# 1. Split each line using comma,
# 2. remove first line (which is the names of each column)
# 3. sort the data based on the "year" attribute

dataRows=range(len(file_list))
dataFrames=range(len(file_list))

path = '/mnt/NCDC-weather/WeatherUncompressed/'
data = sc.textFile(path)
dataRows = data.map(lambda s: s.split(',')) \
                .filter(lambda d: d[0] != 'station') \
                .filter(lambda d:len(d)==368)\
                .map(lambda d: tuple(d[0:2]) + tuple([convert(x) for x in d[2:]])) \
                .sortBy(lambda d: d[2])
dataFrame = sqlContext.createDataFrame(dataRows, index)
```

The resulting Parquet files are 4GB (almost half the original 7GB).

```
dataFrame.write.parquet("/mnt/NCDC-weather/Weather.parquet")

# the smaller file that we use in class was generated as follows:

sampled_df=dataFrame.sample(False,0.001)
sampled_df.write.parquet("/mnt/NCDC-weather/Weather_sampled.parquet")
```

```
In [5]: sc.stop()
```