Filter RDD based on row_number

sc.textFile(path) allows to read an HDFS file but it does not accept parameters (like skip a number of rows, has_headers,...).

in the "Learning Spark" O'Reilly e-book, it's suggested to use the following function to read a CSV (Example 5-12. Python load CSV example)

```
import csv
import StringIO

def loadRecord(line):
"""Parse a CSV line"""
input = StringIO.StringIO(line)
reader = csv.DictReader(input, fieldnames=["name", "favouriteAnimal"])
return reader.next()
input = sc.textFile(inputFile).map(loadRecord)
```

My question is about how to be selective about the rows "taken":

1. How to avoid loading the first row (the headers)
2. How to remove a specific row (for instance, row 5)

I see some decent solutions here: select range of elements but I'd like to see if there is anything more simple.

Thx!

python csv apache-spark

asked Nov 19 '14 at 16:34

Guillaume G
322 2 10

1 Answer

Don't worry about loading the rows/lines you don't need. When you do:

```
input = sc.textFile(inputFile)
```

you are not loading the file. You are just getting an object that will allow you to operate on the file. So to be efficient, it is better to think in terms of getting only what you want. For example:

```
header = input.take(1)[0]
rows = input.filter(lambda line: line != header)
```

Note that here I am not using an index to refer to the line I want to drop but rather its value. This has the side effect that other lines with this value will also be ignored but is more in the spirit of Spark as Spark will distribute your text file in different parts across the nodes and the concept of line numbers gets lost in each partition. This is also the reason why this is not easy to do in Spark(Hadoop) as each partition should be considered independent and a global line number would break this assumption.

If you really need to work with line numbers I recommend that you add them to the file outside of Spark(see here) and then just filter by this column inside of Spark.

Edit: Added `zipWithIndex` solution as suggested by @Daniel Darabos.

```
sc.textFile('test.txt')\
  .zipWithIndex()\            # [(u'First', 0), (u'Second', 1), ...
  .filter(lambda x: x[1]!=5)\ # select columns
  .map(lambda x: x[0])\       # [u'First', u'Second'
  .collect()
```

edited Mar 2 at 20:04

answered Nov 19 '14 at 17:16

elyase
14.8k 1 24 52

Thanks Elyase. With my edit, your code does the trick for the headers! – Guillaume G Nov 19 '14 at 17:37

1 You can use `RDD.zipWithIndex()` to add the line numbers and then you can filter by that too. (Note that `zipWithIndex` is not cheap.) – Daniel Darabos Nov 19 '14 at 18:08