Big Data elemzési eszközök nyílt forráskódú platformokon
Házi feladat

Mátyás-Barta Csongor

VYW0YR
mbcsongor@yahoo.com

2015. november 22.

Választott feladatok és technológiák

- Flight1 - Melyik reptéren gurulnak (TaxiIn és TaxiOut) átlagosan legtöbbet a gépek?

- Flight4 - Melyik 10 légitársaság indul a legtöbbször későn (DepDelay)?

- Flight2 - Honnan szállt fel a legtöbb repülő?

Java MapReduce lépések(1): Átlag számítás

Melyik reptéren gurulnak (TaxiIn és TaxiOut) átlagosan legtöbbet a gépek?

Map:
flightrecord $\rightarrow$ (OriginAirport, TaxiOut), (DestAirport, TaxiIn)

Reduce:
(Airport, list(TaxiTime)) $\rightarrow$ (Airport, Avg(TaxiTime))

Java MapReduce lépések(2):Maximum meghatározás

Map:
(Airport, AvgTaxiTime)$\rightarrow$ (1, (Airport, AvgTaxiTime))

Reduce:
(1, list(Airport, AvgTaxiTime)) $\rightarrow$ (MaxAirport, MaxTaxiTime)

Java MapReduce futtatás

%input fájlok hozzáadása a hdfs-hez

```
bin/hadoop fs -ls input/
```

%mapreduce job futtatás

```
bin/hadoop jar JavaMapReduce-1.0-SNAPSHOT.jar
com.mbcsongor.javamapreduce.Flight1 input/ output/
```

Spark lépések

Melyik 10 légitársaság indul a legtöbbször későn (DepDelay)?

- Releváns mezők kinyerése: Airport, DepDelay

```
map(lambda line: (line[16], line[15]))
```

- Időben elinduló gépek szűrése

```
filter(lambda line: line[1] 0)
```

- Késések előfordulásának összegzése repterek szerint

```
map(lambda line: (line[0],1)
reduceByKey(add)
```

Spark futtatás

```
In [2]: from operator import add
        data = sc.textFile("../big_data_eszkozok/hazi/input/2008.csv")
        header = data.first()
        rows = data.filter(lambda line: line != header)\ #header eldobás
        .map(lambda line: line.split(","))\      #mezők kinyerése
        .filter(lambda line: len(line)>1)\       #üres sorok eldobása
        .map(lambda line: (line[16],line[15]))\ #releváns mezők kiválasztása
        .filter(lambda line: line[1] != "NA")\  #nem teljes sorok kihagyása
        .map(lambda line: (line[0], int(line[1])))\ #mező int konverziója
        .filter(lambda line: line[1] > 0) \      #szűrés késön indulásra
        .map( lambda line : (line[0], 1)) \      #késön indulások számolása
        .reduceByKey(add) \
        .takeOrdered(10, key=lambda x: -x[1])    #legtöbbet késő 10 kiválasztás
        #rows = [i[0] for i in rows]
        rows

Out[2]: [(u'ATL', 175017),
         (u'ORD', 159427),
         (u'DFW', 127749),
         (u'DEN', 104414),
         (u'LAX', 87258),
         (u'IAH', 87139),
         (u'PHX', 82915),
         (u'LAS', 76240),
         (u'EWR', 69612),
         (u'DTW', 59837)]
```

Spark megjegyzések

- Első próba SparkR-ben, jobb a Python ☺
- Spark csak python2-vel megy egyelőre, így a Jupyter-nek(Notebook) szüksége van a python2-es kernelre(ha alapból nem azzal jön, linux disztró függő)

Pig lépések

Honnan szállt fel a legtöbb repülő?

- Nem teljes adatok szűrése

```
completeRecords = FILTER flights BY ( Origin != 'NA');
```

- Repterek listájának előállítása

```
origin = group completeRecords by Origin;
```

- Felszállások számolása repterenként

```
airportLiftoffs =
foreach origin generate COUNT(completeRecords), group;
```

- Rendezés felszállások száma szerint

```
ordered = ORDER airportLiftoffs BY $0 DESC;
```

- Legnagyobb kiválasztása

```
limited = LIMIT ordered 1;
projected = FOREACH limited GENERATE $1;
```

Pig futtatás

```
pig -param inputCSV="input/2008.csv"
-param output="pigResult" -x local Flight2.pig
```

Köszönöm a figyelmet!