Corso di Laurea Magistrale in

Ingegneria Informatica e dell'Automazione

Esercitazione di gruppo MapReduce e Spark AA 2021/2022

Domenico Potena

MapReduce

Dataset - caratteristiche

- Link: https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Video_Games_v1_00.tsv.gz
  - SAMPLE CONTENT:
    - https://s3.amazonaws.com/amazon-reviews-pds/tsv/sample_us.tsv

- Recensioni di prodotti acquistati su Amazon ed inerenti il mondo dei videogames

- TSV: Tab (‘\t’) separated values
- la prima riga contiene l’intestazione del dataset e non va usata nell’esercizio

- Dimensioni:
  - 1.12GB
  - ~1.8M record

Dataset - attributi

- **marketplace:** codice di 2 caratteri identificante la nazione in cui è stato comprato il prodotto
- **customer_id:** codice identificativo dell'utente che ha scritto la review
- **review_id:** codice identificativo della review
- **product_id:** codice identificativo del singolo prodotto recensito
- **product_parent:** codice identificativo della tipologia di prodotti a cui il singolo prodotto appartiene
- **product_title:** titolo del prodotto
- **product_category:** categoria a cui appartiene il prodotto
- **star_rating:** il punteggio, compreso fra 1 e 5, assegnato al prodotto dall'utente che ha scritto la revisione
- **helpful_votes:** numero di volte che la review è stata considerate utile
- **total_votes:** numero totale di voti ricevuti dalla review
- **vine:** la review è stata scritta nell'ambito del programma Vine
- **verified_purchase:** la review è relativa ad un acquisto verificato
- **review_headline:** titolo della review
- **review_body:** testo della review
- **review_date:** data in cui è stata scritta la review

MapReduce: Obiettivi

- Per ogni valore di *star_rating*, determinare la parola che compare in più recensioni.
  - In caso di occorrenze multiple della stessa parola all'interno di una recensione, conteggiarle tutte.
  - Eliminare dal testo le *stop words.*

- In merito alle *stop words*, si utilizzi la lista presente su:
  - https://gist.github.com/sebleier/554280

Note

- Nella macchina che useremo per i test, il dataset è memorizzato nella root di HDFS
  - Percorso dataset: hdfs://192.168.104.45:9000/test.tsv
- L'output del programma dovrà essere scritto in una cartella chiamata "output<numero_gruppo>" posta nella root di HDFS (/)
  - Es.: /output1 per il gruppo 1
- **Nello script per lanciare il job che consegnerete, impostate questi path**

- Nomi da dare alle varie classi: ChallengeXXXX
  - ChallengeDriver, ChallengerMapper, ChallengeReducer, ChallengePartitioner...

Esempio output

- 01 peace
- 02 nice
- 03 battery
- 04 car
- 05 fantasy

Risultati

- Unico file .ZIP (no .RAR, .7Z ecc) contenente:
  - Presentazione della soluzione (powerPoint o PDF)
  - File jar con il codice della soluzione
  - i sorgenti
  - Script con il comando per mandare in esecuzione il jar
    - qualora si abbia bisogno di più MapReduce in successione, dovrà essere prodotto uno script che lanci tutti i Jar in sequenza, specificando dati in input e output in maniera coerente

- Rispettate le indicazioni presenti nelle Note

Spark

Spark: Obiettivi

- Utilizzando pyspark e, se utile, pysparkSQL (versione python 2.7.5)
- Stilare la classifica su un singolo file dei "**product_title**" in base allo "**star_rating**" medio ricevuto (troncato alla prima cifra decimale) e in caso di pareggio si riportino prima i "**product_title**" con il maggior numero di occorrenze, si riportino entrambi i valori con cui si è stilata la classifica. Nel farlo:
  - Si ignorino i "**product_title**" che abbiano meno di 10 occorrenze;
  - Si valutino tutte e sole le review che abbiano **verified_purchase** = "Y " ;
- Inoltre si riporti a fianco a ciascuno "**product_title**" la parola (Intesa come sottostringa separata da " ") di almeno (>=) 5 caratteri, più frequente tra tutte le "**review_headline**" e "**review_body**" di quel "**product_title**" . Di questa parola si riporti anche il numero di volte che occorre;
  - Non contando come occorrenze tutte le sottostringhe che sono anche presenti nel "**product_title**"
- **Trasformare tutte le parole in lower case prima dei conteggio.**
- **Non usare librerie.**

NOTA: i campi "**review_headline**" **e** "**review_body**" potrebbero anche essere vuoti, il codice non deve andare in errore in quell caso.

Spark: Esempio

Input: (hdfs://192.168.104.45:9000/test.tsv)

```
product_title,star_rating,verified_purchase,review_headline,review_body
Monopoly Junior Board Game,5,Y,Great ,This board game is amazing!
Monopoly Junior Board Game,1,N,NO,don't buy it
Monopoly Junior Board Game,4,Y,Great,great great great
Star Wars Clone Wars Clone Trooper Costume,5,Y,<3,this costume is beautiful
Star Wars Clone Wars Clone Trooper Costume,3,Y,Accettable,not so beautiful
Magic Cards Game, 4, Y, Magic game, <3
```

Output:

Star Wars Clone Wars Clone Trooper Costume,4,2,beautiful, 2
Magic Cards Game, 4,1,,0
Monopoly Junior Board Game,3.3,3,great,5

ovvero: "**product_title**" , avg(star_rating), occorrenze("**product_title**"), parola+frequente, conteggio parola+frequent

NOTA : nell'esempio NON si sta considerando il numero minimo di occorrenze di "**product_title**" fissato a 10 per ragioni di spazio.

NOTA2: i campi "**review_headline**" **e** "**review_body**" anche potrebbero essere vuoti

NOTA3: un tsc può essere importato come un csv, usando il tab come separatore

Parti fisse di codice Spark

```
sqlContext = SQLContext(sc)
…. Codice vostro …
lista_risultato = lista da scrivere in output su file

Es:
lista_risultato = [(The Magic Tree,4.4,45,top,5),…]

with open('/home/amircoli/Andrea_Chiorrini/'+str(gruppo)+’.csv', 'w') as filehandle:
        for riga in lista_risultato.collect():
                filehandle.write(str(riga[0])+","+str(riga[1])+…. str(…)+"\n")
```

NOTA: potete anche scrivere con altri modi, purché funzioni.

Spark: alcuni consigli

- Potrebbero servire delle broadcast variables o degli accumulators
- Le lambda funzionano anche con funzioni custom, es:

```
def myFunction(a):
        …
        return f(a), f1(a)…
…
.map(lambda s: myFunction(s))
```

or:

```
.flatMap(lambda s: myFunction(s))
```

or:

….

Punteggi

- Esercizio MapReduce: 0.8
  - fastest: 0.2
- Esercizio Spark: 0.8
  - fastest: 0.2