Homework #4 - MapReduce

Sistemi Informativi Evoluti e Big Data

A.A. 2025-2026

Indice

1. Introduzione

L'obiettivo di questo homework è progettare e sviluppare un programma MapReduce per analizzare i dati relativi ai passaggi ai caselli autostradali. In particolare, si richiede di calcolare il **pedaggio medio** pagato da ogni tipo di veicolo nell'anno 2025 e confrontarlo con i dati di 10 anni fa (2015).

Il dataset è descritto dai seguenti campi: IDVeicolo, TipoVeicolo, Tratta, Pedaggio, DataTransito (formato GG/MM/YYYY), FasciaOraria e Provincia.

Sono state sviluppate due implementazioni del programma MapReduce utilizzando:

- **mrjob**: libreria Python per job MapReduce, ideale per prototipazione e testing locale
- **PySpark**: framework distribuito Apache Spark con API Python, ottimizzato per elaborazioni su larga scala

2. Generazione del Dataset

È stato generato un dataset sintetico di **100.000 record** utilizzando Python, con distribuzione bilanciata tra gli anni 2015 e 2025 (50% ciascuno).

2.1 Caratteristiche del Dataset

I pedaggi base sono stati differenziati per tipo di veicolo, con valori maggiorati per il 2025 per simulare l'inflazione e gli aumenti tariffari decennali:

Tipo Veicolo	Pedaggio Base 2015	Pedaggio Base 2025
Auto	€5.00	€6.50
Moto	€3.00	€4.00
Furgone	€8.00	€10.50
Camion	€15.00	€20.00
Bus	€12.00	€16.00

Il pedaggio finale include variazioni basate sulla tratta autostradale e una componente casuale (+-20%) per maggiore realismo.

3. Progettazione MapReduce

3.1 Strategia di Calcolo della Media

Un aspetto fondamentale nella progettazione è la gestione del calcolo della media in un contesto MapReduce. La media **non è una funzione associativa**, il che significa che non può essere calcolata direttamente nel Combiner.

Esempio del problema: media(1,2,3) = 2, ma media(media(1,2), 3) = media(1.5, 3) = 2.25 ≠ 2

Soluzione adottata: invece di emettere singoli valori, il Mapper emette coppie (somma, conteggio). Il Combiner aggrega somme parziali e conteggi, mentre il Reducer calcola la media finale dividendo la somma totale per il conteggio totale.

3.2 Flusso MapReduce

1. **Map:** Per ogni record, se l'anno è 2015 o 2025, emette (TipoVeicolo_Anno, {sum: costoPedaggio, count: 1})
2. **Combine:** Aggrega localmente somma e conteggio per chiave, riducendo il traffico di rete
3. **Reduce:** Calcola la media finale: media = Σ(somme) / Σ(conteggi)

4. Implementazione con mrjob

L'approccio basato su mrjob segue fedelmente il modello MapReduce classico. La libreria permette di descrivere in Python le tre fasi principali del paradigma, delegando al framework la gestione dell'esecuzione.

Il job è implementato nella classe PedaggioMedioMRJob, che estende MRJob.

1. Il metodo mapper() si occupa della lettura del file CSV: ogni riga viene analizzata, filtrando i record in base all'anno di interesse. Per ciascun record valido viene emessa una coppia chiave-valore, dove la chiave identifica la tratta (o l'attributo di aggregazione) e il valore contiene il pedaggio e un contatore unitario.
2. Il metodo combiner() esegue una prima aggregazione locale, sommando i pedaggi e i contatori associati alla stessa chiave.
3. Il metodo reducer() riceve i valori aggregati, calcola la somma totale e il numero complessivo di transiti, producendo infine il pedaggio medio per ciascuna chiave.

Il job può essere eseguito localmente tramite il comando:

- *python mapreduce_mrjob.py caselli_autostradali.csv*

5. Implementazione con PySpark

L'implementazione in PySpark adotta un approccio differente, basato sull'uso dei RDD/DataFrame anziché sulla definizione esplicita delle fasi Map e Reduce.

Dopo il caricamento del dataset in un RDD/DataFrame, vengono applicate operazioni di filtraggio, raggruppamento e aggregazione utilizzando le API di Spark SQL. Questo consente di esprimere il calcolo del pedaggio medio in modo più compatto e leggibile rispetto al modello MapReduce tradizionale.

Il motore di Spark si occupa automaticamente dell'ottimizzazione del piano di esecuzione, sfruttando l'elaborazione in memoria.

Il job può essere eseguito localmente tramite il comando:

- *python mapreduce_pyspark.py caselli_autostradali.csv*

6. Confrono tra le Implementazioni

6.1 Confronto Qualitativo

In entrambe le implementazioni è stato utilizzato l'equivalente di un Combiner:

- **mrjob:** funzione combiner() esplicita
- **PySpark:** reduceByKey() applica automaticamente aggregazione locale

6.2 Confronto Quantitativo

I tempi di esecuzione misurati su un dataset di 100.000 record mostrano differenze significative:

- **mrjob:** ~3 secondi
- **PySpark:** ~30 secondi (esecuzione come processo esterno)

Il tempo di esecuzione più elevato in PySpark è dovuto all'overhead di inizializzazione (avvio della JVM e creazione della SparkSession). Questo costo fisso viene ammortizzato su dataset di grandi dimensioni e in contesti distribuiti, dove Spark può sfruttare l'elaborazione parallela e in-memory. Nel caso di dataset piccoli e di esecuzioni locali, come nel test considerato, soluzioni più leggere come mrjob risultano più efficienti grazie al minore overhead di avvio.

7. Risultati e Analisi

Entrambe le implementazioni producono risultati identici, confermando la correttezza degli algoritmi:

```
Risultati mrjob:
----------------------------------------
Auto_2015: Media = €7.17, Transiti = 10173
Bus_2015: Media = €17.29, Transiti = 9996
Camion_2015: Media = €21.54, Transiti = 9919
Furgone_2015: Media = €11.46, Transiti = 9928
Moto_2015: Media = €4.31, Transiti = 9984

Auto_2025: Media = €9.33, Transiti = 10012
Bus_2025: Media = €22.99, Transiti = 10068
Camion_2025: Media = €28.60, Transiti = 10055
Furgone_2025: Media = €15.06, Transiti = 9958
Moto_2025: Media = €5.74, Transiti = 9907
```

```
Risultati PySpark:
----------------------------------------
Auto_2015: Media = €7.17, Transiti = 10173
Bus_2015: Media = €17.29, Transiti = 9996
Camion_2015: Media = €21.54, Transiti = 9919
Furgone_2015: Media = €11.46, Transiti = 9928
Moto_2015: Media = €4.31, Transiti = 9984

Auto_2025: Media = €9.33, Transiti = 10012
Bus_2025: Media = €22.99, Transiti = 10068
Camion_2025: Media = €28.60, Transiti = 10055
Furgone_2025: Media = €15.06, Transiti = 9958
Moto_2025: Media = €5.74, Transiti = 9907
```

7.1 Analisi dei Risultati

L'analisi evidenzia un **aumento generalizzato dei pedaggi** in 10 anni per tutti i tipi di veicoli, con variazioni percentuali comprese tra il 30% e il 33%:

- I **Camion** mostrano l'incremento assoluto maggiore (+€7.06), coerente con politiche tariffarie che penalizzano i veicoli pesanti
- L'incremento medio (~32%) è in linea con l'inflazione cumulata decennale e gli investimenti in manutenzione autostradale.

```
============================================================
VARIAZIONE PEDAGGIO MEDIO 2015 -> 2025
============================================================
Tipo Veicolo       2015       2025   Var.Ass.      Var.%
------------------------------------------------------------
Auto               7.17€      9.33€     +2.16€     +30.1%
Bus               17.29€     22.99€     +5.70€     +33.0%
Camion            21.54€     28.60€     +7.06€     +32.8%
Furgone           11.46€     15.06€     +3.60€     +31.4%
Moto               4.31€      5.74€     +1.43€     +33.2%

============================================================
```