Sistemi e Architetture per Big Data

A.Y. 2020-2021

Analisi del dataset delle vaccinazioni anti Covid-19

Tiziana Mannucci 0285727
ACM Member
Master Degree Student
Università degli Studi di Roma Tor Vergata
Via del Politecnico 1
Roma, Lazio, Italia
Email:
tiziana.mannucci@alumni.uniroma2.eu

***Abstract*— In questo articolo viene argomentata l'analisi fatta sul dataset riguardante le vaccinazioni anti Covid-19 andando a rispondere a delle query di interesse utilizzando un framework per il processamento dei dati. Vengono illustrati infine i risultati dello studio ed i tempi di esecuzione.**

1 Introduzione

Nella presente relazione vengono esposte le soluzioni per rispondere a due query di interesse utilizzando Spark per il processamento batch dei dati. È stato scelto quest'ultimo e non MapReduce in quanto Spark è più veloce (distribuito in-memoria), non si è limitati all'uso delle sole funzioni di Map e Reduce ma è possibile utilizzare un modello di programmazione basato su trasformazioni e azioni più programmer friendly.

Nelle sezioni 4 e 5 sono argomentate le soluzioni proposte rispettivamente per le query 1 e 2 e per ciascuna viene allegato il DAG ed ogni suo stage è stato descritto nelle sottosezioni specificando ogni trasformazione e scelta progettuale effettuata.

Nella sezione 6 viene fatta l'analisi dei tempi di esecuzione.

2 Architettura del sistema

Il dataset viene importato dal filesystem (non viene effettuato data ingestion) ed il processamento avviene su un nodo standalone su cui è in azione lo Spark-master che prende in carico la gestione del Job. Alla fine del processamento dei dati i risultati vengono esportati su HDFS. Quest'ultimo è costituito da un nodo master e tre nodi worker che formano una rete di container. L'immagine Docker usata per l'HDFS è quella di "efferre/hadoop" a cui sono state apportate delle modifiche relativamente al file di bootstrap per creare le directories di output[1]. Per facilitare la leggibilità dei risultati questi sono stati esportati anche su filesystem locale in directories omologhe a quelle usate per l'HDFS.

3 Analisi del Dataset

"Per gli scopi di questo progetto viene utilizzato il dataset disponibile all'URL:

https://github.com/italia/covid19-opendata-vaccini/tree/master/dati

In particolare, per il progetto si utilizzano i seguenti file in formato CSV:

1. *punti-somministrazione-tipologia.csv*
2. *somministrazioni-vaccini-latest.csv*
3. *somministrazioni-vaccini-summary-latest.csv"*[2]

È stata fatta un'analisi preliminare del dataset e sono stati trovati dei centri "duplicati" per il file riguardante i punti di somministrazione. Nel dettaglio sono presenti delle righe aventi tutti i campi coincidenti (stessa regione e stesso nome) ad eccezione della tipologia Ospedaliero/Territoriale. Con l'ausilio di personale medico sono state fatte delle ricerche su tali punti di somministrazione con lo scopo di verificare se fossero corrispondenti ad un unico centro oppure ad un presidio Ospedaliero con delle succursali Territoriali. Effettivamente ciascun punto di somministrazione tra quelli analizzati risulta essere un presidio Ospedaliero unico senza alcuna succursale, pertanto nella computazione della Query 1 le occorrenze duplicate vengono considerate singolarmente.

I file somministrazioni-vaccini-summary-latest.csv e somministrazioni-vaccini-latest.csv risultano essere omogenei e non sono state trovate anomalie.

4 Query 1

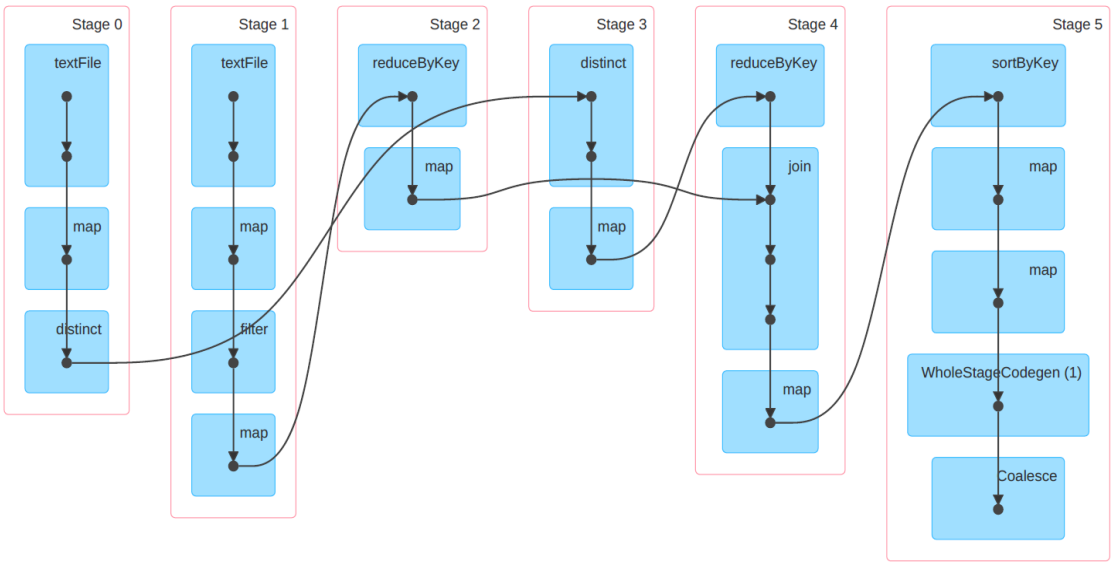
4.1 Specifica della query

"Utilizzando somministrazioni-vaccini-summary-latest.csv e punti-somministrazione-tipologia.csv, per ogni mese solare e per ciascuna area, calcolare il numero medio di somministrazioni che è stato effettuato giornalmente in un centro vaccinale generico in quell'area e durante quel mese. Considerare i dati a partire dal 1-01-2021."[2]

[1] "/Results/query1", "/Results/query2"

[2] Valeria Cardellini, SABD2021_progetto1.pdf

Figura 1DAG Query 1



4.2 *Sviluppo*

Per ottenere la media giornaliera delle vaccinazioni effettuate in un generico centro vaccinale per ogni regione e mese servono tre informazioni:

1. Numero di centri in una regione: x_w, $w \equiv regione$
2. Numero di giorni attivi di vaccinazione per mese e regione che potrebbe essere minore rispetto ad i giorni del mese effettivi: y_{wz}, $z \equiv mese$
3. Il totale delle somministrazioni per ciascun mese e regione, sapendo il totale giornaliero: d_i

$$avg_{wz} = \frac{1}{y_{wz}} \sum_i \frac{d_i}{x_w} \quad , \forall\, w, z$$

Per la proprietà distributiva rispetto all'addizione della sommatoria l'equazione può essere modificata nella seguente:

$$avg_{wz} = \frac{1}{y_{wz} * x_w} \sum_i d_i \quad , \forall\, w, z$$

Nell'ottica di operazioni su RDD applicare questa seconda formula comporta avere minor overhead computazionale non solo perché si riduce il numero di divisioni effettuate ma anche perché il calcolo avviene dopo aver aggregato i dati, quindi su RDD di dimensione inferiore.

4.3 *Soluzione Step-By-Step*

4.3.1 *DAG: Figura 1*

Per la risoluzione della query si è cercato di minimizzare il numero di trasformazioni che provocassero shuffle & sort dei dati ed il corrispondente cambio di stage. Tale scelta è motivata dal fatto che avere molti stage è dannoso dal punto di vista delle prestazioni in quanto risulta essere una sorta di punto di sincronizzazione per i task.

4.3.2 *Stage 0*

I dati riguardanti i punti di somministrazione vengono importati sottoforma di testo e successivamente mappati in oggetti custom che mantengono per ciascuna linea del csv solo i campi utili alla risoluzione della query: codice della regione, nome completo della regione, nome del centro e sua tipologia.
Viene mantenuta un'informazione ridondante sul nome della regione insieme al codice per facilitare l'operazione di join successiva che viene fatta proprio sul codice a 3 caratteri e non sul nome esteso della regione poiché quest'ultimo potrebbe presentare delle discrepanze tra i due file forniti.
L'operazione di distinct serve per eliminare i centri duplicati descritti nella sezione 3.
Quest'ultima trasformazione nel piano logico avviene immediatamente dopo la map, ma nel piano fisico di esecuzione vengono effettuate delle ottimizzazioni che rimandano la trasformazione allo stage 3 (sezione 4.3.5) in modo che sia più vicina all'operazione di Join.

4.3.3 *Stage 1*

L'importazione dei dati per il file somministrazioni-vaccini-summary-latest.csv viene fatta in modo analogo a quella per i punti di somministrazione appena descritto. Si mantengono solo i campi utili alla risoluzione della query: la data di somministrazione, il codice della regione ed il totale giornaliero delle vaccinazioni effettuate. Gli oggetti appena mappati vengono filtrati per eliminare tutte le vaccinazioni effettuate nel 2020.
L'RDD risultante viene mappato a sua volta per ottenere un nuovo RDD composto da tuple aventi i seguenti campi:

1. Tuple2(codice regione, mese)
2. Tuple2(totale giornaliero: d_i, 1)

4.3.4 *Stage 2*

Viene effettuata la reduceByKey che provoca shuffle dei dati e passaggio di stage; e la chiave corrisponde a: Tuple2(codice regione, mese). La funzione effettuata dalla reduceByKey fa la *somma del totale giornaliero* e la somma dei *giorni di vaccinazione attivi* per quel mese e quella regione:

$$tot_{wz} = \sum_i d_i\,, \qquad \forall\, i \equiv giorno\ vaccinale\ attivo$$

$$y_{xz} = \sum_i 1, \qquad \forall\, i \equiv giorno\ vaccinale\ attivo$$

In vista della successiva operazione di join si mappa il risultato della reduceByKey in un RDD di tuple aventi come chiave il codice della regione e come valore una tupla a tre campi:

- Mese
- Totale mensile vaccinazioni tot_{wz}
- Giorni di vaccinazione y_{wz}

4.3.5 *Stage 3*

Si riprende il processamento dell'RDD riguardante i punti di somministrazione con una distinct seguita da map. Quest'ultima trasformazione serve per ottenere un RDD di tuple con i seguenti campi:

- Codice della regione
- Tuple2(1, nome esteso della regione)

Non si mantiene anche il nome del centro vaccinale poiché si è interessati solamente al totale per regione.

4.3.6 *Stage 4*

Viene eseguita una reduceByKey sull'RDD riguardante i centri vaccinali per ottenere il totale dei centri vaccinali nella stessa regione.

$$x_w = \sum_1^n 1\,, \forall\, w \equiv regione$$

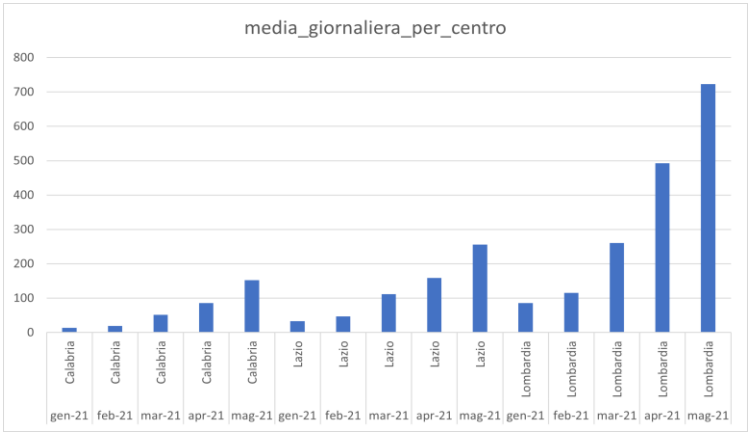
Finora sono stati generati due RDD rispettivamente composti dalle tuple:

1. (codice regionale, (mese, tot_{wz}, y_{wz}))
2. (codice regionale, (x_w, nome regione))

Il Join viene fatto sul codice regionale per ottenere un RDD che unisce tutte le informazioni necessarie per applicare, nella seguente operazione di map, la formula della media giornaliera

$$\frac{tot_{wz}}{y_{wz} * x_w}$$

Figura 2 Istogramma risultati



4.3.7 *Stage 5*

L'RDD risultante viene ordinato e mappato per essere esportato su HDFS in un file csv e nel formato desiderato: mese, regione, media giornaliera.

4.4 *Analisi Risultati*

Nella Figura 2 sono rappresentati gli istogrammi relativi a 3 regioni: Calabria, Lazio e Lombardia.
Si può notare che per tutte le regioni il trend delle vaccinazioni è in crescita da gennaio a maggio con la differenza che la regione Lombardia ha una media notevolmente superiore per tutti i mesi rispetto ai valori del Lazio il quale, a sua volta, ha dei valori medi superiori rispetto alla Calabria. Questo gap tra le regioni in parte è determinato dal fatto che non si tiene in considerazione la densità di popolazione, oltre che alla differente collocazione geografica e quindi al trasporto dei vaccini ed i suoi tempi, con il conseguente rallentamento della stessa campagna vaccinale.

5 QUERY 2

5.1 *Specifica della query*

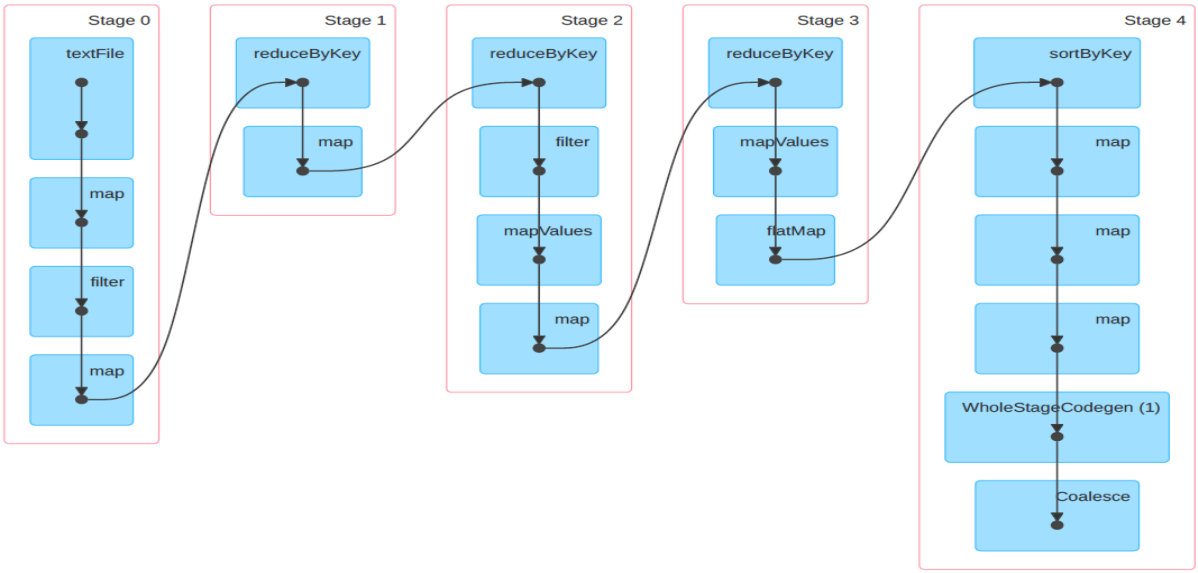
"Utilizzando somministrazioni-vaccini-latest.csv, per le donne, per ogni fascia anagrafica e per ogni mese solare, determinare le prime 5 aree per le quali è previsto il maggior numero di vaccinazioni il primo giorno del mese successivo. Per determinare la classifica mensile e prevedere il numero di vaccinazioni, considerare la retta di regressione che approssima l'andamento delle vaccinazioni giornaliere. Per la risoluzione della query, considerare le sole fasce anagrafiche per cui nel mese solare in esame vengono registrati almeno due giorni di campagna vaccinale. Viene inoltre richiesto di calcolare la classifica per ogni mese e fascia anagrafica a partire dai dati raccolti dal 1-02-2021."[2]

5.2 *Sviluppo*

La risoluzione della query fa affidamento ad un modello di regressione lineare per fare una previsione sul numero di vaccinazioni del primo giorno di un mese basandosi sui dati del precedente. Per farlo si utilizza un modello offerto dalla libreria Java math3: SimpleRegression.
Tale modello prende in input un insieme di punti come delle coordinate (x, y) in cui l'ascissa sarà rappresentata dalla data e l'ordinata dal numero di vaccinazioni effettuato alle donne in quel giorno.
Questa previsione deve essere fatta dopo aver raggruppato i dati in base ad una chiave formata da regione, mese e fascia di età.
Nella sezione 5.3 si illustra un pattern utilizzato per evitare l'uso della groupByKey [3] in quanto è considerata un'operazione onerosa e che provoca shuffle di una grande quantità di dati rispetto alla reduceByKey. Quest'ultima ha il vantaggio di applicare la funzione di reduce prima che avvenga lo shuffle dei dati similmente al Combiner per MapReduce.

[3] databricks-avoid-groupByKey

Figura 3 DAG query 2



5.3 *Evitare GroupByKey con pattern map & reduceByKey*

Lo scenario in cui vale la pena usare il seguente pattern è quello in cui si hanno delle coppie (key, val) e si vuole ottenere una lista di tutti i valori che condividono la medesima chiave: $(key, [val_1, val_2, \dots, val_n])$.
Per farlo si esegue prima una map che trasforma la coppia originaria nella coppia $(key, [\ val_i\ ])$ e successivamente una reduceByKey dove la funzione di reduce consiste nell'appendere un valore nella lista dell'altro:
e. g.
in: $(key, [val_1])$ *e* $(key, [val_2])$
out:$(key, [val_1, val_2])$

5.3.1 *Performance groupByKey*

Tabella 1

Metric	Min	25th percentile	Median
Duratin	0.1 s	0.1s	0.1s
GC Time	0.0	0.0	0.0
Shuffle Size/Record	371.5Kib/18650	371.5Kib/18650	371.5Kib/18650

5.3.2 *Performance map & reduceByKey*

Tabella 2

Metric	Min	25th percentile	Median
Duratin	0.1 s	0.1s	0.1s
GC Time	6.0 ms	6.0 ms	6.0 ms
Shuffle Size/Record	169.6 Kib/755	169.6 Kib/755	169.6 Kib/755

Si può notare che a parità di tempo di esecuzione la quantità di dati/record di cui viene fatto shuffle usando il pattern è notevolmente inferiore. Per dataset più grandi ci si aspetta un impatto prestazionale maggiore.

Il pattern proposto introduce un overhead per quanto riguarda la Garbage Collection e per essere vantaggioso si devono usare oggetti di piccola dimensione, serializzabili e assimilabili ad una lista. Nel caso specifico è stato implementato un oggetto Custom chiamato MyIterable che contiene una lista ed alcuni metodi per l'interazione con essa. I risultati in Tabella 2 appartengono a test che fanno uso di questo oggetto.

5.4 *Soluzione Step-By-Step*

5.4.1 *Stage 0*

In modo simile a quanto fatto per la query 1 i dati vengono importati e mappati in oggetti custom e si fa una filter iniziale per eliminare tutti gli oggetti non utili alla risoluzione della query. Nello specifico si tolgono tutte le istanze antecedenti al 1-02-2021 o in cui non è stata fatta neanche una vaccinazione per le donne. L'RDD risultante dal filtraggio viene mappato per ottenere un nuovo RDD di Tuple2(Tuple3(data, area, fascia anagrafica), totale donne).

5.4.2 *Stage 1*

Poiché ci sono diverse occorrenze per la stessa chiave (data, area, fascia anagrafica) ma relative a tipologie di vaccini differenti se ne fa la somma per ottenere il totale usando una reduceByKey.

5.4.3 *Stage 2*

Viene applicato il pattern descritto nella sezione 5.3 per raggruppare i dati che fanno riferimento allo stesso mese, regione e fascia anagrafica. In questo modo sono stati preparati i punti da passare al modello di regressione lineare per fare la predizione. Viene applicata una filter per scartare tutte le occorrenze aventi meno di due punti altrimenti il modello stesso restituirebbe risultati non significativi.
Il predittore è implementato in una nested class che viene invocata in una MapValues. Ogni valore in input al predittore è costituito da una lista di punti che vengono aggiunti al modello che ritorna una coppia: (data, valore predetto).

5.4.4 *Stage 3*

Per stilare la classifica dei valori predetti si devono raggruppare i dati per mese e fascia anagrafica ottenendo una lista di coppie (regione, valore predetto) per ciascuna chiave.

Per farlo si utilizza il pattern osservato nella sezione 5.3
Ogni elemento dell'RDD è costituito da una chiave (mese, età) e da una lista i cui elementi sono a loro volta delle coppie (regione, valore predetto). Ogni lista ha inoltre una lunghezza massima di 22 elementi, tanti quanti le regioni italiane. Per questa caratteristica non è stata usa la sortByKey (provoca shuffle dei dati) allo scopo di ottenere la top5 per ciascun mese e fascia anagrafica ma una mapValues in cui si applica una funzione dell'oggetto MyIterable, usato per implementare la lista, per fare il sorting degli elementi e mantenerne solo i primi 5.
In questo modo sono state ottenute delle coppie:

- Key: (mese, fascia anagrafica)
- Valore: lista delle 5 regioni in classifica con i rispettivi valori predetti.

A partire da ogni lista (classifica) si vogliono ottenere delle coppie: (Tuple3(mese, fascia anagrafica, valore predetto), Regione).
Per farlo si usa la flatMap in quanto permette di ritornare un nuovo RDD avente differente cardinalità rispetto a quello di partenza.

5.4.5 Stage 4

L'RDD risultante dalla flatMap viene ordinato e mappato per essere esportato su HDFS in un file csv e nel formato desiderato.

5.5 Analisi dei risultati

Nel grafico in Figura 4 viene rappresentata una stima del totale delle vaccinazioni previste per le donne il 1/06/2021 nelle diverse regioni. Tale valore è stato ottenuto sommando le previsioni aventi stessa regione ma differente fascia anagrafica.
Si osserva che la classifica generale delle vaccinazioni femminili in base ai valori predetti non è molto distante da quella ufficiale e disponibile al link.
I numeri sono inferiori rispetto a quelli ufficiali in quanto la classifica dei valori predetti si basa solo sulle vaccinazioni alla popolazione di sesso femminile.

Figura 4: valori predetti popolazione femminile 1/06/2021

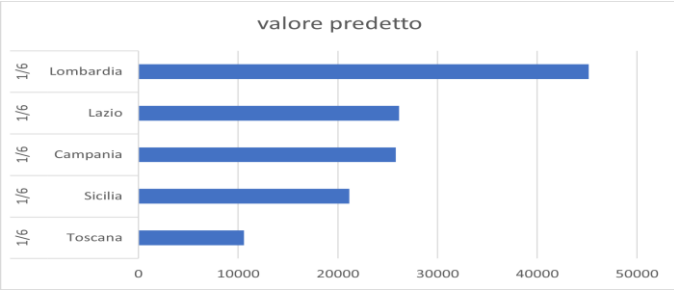
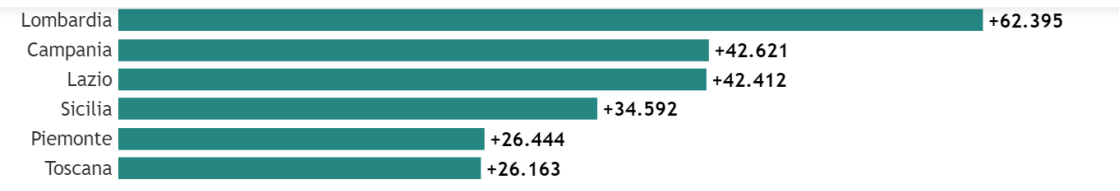


Figura 5: classifica ufficiale vaccinazioni (popolazione maschile e femminile) del 1/06/2021



6 Analisi Tempi di Esecuzione

Tabella 3 Tempi di esecuzione query 1

Stage	0	1	2	3	4	5
Durata	79ms	0.2 s	43ms	41ms	27ms	0.2 s

Tabella 4 Tempi di esecuzione query 2

Stage	0	1	2	3	4
Durata	0.6 s	0.2 s	0.2 s	31ms	0.2s

Nella Tabella 3 e nella Tabella 4 vengono riportati i risultati del report generato da Spark[4], nello specifico i tempi di esecuzione stage by stage. La "query 2" impiega più tempo per il suo completamento rispetto alla "query 1", ciò è dovuto in parte al dataset di dimensioni superiori ed in parte alla maggiore complessità della "query 2" stessa.

Tabella 5 Tempi esecuzione programma

Programma	query1	query2	query1&2
Durata Media	2.83 s	3.55 s	3.76 s
Query1	2.83 s	-	2.84 s
Query2	-	3.55 s	0.92 s

Nella Tabella 5 vengono riportati i tempi di esecuzione calcolati facendo la media su 10 differenti run. Si può notare che rispetto alla durata dei Job effettivi, riportati in Tabella 3 e Tabella 4, la durata complessiva del programma è nettamente superiore poiché comprende il tempo di inizializzazione dello Spark environment. Tale differenza è evidente se si lancia il programma che esegue entrambe le query. La "query 2" che viene eseguita dopo la "query 1" non soffre dell'overhead dovuto alla fase di inizializzazione iniziale ed infatti la sua durata non è lontana da quella che si ottiene sommando le durate riportate in Tabella 4.
Questo aspetto fa riflettere sul fatto che si dovrebbe usare Spark per il processamento batch dei dati quando il tempo di inizializzazione del suo ambiente è non significativo rispetto all'esecuzione del Job stesso.

[4] vedere report allegato nel repository contenente il sorgente