Big Data

Progetto

Nicola Giancecchi - m. 0900057545
nicola.giancecchi@studio.unibo.it
a.a. 2018/2019

Input

Il dataset relativo a questo progetto è tratto da **The Data Expo 2009** dell'*American Statistical Association*, pubblicamente disponibile all'indirizzo http://stat-computing.org/dataexpo/2009.
Il dataset preso in considerazione riguarda tutti i voli commerciali nazionali interni degli Stati Uniti effettuati nell'anno 2007. È composto da:

- un file CSV principale con tutti i dati relativi ai voli;
- 3 file CSV supplementari relativi agli aeroporti, alle compagnie aeree e ai modelli di aerei utilizzati.

I file di input sono disponibili nella cartella HDFS `hdfs://user/ngiancecchi/dataset`.

File principale (2007.csv)

Il file principale è composto da 7.453.216 record ed ha una dimensione compressa di 121 MB, non compressa a 703 MB. Contiene pressochè tutti i dati utili ad identificare un volo, inclusa la data, gli orari, il numero del volo e la destinazione. Sono presenti anche dati utili a svolgere analisi, come i ritardi alla partenza e all'arrivo comprensivi di tipologia, la distanza in km, il tempo in volo e il tempo di taxiing. I campi `UniqueCarrier`, `TailNum`, `Origin` e `Dest` fanno riferimento a file CSV secondari, come indicato in tabella.

(Visualizzazione trasposta)

Year	**2007**	*Note*
Month	1	
DayofMonth	1	
DayOfWeek	1	
DepTime	1232	
CRSDepTime	1225	

ArrTime	1341	
CRSArrTime	1340	
UniqueCarrier	WN	collegato a airports.csv
FlightNum	2891	
TailNum	N351	collegato a plane-data.csv
ActualElapsedTime	69	
CRSElapsedTime	75	
AirTime	54	
ArrDelay	1	
DepDelay	7	
Origin	SMF	collegato a airports.csv
Dest	ONT	collegato a airports.csv
Distance	389	
TaxiIn	4	
TaxiOut	11	
Cancelled	0	
CancellationCode		
Diverted	0	
CarrierDelay	0	
WeatherDelay	0	
NASDelay	0	
SecurityDelay	0	
LateAircraftDelay	0	

Aeroporti (airports.csv)

Il file degli aeroporti contiene la lista di tutti gli aeroporti statunitensi, contraddistinti dall'identificativo IATA (es. SFO). Sono presenti anche le coordinate geografiche.

iata	airport	city	state	country	lat	long
SFO	San Francisco International	San Francisco	CA	USA	37.619001 94	-122.374843 3

Compagnie aeree (carriers.csv)

Il file delle compagnie aeree è molto semplice. È composto dal codice IATA della compagnia aerea (es. *KL* per KLM) e dal nome della compagnia.

Code	Description
KL	Klm Royal Dutch Airlines

Aerei (plane-data.csv)

Il file relativo agli aerei contiene i dati tecnici relativi all'aereo utilizzato durante il volo, inclusa la casa costruttrice, la data di messa in servizio, il tipo di aereo e di motore.

tailnum	type	manufacturer	issue_date	model	status	aircraft_type	engine_type	year
N901WN	Corporation	BOEING	12/31/2007	737-7H4	Valid	Fixed Wing Multi-Engine	Turbo-Fan	2007

Job 1

Compagnie aeree che hanno accumulato più ritardo all'arrivo per ogni mese

Il job è stato realizzato tramite **MapReduce** e scritto in **Java**. È composto da quattro classi mapper, reducer, combiner e partitioner, oltre che dal job stesso (`FlightDelaysAnalyzer`).

- `FlightDelaysMapper` esegue il mapping dalle singole linee del file CSV a coppie chiave-valore che contengono solamente i dati necessari per l'analisi, ovvero per ciascun volo il mese (colonna "`Month`"), la compagnia aerea (colonna "`UniqueCarrier`") e il ritardo accumulato all'arrivo da tale volo (colonna "`ArrDelay`"). Il risultato del mapping è un insieme di coppie composte dal numero del mese (1...12) come chiave e una stringa composta dal codice della compagnia aerea e il ritardo (es. "AA,12") come valore.
- `FlightDelaysCombiner` pre-aggrega i dati in uscita dal mapper per alleggerire il lavoro dei reducer. Il tipo di output è identico a quello prodotto dal mapper, l'unica operazione effettuata è l'aggregazione del ritardo di più voli della stessa compagnia aerea in un unico record tramite la funzione statica `accumulateDelays(values)` presente nel reducer.

- `FlightDelaysReducer` aggrega i dati utilizzando la stessa funzione del combiner, dopodichè stila la classifica delle compagnie aeree del mese ordinando in modo discendente i valori e considerando solo i primi tre. L'output del job è composto da 3 coppie chiave-valore per ciascun mese, ad esempio per il mese di agosto avremo:
    - `8 WN` (Southwest Airlines, 1° compagnia per ritardo accumulato)
    - `8 AA` (American Airlines, 2°)
    - `8 UA` (United Airlines, 3°)

  Oltre alla compagnia aerea viene anche indicato il totale di minuti di ritardo.
- `FlightDelaysPartitioner` gestisce l'organizzazione del partizionamento del job. Si è scelto di suddividere i task in base al mese e al numero di task Reduce.

Performance

Dai test effettuati si è notato un sensibile miglioramento delle performance utilizzando il combiner e il partitioner, passando da 42 a 30 secondi di elaborazione. L'introduzione del combiner ha permesso di abbattere il carico di lavoro in capo al task Reduce sia in termini di tempi (da 19 a 5,5 secondi) che di quantità di dati: da 14 MB (7 milioni di record) a 3 KB (253 record).

Risultati

I risultati di questo job sono i seguenti:

Mese	Rank	Codice	Compagnia aerea	Rit. accumulato (min.)
1	**1**	AA	American Airlines Inc.	987.572
	2	OO	Skywest Airlines Inc.	958.223
	3	MQ	American Eagle Airlines Inc.	802.626
2	**1**	WN	Southwest Airlines Co.	946.839
	2	AA	American Airlines Inc.	909.953
	3	OO	Skywest Airlines Inc.	814.782
3	**1**	US	US Airways Inc.	962.082
	2	AA	American Airlines Inc.	942.645
	3	WN	Southwest Airlines Co.	916.754
4	**1**	AA	American Airlines Inc.	902.600
	2	WN	Southwest Airlines Co.	854.334
	3	US	US Airways Inc.	747.531

5	**1**	AA	American Airlines Inc.	891.826
	2	WN	Southwest Airlines Co.	879.559
	3	MQ	American Eagle Airlines Inc.	699.559
6	**1**	AA	American Airlines Inc.	1.474.860
	2	WN	Southwest Airlines Co.	1.424.455
	3	MQ	American Eagle Airlines Inc.	1.063.298
7	**1**	WN	Southwest Airlines Co.	1.434.623
	2	AA	American Airlines Inc.	1.286.772
	3	MQ	American Eagle Airlines Inc.	924.774
8	**1**	WN	Southwest Airlines Co.	1.266.214
	2	AA	American Airlines Inc.	1.013.147
	3	UA	United Air Lines Inc.	915.979
9	**1**	AA	American Airlines Inc.	663.724
	2	WN	Southwest Airlines Co.	646.211
	3	EV	Atlantic Southeast Airlines	543.164
10	**1**	WN	Southwest Airlines Co.	865.255
	2	AA	American Airlines Inc.	754.499
	3	UA	United Air Lines Inc.	604.281
11	**1**	WN	Southwest Airlines Co.	772.442
	2	AA	American Airlines Inc.	689.010
	3	XE	Expressjet Airlines Inc.	547.788
12	**1**	WN	Southwest Airlines Co.	1.384.935
	2	AA	American Airlines Inc.	1.239.663
	3	MQ	American Eagle Airlines Inc.	1.120.546

Job 2

Lista dei 10 aeroporti con maggior traffico notturno durante i mesi estivi

Il secondo job è stato realizzato tramite **SparkSQL** e scritto in Scala. In questo job viene presa in considerazione la frequenza di voli notturni negli aeroporti durante la stagione estiva, da giugno a settembre.

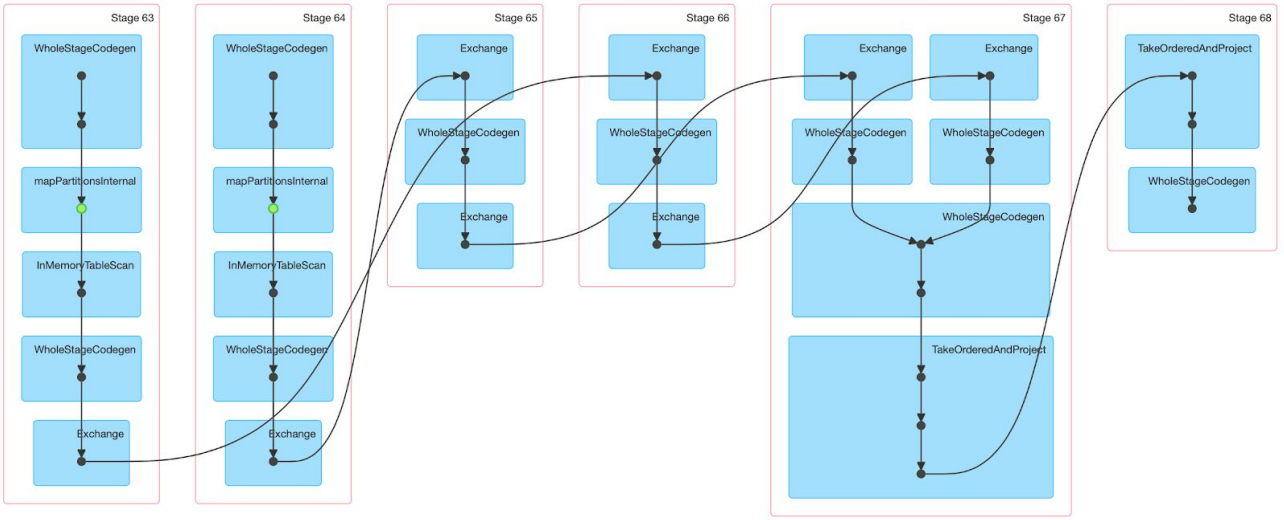
Le operazioni che vengono svolte sono le seguenti:

- viene caricato il file CSV dei voli sotto forma di DataFrame, al quale viene associato lo schema derivato dall'header del file. Vengono subito selezionate solamente le colonne di nostro interesse (`Month, Origin, Dest, DepTime, ArrTime`) e rinominate (`month, origin, destination, departure, arrival`);
- vengono filtrati i voli strettamente compresi tra giugno (6) e settembre (9). Il DataFrame risultante viene inserito in cache poichè verrà riutilizzato nel punto successivo;
- si filtrano poi i voli in con partenze e arrivi notturni, ovvero tutti quelli effettuati dopo le 21:00 e prima delle 6:00. Considerando che il focus è sugli aeroporti, viene considerato sia l'aeroporto di partenza che quello di arrivo, se soddisfa i requisiti. Ad esempio, un volo in partenza da San Francisco alle 22:00 e in arrivo a Los Angeles alle 0:00 viene conteggiato due volte.
  I DataFrame `departures` e `arrivals` contengono i voli selezionati rispettivamente in base all'orario di partenza e di arrivo, dopodichè i dati vengono raggruppati e sommati in base all'aeroporto.
- si uniscono infine i due DataFrame mantenendo solamente il codice IATA dell'aeroporto ("ATL", "SFO", ecc.) e la quantità di voli. Si esegue quindi l'ordinamento in base al valore e si prendono i primi 10 elementi;
- dato che nel file non sono elencati, vengono caricati i nomi completi degli aeroporti dal file CSV supplementare (`airports.csv`) e si esegue il join utilizzando il codice IATA.

Infine, i dati vengono salvati nella cartella `exsql-output` in formato CSV.

```
code,airport,movements
ATL,William B Hartsfield-Atlanta Intl,38047
LAX,Los Angeles International,25042
LAS,McCarran International,24281
ORD,Chicago O'Hare International,24054
...
```

Output CSV del DataFrame



DAG del job Spark SQL

Performance

L'utilizzo della cache per il DataFrame `summerFlights` ha permesso di ridurre sensibilmente il tempo impiegato nei calcoli (fasi 23 e 31, "*show*"), passando da 1:40 a 0:49.

Prestazioni senza cache

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
23	show at <console>:49	2019/02/22 13:48:28	52 s	6/6	613/613
22	run at ThreadPoolExecutor.java:1145	2019/02/22 13:48:28	0.1 s	1/1	1/1
21	save at <console>:49	2019/02/22 13:47:34	54 s	6/6	613/613
20	run at ThreadPoolExecutor.java:1145	2019/02/22 13:47:33	0.1 s	1/1	1/1
19	load at <console>:28	2019/02/22 13:47:31	75 ms	1/1	1/1
18	load at <console>:28	2019/02/22 13:47:31	0.1 s	1/1	1/1
17	load at <console>:28	2019/02/22 13:47:27	80 ms	1/1	1/1
16	load at <console>:28	2019/02/22 13:47:27	0.1 s	1/1	1/1

Prestazioni con cache

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
31	show at <console>:50	2019/02/22 13:51:14	8 s	6/6	613/613
30	run at ThreadPoolExecutor.java:1145	2019/02/22 13:51:13	0.1 s	1/1	1/1
29	save at <console>:50	2019/02/22 13:50:32	41 s	6/6	613/613
28	run at ThreadPoolExecutor.java:1145	2019/02/22 13:50:31	90 ms	1/1	1/1
27	load at <console>:29	2019/02/22 13:50:29	72 ms	1/1	1/1
26	load at <console>:29	2019/02/22 13:50:29	0.1 s	1/1	1/1
25	load at <console>:29	2019/02/22 13:50:26	88 ms	1/1	1/1
24	load at <console>:29	2019/02/22 13:50:26	87 ms	1/1	1/1

Dai test effettuati, il partizionamento ottimale dei dati in input è tra 6 e 12 unità. Un partizionamento troppo alto o troppo basso può compromettere le performance ed allungare i tempi.

Risultati

I risultati di questo job sono i seguenti:

Rank	IATA	Aeroporto	Totale movimenti
1	ATL	Hartsfield-Jackson Atlanta International Airport	38.047
2	LAX	Los Angeles International Airport	25.042
3	LAS	McCarran International Airport	24.281
4	ORD	O'Hare International Airport	24.054
5	DFW	Dallas-Fort Worth International Airport	18.114
6	SFO	San Francisco International Airport	16.097
7	SEA	Seattle-Tacoma International Airport	15.898
8	DEN	Denver International Airport	15.436
9	EWR	Newark Liberty International Airport	14.923
10	DTW	Detroit Metropolitan Wayne County Airport	14.619

Differenze tra i job

Nonostante i due job lavorino sullo stesso dataset, presentano alcune differenze sostanziali sia per la natura delle query che per i framework utilizzati.

Le tre principali riguardano:

- **calcoli svolti.** Il primo job accumula i ritardi in base alla compagnia aerea in una sola fase (due considerando anche il combiner), rendendolo così un semplice sommatore. Il secondo job invece estrae i dati in in base a due condizioni diverse (gli arrivi e le partenze) che necessitano di due scansioni diverse. I due set vengono poi uniti assieme e sommati.
- **dati utilizzati.** Il primo job utilizza i dati di tutto l'anno e li raggruppa per mese e compagnia aerea, il secondo invece solo quelli dei mesi estivi (equivalenti a circa ⅓ del dataset) e stila una classifica generale.
- **sorgenti ausiliarie.** Il job 1 fa uso del solo file CSV relativo ai voli, il job 2 utilizza anche il file degli aeroporti al fine di avere un output più completo. Il join di tale

file con i dati principali viene effettuato solamente dopo la selezione dei primi 10 aeroporti, così da applicare i nomi solamente ai dati strettamente necessari.

Difficoltà riscontrate

Ho trovato abbastanza familiare lavorare su MapReduce e Spark SQL sia grazie ai linguaggi di programmazione utilizzati (Java, Scala) che ai paradigmi propri dei tre framework. Essi permettono di separare il **cosa** si vuole ottenere, ovvero i dati, dal **come**, astraendo la maggior parte delle problematiche relative alla programmazione distribuita e lasciando comunque allo sviluppatore dei margini di manovra per tarare alcuni aspetti (numero di reducer, executor, combiner, ecc.) che possono variare in base alle caratteristiche del dataset e del cluster, al fine di ottenere le massime prestazioni da Hadoop.

Le principali difficoltà riscontrate, superata la fisiologica fase iniziale di apprendimento, hanno riguardato:

- **debugging.** Ho riscontrato alcune difficoltà nel debugging sia a causa di QuickStart che per l'output a volte non chiaro.
- **utilizzo di Cloudera QuickStart.** Nonostante fornisca una buona base per approcciarsi ad Hadoop, ho avuto diverse difficoltà nell'utilizzo della macchina virtuale sul computer personale, causate dalle scarse risorse a disposizione (4 core, RAM 8GB, HDD 64GB) e dal fatto che la VM simula un cluster a nodo singolo. Alcuni servizi infatti non hanno funzionato, come l'History Server di Spark o alcune sezioni di Cloudera Manager, e la mancanza di distribuzione ha rallentato i job rispetto all'esecuzione sul cluster. A tal fine, tutte le analisi e le prove finali sono state svolte utilizzando il cluster del corso.

Possibili implementazioni

È sicuramente interessante ragionare su entrambi i job con una quantità maggiore di dati e con dati più aggiornati, sia per quanto riguarda l'analisi dei dati (comprensione dei fenomeni) che delle performance. Potrebbero essere svolte analisi includendo i voli intercontinentali ed analizzare l'impatto sui ritardi di voli più lunghi e con distanze maggiori, o il comportamento compagnie aeree di altre paesi. Inoltre si potrebbero confrontare le prestazioni dei voli anno per anno.