Progetto 1 [SABD]

DAMIANO NARDI, TorVergata, Corso Di Informatica

ACM Reference Format:
Damiano Nardi. 2020. Progetto 1 [SABD]. 1, 1 (May 2020), 4 pages.

1 INTRODUZIONE

In questa relazione si descriverà il lavoro svolto che consiste nella realizzazione delle query(1,2) e tutta l'infrastruttura composta da vari framework istanziati su container docker per l'ingest, process e store dei dati.

2 PROCESSAMENTO DELLE QUERY

Per processare le entrambe le query è stato usato il framework Spark usando le api java, istanziato su cluster yarn all'interno di container docker con un nodo master e due worker. Si è utilizzata una macchina virtuale con Ubuntu 20.04 con allocatevi 4 core della cpu (i7-8750H) e 8gb di ram (ddr4 2400 MHz).

2.1 Query1

> Per ogni settimana, calcolare il numero medio di guariti e dei tamponi effettuati in Italia in quella settimana.

Una volta caricato il dataset come JavaRDD viene fatta una

2.1.1 flatMapToPair. in cui viene restituito come chiave il numero della settimana dell'anno e come valore una tupla composta da positivi,tamponi (vengono ignorati i giorni in mezzo alla settimana in base allo startingDay selezionato)

2.1.2 reduceByKey. vengono presi solo i giorni con la stessa settimana dell'anno e viene calcolata la media dei tamponi e guariti facendo la differenza dei giorni di inizio e fine settimana.

2.1.3 ricreazione schema. Viene poi ricreato lo schema e viene salvato il dataset processato in formato parquet su HDFS

2.1.4 Tempi. è stata tenuta traccia di due tempi: tempo di processamento della query (da prima del caricamento a dopo il caricamento dei dati processati)e tempo del caricamento dello spark contex nel driver program (i tempi sono in media su 5 misurazioni).

	tempo (sec.)
Query processing	12,5
Spark loading	1,5

2.2 Query2

> Per ogni continente, calcolare la media, la deviazione standard, il minimo e il massimo del numero di casi confermati giornalmente per ogni settimana. Nel calcolo delle statistiche, considerare solo i 100 stati piu colpiti dalla pandemia. Qualora lo stato non fosse indicato, considerare la nazione. Per de- ‘ terminare gli stati piu colpiti nell'intero

Author's address: Damiano Nardi, damiano6276@gmail.com, TorVergata, Corso Di Informatica.

dataset, si consideri l'andamento degli incrementi giornalieri ‘ dei casi confermati attraverso il trendline coefficient. Per stimare il trendline coefficient, si calcoli la pendenza della retta di regressione che approssima la tendenza degli incrementi giornalieri. Nota: il continente a cui appartiene ogni nazione non viene indicato in modo esplicito nel dataset, ma deve essere ricavato. Si considerino 6 continenti: Africa, America, Antartide, Asia, Europa, Oceania.

Una volta caricato il dataset come JavaRDD viene fatta una

2.2.1 flatMapToPair. in cui viene restituito come chiave il coefficente di tread line e come valore un oggetto "CovidGlob", all'interno ha la lista dei contagiati, la regione e la nazione, durante questa fase viene calcolato il coefficente di di tread line, la lista degli infetti per giorno viene trasformata da cumulativa alla lista dei nuovi casi (per giorno).

2.2.2 top 100. Vegono presi i top 100 stati con il coefficente di tread line più alto

2.2.3 mapToPair. La chiave vine cambiata dal coefficente di tread line alla nazione, il valore rimane l'oggetto "CovidGlob"

2.2.4 Join. Viene caricato come javaRDD un dataset esterno al progetto da HDFS preso da https://github.com/dbouquin/IS_608/blob/master/NanosatDB_munging/Countries-Continents.csv essenzialmente è mapping continete-nazione, prima di fare il join viene fatta una mapToPair per rendere la nazione chiave e poi viene fatto il join con i top 100

2.2.5 mapToPair. Per rendere il continente key, il valore invece è la lista degli infetti per giorno

2.2.6 reduceByKey. Per sommare le liste di infetti con lo stesso continente

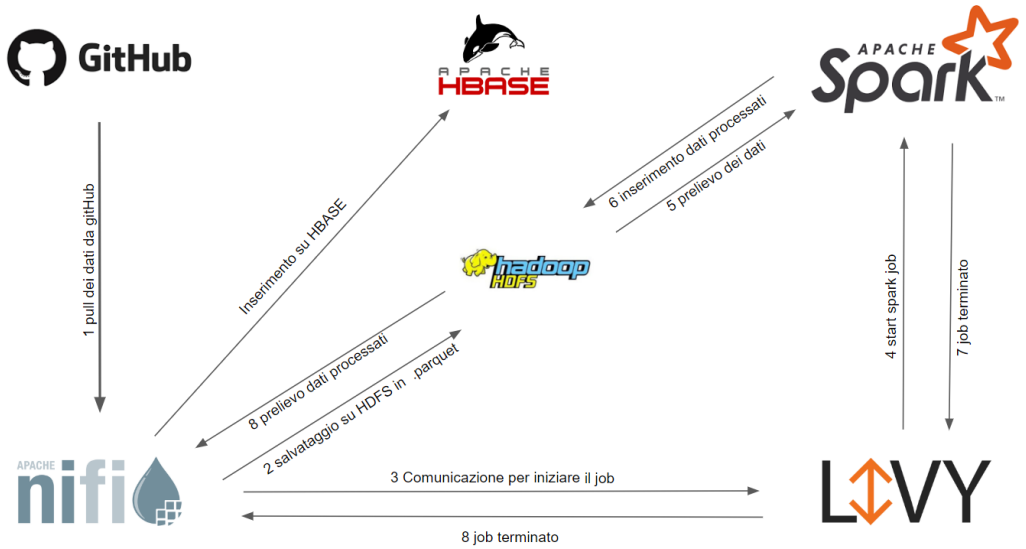
2.2.7 flatMap. Per calcolare tutte le statistiche richieste per settimana, viene usata la flatMap in quanto da ogni RDD element verranno generate 4 liste: lista delle medie,massimi,minimi,deviazioni standard tutte quante per settimana

2.2.8 ricreazione schema. Viene poi ricreato lo schema e viene salvato il dataset processato in formato parquet su HDFS

2.2.9 Tempi. è stata tenuta traccia di due tempi: tempo di processamento della query e tempo delcaricamento dello spark contex nel driver program (i tempi sono in media su 5 misurazioni).

	tempo (sec.)
Query processing	16,63
Spark loading	1,3

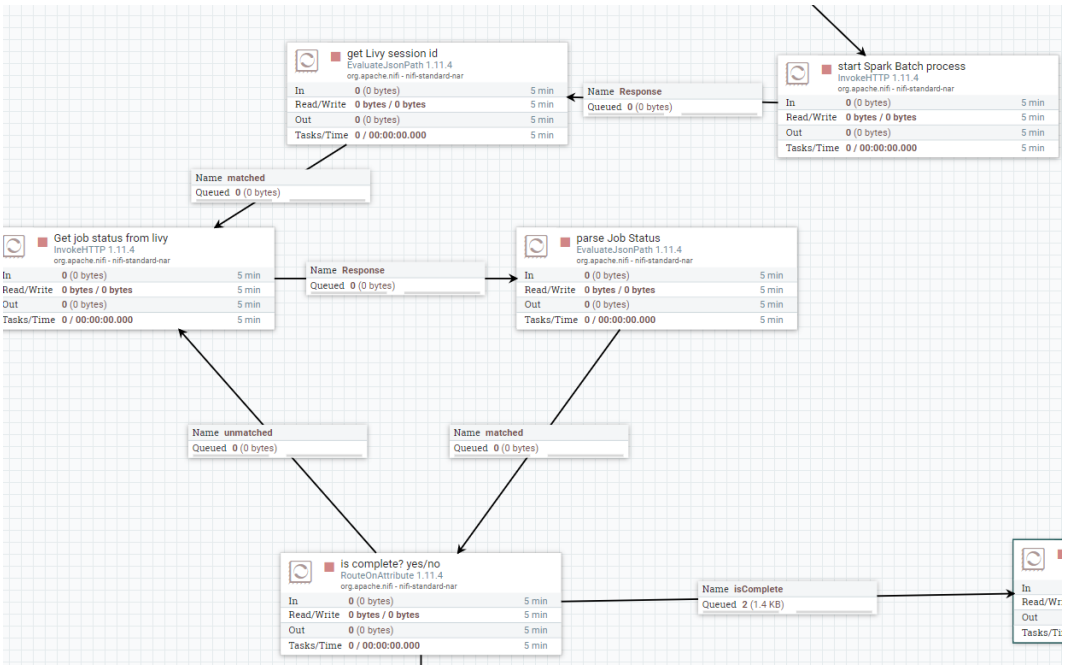
3 INFRASTRUTTURA DI PROCESSAMENTO



3.1 Framework utilizzati

3.1.1 Spark: processing. come visto prima si è utilizzato Spark per il processamento delle query, nello specifico è stato creato un cluster yarn composto da due worker e un master istanziati su container docker.

3.1.2 NIFI: ingestion. Si è utilizzato NiFi per fare data ingestion, nello specifico i dati vengono scaricati da gitHub ogni settimana, sono scremati attraverso una query sql, convertiti in formato parquet e caricati su HDFS (questo per la query1, Nifi non è riuscito a convertire in parquet il dataset della query2 quindi è stato caricato direttamente il csv per quest'ultima). Di default Nifi non mette a disposizione un modo per comunicare con Spark, Livy viene in nostro (mio) aiuto permettendo a Nifi di inviare job a Spark e di sapere lo stato della computazione, in questo modo Nifi dopo aver finito l'upload dei dati su HDFS puo far iniziare a Spark il processamento e una volta finito Nifi può prelevare dati processati da HDFS e passarli al livello di storage



3.1.3 Livy: layer di comunicazione. Livy è un framework che mette a disposizione delle REST API per gestire Spark. In questo progetto è stato usato per far comunicare Nifi con Spark, nello specifico Nifi fa una post con dentro la classe main e la locazione del file jar nell' HDFS da eseguire; successivamente NiFi ogni 5 secondi controlla con una get a Livy lo stato del JoB, quando lo stato risulterà "succeed" Nifi continuerà il suo flusso

3.1.4 Hbase: Storage. Per il layer di storage è stato utilizzato Hbase una volta che Spark ha finito la computazione Nifi si occupa di prendere i file parquet salvati da spark su HDFS processarli come parquet specificare la colonna che si vuole come key e caricarli su hbase. Per la query1 come key è stata scelta la data (formato annoSettimana) in modo tale da ottimizzare lo scan inquanto i dati saranno memorizzati su posizioni vicine memoria, di contro qualora i dati dovessero diventare troppi si andrebbe a sovraccaricare un singolo region Server però considerata la quantità dei dati non è il caso. Per la seconda query dopo il processamento abbiamo righe di questo tipo :

AAAEuropa	europa	MAX_WEEK	...
ZZZamerica	america	MIN_WEEK	...
IIIEuropa	europa	DEV_WEEK	...
RRREuropa	europa	AVG_WEEK	...

come key è stato scelto il campo della prima colonna, in questo modo le righe con la stessa funzione statica (cioè tutte le righe con il campo MAX_WEEK inizieranno con AAA nella prima colonna) si troveranno sullo stesso regionserver e vicine tra loro ,andando ad ottimizare lo scan di tutti i continenti con una determinata funzione statistica