Progetto 2 [SABD]

DAMIANO NARDI, TorVergata, Corso Di Informatica

ACM Reference Format:
Damiano Nardi. 2020. Progetto 2 [SABD]. 1, 1 (June 2020), 3 pages.

1 INTRODUZIONE

In questa relazione si descriverà il lavoro svolto che consiste nella realizzazione delle query(1,2)

2 PROCESSAMENTO DELLE QUERY

Per processare le query 1 e 2 è stato usato il framework apache flink usando le api java, inoltre per confrontare flink con kafka streams è stata fatta la anche query 1 usando kafka streams. 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 Stremming del dataset

Il dataset viene letto e inviato riga per riga a una topica kafka che viene poi consumata da apache flink o da kafka streams.

2.2 Query1

query 1 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.

Lo stream del dataset viene letto da una topica kafka la prima operazione fatta è una

2.2.1 flatMap. In questa flatMap viene processata la singola ringa del dataset andando a mettere come chiave il campo Boro (il quartiere) e come valore un ogetto con questi campi: OccurredOn (data del ritardo), HowLongelayed (ritardo in minuti) e altri campi meno importanti; le righe del dataset che non hanno il campo HowLongDelayed o lo hanno in un formato non previsto vengono ignorate.

2.2.2 EventTime e TumblingEventTimeWindows. A questo punto flink estrare l'event time dai dati processati della flatMap e subito dopo crea una TumblingWindows della durata preimpostata tra 1,7,30 giorni

2.2.3 Reduce. Facciamo una reduce (abbiamo che chiave Boro), essenzialmente andiamo a sommare i ritardi di un quartiere (nella stessa finestra) teniamo anche in considerazione la "somma" delle reduce fatte e la data più vecchia in ogni reduce.

2.2.4 Map. In questa Map andiamo a calcolare la media dei ritardi sommati per ogni quartiere nella precedente reduce

2.2.5 Apply. In fine vengono raggruppati tutti i dati nella stessa finestra temporale e viene formattato l'output come richiesto dalle specifiche.

2.2.6 addSink. Infine viene aggiunto un sink che manda lo stream processato a una topica kafka

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

2.3 Query 1 kafka streams

La query 1 è stata fatta anche su kafka streams le operazioni fatte sono più o meno le stesse, cambia la sintassi, quindi per evitare ridondanza non verranno riportate.

2.4 Query2

query2 .

Lo stream del dataset viene letto da una topica kafka la prima operazione fatta è una

2.4.1 flatMap. In questa flatMap viene processata la singola riga del dataset andando a mettere come chiave il campo Reason con il prefisso "fascia5-11" o "fascia12-19" scelto in base al campo OccuredOn mentre come valore un ogetto con i seguenti attributi OccurredOn,Reason,rank(intero inizialmente pari a 1), list(inizialmente vuota).

2.4.2 EventTime e TumblingEventTimeWindows. A questo punto flink estrare l'event time dai dati processati della flatMap e subito dopo crea una TumblingWindows della durata preimpostata tra 1 o 7 giorni

2.4.3 Reduce. In questa reduce vengono sommati i campi rank e viene presa la data (OccurredOn) minima tra le varie operazioni di reduce

2.4.4 Map. Viene messa la tupla (Reason,rank) dentro la lista e mettiamo come chiave "fascia5-11" o "fascia12-19"

2.4.5 TumblingEventTimeWindows. specifichiamo ancora la finestra temporale

2.4.6 Reduce. In questa reduce viene fatto il l'unione del campo lista (che avevamo iniziato a popolare nella map precedente) per le due fascie "fascia5-11", "fascia12-19"

2.4.7 Map. Adesso prendiamo la dal campo list le 3 Reason che hanno il rank più alto

2.4.8 TumblingEventTimeWindows. Specifichiamo ancora la finestra temporale

2.4.9 Reduce e Map. In fine facciamo una Reduce e una Map ai fine formattare l'output come richiesto dalle specifiche

2.4.10 addSink. Viene aggiunto un sink che manda lo stream processato a una topica kafka

3 TEMPI

Sono stati misurati i tempi di latenza e throughput di messaggi al secondo, In questo paragrafo descriveremo come sono stati calcolati

3.1 Producer

Come accennato prima abbiamo un producer che legge il dataset riga per riga e li passa a una topica kafka durante questa operazione appena prima di inserire la riga sulla topica viene aggiunto a inizio riga il timestamp corrente.

3.2 2 tipo di latenza

Quando viene fatta una reduce andiamo ad unificare più righe del dataset che sono state inserite nella topica a tempi di processamento diversi, durante queste operazioni viene tenuta traccia del tempo di processamento più vecchio e più nuovo, quindi ogni output prodotto da flink e da kafka conterrà 3 tempi: event time,processing time riga più vecchia,processing time riga più nuova.

Abbiamo la latenza del record più vecchio e la latenza del record più nuovo per ogni output da flink/kafka

3.3 Consumer

Il consumer non fa altro che leggere da una topica kafka (output di flink o kafka streams) registra il timestamp corrente ad ogni record prelevato da quest'ultima e calcola i due tempi di latenza in secondi andando a fare la differnza tra quello corrente e quelli presnti nell'output,

3.4 throughput

Per misurare il throughput ho utilizzato questo comando di kafka streams

```
/home/naddi/kafka/bin/kafka-consumer-perf-test.sh
--topic output-stream
 --broker-list "localhost:9092,localhost:9093,localhost:9094"
 --messages 900 --threads 1
```

che tra le misure che ritorna vi è anche la latenza