Analisi di dati marittimi geo-spaziali con Flink

Relazione Secondo Progetto

Petrolo Melissa

Corso di Laurea Magistrale in Ingegneria Informatica
Università degli Studi di Roma Tor Vergata
Matricola: 0286160
emelis.ptr@gmail.com

1. ABSTRACT

Questo documento riporta i dettagli implementativi riguardanti l'analisi mediante Flink del dataset contenente informazioni su dati marittimi geo-spaziali. In particolare, si fa riferimento a dei dispositivi di identificazione automatica (AIS) utilizzati per garantire la sicurezza delle navi in mare e nei porti, permettendo lo scambio di informazioni riguardo lo stato delle navi in movimento.

Verrà descritta, insieme a tali dettagli implementativi, anche l'architettura a supporto dell'analisi.

2. INTRODUZIONE

L'analisi effettuata si pone lo scopo di valutare la media e il numero di navi con maggiore transizione in una particolare area marittima.

Il dataset, preso in considerazione, contiene i dati riguardanti un'area marittima geo-spaziale e l'area del mar Mediterraneo.

Il dataset, campionato con la granularità del minuto, si compone come segue:

- SHIP_ID: identificativo della nave;
- SHIP_TYPE: tipologia della nave;
- SPEED: velocità misurata in nodi a cui precede la nave all'istante di segnalazione dell'evento;
- LON: coordinata cartesiana in gradi decimali della longitudine data dal GPS;
- LAT: coordinata cartesiana in gradi decimali della latitudine data dal GPS;
- COURSE: direzione del movimento; definito come angolo in senso orario tra il nord Vero e il punto di destinazione;
- HEADING: direzione verso cui la nave è orientata ed è espresso in gradi;
- TIMESTAMP: istante temporale della segnalazione all'evento AIS;
- DEPARTURE_PORT_NAME: identificativo del porto di partenza del viaggio in corso;
- REPORTED_DRAUGHT: profondità della parte immersa della nave tra la linea di galleggiamento e la chiglia;
- TRIP_ID: identificativo del viaggio.

2.1 Query uno

Per la prima query si richiede di calcolare per ogni area del Mar Mediterraneo il numero medio giornaliero di navi di ogni tipologia su finestre temporali settimanali e mensili.

L'output deve essere riportato nel seguente modo: ts, id_cella, ship_t35, avg_t35, ..., ship_to, avg_to; dove ts rappresenta il timestamp relativo all'inizio del periodo in cui si è calcolata la media.

2.2 Query due

Per la seconda query si richiede di fornire, per il Mar Mediterraneo e per il Mar Orientale, una classifica delle tre celle più frequentate nelle due fasce orarie di servizio 00:00-11:59 e 12:00-23:59 su finestre temporali settimanali e mensili.

L'output deve essere riportato nel seguente modo: ts, sea, slot_a, rank_a, slot_p, rank_p; dove ts cui si è calcolata la media.

3. ARCHITETTURA E FRAMEWORK

Per poter eseguire e realizzare il progetto è stato utilizzato Docker, una piattaforma software che

permette di creare, testare e distribuire applicazioni con la massima rapidità.

È stato creato un file docker-compose per la creazione di container dei servizi utili

- Zookeeper
- Kafka [1]

Mentre Flink crea un ambiente locale con una web UI.

3.1 Zookeeper

Zookeeper viene utilizzato da Apache Kafka per memorizzare informazioni relative al cluster Kafka e alle informazioni degli utenti; in breve, possiamo dire che Zookeeper memorizza i metadati sul cluster Kafka.

3.2 Kafka

Kafka è il sistema di messaggistica di tipo publish-subscribe utilizzato per l'ingestion di dati nei sistemi di processamento e per l'export dei risultati.

Il cluster prevede un container per Zookeeper, necessario per la coordinazione e tre Kafka brokers.

Vengono create cinque topiche, di cui una per l'input dei dati mentre le altre riguardano i risultati ottenuti sia per la prima che la seconda query.

Inoltre, ogni Kafka topic ha una sola partizione in modo tale da mantenere l'ordine dei record che vengono inseriti.

3.2.1 Producer

Il producer si occupa di leggere il dataset e di pubblicarne il risultato su un topic "dataset-source" di Kafka.

La scrittura viene effettuata a intervalli di 10 millisecondi al fine di simulare una sorgente di dati real time.

3.2.2 Consumer

Il consumer si occupa di sottoscriversi ad un topic e di leggerne il contenuto.

Inoltre, si occupa anche di salvare i risultati in un file csv attraverso il topic in cui sono stati salvati i record ottenuti.

3.3 Flink

Flink è il framework di data streaming utilizzato per l'esecuzione delle due query.

Nel momento in cui viene eseguita l'applicazione, vengono create quattro task e un task manager di default e le operazioni vengono eseguite con parallelismo pari a quattro.

Di seguito, viene rappresentata la topologia delle due queries.

zzati:

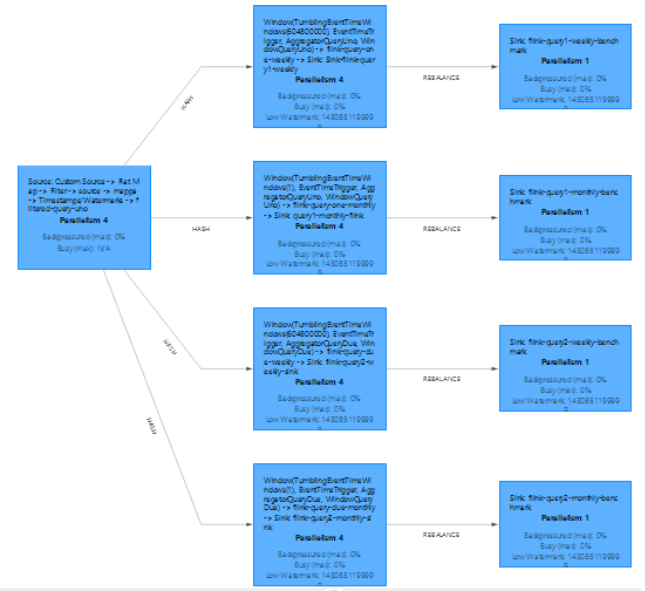

Figura1: Schema topologia in Flink

4. IMPLEMENTAZIONE QUERY

Ai fini del progetto, è stata considerata solamente l'area marittima con una latitudine compresa tra [32.0, 45.0] e una longitudine compresa tra [-6.0, 37.0].

Inoltre, l'area considerata è stata divisa in celle rettangolari di dimensione uguale. Le celle sono state ottenute dividendo la LAT in 10 settori, identificate dalle lettere che vanno da A a J, e dividendo la LON in 40 settori, identificate da numeri interi che vanno da 1 a 40.

Per ottenere ciò è stata calcolata la dimensione di ogni cella, sia in base alla longitudine che latitudine; la prima è stata ottenuta come la differenza tra latitudine massima e latitudine minima tutto diviso per il numero di settori e la seconda è stata ottenuta come differenza tra longitudine massima e longitudine minima tutto diviso per il numero di settori.

Ad ogni nave viene assegnata una cella in base alla longitudine e latitudine.

Per ogni nave è stata definita la tipologia:

- 35: navi militari;
- 60-69: navi per trasporto passeggeri;
- 70-79: navi cargo;
- Others: navi che non rientrano nei casi precedenti.

Inoltre, per distinguere il mar Mediterraneo Occidentale da quello Orientale; se la longitudine di ogni SHIP_ID è minore di 12 allora viene considerata come mar Occidentale altrimenti Orientale.

Per l'esecuzione delle due queries vengono presi solo i campi necessari: TRIP_ID, SHIP_ID, SHIP_TYPE, TYPE, LAT, LON, TIMESTAMP.

4.1 Finestre temporali

Vengono sfruttate delle finestre temporali, in particolare quelle settimanali e mensili.

Per la prima è stata utilizzata *TumblingeEventTimeWindows* di 7 giorni.

Per la seconda è stata sviluppata una *TumblingeEventTimeWindows* in modo tale da suddividere correttamente le tuple a prescindere dal numero di giorni presenti nello specifico mese. Il funzionamento si basa sull'utilizzo di un *assigner* che sfrutta l'*event time* per ottenere il mese di riferimento assegnando ad ogni dato una finestra con tempo di inizio impostato al primo giorno del mese e all'ultimo.

Su tutte le finestre viene applicata una funzione di aggregate *AggregatorFunction* associata con una *ProcessWindowFunction*, per aggregare i valori in modo incrementale con il loro arrivo alla finestra

4.2 Query uno

L'esecuzione della prima query inizia con il filtraggio dei dati considerando solo i valori che sono presenti nell'area del mar Mediterraneo Occidentale.

Il processamento avviene raggruppando i valori in base al numero di cella e per ciascuna tipologia di nave viene effettuata una media giornaliera.

Vengono sfruttate delle finestre temporali, in particolare quelle settimanali e mensili.

Come detto in precedenza, sono state sfruttate delle finestre temporali *TumblingeEventTimeWindows* di 7 giorni e di un mese.

La funzione di aggregazione si occupa, quindi, di mantenere aggiornata, per ogni SHIP_TYPE, il numero di SHIP_TRIP.

Infine, i valori prodotti vengono mappati in stringhe e indirizzati ai Sink per poterli pubblicare sulle rispettive topiche. Dato che bisogna calcolare le medie su due finestre temporali diverse, queste vengono indirizzate a due Sink diversi e in più altri due Sink per valutare le performance della query in termini di throughput medio e latenza.

4.3 Query due

L'esecuzione della seconda query avviene raggruppando i valori in base al tipo di mare: mar Occidentale o Orientale.

Come detto in precedenza, sono state sfruttate delle finestre temporali *TumblingeEventTimeWindows* di 7 giorni e di un mese.

La funzione di aggregazione si occupa, quindi, di mantenere aggiornata, per ogni fascia oraria 00:00-11:59 e 12:00-23:59 e per ogni cella il numero di frequentazione.

Per ottenere la classifica delle celle più frequentate, viene considerata la data di ogni SHIP_ID impostata a 12:00 e la data effettiva del transito. Se la data effettiva precede le 12:00 viene inserita all'interno del Set della mattina compresa tra 00:00-11:59, altrimenti viene inserita nel Set del pomeriggio compreso tra 12:00-23:59. Il Set permette di evitare che per ogni cella ci siano SHIP_ID duplicati.

Infine, viene calcolata la frequentazione di ogni cella in base al numero di navi che vi hanno attraversato dalle liste precedenti.

Per ogni cella, i valori sono stati ordinati in base al numero di frequentazione e presi in considerazione solamente i tre valori più alti.

Infine, i valori prodotti vengono mappati in stringhe e indirizzati ai Sink per poterli pubblicare sulle rispettive topiche. Dato che bisogna calcolare le medie su due finestre temporali diverse, queste vengono indirizzate a due Sink diversi e in più altri due Sink per valutare le performance della query in termini di throughput medio e latenza.

5. ANALISI DELLE PERFORMANCE

Le valutazioni sulle prestazioni sono state effettuate su un processore Intel(R) Core (TM) i7-6500U CPU @ 2.50GHz 2.60 GHz.

Le prestazioni vengono valutate con un numero di parallelismo pari a 4 e in base alla finestra temporale.

Per determinare le prestazioni, viene definito uno startTime nel momento in cui la window processa il primo dato e ogni qualvolta che ne processa un altro viene incrementato un contatore.

La latenza viene definita come: currentTime / contatore.

Il troughput viene definito come: contatore / currentTime.

Inoltre, vengono presi gli ultimi valori restituiti.

Le prestazioni della prima query vengono riportate di seguito:

Query uno	Settimanale	Mensile
Latenza	6785,45	8882,909091
Throughput	0,00014737	0,3530337

Le prestazioni della seconda query vengono riportate di seguito:

Query due	Settimanale	Mensile
Latenza	8194,5	15386,34
Throughput	0,000122	0,0016248

Confronto in termini di latenza tra le due query:

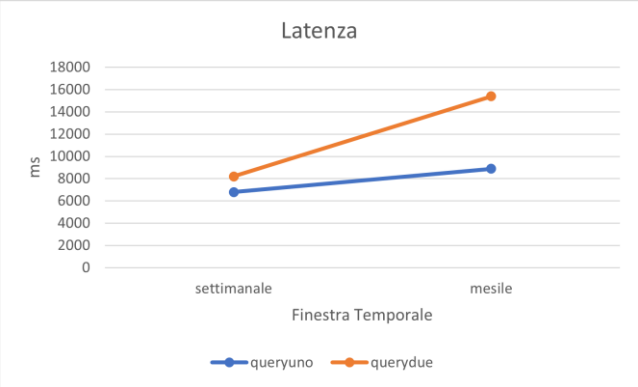

Figura2: Latenza

Confronto in termini di throughput tra le due query:

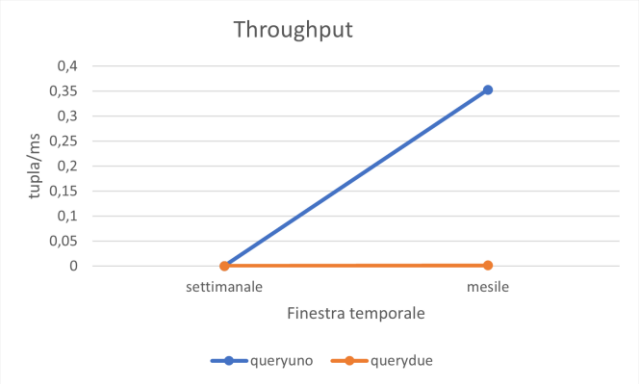

Figura3: Throughput

6. Conclusione

In conclusione, possiamo notare che per quanto riguarda la latenza essa aumenta all'aumentare della dimensione della finestra temporale e la latenza della seconda query risulta essere maggiore della prima.

Mentre, per quanto riguarda il throughput essa diminuisce all'aumentare della dimensione della finestra temporale e il throughput della prima query risulta essere maggiore della seconda.

6.1 References

[1] https://hub.docker.com/r/bitnami/kafka