Sistemi e Architetture per Big Data - A.A. 2017/18
Progetto 2: Analisi del dataset ACM DEBS Grand Challenge 2016 con Storm/Flink

Docenti: Valeria Cardellini, Matteo Nardelli
Dipartimento di Ingegneria Civile e Ingegneria Informatica
Università degli Studi di Roma "Tor Vergata"

Requisiti del progetto

Lo scopo del progetto è rispondere ad alcune query riguardanti il dataset DEBS 2016 Grand Challenge [2], utilizzando il framework Apache Storm o, in alternativa, Apache Flink.

Il dataset Grand Challenge 2016 contiene dati provenienti da una rete sociale, che rappresentano un grafo dinamico. I dati per il Grand Challenge 2016 sono organizzati in quattro flussi separati, ciascuno dei quali è fornito come un file di testo.

Il primo flusso di input, contenuto nel file `friends.dat` indica quando due utenti attivano una relazione di amicizia. Ogni riga del file ha il formato:

```
ts, user_id_1, user_id_2
```

dove:

- `ts` è il timestamp che indica quando la relazione di amicizia è stata stabilita;
- `user_id_1` è l'identificatore del primo utente (intero senza segno a 32 bit);
- `user_id_2` è l'identificatore del secondo utente (intero senza segno a 32 bit).

Il secondo flusso di input, contenuto nel file `posts.dat` indica quando un utente crea un nuovo post. Ogni riga del file ha il formato:

```
ts, post_id, user_id, post, user
```

dove:

- `ts` è il timestamp che indica quando il post è stato creato;
- `post_id` è l'identificatore del post (intero senza segno a 32 bit);
- `user_id` è l'identificatore dell'utente che ha creato il post (intero senza segno a 32 bit);
- `post` è il contenuto del post (stringa);
- `user` è il nome dell'utente che ha creato il post (stringa).

Il terzo flusso di input, contenuto nel file `comments.dat` indica quando un utente commenta un post. Ogni riga del file ha il formato:

```
ts, comment_id, user_id, comment, user, comment_replied, post_commented
```

dove:

- `ts` è il timestamp che indica quando il commento è stato creato;
- `comment_id` è l'identificatore del commento (intero senza segno a 32 bit);
- `user_id` è l'identificatore dell'utente che ha creato il commento (intero senza segno a 32 bit);
- `comment` è il contenuto del commento (stringa);
- `user` è il nome dell'utente che ha creato il commento (stringa).
- `comment_replied` è l'identificatore del commento che viene commentato (intero senza segno a 32 bit), il campo è vuoto se la riga rappresenta un commento ad un post;
- `post_commented` è l'identificatore del post che viene commentato (intero senza segno a 32 bit), il campo è vuoto se la riga rappresenta un commento ad un commento.

Il quarto flusso di input (`likes.dat`) non è usato nel progetto. Nei file di input i timestamp sono rappresentati nel formato `YYYY-MM-DDTHH:MI:Sec.SSS+ZZZZ`, ad es. 2010-02-22T13:56:38.686+0000.

Per gli scopi di questo progetto, si utilizza la versione ridotta del dataset di DEBS 2016 [1]. Le misure riportate nel dataset ricoprono più di 2 anni ed 8 mesi (inizio: 2010-02-01T05:12:32, fine: 2012-10-20T15:47:53). Tutti gli eventi sono ordinati in base al proprio timestamp; gli eventi aventi lo stesso timestamp sono ordinati in modo casuale.

Il progetto è dimensionato per un gruppo composto da **2 studenti**; per gruppi composti da 1 oppure 3 studenti, si vedano le indicazioni specifiche. Supponendo di effettuare il replay dei dati, si chiede di rispondere alla seguenti query in tempo reale:

1. Analizzare le relazioni di amicizia che si creano all'interno del social network, per ricavare statistiche sulla fascia oraria in cui queste relazioni si creano. L'output della query riporta il numero di relazioni che si formano in ogni ora, secondo lo schema:

   ```
   ts , count_h00 , count_h01 , ... , count_h22 , count_h23
   ```

   dove

   ```
   ts          // timestamp di inizio statistica
   count_h00   // numero di relazioni create da 00:00:00 a 00:59:59
   count_h01   // numero di relazioni create da 01:00:00 a 01:59:59
   ...
   count_h23   // numero di relazioni create da 23:00:00 a 23:59:59
   ```

   Tali statistiche aggregate dovranno essere calcolate sulle finestre temporali:

   - 24 ore (di event time),
   - 7 giorni (di event time),

- dall'avvio del social network.

Si gestisca opportunamente il caso di relazione bidirezionale/monodirezionale, in modo da conteggiare una sola volta la creazione di un'eventuale relazione bidirezionale (ad es. relazione di amicizia da A a B e relazione di amicizia da B ad A).

2. Fornire la classifica aggiornata in tempo reale dei 10 post che ricevono il numero maggiore di commenti (diretti). L'output della classifica ha il seguente schema:

```
ts, post_id_1, num_comments_1, post_id_2, num_comments_2, ...,
        post_id_10, num_comments_10
```

dove

```
ts              // timestamp di inizio statistica
post_id_1       // id del post classificato primo
num_comments_1  // numero di commenti del post classificato primo
...
post_id_10      // id del post classificato decimo
num_comments_10 // numero di commenti del post classificato decimo
```

La classifica dovrà essere calcolata sulle finestre temporali:

- 1 ora (di event time),
- 24 ore (di event time),
- 7 giorni (di event time).

3. Fornire la classifica aggiornata in tempo reale dei 10 utenti più attivi all'interno del social network. Il punteggio calcolato per ciascun utente si compone di tre parti ed è espressa nel formato (a, b, c), dove:

   - a è il numero di relazioni d'amicizia create dall'utente,
   - b è il numero di post scritti dall'utente,
   - c è il numero di commenti scritti dall'utente.

   Per la relazione d'ordine tra gli utenti, si consideri il punteggio complessivo associato all'utente, ovvero si consideri $a + b + c$ come punteggio per l'utente: ad es. dato l'utente U_1 con (a_1, b_1, c_1) e l'utente U_2 con (a_2, b_2, c_2), si ha che $U_1 > U_2$ se e solo se $a_1 + b_1 + c_1 > a_2 + b_2 + c_2$.

   L'output della classifica ha il seguente schema:

```
ts, user_1, rating_1, user_2, rating_2, ..., user_10, rating_10
```

   dove

```
ts          // timestamp di inizio statistica
user_1      // id dell'utente classificato primo
rating_1    // punteggio complessivo dell'utente classificato primo
...
user_10     // id dell'utente classificato decimo
rating_10   // punteggio complessivo dell'utente classificato decimo
```

La classifica dovrà essere calcolata sulle finestre temporali:

- 1 ora (di event time),
- 24 ore (di event time),
- 7 giorni (di event time).

Si chiede inoltre di valutare sperimentalmente i tempi di latenza ed il throughput delle tre query durante il processamento sulla piattaforma di riferimento usata per la realizzazione del progetto, riportando tali tempi nella presentazione. La piattaforma di data stream processing può essere un nodo standalone con Apache Flink o Apache Storm oppure in alternativa è possibile utilizzare un servizio Cloud per Hadoop (ad es. Amazon EMR o Google Dataflow), usando i rispettivi grant a disposizione. Gli studenti interessati ad usare Google Dataflow o comunque un servizio Cloud di Google per il deployment dei framework usati sono invitati a richiedere l'attivazione del grant (che non richiede un numero di carta di credito) ai docenti, segnalando lo username del proprio account Google.

Opzionale: rispondere alla query 1 usando Kafka Streams e confrontare, sulla stessa piattaforma di riferimento, le prestazioni in termini di tempo di latenza e throughput delle query ottenute dai due framework.

Per gruppi composti da 1 studente: si richiede di rispondere alle query 1 e 2.

Per gruppi composti da 3 studenti: in aggiunta ai requisiti sopra elencati, si richiede di utilizzare un altro framework di data stream processing per rispondere ad una delle tre query (si suggerisce la query 2) e di confrontare le prestazioni in termini di latenza e throughput con quelle ottenute dal primo framework scelto.

Svolgimento e consegna del progetto

Comunicare ai docenti la composizione del gruppo entro **giovedì 21 giugno 2018**.

Per ogni comunicazione via email è necessario specificare *[SABD]* nell'oggetto (subject) dell'email. Il progetto è valido **solo** per l'A.A. 2017/18 e deve essere consegnato **entro venerdì 6 luglio 2018** per poter raggiungere il punteggio massimo.

La consegna del progetto consiste in:

1. link a spazio di Cloud storage o repository contenente il codice del progetto;
2. lucidi della presentazione orale, da inviare via email ai docenti *dopo* lo svolgimento della presentazione.
3. *opzionale*: relazione di lunghezza compresa tra le 4 e le 6 pagine, usando il formato ACM proceedings (`https://www.acm.org/publications/proceedings-template`) oppure il formato IEEE proceedings (`https://www.ieee.org/conferences_events/conferences/publishing/templates.html`).

La presentazione si terrà **mercoledì 11 luglio 2018**; ciascun gruppo avrà a disposizione **massimo 15 minuti**.

Valutazione del progetto

I principali criteri di valutazione del progetto saranno:

1. rispondenza ai requisiti;
2. originalità;
3. organizzazione del codice;
4. efficienza;
5. organizzazione, chiarezza e rispetto dei tempi della presentazione orale.

Riferimenti bibliografici

[1] DEBS 2016 Grand Challenge: Small sample set. `https://www.dropbox.com/s/vo83ohrgcgfqq27/data.tar.bz2`, 2018.

[2] V. Gulisano, Z. Jerzak, S. Voulgaris, and H. Ziekow. The DEBS 2016 Grand Challenge. In *Proc. of 10th ACM Int'l Conf, on Distributed and Event-based Systems*, DEBS '16, pages 289–292, 2016.