The DEBS 2016 Grand Challenge

Vincenzo Gulisano
Chalmers University of Technology
Hörsalsvägen 11
41296 Gothenburg, Sweden
vincenzo.gulisano@chalmers.se

Zbigniew Jerzak
SAP SE
Münzstraße 15
10178 Berlin, Germany
zbigniew.jerzak@sap.com

Spyros Voulgaris
Vrije Universiteit Amsterdam
De Boelelaan 1081A
1081HV Amsterdam, The Netherlands
spyros@cs.vu.nl

Holger Ziekow
Hochschule Furtwangen
Robert-Gerwig-Platz 1
78120 Furtwangen, Germany
zie@hs-furtwangen.de

ABSTRACT

Pending...

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*

General Terms

Algorithms, Design

Keywords

event processing, streaming, utilities, geo-spatial

1. INTRODUCTION

The ACM DEBS 2016 Grand Challenge is the sixth in a series [4, 9, 5, 6] of challenges which seek to provide a common ground and uniform evaluation criteria for a competition aimed at both research and industrial event-based systems. The goal of the 2016 DEBS Grand Challenge competition is to evaluate event-based systems for real-time analytics over high volume data streams in the context of graph models.

The underlying scenario addresses the analysis metrics for a dynamic (evolving) social-network graph. Specifically, the 2016 Grand Challenge targets following problems: (1) identification of the posts that currently trigger the most activity in the social network, and (2) identification of large communities that are currently involved in a topic. The corresponding queries require continuous analysis of a dynamic graph under the consideration of multiple streams that reflect updates to the graph.

The data for the DEBS 2016 Grand Challenge is based on the dataset provided together with the LDBC Social Network Benchmark [2]. DEBS 2016 Grand Challenge takes up the general scenario from the 2014 SIGMOD Programming Contest [1], however, in contrasts to the SIGMOD contest, it explicitly focuses on processing streaming data and thus dynamic graphs. Details about the data, queries for the Grand Challenge, and information about evaluation are provided below.

2. DATA

The data for the 2016 Grand Challenge is organized in four separate streams, each provided as a text file. The first input stream indicates when two users enter a ”friendship” relationship – see Table 1 and Listing 1. The first input stream file name is *friendships.dat*.

Table 1: The set of attributes used in the *friendships.dat* input file

Attribute	Description
ts	timestamp indicating when a friendship was established
user_id_1	id of one of the users
user_id_2	id of the other user

Listing 1: First line from the *friendships.dat* file – one attribute per line of listing

```
XXX
YYY
ZZZ
PLEASE ADD ACTUAL DATA HERE
```

The second input stream indicates when a users creates a new post – see Table 2 and Listing 2. The second input stream file name is *posts.dat*.

The third input stream indicates when a users commnets on a post – see Table 3 and Listing 3. The third input stream file name is *comments.dat*.

Table 2: The set of attributes used in the *posts.dat* input file

Attribute	Description
ts	timestamp indicating when a post was created
post_id	unique id of the post
user_id	unique id of the user who created the post
post	string containing the post's content
user	string containing the user name of the post creator

Listing 2: First line from the *posts.dat* file – one attribute per line of listing

```
XXX
YYY
ZZZ
PLEASE ADD ACTUAL DATA HERE
```

The fourth input stream indicates when a users likes a comment – see Table 4 and Listing 4. The fourth input stream file name is *likes.dat*.

Each of the data files is sorted chronologically based on the timestamp (*ts*) attribute. Please note that the logical time of the processing engine is advanced through the timestamps in the respective tuples in the input streams. Specifically, the logical clocks of different queries may be advancing independently from each other. Input tuples with the same time stamp must be processed in the order of arrival.

3. THREE TOP SCORING POSTS QUERY

The goal of the first query is to compute the top three scoring active posts, producing an updated result every time the list changes. The total score of an active post P is computed as the sum of its own score plus the score of all its related comments. Active posts having the same total score should be ranked based on their timestamps – in descending order. And if their timestamps are also identical, they should be ranked based on the timestamps of their last received related comments – in descending order. A comment C is related to a post P if it is a direct reply to P or if the chain of C's preceding messages links back to P.

Each new post has an initial own score of 10 which decreases by 1 each time another 24 hours elapse since the post's creation. Each new comment's score is also initially set to 10 and decreases by 1 in the same way, i.e., every 24 hours since the comment's creation. Both post and comment scores are non-negative numbers, that is, they cannot drop below zero. A post is considered no longer active (that is, no longer part of the present and future analysis) as soon as its total score reaches zero, even if it receives additional comments in the future. The output format for the result stream is shown in Table 5.

Table 3: The set of attributes used in the *comments.dat* input file

Attribute	Description
ts	timestamp indicating when a comment was created
comment_id	unique id of the comment
user_id	unique id of the user who created the comment
comment	string containing the comment's content
user	string containing the user name of the comment creator
comment_replied	id of the comment being commented (-1 if this is a comment to a post)
post_commented	id of the post being commented (-1 if this is a comment to a comment)

Listing 3: First line from the *comments.dat* file – one attribute per line of listing

```
XXX
YYY
ZZZ
PLEASE ADD ACTUAL DATA HERE
```

Results must be sorted by their timestamp (*ts*) field. For updates caused by a timeout event, e.g., a post's score dropping every 24 hours, *ts* is the logical time of the event's generation, plus the logical time duration of the time window. For example, if a comment's expiration should produce some output at logical time 10:00, the logical clock goes past 10:00 only once the first tuple with a later timestamp is processed.

Please note that furthermore, input tuples with the same time stamp must be processed in the order of arrival. For example: if a post A expires at time X and another post B arrives at time X, followed by a comment C (for post A) arriving at the same time X, then the post A will expire. The reason for this is that post B will be processed before comment C. Processing of post B will trigger the expiration of the old post A. The subsequent comment C (for post A) will not have a post A to reference.

Specifically, when processing a new input tuple, processing steps should be performed in this order: (1) decrease the score of all previous posts (given the semantics of the query) (2) increase score of post related to the input tuple, (3) decrease score of posts expiring precisely on this timestamp, if any, and (4) discard posts with score equal to zero. Such a post would, thus, survive the transient state. However, a post whose score reached zaero at a timestamp earlier than the current input tuple, will not survive, even if the processing of that timeout happens to be triggered by the current input tuple.

Table 4: The set of attributes used in the *likes.dat* input file

Attribute	Description
ts	timestamp indicating when user liked a comment
user_id	unique id of the user who liked the comment
comment_id	unique id of the comment that was liked

Listing 4: First line from the *comments.dat* file – one attribute per line of listing

```
XXX
YYY
ZZZ
PLEASE ADD ACTUAL DATA HERE
```

The character „–" (a minus sign without the quotation marks) should be used for each of the fields (topX_post_id, topX_post_user, topX_post_commenters) of any of the top three positions that has not been defined. The logical time of the query advances based on the timestamps of the input tuples, not the system clock. Listing 5 shows the example output for the first query.

Listing 5: Output example for the three top scoring posts query

```
2010−09−19 12:33:01.923+0000,
   25769805561,Karl Fischer,115,10,
   25769805933,Chong Liu,83,4,
   −,−,−,−
2010−10−09 21:55:24.943+0000,
   34359739095,Karl Fischer,58,7,
   34359740594,Paul Becker,40,2,
   34359740220,Chong Zhang,10,0
2010−12−27 22:11:54.953+0000,
   42949673675,Anson Chen,127,12,
   42949673684,Yahya Abdallahi,69,8,
   42949674571,Alim Guliyev,10,0
```

An update to the result stream must be produced whenever any of the top three comments changes. Specifically, if any of the scores changes, but the comments remain the same, no output must be produced. When the end of the input stream is reached (defined as EOF value) all outstanding tuples (in the window) for each query must be processed and results must be output.

4. COMMUNITY INTEREST QUERY

The second query addresses the change of interests with large communities. This query represents a version of query type 2 from the 2014 SIGMOD Programming contest. Unlike in the SIGMOD problem, the version for the DEBS Grand

Table 5: The output format for the first query.

Attribute	Description
ts	the timestamp of the event that triggers a change in the top-3 scoring active posts appearing in the rest of the tuple
topX_post_id	the unique id of the top-X post
topX_post_user	author of top-X post
topX_post_commenters	number of unique users commenting on the top-X post, excluding the post author

Challenge focuses on the dynamic change of query results over time, i.e., calls for a continuous evaluation of the results.

The goal of the query is given an integer k and a duration d (in seconds), to find the k comments with the largest range. A range of a comment is defined as the size of the largest connected component in the graph defined by:

1. users who have liked that comment – see files *likes.dat* and *comments.dat*
2. the comment was created not more than d seconds ago
3. users who liked that comment who know each other – see *friendships.dat*

Specifically, the size of a component is determined by group members who: (1) all like the comment in question, (2) are all friends with each other, i.e., they form a clique.

The output of the second query includes a single timestamp *ts* and exactly k strings per line. The timestamp and the strings should be separated by commas. The k strings represent comments, ordered by range from largest to smallest, with ties broken by ascending lexicographical order. The k strings and the corresponding timestamp must be printed only when an input event triggers a change of the output. If less than k strings can be determined, the character „–" (a minus sign without the quotation marks) should be printed in place of each missing string. The output format for the result stream is shown in Table 6.

Table 6: The output format for the second query.

Attribute	Description
ts	the timestamp of the event that triggers a change in the output
comment_X	top k comments ordered by range

The field *ts* corresponds to the timestamp of the input data item that triggered an output update. For instance, a new friendship relation may change the size of a community with a shared interest and hence may change the k strings. The timestamp of the event denoting the added friendship relation is then the timestamp *ts* for that line's output. Also, the output must be updated when the results change due to the progress of time, e.g., when a comment is older that d.

Specifically, if the update is triggered by an event leaving a time window at $t2$, i.e., $t2$ = timestamp of the event + window size, the timestamp for the update is $t2$. As in the first query (see Section 3), timestamps refer to the logical time of the input data streams, rather than on the system clock.

Listing 6 shows the example output for the first query with a k value equal to three.

Listing 6: Output example for the community interest query

```
2010-10-28T05:01:31.022+0000,
    I love strawberries,
    _,
    _
2010-10-28T05:01:31.024+0000,
    I love strawberries,
    what a day!,
    _
2010-10-28T05:01:31.027+0000,
    I love strawberries,
    what a day!,
    well done
2010-10-28T05:01:31.032+0000,
    what a day!,
    I love strawberries,
    well done
```

5. ADDITIONAL REMARKS

For both queries it is assumed that the result is calculated in a streaming fashion - i.e.: (1) solutions must not make use of any pre-calculated information, such as indices and (2) result streams must be updated continuously.

The ranking of the submissions is done using the total duration of the execution and the average delay per stream. Specifically, solutions are ranked (a) by the total execution time as well as (b) by the average latency. The final result is calculated as a sum of average results from both query result streams. The average latency should be computed as the average of each output tuple latency, the latter measured as the difference between the system clock taken when logging an output tuple and the system clock taken when the processing of the input tuple triggering the result starts. The latency measurement always refers to the real time it takes to process a single input tuple. Even when an input tuple triggers an update caused by the timeout of some event, it is the present input tuple's latency that should be measured, which has nothing to do with the logical or real time elapsed since the past event that just timed out.

The final ranking is determined by a score that is the sum of the rank for execution time and the rank for delay. For solutions with the same overall score the solution with the lower execution time will be ranked higher.

6. LICENSE

All solutions submitted to the DEBS 2016 Grand Challenge are open source under the BSD license: https://opensource.org/licenses/BSD-3-Clause. A solution incorporates concepts, queries, and code developed for the purpose of solving the Grand Challenge. If a solution is developed within the context of, is built on top of, or is using an existing system or solution which is licensed under a different license than BSD, then such an existing solution or system maintains its existing license.

7. ACKNOWLEDGEMENTS

The DEBS Grand Challenge Organizing Committee would like to explicitly thank WSO2 (http://wso2.com) for sponsoring the DEBS 2016 Grand Challenge prize and the LDBC Council (http://www.ldbcouncil.org) for their help in preparing the test data set.

8. REFERENCES

[1] Curtis E. Dyreson, Feifei Li, and M. Tamer Özsu, editors. *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*. ACM, 2014.

[2] Orri Erling, Alex Averbuch, Josep-Lluis Larriba-Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat-Pérez, Minh-Duc Pham, and Peter A. Boncz. The LDBC social network benchmark: Interactive workload. In Timos K. Sellis, Susan B. Davidson, and Zachary G. Ives, editors, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 619–630. ACM, 2015.

[3] Ihab F Ilyas, George Beskales, and Mohamed A Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, 40(4):11, 2008.

[4] Zbigniew Jerzak, Thomas Heinze, Matthias Fehr, Daniel Gröber, Raik Hartung, and Nenad Stojanovic. The DEBS 2012 grand challenge. In François Bry, Adrian Paschke, Patrick Th. Eugster, Christof Fetzer, and Andreas Behrend, editors, *Proceedings of the Sixth ACM International Conference on Distributed Event-Based Systems, DEBS 2012, Berlin, Germany, July 16-20, 2012*, pages 393–398. ACM, 2012.

[5] Zbigniew Jerzak and Holger Ziekow. The DEBS 2014 grand challenge. In Umesh Bellur and Ravi Kothari, editors, *The 8th ACM International Conference on Distributed Event-Based Systems, DEBS '14, Mumbai, India, May 26-29, 2014*, pages 266–269. ACM, 2014.

[6] Zbigniew Jerzak and Holger Ziekow. The DEBS 2015 Grand Challenge. In Frank Eliassen and Roman Vitenberg, editors, *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems, DEBS '15, Oslo, Norway, June 29 - July 3, 2015*, pages 266–268. ACM, 2015.

[7] Nikos Mamoulis, Huiping Cao, George Kollios, Marios Hadjieleftheriou, Yufei Tao, and David W Cheung. Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 236–245. ACM, 2004.

[8] Mohamed F Mokbel, Xiaopeing Xiong, and Walid G Aref. Sina: Scalable incremental processing of continuous queries in spatio-temporal databases. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 623–634. ACM, 2004.

[9] Christopher Mutschler, Holger Ziekow, and Zbigniew Jerzak. The DEBS 2013 grand challenge. In Sharma Chakravarthy, Susan Darling Urban, Peter Pietzuch, and Elke A. Rundensteiner, editors, *The 7th ACM International Conference on Distributed Event-Based Systems, DEBS '13, Arlington, TX, USA - June 29 - July 03, 2013*, pages 289–294. ACM, 2013.

[10] Donna J Peuquet and Niu Duan. An event-based spatiotemporal data model (estdm) for temporal analysis of geographical data. *International journal of geographical information systems*, 9(1):7–24, 1995.