Proceedings of the 9th International Conference on Applied Informatics
Eger, Hungary, January 29–February 1, 2014. Vol. 1. pp. 181–187
doi: 10.14794/ICAI.9.2014.1.181

Virtual observatory for twitter messages

Gergő Gombos, Zoltán Vincellér, Attila Kiss

Eötvös Loránd University
{ggombos,vzoli,kiss}@inf.elte.hu

Abstract

A new research direction has emerged as the investigation of On-line Social Networks. Twitter is one of the most well-known social networks. Analysis of the Twitter is easier than other social networks because it provides the opportunity for collecting and downloading of a certain percentage of the messages without any restrictions. There are several researches on topics as detecting news and events, human behaviors, analyzing and mining of opinions. The on-line messages are available only through a continuous stream. To store the messages from the stream effectively and efficiently is a serious challenge against software system design and architecture. Every day about 10 GBs data are generated by this way and storing of this volume of data is not an easy task. In this paper we present a technique and architecture for collecting and storing the messages of the Twitter, and we present a prototype where data can be accessed for further analysis. Our system makes use specific techniques and methods of Oracle environment. Our software architecture approach is in contrast to previous solutions in which the systems use MSSQL or MySQL DBMS. We demonstrate that indexing and Job scheduler of the Oracle provide advantages to retrieve and handle large amounts of data.

Keywords: Database, Architecture, Social network

MSC: 68P20,68M01,68M11

1. Introduction

The usage of the social networks increased because the availability of the internet grew. Everyone uses some social network. The most well known systems are the Facebook, where we can share information with our friends, we can play some games with our friends or we can create groups. Another network is the FourSquare, where we can share our location, we can share in which restaurant we have lunch. Another network is the Twitter, where we can share short messages. These social networks provide an opportunity for researchers, which previously have been done only with surveys or questionnaires. Previously, if we had to know how many friends a certain person has, it could only take with questionnaire. Now it is enough to harvest social networks. If we want to know what the opinions of the customers about a product, then we need a questionnaire. Now we can find this information on the social networks. In paper [1] the authors presented a warehouse architecture. The technique has two phase, first phase build the short-term data cubes. The second phase stores the data with new compression method. This architecture is a data analyze and store implementation like our Virtual Observatory.

The most research work on the messages of the Twitter, because it gives an opportunity to download a small percentage of the messages. This small data come from the real messages of the twitter. If we do not store these messages, then we will not get again later. Therefore, it is important to build a system where data are collected and stored. Another problem is the size of data are to large. Many gigabytes of data arrive daily. These data are needed to manage effectively. In this paper we present an architecture and a pilot on this problem. In paper [2] the authors mentioned the full-text search required for text mining, but the it is very costly. They present an Oracle solution for this problem. In our system need full-text search for analyze twitter messages.

The paper is organized as follows: In the Section 2 we show some related systems, architecture plan. In the Section 3 we describe shortly, what is Twitter? In the Section 4 we present the main components of our system. Finally, in the Section 5 we conclude the result and describe our future plans.

2. Related work

In a previous paper [3] we described an effective system plan that can be used for storing and collecting the social networks. We summarized the requirements of the system in it. These are the basics of our system. The system is similar to two other systems, where the main goal is the collect of twitter messages.

The first system uses MySQL [4] backends. The system is designed for fast and effective search and query. This system uses Lucene and WordNet for the word search and it uses PostGIS for localized search.

The other system [5] uses MSSQL for data storage. They use the advantages of MSQL to build indices for localization. They show the process of collection of the messages, the structure of the database, the indices for language and localization.

Our system is based on the previous paper, but we use the Oracle database to store data.

3. Twitter

Twitter is the most widely used social network. On Twitter we can share 140 characters long messages. These call tweets. These messages can contain also images or links. These messages are received by users, that are following us. The users can share our messages and this process call retweet. For example, we can test a news spread with this process. When a news appears, we can see how fast increase the retweets on Twitter. Another feature of the tweets the geo information if the user allows that. This information is useful for analyzing where to spread the tweet or which areas they like or do not like things. Twitter gives us a Sample API from where we can get a small amount of the messages. The tweets from Sample API are the 1% of the real tweets. This small percentage is enough to be able to analyze these. Sample API is a stream, so if we do not store some messages, it is difficult to achieve again.

The messages can be used for marketing purposes where the main goal is to find those people who can be influenced to sell our products [6][7]. Other research analyzes the social behavior, these try to identify human behaviors: criminal propensity, depression, etc. [8][9]. The feedback of the US presidential election speeches are analyzed by twitter [10]. They are observed the opposition that is positive or negative impact on the campaign. Other research analyzes the news spread [11]. They looked how quickly spread the different types of news.

4. Architecture

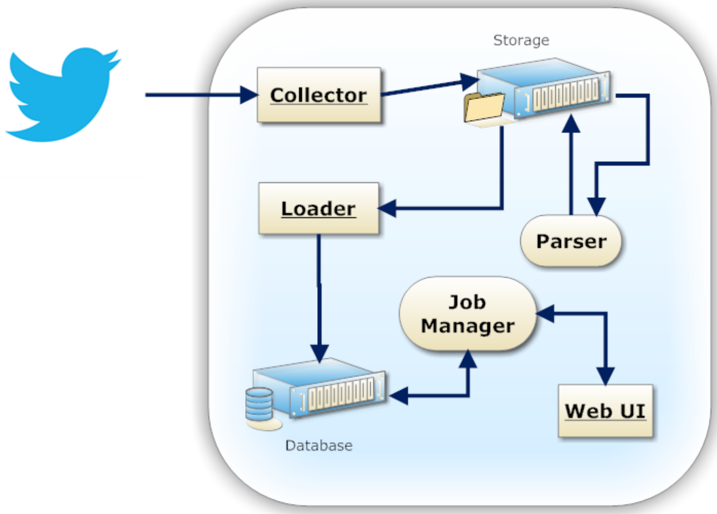


Figure 1: Architecture

The system contains five main components. Two main parts of the components are the loading part and the querying part. The Collector component collects the messages from the Twitter. Since these messages are in JSON format, so the Parser component transforms these to the right format, that need to the Loader. Then the Loader component loads the messages and user information into the database. On the query side the Job Manager schedules the queries that are coming from via the Web interface.

4.1. Collector

The collector is a key part of the system that collects the tweets from Sample API. Sample API is a stream, so storing the data is continuous. We receive that in JSON format. Each tweet message is a JSON object. The object contains the message, geo information (if enabled), the user and the user information. It the tweet is a retweet, it also includes the original tweet. The Collector collects these objects to files. It writes 400000 lines per file. If this lose the internet connection or the computer is turned off, the tweets cannot be recovered later. Therefore we use two Collector components on two different machines. Both collects the tweets, but the backup Collector deletes files that are older than a week. If something happens to any collection process, we found the tweets on the other machine. This component collects 12GB JSON tweets per day, and we need to store this data.

4.2. Parser and Storage

The Parser component generates csv files from the JSON files. This is necessary because the bulk load can handle it. The task of the Parser is to separate the tweet informations. These information is stored in several tables. One table stores the tweet messages, one store the users of the twitter and their information. On twitter we can tweet a message to a topic or we can mention another user. For this we have to use some special character. We use the # for the topic and @ for the user mention. The twitter API interprets these and the JSON Object has mention part where we see the mentioned users or topics. We store these informations in separate tables. Some tweets have media content such as images, videos, links. These we can get as another Objects of the JSON message. We store these also in separate tables. Another information that is the twitter send me is the retweet information. We store the connections between two tweets based on these informations. In this situation we get the original tweet message and not the retweeted message and we store this message too. We store the data in Oracle relation database.

4.3. Loader and Switch

A prepared csv files will be loaded into the database with the Loader component. During the load the database builds indexes on the data. For this reason, we drop the indexes before load, and we rebuild after the load. Because the tables have not indexes the response of the queries are slow. So, we use a second table, where the data are stored and it has indexes. This table can use the users and it call HOT table. The another table, where we load, we call COLD table. It has only indexes, that helps in the merge process, and this table stores the new messages. Weekly we synchronize the tables and switch the COLD and the HOT table.

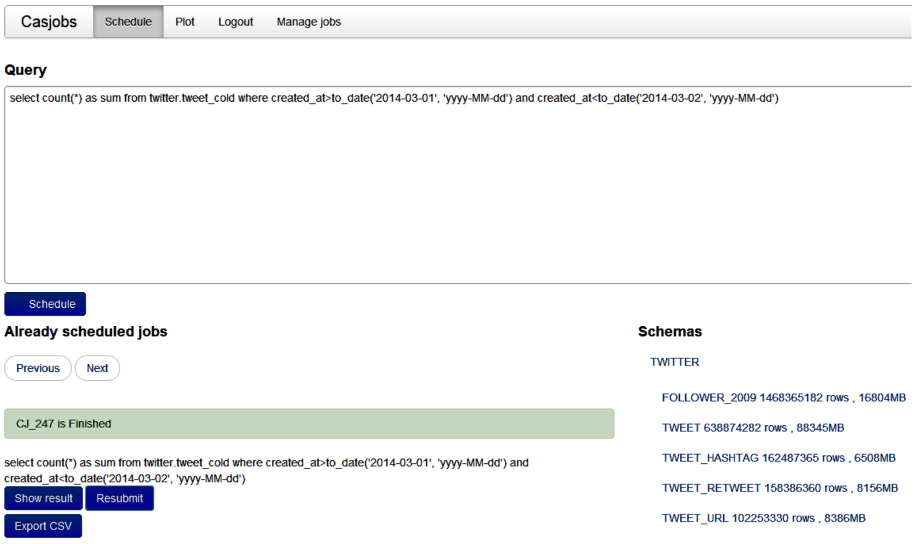
The loading process is as follows:
1 - to load the data into the DIFF tables
2 - to merge the DIFF tables to the COLD tables
3 - to build indexes for querying on COLD tables
4 - to switch the COLD and the HOT tables
5 - to drop the query indexes on the COLD tables
6 - to build indexes for the merge process on COLD tables (the COLD tables are the old HOT tables)
7 - to merge the DIFF tables to the COLD tables
8 - to truncate the DIFF tables
We split this process in three phases. The first phase prepares the switch of the COLD and HOT tables. The second phase switches the tables. The third phase synchronizes the COLD and HOT tables. The critical step is the second phase, because the users should not observe that the database is shut down. So we just rename the tables in this step and this is a fast operation. The two phases are slow processes. Table 1 shows the time of the process 75 JSON files. All JSON files are ∼1 GByte.

Process	Time (h:m:s)
Load	5:30:00
Merge to COLD	2:30:00
Build merge index	2:10:00
Build query index	4:10:00
Switch	0:0:2

Table 1: Time of the Load and Switch process

4.4. Job Manager and Web Interface

The data are available through a web interface (Figure 2), where we can send queries to the system. Because the query can run up to several days, the query does not run directly. The query is run by a Job Scheduler. We use the DBMS_JOB and the DBMS_SCHEDULER packages of the Oracle. These packages schedule the queries. The advantage of the system is the result of the query will not be lost when we leave the page. The results are placed in temporary tables. All users have quotas, which controls the size of the tables. If the user reaches this quota, it cannot send a new query until it makes free space. The interface makes it possible to export the results in CSV format and it provides information on the available schemas.



Query

```
select count(*) as sum from twitter.tweet_cold where created_at>to_date('2014-03-01', 'yyyy-MM-dd') and created_at<to_date('2014-03-02', 'yyyy-MM-dd')
```

Already scheduled jobs

CJ_247 is Finished

```
select count(*) as sum from twitter.tweet_cold where created_at>to_date('2014-03-01', 'yyyy-MM-dd') and
created_at<to_date('2014-03-02', 'yyyy-MM-dd')
```

Schemas

TWITTER

FOLLOWER_2009 1468365182 rows , 16804MB
TWEET 638874282 rows , 88345MB
TWEET_HASHTAG 162487365 rows , 6508MB
TWEET_RETWEET 158386360 rows , 8156MB
TWEET_URL 102253330 rows , 8386MB

Figure 2: User Interface

5. Conclusion and future work

In this paper we presented an architecture that is suitable for download tweet messages and store the messages for later analysis. We introduced some components of the system: the Collector, which collects the tweet messages, the Parser, which processes the JSON files, the Loader, which loads them in the database, the Job Manager, which schedules the queries.

The system has been running for several months and now includes more than 650 million tweet messages, 70 million twitter users, over 150 million retweet information and the system is collecting continuously the tweets.

Our further plans are making the three phase automatic. We would like to start the loading every week and the new tweets will be always available. In addition, we would like to change the scheduler, we would like to implement an another queue, where the fast queries can run. In this queue the time limit is 1 minute. If the query run over than 1 minute the system stop the query, but the query can run fast, then the user does not need to wait for the finish of the long running queries. In addition, we would like to provide an opportunity for plotting, which shows the result of the query on a diagram. Finally we would like to examine the various indexes that need for fast queries.

Acknowledgements. This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013). This work was completed with the support of the

Hungarian and Vietnamese TET (grant agreement no. TET 10-1-2011-0645). We are grateful to Mihály Morvai and Száva Maczika for helpful implementation.

References

[1] B. Rácz, C. I. Sidló, A. Lukács, and A. A. Benczúr Two-phase data warehouse optimized for data mining. In *Business Intelligence for the Real-Time Enterprises* (pp. 63-76). Springer Berlin Heidelberg. (2007)

[2] L. Kovács and D. Tikk Full-text Search Engines for Databases, *Encyclopedia of Artificial Intelligence*, IGI Global Publisher, Hersey (USA) 2008, ISBN 978-1-59904-849-9

[3] B. Molnár, Z. Vincellér Comparative study of Architecture for Twitter Analysis and a proposal for an improved approach, *CogInfoCom*, (2013)

[4] M. Oussalah, F. Bhat, K. Challis, and T. Schnier. A software architecture for twitter collection, search and geolocation services., *Knowl.-Based Syst.*, (2013), 105–120.

[5] L. Dobos, J. Szüle, T. Bodnár, T. Hanyecz, T. Sebők, D. Kondor, Zs. Kallus, J. Stéger, I. Csabai, G. Vattay A multi-terabyte relational database for geo-tagged social network data, *CogInfoCom* (2013)

[6] C. J. Case, D. L. King Twitter Usage in the Fortune 50: A Marketing Opportunity?. *Journal of Marketing Development and Competitiveness*, 5(3), (2011)

[7] E. Bakshy, J. M. Hofman, W. A. Mason, D. J. Watts Everyone's an influencer: quantifying influence on twitter. *In Proceedings of the fourth ACM international conference on Web search and data mining* (pp. 65-74). ACM. (2011).

[8] X. Wang, M. S. Gerber, and D. E. Brown Automatic crime prediction using events extracted from twitter posts. *In Social Computing, Behavioral-Cultural Modeling and Prediction* (pp. 231-238). Springer Berlin Heidelberg. (2012)

[9] J. Bollen, H. Mao, A. Pepe Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *In ICWSM.* (2011, July)

[10] H. Wang, D. Can, A. Kazemzadeh, F. Bar, S. Narayanan A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. *In Proceedings of the ACL 2012 System Demonstrations* (pp. 115-120). Association for Computational Linguistics. (2012, July)

[11] M. Hu, S. Liu, F. Wei, Y. Wu, J. Stasko, K. L. Ma Breaking news on twitter. *In Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems* (pp. 2751-2754). ACM. (2012, May)