Evaluating Scheduling Strategies in LOD Based Application

Usman Akhtar, Muhammad Bilal Amin ‡, and Sungyoung Lee
Department of Computer Science and Engineering,
Kyung Hee University, South Korea, Yonsin-si, 446–701
‡Korea Research Foundation (KRF) and UCLab, Kyung Hee University
Email: {usman,sylee}@oslab.khu.ac.kr, m.b.amin@ieee.org

***Abstract*—In this paper, we have evaluated the effectiveness of scheduling strategies in Linked Open Data based application for keeping local data caches up-to-date. We argue that the healthcare organizations are publishing data publicly, but consuming of data is difficult due to rapid growth of the linked data cloud. Most of the applications that are consuming linked data suffer from challenges such as change estimation and accuracy of index for keeping the data fresh for visualization. In this work, we have evaluated the quality of the data updates performed by the scheduling strategies. We have implemented the state-of-the-art web scheduling approaches; *ChangeRatio* and *ChangeRate* on linked dataset. We have concluded our evaluation that the strategies based on *ChangeRate* performed better than the *ChangeRatio*.**

I. Introduction

Linked Open Data (LOD) is a global information space for structurally represented and connect data. The LOD cloud provides a flexible way to integrate the data according to the LOD principles. Therefore, LOD has grown significantly with new content getting added and removed regularly. The changing behavior of Linked Data is extremely important in the applications such as data caching [1], web incremental crawlers [2] and linked data based health information systems. However, there exist some important challenges that need to be tackled by the scheduling strategies.

A. Identified Challenges

The first challenge is the sensitivity of the index model. Change in the LOD sources are not automatically propagated to the local copies; therefore due to the updated information there is a risk of having an inaccurate index and cache. To avoid the communication overhead most of the applications rely on the local copies, consequently the application has to synchronize the local copy with the data at the origin. Index models also rely on other resources like network bandwidth as spending efforts on updating requires more bandwidth to achieve the higher accuracy.

The second challenge is incomplete change history. In many of the scenarios, complete change information of the element is unavailable. As LOD source do not propagate change automatically, they rely on scheduling strategies to identify the changes that have occurred. There are several quality measures that have been proposed in the literature that are helpful in detecting the change in the LOD when updated. [3].

LOD sources have unreliable availability, as access to their repositories regularly turns offline, producing a denial of service. This plausible nature of LOD is unknown to the update strategies. Furthermore, as resources are often moved, removed, or even updated, thus, resulting in broken repository links.

B. Motivation

There is a need of user-friendly techniques to represent, query, and visualize the linked data. Healthcare organization such as World Health Organization (WHO), publish health related data online. However, most of the data are either available in Excel or PDF format by the portal. On the other hand, linked data provides an efficient way to publish the data and alleviate many challenges.

Figure 1. shows the typical architecture of LOD based application which provides the flexibility and usability where the data consumer such an expert is able to explore and visualize the data. Whenever the change occurs in the sources the scheduling strategies should keep the local copy of the data up-to-date. The main core of the architecture is the service and the model layer where the external sources are converted into the RDF triples and stored in the triplestore. As the data keep on changing the effective change estimation is needed to update the local copies of the data considering the limitation on the bandwidth. The purpose of this work is to evaluate the effectiveness of these scheduling strategies.

C. Solution Strategies

There are numerous applications that are related to the change detection and notification services [4] but the working implementations are rare. The estimation of the change proposed by [5], which estimate the change frequency of the web and to improve the incremental web crawler. Identified solution strategies such as *HTTP Meta-data Monitoring* [6] uses the header timestamps to detect the change in the data. Another solution strategy to estimate the change is Dynamic Linked Data Observatory (DYLDO) [7] which is fetching the entire content and determining locally. Hence, the existing

978-1-5386-1101-2/17/$31.00

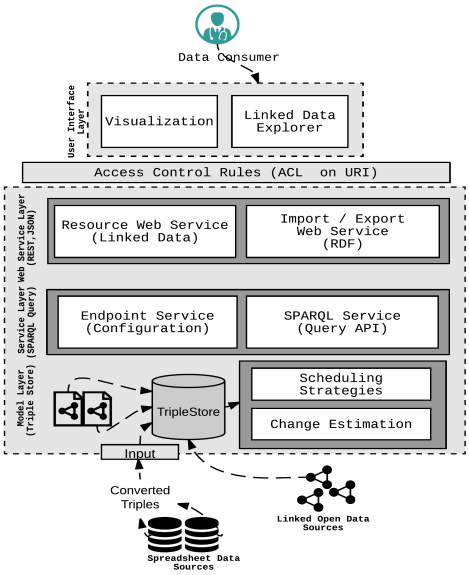

Fig. 1. Typical Workflow diagram of medical LOD application from querying, conversion and visualization

solution strategies are unable to cope the scalability and dynamics of the LOD cloud in an effective way.

In this paper, we have evaluated the effectiveness of the scheduling strategies on evolving Linked Dataset. We have implemented the *ChangeRatio* and *ChangeRate* scheduling strategies. Our main contribution is to measure the effectiveness of the updates performed by these scheduling strategies. We have evaluated the effectiveness of the updates considering the limitation on the bandwidth. Our goal is to show which updated strategies produce better updates in term of the data accuracy and freshness.

II. Related Work

There are various implementations that looked into the characteristics of the LOD cloud. Some have conducted the structural analysis in order to obtain the characteristics of the data [8]. In literature, there are various works that have investigated on the characteristics of LOD and estimation of the change. Researchers in [5] have estimated the change frequency of data to improve the web crawlers, web caches and to help the data mining tasks. Other works are related to change in content of a RDF documents that is crawled for the period of 24 weeks. According to the author Etag and Last-modified HTTP header were applied to typically indicate the change. There is a variety of existing works related to the change detection of the query results on dynamic data sets. Among them the most prominent work on the query caching is [1], but the working implementations are rare. Currently, there is a limited research that focuses on the impact of the cache on LOD. Most of the available work is rich in database literature where query cache occurs; however cache with the SPARQL engine is not considered relevant [9].

Since, LOD cloud is a global information space and it is structurally connect data items. The distributed web based nature of data motivates many application to keep local copies of the data. Due to the dynamic nature of the linked data many applications need to keep updating the local copy of the data. The main problem is when to perform the updates. To solve this problem researchers have investigated scheduling update strategies to periodically update the LOD caches. We also investigate update strategies that are proposed in the literature for updating linked data caches. Another worth mentioning work by Magnus Knuth et al. [10] discusses the problem of scheduling refresh queries for large number of registered SPARQL queries. They have investigated various scheduling strategies and compared them experimentally. The main contribution of their work is an empirical evaluation on the real world SPARQL queries.

III. Existing Scheduling Strategies

The aim of the scheduling strategies is to prioritize what data sources should be first considered in order to keep the local data caches up-to-date.

A. Scheduling Strategies Based on ChangeRatio

ChangeRatio [11] scheduling strategies are based on the past change history of the LOD sources. These strategies can estimate "how often" the item changes and then decide when to schedule the sources. For instance, if the scheduled strategies identifies that the items have changed then it is easy to estimate how often the sources changed based on the previously observed history of the item. Informally, *ChangeRatio* strategies count how many items in the LOD cloud changes over each snapshot. *ChangeRatio* defines the number of the changes in the LOD and is represented as:

$$ChangeRatio = \sum_{i \in R} w_i.1(o_i) \tag{1}$$

In equation 1, the function $1(o_i)$ indicate the change occurred in an item. *ChangeRatio* is useful to store the change history of the items in the Linked Open Data (LOD).

B. Scheduling Strategies Based on ChangeRate

LOD sources change frequently, instead of finding the change between the two snapshots it is better to find the evolution over the period of time. Scheduling strategies based on the *ChangeRate* quantify the change and measure the change of the LOD cloud. The *ChangeRate* of the LOD can be represented as Δ that quantifies the changes between the two datasets such as different snapshot of the LOD sources. The strategies based on the *ChangeRate* quantify the evolution of the LOD over the period of the time [10]. It is represented as follows:

$$ChangeRate = \sum_{i=0}^{j} \frac{\Delta\left(X_i, X_{t_{i-1}}\right)}{t_i - t_{i-1}} \tag{2}$$

TABLE I
THEORETIC COMPARISON OF LINKED DATA SCHEDULING STRATEGIES

Strategies	Feature	Strength	Weakness
ChangeRatio	Measure how much data items in cache that are out-to-date.	Update from most to the least change linked data source.	Does not quantify the evolution over a period of time.
ChangeRate	Compare the data sets based on the distance function such as Jaccard and dice coefficient.	Prioritize the sources based on the Jaccard distance from most to least data source.	Does not capture the change behaviour of the data source.

These scheduling strategies calculate the aggregation of the results changes over time by using the Jaccard distance between the subsequent results. In Table 1, we compare both the state-of-the-art strategies on theoretical point of view. One of the interesting finding during the comparison is the limitation on the evolving dataset. Both the strategies unable to cope the dynamics of the linked data.

IV. EXPERIMENTAL SETUP

In this section, we have evaluated the effectiveness of the scheduling strategies. Our main goal is to find out the most appropriate strategies for keeping LOD caches up-to-date.

A. Dataset

For our experiments, we use the Dynamic Linked Data Observatory (DYLDO) [7], which monitor the fixed set of the Linked Open Data documents on weekly basis. In DYLDO [1], each week fixed set of documents are retrieved and stored. The DYLDO datasets contain a large number of the well-known LOD sources such as dbpedia.com. On average, DYLDO contains more than 600 data sources. On every snapshot the number of the sources goes online and offline, due to the dynamic nature of the linked data cloud. For better insights into our evaluation, we have chosen the dbepedia.com in every snapshot over the three months. In each snapshot of the dbpedia.com the number of the sources increases or decrease. On average there are 3981 sources available in dbpedia.com. In every snapshot, many of the statements are added and removed. On average 4796 statements are added, and 4800 statements are removed from the three months period.

B. Queries

It is essential to use queries that matches with the dataset. We use the Linked SPARQL Queries dataset (LSQ)[2] that contain query for Dbpedia and matches the current structure of the data. These LSQ provide a support to query different snapshots of the datasets and check the dynamics of the sources. Whenever data sources change the scheduling strategies need to prioritize which data sources need to update.

[1]http://swse.deri.org/dyldo/data/

[2]http://aksw.github.io/LSQ/

C. Evaluation Matrix

We have evaluated *ChangeRatio* and *ChangeRate* to check the quality of the local data caches updates. The quality of the update strategy is measured in terms of the recall and precision.

—***Effectiveness:*** Scheduling strategies should only evaluate the data that have changed, which reduced the load on the SPARQL endpoints.

—**Quality Measures:** The quality of the updates can be measured in term of the Precision and Recall [12]. The purpose is to find out the strategies that perform better local update of the data caches. We assume that the data is fetched at the fixed point on time t and we denote the size of the dataset that is fetched from the source by $|X_{c,t}|$ that contain the number of the triples in a dataset at context c at the point in time t. Whereas precision is defined as the portion of the cached data that are up-to-dated and recall is the portion in the LOD cloud that is identical to the cached data.

$$Precision = \frac{\sum |\chi_{c,t} \bigcap \chi^{'}_{c,t}|}{\sum x^{'}_{t} |\chi^{'}_{c,t}|} \tag{3}$$

$$Recall = \frac{\sum |\chi_{c,t} \bigcap \chi^{'}_{c,t}|}{\sum x^{'}_{t} |\chi_{c,t}|} \tag{4}$$

When the application wants to update the local copies from the data sources $c \epsilon C$ at the time t_{i+1}. The complete dataset can be fetched in time $X_{t_{i+1}}$. However, this would also consider to retrieve the most recent version of the data $X_{c,t_{i+1}}$. Due to the limitation on the bandwidth only the fraction of the data can be retrieved from the LOD cloud.

D. Experimental Results

Overall the *ChangeRate* strategies perform better on the datasets considering the limitation on the bandwidth. However, the *ChangeRatio* show very similar performance as LOD sources vary in size. Most of the time the changes are random and both the *ChangeRate* and the *ChangeRatio* strategies work very well on these data sources. Both of these strategies are also performing well on the limited bandwidth provided. A strategy based on the *ChangeRate* relatively detects many updates but neglects the less evolving sources. On the other hand, *ChangeRatio* detects the fair amount of changes. It is also observed that strategies that are based on the previously observed changes are able to produce better results. Both of the *ChangeRate* and the *ChangeRatio* neglect the less frequent changes of the sources.

Table.II shows that scheduling strategies based on the *ChangeRate* outperform the *ChangeRatio* as it rely on the past result history of the query. Although *ChangeRatio* executes lot of irrelevant queries, it performs better and it detects relatively many updates without execution of too many irrelevant queries. The quality of the scheduling strategies are measured in term of the precision and recall as shown in the Fig.2. where *ChangeRate* scheduling strategies perform better quality updates than the *ChangeRatio* for very small bandwidth usage.

TABLE II
EVALUATION EFFECTIVENESS OF THE SCHEDULING STRATEGIES

Scheduling Strategies	total query execution	Irrelevant	Relevant	Maximum miss
Change Ratio	2,395,472	2,381,306	14,166	7
Change Rate	1,986,100	1,970,895	15,205	5

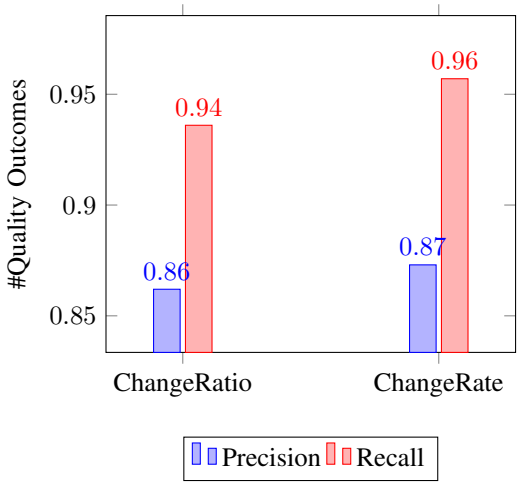

Fig. 2. Showing the Quality Outcomes of *ChangeRatio* and *ChangeRate* of Scheduling Strategies

We started with the perfect caches and after some time when the data evolve we rescheduled both the strategies as our main goal is to only capture the change part of the sources instead of download all the resources.

A very similar trend confirms the output of the evaluation for the fixed bandwidth scenario. The update is performed by the scheduling strategies and it is observed that even in the limited bandwidth *ChangeRate* perform better quality of the updates. Overall, the results from the evaluation confirm that the strategies based on the *ChangeRate* are more appropriate to estimate and update the local data caches.

V. CONCLUSION

Linked open data provide an efficient way to publish the data. Although these data are evolving, hence it brings some challenges related to the sensitivity of the index model, incomplete change history and unavailability of the data sources. A lot of LOD applications use scheduling strategies to keep the local data caches up-to-date. In the limited bandwidth, scheduling strategies need to prioritize which sources to schedule and predict for changes. We have evaluated the effectiveness of different strategies and observed that the *ChangeRate* based strategies are more appropriate to estimate the changes even in the limited resource provided. Our evaluation results shows that the strategies based on the *ChangeRate* reduces the load on the SPARQL endpoint and effectively execute the queries. Whenever the change occurs in the LOD cloud *ChangeRate* based scheduling perform quality updates on the local data caches. In the future, we will combine the different scheduling into a hybrid scheduler to overcome the shortcoming of the *ChangeRate* and *ChangeRatio* strategies.

Acknowledgments: This work was supported by the Industrial Core Technology Development Program (10049079, Develop of mining core technology exploiting personal big data) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea), this research was also supported by Korea Research Fellowship program funded by the Ministry of Science, ICT and Future Planning through the National Research Foundation of Korea(NRF-2016H1D3A1938039) and this research was also supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2017-01629) supervised by the IITP(Institute for Information & communications Technology Promotion).

REFERENCES

[1] M. Martin, J. Unbehauen, and S. Auer, "Improving the performance of semantic web applications with sparql query caching," *The Semantic Web: Research and Applications*, pp. 304–318, 2010.
[2] J. Cho and H. Garcia-Molina, "The evolution of the web and implications for an incremental crawler," Stanford, Tech. Rep., 1999.
[3] H. Cho, Junghoo, "Synchronizing a database to improve freshness."
[4] A. Passant and P. N. Mendes, "sparqlpush: Proactive notification of data updates in rdf stores using pubsubhubbub." in *SFSW*, 2010.
[5] J. Cho and H. Garcia-Molina, "Estimating frequency of change," *ACM Transactions on Internet Technology (TOIT)*, vol. 3, no. 3, pp. 256–290, 2003.
[6] R. Dividino, A. Scherp, G. Gröner, and T. Grotton, "Change-a-lod: does the schema on the linked data cloud change or not?" in *Proceedings of the Fourth International Conference on Consuming Linked Data-Volume 1034*. CEUR-WS. org, 2013, pp. 87–98.
[7] T. Käfer, J. Umbrich, A. Hogan, and A. Polleres, "Towards a dynamic linked data observatory," *LDOW at WWW*, 2012.
[8] S. Auer, J. Demter, M. Martin, and J. Lehmann, "Lodstats–an extensible framework for high-performance dataset analytics," in *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 2012, pp. 353–362.
[9] K. Kjernsmo, "A survey of http caching implementations on the open semantic web," in *European Semantic Web Conference*. Springer, 2015, pp. 286–301.
[10] M. Knuth, O. Hartig, and H. Sack, "Scheduling refresh queries for keeping results from a sparql endpoint up-to-date (short paper)," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2016, pp. 780–791.
[11] J. Cho and A. Ntoulas, "Effective change detection using sampling," in *Proceedings of the 28th international conference on Very Large Data Bases*. VLDB Endowment, 2002, pp. 514–525.
[12] W. Zheng, L. Zou, W. Peng, X. Yan, S. Song, and D. Zhao, "Semantic sparql similarity search over rdf knowledge graphs," *Proceedings of the VLDB Endowment*, vol. 9, no. 11, pp. 840–851, 2016.