Cloud storage platform for efficient RDF compression

Y X Sun¹, S H Lee² and Y J Lee³

¹,²,³ School of Computer Science and Engineering, Kyungpook National University, Daegu, Korea.
syx921120@gmail.com

Abstract. In current web environment, a large of RDF datasets are producing. Thus, the storage of RDF data is becoming an important part of Semantic Web development. Due to the triple structure of RDF data, how to efficiently compress RDF data as much as possible without breaking the integrity of data is become one of the important issues. By Comparing previous research, we found that people had made many efforts in the compression of static, large RDF datasets. In this paper, we not only propose a cloud-based compression approach based on previous research, but also made efforts in data security. Due to the openness of the internet, anyone can publish their Linked Data. At this time, the security of personal data will become very important. Our cloud storage platform not only provides publishers with efficient compression services but also maximizes the security of data. To improve the openness of the platform, we adopt REST (REpresentational State Transfer) architecture to exchange data between publishers and cloud.

1. Introduction

The principles of Linked Data were first outlined by Berners-Lee in 2006 [1]. Linked Data is about utilizing the Resource Description Framework (RDF) and the Hypertext Transfer Protocol (HTTP) to publish structured data on the Web and to connect data between different datasets. In LOD (Linked Open Data) cloud [2], From 2007 to 2017, the number of datasets had increased by nearly 100 times. Although the growth of datasets is beneficial for the development of semantic web, we must face the storage pressure of RDF data. In [3], three basic RDF compression approaches are proposed. By analysing the experimental results, we find that RDF data is highly compressible. In [4] a binary RDF representation was proposed, because it was consisted by Header information, a Dictionary, and the actual Triples structure, thus it was named HDT. It is mainly about how to improve high levels of verbosity/redundancy and weak machine-processable capabilities in the description of these datasets. According to the authors, it outperforms existing compression solutions for efficient RDF exchange. In [5] authors proposed a logical Linked Data compression approach. It is a novel lossless compression technique for RDF datasets, called Rule Based Compression (RB Compression) that compresses datasets by generating a set of new logical rules from the dataset and removing triples that can be inferred from these rules. It can prune more than 50% of the original triples without affecting data integrity. Although these approaches can compress RDF datasets efficiently, their datasets have a common point is large and static. The user-generated datasets have an important feature is dynamic and need to pay more attention to data security. Thus, for addressing these issues, we proposed a dictional-based cloud storage platform. Not only we can reduce local storage pressure but also improve the security of published data. Users don’t need to access local server to modify published data, just only by REST-APIs[6] to modify dictionary data in the cloud platform.
In related work, we will summarize these compression approach’s feature. Later, we will propose cloud storage platform’s system architecture and service. In experiment part, by experiment analysis we can obtain actual compression performant. Finally, in conclusion part we will make an evaluation about cloud storage platform.

2. Related work
Table 1 shows three basic compression approaches in [3]. Direct Compression uses three well-known techniques such as gzip, bzip2, and ppmdi. Adjacency Lists focus the data repeatability. For example, the set of triples can be converted into the adjacency list \( s \rightarrow [(p_1, \text{ObjList}_1), \ldots, (p_k, \text{ObjList}_k)] \). Dictionary+Triples split the data into dictionary of elements and the triples substituting for each element. Triples can be represented by adjacency lists. In order to obtain better reflect the influence of data structure on the compression results, we adopt the same compression approach to compress data of different structures. Here, gzip approach was adopted.

| Approach               | Description                                      |
|------------------------|--------------------------------------------------|
| Direct Compression     | Compression with gzip, bzip2, ppmdi              |
| Adjacency Lists        | Convert RDF data to adjacency lists              |
| Dictionary + Triples   | Split data into dictionary and triples           |

In figure 1, Direct Compression, Adjacency Lists, and Dictionary+Triples compressions refer to as DC, AL, and DT. We analyze the three basic approaches and test them with a well-known real world RDF data set, DBpedia[7]. From this test, we found that the using of AL greatly reduce storage space. DT shows slightly worse performance than other two approaches. The main reason of the DT low compression is dictionary itself. Nevertheless, we decide to adopt DT, because DT is focused on the graph nature of RDF. To reduce the dictionary storage pressure, therefore, we implement a cloud storage platform for the dictionary management. We could achieve the best storage performance by separating the dictionary and adjacency lists. This conclusion is also verified in [3]. This provides a powerful theoretical basis for the development of our cloud storage platform.

![Figure 1. Performance of different compression approaches.](image-url)
In [8] Because each RDF statement is made of three different terms: a subject, a predicate, and an object. Thus, authors proposed a scalable compression approach for large RDF datasets by using the MapReduce programming model [9] and dictionary encoding technique. According author’s description this technique can be used by all RDF applications that need to efficiently process a large amount of data, such as RDF storage engines, network analysis tools, and reasoners.

The aforementioned approaches are focus on large, static RDF databases, some of these technologies are very helpful for the development of our cloud storage platform. In this paper, we not only define a lossless compression approach but also the dynamic modification and data security can be realized.

3. System architecture

Figure 2 shows the architecture of the cloud storage platform. The cloud platform is divided into three layers, DAL (Data Analysis Layer), DFL (Data Forward Layer) and DSL (Data Storage Layer). DSL is divided into a register database and a triples database. The register database mainly stores user’s ID and forward URL for target server’s REST-API. The triples database mainly stores hash values and string values of triples and they are independent of each other. Notice, server A and server B are not part of the cloud platform. It is user's personal server, that's why we set up the register database. Therefore, users can change the access path of the server at any time by modifying the information of the register database. The following explains two main services for the cloud storage platform.

Data Parsing and Registration Service: Registration must be completed before users upload original RDF datasets. In DSL, the cloud platform will parse the original data and stores it in a triples database. In the meantime, a compression file is generated. In this case, the compressed file is an adjacency list based on hash values. Users can download this compression file by REST-API and stores it in local server.

Data Query Service: If users want to query data that they only need to access the cloud platform, in register database they will obtain the URL of target server, then they can obtain result by DFL. At this point, the result is an array of hash values. If users want to obtain final results they need to convert the hash value into a real value in the triples database. Complete above steps, users can get the final results that they want.

4. Experiment

For making the results more convincing, we downloaded datasets from three different fields. The DBpedia dataset contains RDF information extracted from Wikipedia. The DrugBank dataset contains information on drugs and drug targets. Table 3 gives the characteristics of the collected dataset. It is...
widely used by the drug industry, medicinal chemists, pharmacists, physicians, students and the public institutions. The LinkedGeoData dataset is a large spatial knowledge base which has been derived from OpenStreetMap for the Semantic Web. Table 2 shows detailed performance of compression, our adjacency list structure uniformly adopts $s \rightarrow \{[p_1, \text{ObjList}_1], \ldots, [p_k, \text{ObjList}_k]\}$.

|                     | DBpedia | DrugBank | LinkedGeoData |
|---------------------|---------|----------|---------------|
| Original size (MB)  | 3.94    | 144      | 327           |
| Number of triples   | 31,050  | 766,920  | 2,207,295     |
| Compressed size (KB)| 173     | 2,821    | 12,182        |
| Compression rate    | 4.29%   | 1.91%    | 3.63%         |

Table 2. Result of cloud storage platform compression.

From the results of the experiment, we can find that we can greatly reduce the personal storage pressure for RDF datasets by utilizing the cloud storage platform. The compression rate is not proportional to the number of triples. This is because the data content of each RDF datasets is different. On the whole, our experiment has achieved satisfactory results.

5. Conclusion

Our proposed solution not only achieves high performance lossless compression but also provides an open cloud storage platform. Due to our processing is done over the network, the user experience is not as good as centralized storage approach. But we believe that with the development of the network this gap will become more and more small. In terms of the storage security, our cloud storage approach is more secure compared to traditional compression approaches. Because subjects, predicates, and objects are stored in databases separately in the cloud, and their logical relationships are stored in the personal server, the eavesdropper cannot obtain complete information even if the cloud data is hacked. Security is one of the important factors for developing Semantic Web. Especially, the cloud platform provides standard interface to the update and delete data, it means that we don’t need to access the personal server frequently. It is useful for IoT(Internet of Thing) devices that have less processing capacity and data is dynamic.

References

[1] Bizer C, Heath T, Idehen K and Berner-Lee T 2008 Linked Data on the Web (LDOW2008) Beijing China p21
[2] LOD (linked open data) sets. From website: http://lod-cloud.net/.
[3] Javier D. F, Claudio G and Miguel A. M 2010 RDF Compression: Basic Approaches p 1091
[4] Fernández J D, Martínez-Prieto M A, Gutiérrez C, Polleres A and Arias M 2013 Binary RDF representation for publication and exchange (HDT). Web Semantics: Science, Services and Agents on the World Wide Web p22
[5] Joshi A, Hitzler P and Dong G 2013 Logical linked data compression. In The Semantic Web: Semantics and Big Data p 170
[6] Fielding R T, 2000 Representational State Transfer (REST): Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine.
[7] Jens L., Robert I, Max J, Anja J, et al. 2012 DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia pp 1-5
[8] Urbani J, Maassen J, Drost N, Seinstra F and Bal H 2013 Scalable RDF data compression with MapReduce P 24
[9] Dean J and Ghemawat S M 2004 simplified data processing on large clusters In Proceedings of the USENIX Symposium on Operating Systems Design & Implementation (OSDI) p 137–147
Acknowledgments
This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2016R1D1B02008553).