International Journal of Research
Available at https://edupediapublications.org/journals
Special Issue on Conference Papers
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 06
March 2018

Secured Distributed Rdf Data In Cloud Storage

Ms.Syed Thisin
Assistant Professor – CSE, Visvesvaraya College of Engineering & Technology
syedthaisin@gmail.com

Abstract: **Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. Consumer-oriented applications such as financial portfolios to deliver personalized information, to provide data storage.Speculate that large scaledata analysis tasks, decision support systems, and application specific data marts are more likely to take advantage of cloud computing platforms than operation. A distributed system is a model in which components located on networked computers communicate and coordinate their actions by passing messages. Efficient RDF data management is one of the cornerstones in realizing the Semantic Web vision. In the past, different RDF storage advanced techniques like clustering or vertical partitioning on the predicates.RDF actually encodes rich and complex graphs mixing both instance and schema-level data. Sharding such data using classical techniques or partitioning the graph using traditional min-cut algorithms leads to very inefficient distributed operations and to a high number of joins. In this paper, we describe DiploCloud, an efficient and scalable distributed RDF data management system for the cloud. Contrary to previous approaches, DiploCloud runs a physiological analysis of both instance and schema information prior to partitioning the data. In this paper, we describe the architecture of DiploCloud, its main data structures, as well as the new algorithms we use to partition and distribute data. We also present an extensive evaluation of DiploCloud showing that our system is often two orders of magnitude faster than state-of-the-art systems on standard workloads.**

Index terms--RDF, triple stores, cloud computing, secured data, Big data

1. INTRODUCTION

1.1 CHARACTERISTICS AND SERVICES MODELS

The salient characteristics of cloud computing based on the definitions provided by the National Institute of Standards and Terminology (NIST) are outlined below:

On-demand self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.

Broad network access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).

Resource pooling: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

Rapid elasticity: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

Measured service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some

International Journal of Research
Available at https://edupediapublications.org/journals
Special Issue on Conference Papers
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 06
March 2018

level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

1.2 PREAMBLE

The cheaply provision computing resources, for example to advent of cloud computing enables to easily and test a new application or to scale a current software installation elastically. The complexity of scaling out an application in the cloud (i.e., adding new computing nodes to accommodate the growth ofsome process) very much depends onthe process to be scaled. Often, the task at hand can be easily split into a large series of subtasks to be run independently and concurrently. Such operations are commonly called embarrassingly parallel. Embarrassingly parallel problems can be relatively easily scaled out in the cloud by launching new processes on new commodity machines. There are however many processes that are much more difficult to parallelize, typically because they consist of sequential processes (e.g., processes based on numerical methods such as Newton's method). Such processes are called inherently sequential as their running time cannot be sped up significantly regardless of the number of processors or machines used. Some problems, finally, are not inherently sequential but are difficult to parallelize in practice because of the profusion of inter-process traffic they generate. Scaling out structured data processing often falls in the third category.

Traditionally, relational dataprocessingisscaled out by partitioning the relations and rewriting the query plans to reorder operations and use distributed versions of the operators enabling intra-operator parallelism. While some operations are easy to parallelize (e.g., large-scale, distributed counts), many operations, such as distributed joins, are more complex to parallelize because of the resulting traffic they potentially generate. While much more recent than relational data management, RDF data management has borrowed many relational techniques; Many RDF systems rely on hash-partitioning(on triple or property tables, see below Section 2) and ondistributedselections, projections, and joins. Our own Grid Vine system was one of the first systems to do so in the context of large-scale decentralized RDF management. Hash partitioning has many advantages, including simplicity and effective load-balancing. However, it also generates much inter-process traffic, given that related triples (e.g., that must be selected and then joined) end up being scattered on all machines.

2. RELATED WORKS

With the reference of**,** "tracking rdf graph provenance using rdf molecules"“l. Ding", "y. Peng", "p. P. Da silva", and "d. L. Mcguinness"The Semantic Web facilitates integrating partial knowledge and finding evidence for hypothesis from web knowledge sources. However, the appropriate level of granularity for tracking provenance of RDF graph remains in debate. RDF document is too coarse since
it could contain irrelevant information. RDF triple will fail when two triples share the same blank node. Therefore, this paper investigates lossless decomposition of RDF graph and tracking the provenance of RDF graph using RDF molecule, which is the finest and lossless component of an RDF graph. A sub-graph is {em lossless} if it can be used to restore the original graph without introducing new triples. A sub-graph is {em finest} if it cannot be further decomposed into lossless sub-graphs. The lossless decomposition algorithms and RDF molecule have been formalized and implemented by a prototype RDF graph provenance service in Swoogle project.

As given in the referenceof**“**dogma: a disk-oriented graph matching algorithm for rdf databases", "m. Brocheler", "a. Pugliese", and "v. Subrahmanian"RDF is an increasingly important paradigm for the representation of information on the Web. As RDF databases increase in size to approach tens of millions of triples, and as sophisticated graph matching queries expressible in languages like SPARQL become increasingly important, scalability becomes an issue. To date, there is no graph-based indexing method for RDF data where the index was designed in a way that makes it disk-resident. There is therefore a growing need for indexes that can operate efficiently when the index itself resides on disk. In this paper, we first propose the DOGMA index for fast subgraph matching on disk and then develop a basic algorithm to answer queries over this index. This algorithm is then significantly sped up via an optimized algorithm that uses efficient (but correct) pruning strategies when combined with two different extensions of the index. We have implemented a preliminary system and tested it against four existing RDF database systems developed by others. Our experiments show that our algorithm performs very well compared to these systems, with orders of magnitude improvements for complex graph queries.

As mention in this paper **"**the design and implementation of a clustered rdf store","s. Harris", "n. Lamb", and "n. Shadbolt**"**This paper describes the design and implementation of the 4store RDF storage and SPARQL query system with respect to its cluster and query processing design. 4store was originally designed to meet the data needs of Garlik, a UK-based semantic web company. This paper describes the design and performance characteristics of 4store, as well as discussing some of the trade-offs and design decisions. These arose both from immediate business requirements and a desire to engineer a scalable system capable of reuse in a range of experimental contexts where we were looking to explore new business opportunities.

3. SYSTEM IMPLEMENTATION

3.1 CLOUD SERVERS MODULE

In this module, we develop Cloud Service Provider module. This is an entity that provides a data storage service in public cloud.

For Example,where "Student" is the the root node of

The CS provides the data outsourcing service and stores data on behalf of the users.

To reduce the storage cost, the CS eliminates the storage of redundant data via deduplication and keeps only unique data.

In this paper, we assume that CS is always online and has abundant storage capacity and computation power.

3.2 DATA USERS MODULE

A user is an entity that wants to outsource data storage to the S-CSP and access the data later.

In a storage system supporting deduplication, the user only uploads unique data but does not upload any duplicate data to save the upload bandwidth, which may be owned by the same user or different users.

In the authorized deduplication system, each user is issued a set of privileges in the setup of the system. Each file is protected with the convergent encryption key and privilege keys to realize the authorized deduplication with differential privileges.

There is two process of searching by the user in Diplo cloud:

Template

Template roots are used to determine which literals to store in template lists. Based on the storage patterns, the system handles two main operations in

our system: i) it maintains a schema of triple templates in main-memory and ii) it manages template lists.

Molecule

All molecules are template-based, and hence store data extremely compactly.Similarly to the template lists, the molecule clusters are serialized in a very compact form, both on disk and in main-memory.

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 06
March 2018

and "StudentID" is the root node for the template list.

3.3 DIPLO CLOUD MODULE

I say that DiploCloud is a hybrid system.DiploCloud is a native, RDF database system. It was designed to run on clusters of commodity machines in order to scale out gracefully when handling bigger RDF file.Our system design follows the architecture of many modern cloud-based distributed systems.
Where one (Master) node is responsible for interacting with the clients and orchestrating the operations performed by the other (Worker) nodes.

Master

The Master node is composed of three main subcomponents: A key index in charge of encoding URIs and literals into compact system identifiers and of translating them back, a partition manager responsible for the partitioning the RDF data and a distributed molecule,

query executor,responsible for parsing the incoming query, rewriting the query plans into the Workers.

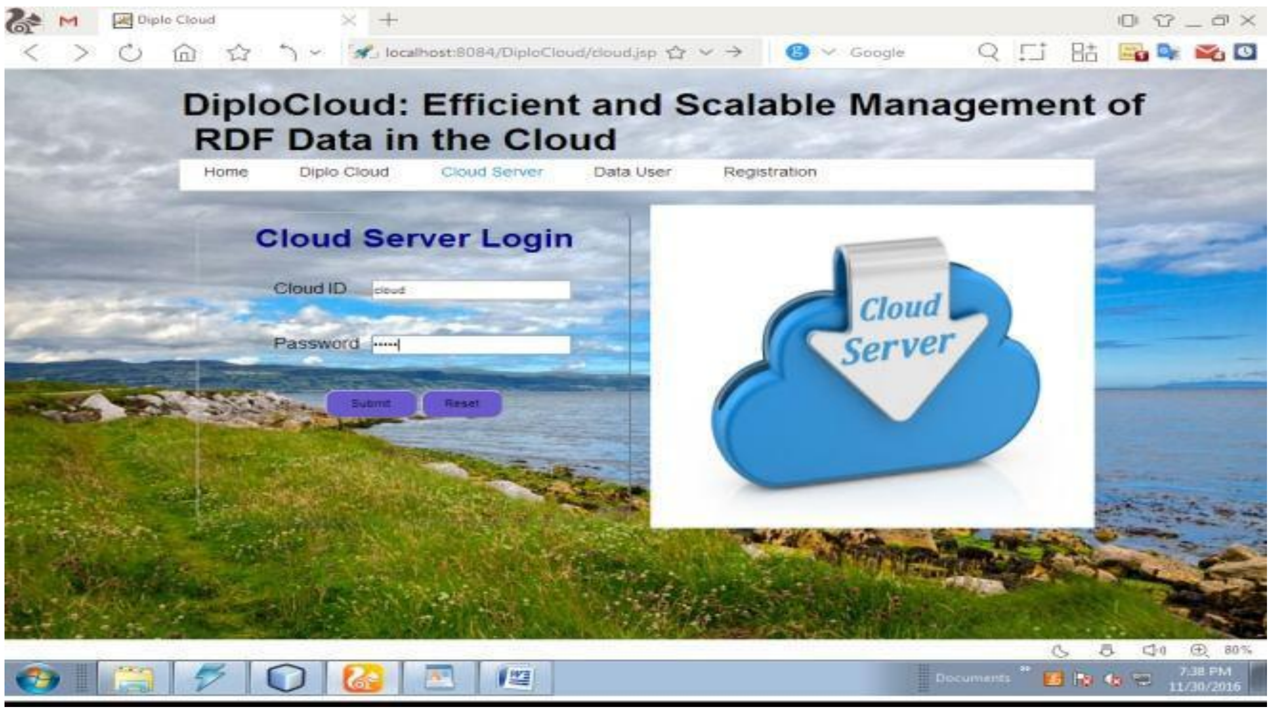
Worker

The Worker nodes hold the partitioned data and its corresponding local indices, and are responsible for running subqueries and sending results back to the Master node. Conceptually, the Workers are much simpler than the Master node and are built on three main data structures:
i) A type index, clustering all keys based on their types
ii)A series of RDF molecules, storing RDF data as very compact subgraphs. iii) A molecule index, storing for each key the list of molecules where the key can be found.

4. SIMULATION

4.1 CLOUD LOGIN

This Screen Shot is a Cloud Server Login page. This page is used for login a Cloud
.



DiploCloud: Efficient and Scalable Management of RDF Data in the Cloud

Cloud Server Login

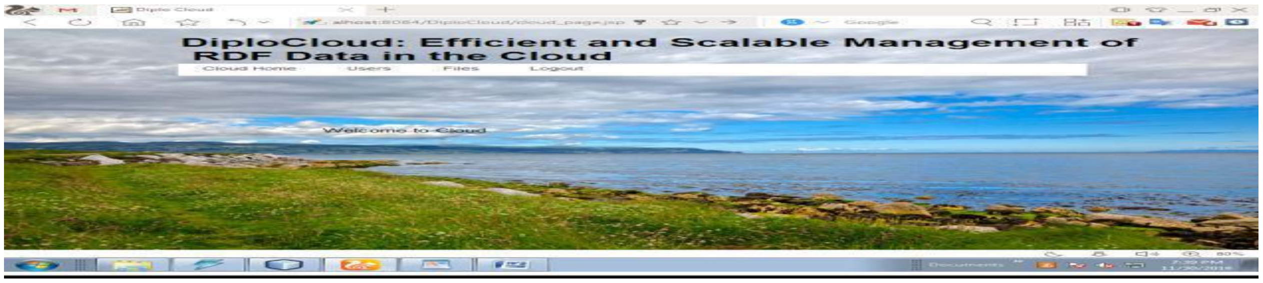
Cloud ID cloud

Password •••••

Fig 1.Cloud Login

4.2 Cloud Home

This Screen Shot is a home page cloud for user interface.



DiploCloud: Efficient and Scalable Management of RDF Data in the Cloud

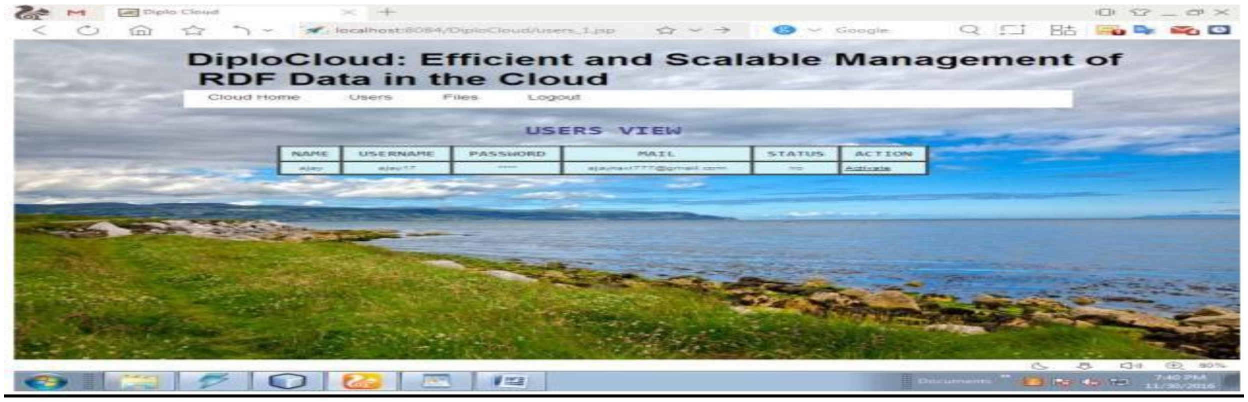
Cloud Home Users Files Logout

Welcome to Cloud

Fig 2.Cloud Home

4.3 User Details

This Screen Shot is a user page in Cloud. This page is used for view a user details by the Cloud.



DiploCloud: Efficient and Scalable Management of RDF Data in the Cloud

Cloud Home Users Files Logout

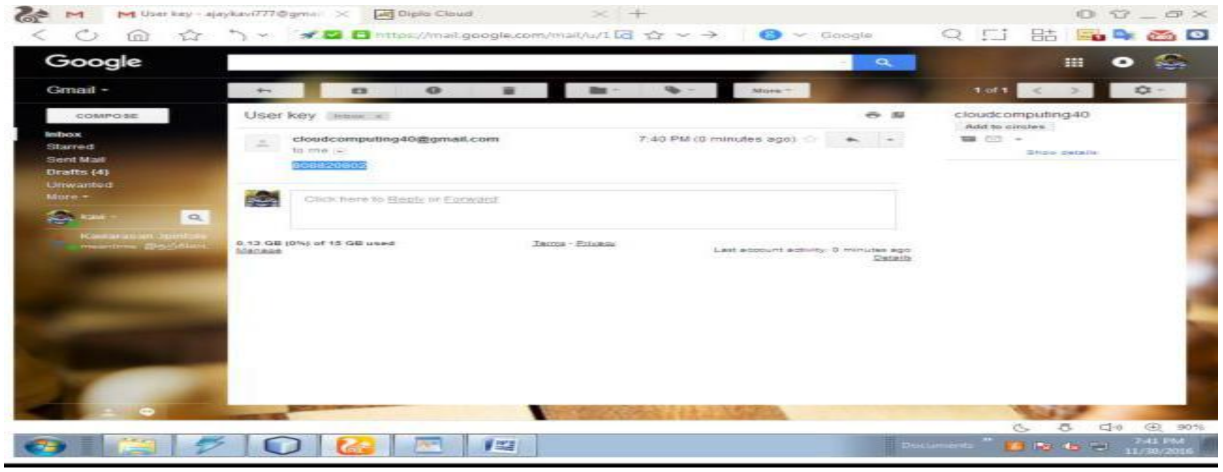
USERS VIEW

NAME	USERNAME	PASSWORD	MAIL	STATUS	ACTION
ajay	ajay17	****	ajaykavi777@gmail.com	no	Activate

Fig 3.User Details

4.4 User Token

This Screen Shot is a user token and it is used for cloud send a unique token id to users email.



User key

cloudcomputing40@gmail.com
to me
7:40 PM (0 minutes ago)

8088820802

Click here to Reply or Forward

0.13 GB (0%) of 15 GB used
Manage
Terms - Privacy
Last account activity: 0 minutes ago
Details

Fig 4.User Token

4.5 User Login

This Screen Shot is a User login page.The User Login page attempts to deal with authenticating the authorized Users to allow the User page.

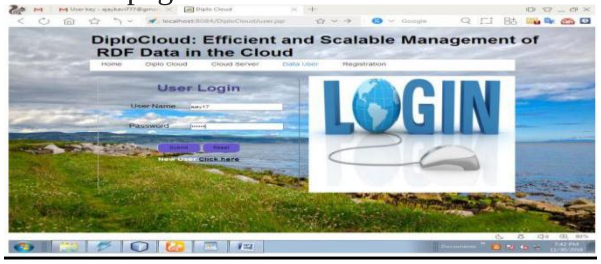


Fig 5.User Login

4.6 User Token

This Screen Shot shows the user token key verification page .This page verify the user token id for security.

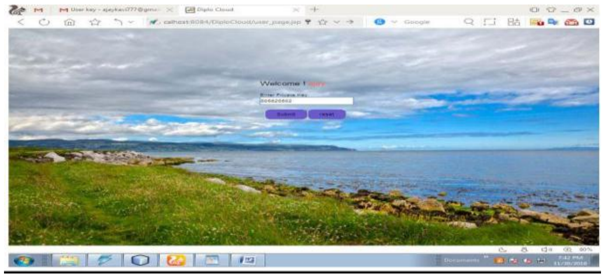


Fig 6.User Token

4.7 File Upload to cloud

This Screen Shot is a upload page in User home.This upload page is used for user upload a file to cloud.

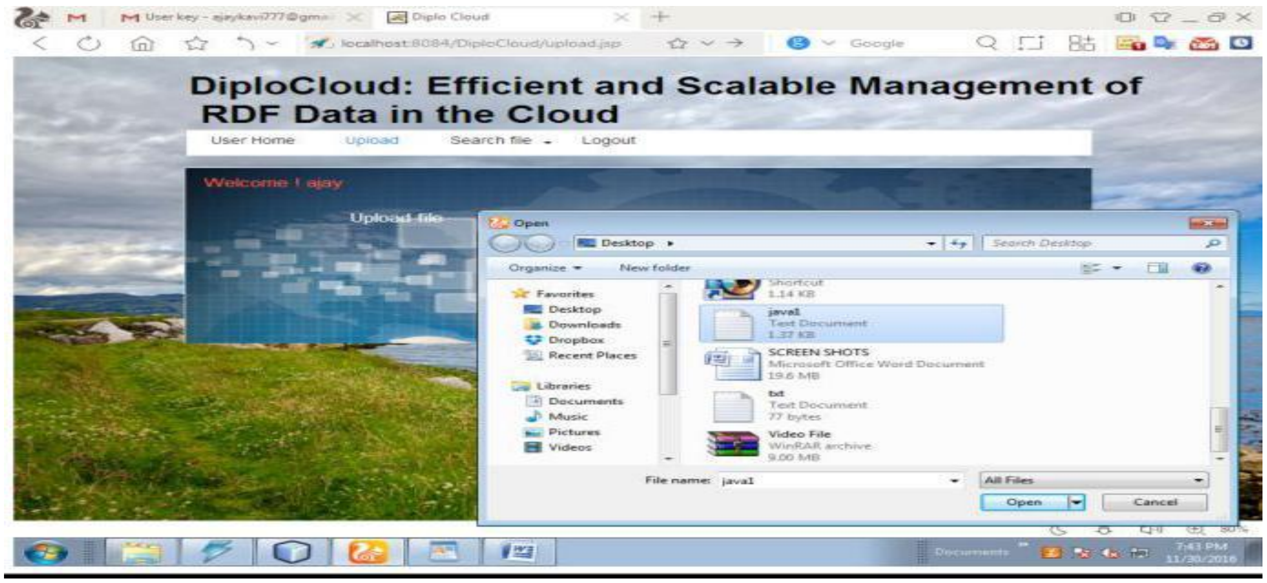


Fig 7.File Upload to cloud

5. CONCLUSION & FUTURE ENHANCEMENTS

5.1 CONCLUSION

DiploCloud is an efficient and scalable system for managing RDF data in the cloud. From our perspective, it strikes an optimal balance between intra-operator parallelism and data collocation by considering recurring, fine-grained physiological RDF partitions and distributed data allocation schemes, leading however to potentially bigger data (redundancy introduced by higher scopes or adaptive molecules) and to more complex inserts and updates. DiploCloud is particularly suited to clusters of commodity machines and cloud environments where network latencies can be high, since it systematically tries to avoid all complex and distributed operations for query execution. Our experimental evaluation showed that it very favorably compares to state-of-the-art systems in such environments. We plan to continue developing DiploCloud in several directions: First, we plan to include some further compression mechanism. We plan to work on an automatic templates discovery based on frequent patterns and un typed elements. Also, we plan to work on integrating an inference engine into DiploCloud to support a larger set of semantic constraints and queries natively. Finally, we are currently testing and extending our system with several partners in order to manage extremely large scale, distributed RDF datasets in the context of bioinformatics applications.

5.2 FUTURE ENHANCEMENT

Many RDF systems rely on hash-partitioning and on distributed selections, projections, and joins. Our own Grid Vine system was one of the first systems to do so in the context of large-scale decentralized RDF management.

REFERENCES

[1] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt,“GridVine: Building Internet-scale semantic overlay networks,” inProc. Int. Semantic Web Conf., 2004, pp. 107–121.

[2] P. Cudr_e-Mauroux, S. Agarwal, and K. Aberer, “GridVine: An infrastructure for peer information management,” IEEE InternetComput., vol. 11, no. 5, pp. 36–44, Sep./Oct. 2007.

[3] M. Wylot, J. Pont, M. Wisniewski, and P. Cudr_e-Mauroux. (2011). dipLODocus[RDF]: Short and long-tail RDF analytics for massivewebs of data. Proc. 10th Int. Conf. Semantic Web - Vol. Part I, pp. 778–793 [Online]. Available: http://dl.acm.org/citation.cfm?id=2063016.2063066

[4] M. Wylot, P. Cudre-Mauroux, and P. Groth, “TripleProv: Efficient processing of lineage queries in a native RDF store,” in Proc. 23rd Int. Conf. World Wide Web, 2014, pp. 455–466.

[5] M. Wylot, P. Cudr_e-Mauroux, and P. Groth, “Executing provenance- enabled queries over web data,” in Proc. 24th Int. Conf.World Wide Web, 2015, pp. 1275–1285.

[6] B. Haslhofer, E. M. Roochi, B. Schandl, and S. Zander.(2011).Europeana RDF store report.Univ. Vienna, Wien, Austria, Tech.Rep. [Online]. Available: http://eprints.cs.univie.ac.at/2833/1/europeana_ts_report.pdf

[7] Y. Guo, Z. Pan, and J. Heflin, “An evaluation of knowledge base systems for large OWL datasets,” in Proc. Int. Semantic Web Conf.,2004, pp. 274–288.

[8] Faye, O. Cure, and Blin, “A survey of RDF storage

approaches," ARIMA J., vol. 15, pp. 11–35, 2012.

[9] B. Liu and B. Hu, "An Evaluation of RDF Storage Systems for Large Data Applications," in Proc. 1st Int. Conf. Semantics, Knowl.Grid, Nov. 2005, p. 59.

[10] Z. Kaoudi and I. Manolescu, "RDF in the clouds: A survey," VLDB J. Int. J. Very Large Data Bases, vol. 24, no. 1, pp. 67–91, 2015.

[11] C. Weiss, P. Karras, and A. Bernstein, "Hexastore: sextuple indexing for semantic web data management," Proc. VLDB Endowment,vol. 1, no. 1, pp. 1008–1019, 2008.

[12] T. Neumann and G. Weikum, "RDF-3X: A RISC-style engine for RDF," Proc. VLDB Endowment, vol. 1, no. 1, pp. 647–659, 2008.

[13] A. Harth and S. Decker, "Optimized index structures for querying RDF from the web," in Proc. IEEE 3rd Latin Am. Web Congr., 2005,pp. 71–80.

[14] M. Atre and J. A. Hendler, "BitMat: A main memory bit-matrix of RDF triples," in Proc. 5th Int. Workshop Scalable Semantic WebKnowl. Base Syst., 2009, p. 33.

[15] K. Wilkinson, C. Sayers, H. A. Kuno, and D. Reynolds, "Efficient RDF Storage and Retrieval in Jena2," in Proc. 1st Int. WorkshopSemantic Web Databases, 2003, pp. 131–150.

[16] A. Owens, A. Seaborne, N. Gibbins, et al., "Clustered TDB: A clustered triple store for Jena," 2008.

[17] E. PrudHommeaux, A. Seaborne, et al., "SPARQL query language for RDF," W3C Recommendation, vol. 15, 2008.

[18] Y. Yan, C. Wang, A. Zhou, W. Qian, L. Ma, and Y. Pan. (2009). Efficient indices using graph partitioning in RDF triple stores. Proc.IEEE Int. Conf. Data Eng., pp. 1263–1266 [Online]. Available: http://portal.acm.org/citation.cfm?id=1546683.1547484