Efficient and Scalable Management of RDF Data Using Diplo Cloud

G.Anurag, M.Devi Sri, P.Kiran Mai, A.Ramya,
Under the Guidance of G.Pushpa Rajitha
B.Tech, Department of Computer Science and Engineering,
St.Martin's Engineering College, Hyderabad, Telangana, India.

ABSTRACT

Despite recent advances in distributed RDF data management, processing large-amounts of RDF data in the cloud is still very challenging. In spite of its seemingly simple data model, RDF actually encodes rich and complex graphs mixing both instance and schema-level data. Sharing 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**.**

1. INTRODUCTION

In the era of big data, a huge amount of data can be generated quickly from various sources (e.g., smart phones, sensors, machines, social networks, etc.). Towards these big data, conventional computer systems are not competent to store and process these data. Due to the flexible and elastic computing resources, cloud computing is a natural fit for storing and processing big data. With cloud computing, end-users store their data into the cloud, and rely on the cloud server to share their data to other users (data consumers). In order to only share end-users' data to authorized users, it is necessary to design access control mechanisms according to the requirements of end-users.

When outsourcing data into the cloud, end-users lose the physical control of their data. Moreover, cloud service providers are not fully-trusted by end-users, which make the access control more challenging. For example, if the traditional access control mechanisms (e.g., Access Control Lists) are applied, the cloud server becomes the judge to evaluate the access policy and make access decision. Thus, end-users may worry that the cloud server may make wrong access decision intentionally or unintentionally, and disclose their data to some unauthorized users. In order to enable end-users to control the access of their own data, some attribute-based access control schemes are proposed by leveraging attribute-based encryption. In attribute-based access control, end-users first define access policies for their data and encrypt the data under these access policies. Only the users whose attributes can satisfy the access policy are eligible to decrypt the data. Although the existing attribute-based access control schemes can deal with the attribute revocation problem, they all suffer from one problem: *the access policy may leak privacy*. This is because the access policy is associated with the encrypted data in plaintext form. From the plaintext of access policy, the adversaries may obtain some privacy information about the end-user. For example, Alice encrypts her data to enable the "Psychology Doctor" to access. So, the access policy may contain the attributes "Psychology" and "Doctor". If anyone sees this data, although he/she may not be

able to decrypt the data, he/she still can guess that Alice may suffer from some psychological problems, which leaks the privacy of Alice.

To prevent the privacy leakage from the access policy, a straightforward method is to hide the attributes in the access policy. However, when the attributes are hidden, not only the unauthorized users but also the authorized users cannot know which attributes are involved in the access policy, which makes the decryption a challenging problem. Due to this reason, existing methods do not hide or anonymize the attributes. Instead, they only hide the values of each attribute by using wildcards, Hidden Vector Encryption, and Inner Product Encryption. Hiding the values of attributes can somehow protect user privacy, but the attribute name may also leak private information. Moreover, most of these partially hidden policy schemes only support specific policy structures (e.g., AND-gates on multi-valued attributes). In this paper, we aim to hide the whole attribute instead of only partially hiding the attribute values. Moreover, we do not restrict our method to some specific access structures.

The basic idea is to express the access policy in LSSS access structure (M*;r*) where M is a policy matrix and *r* matches each row *Mi* of the matrix M to an attribute [6], and hide the attributes by simply removing the attribute matching function *r*. Without the attribute matching function *r*, it is necessary to design an attribute localization algorithm to evaluate whether an attribute is in the access policy and if so find the correct position in the access policy. To this end, we further build a novel Attribute Bloom Filter to locate the attributes to the anonymous access policy, which can save a lot of storage overhead and computation cost especially for large attribute universe.

Our contributions are summarized as follows.

1) We propose an efficient and fine-gained big data access control scheme with privacy-preserving policy, where the whole attributes are hidden in the access policy rather than only the values of the attributes.

2) We also design a novel Attribute Bloom Filter to evaluate whether an attribute is in the access policy and locate the exact position in the access policy if it is in the access policy.

3) We further give the security proof and performance evaluation of our proposed scheme, which demonstrate that our scheme can preserve the privacy from any LSSS access policy without employing much overhead.

2. Literature Survey

1) Tracking RDF graph provenance using RDF molecules

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.

2) DOGMA: A Disk-Oriented Graph Matching Algorithm for RDF Databases

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.

3) The design and implementation of a clustered RDF store

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.

4) WARP: Workload-aware replication and partitioning for RDF

This paper proposes a distributed SPARQL engine that combines a graph partitioning technique with workload-aware replication of triples across partitions, enabling efficient query execution even for complex queries from the workload.

5) Scaling queries over big RDF graphs with semantic hash partitioning

In this paper we present a novel semantic hash partitioning approach and implement a Semantic HAsh Partitioning-Enabled distributed RDF data management system, called Shape.

3. OVERVIEW OF THE SYSTEM

ARCHITECHTURE

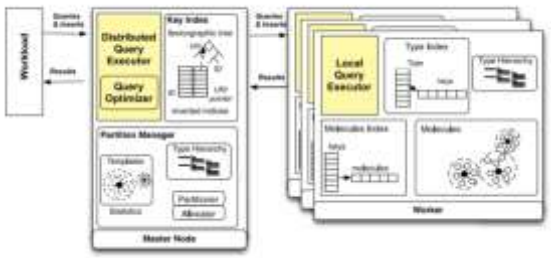

Fig 3.1

System Architecture

EXISTING SYSTEM:

- While much more recent than relational data management, RDF data management has borrowed many relational techniques; Many RDF systems rely on hash-partitioning and on distributed selections, projections, and joins.
- Grid-Vine system was one of the first systems to do so in the context of large-scale decentralized RDF management.
- Approaches for storing RDF data can be broadly categorized in three subcategories: triple-table approaches, property-table approaches, and graph-based approaches.
- Hexastore suggests to index RDF data using six possible indices, one for each permutation of the set of columns in the triple table. RDF-3X and YARS follow a similar approach.
- BitMat maintains a three-dimensional bit-cube where each cell represents a unique triple and the cell value denotes presence or absence of the triple. Various techniques propose to speed-up RDF query processing by considering structures clustering RDF data based on their properties.

DISADVANTAGES OF EXISTING SYSTEM:

- Existing system 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.
- RDF actually encodes rich and complex graphs mixing both instance and schema-level data. Sharing 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.
- Existing system are not efficient and not scalable system for managing RDF data in the cloud.
- In existing system lot of complex while query execution.
- Existing system are slower while handling the standard workloads.

PROPOSED SYSTEM:

- In this article, we propose DiploCloud, an efficient, distributed and scalable RDF data processing system for distributed and cloud environments. Contrary to many distributed systems, DiploCloud uses a resolutely non-relational storage format, where semantically related data patterns are mined both from the instance-level and the schema-level data and get co-located to minimize internode operations. The main contributions of this article are:
- A new hybrid storage model that efficiently and effectively partitions an RDF graph and physically co-locates related instance data;

- A new system architecture for handling fine-grained RDF partitions in large-scale
- Novel data placement techniques to co-locate semantically related pieces of data
- New data loading and query execution strategies taking advantage of our system's data partitions and indices
- An extensive experimental evaluation showing that our system is often two orders of magnitude faster than state-of-the-art systems on standard workloads

ADVANTAGES OF PROPOSED SYSTEM:

- DiploCloud is an efficient and scalable system for managing RDF data in the cloud.
- 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.

4. MODULES

- System Construction Module
- Cloud Servers
- Data Users Module
- Diplo Cloud

MODULES DESCRIPTON:-

System Construction Module

- In the first module, we develop the system with the entities required to prove and evaluate the proposed system module. So first we develop the User process in this module.
- Every user need to register to access the data in the diplo cloud.
- Every user will activate by Cloud server.
- After activate by the cloud server, for each user the private key will be send to corresponding user mail ID

Cloud Service Provider

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

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.

- For Example, where "Student" is the root node of the molecule, and "StudentID" is the root node for the template list.

Diplo Cloud:

- We 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 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 sub-queries 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, and iii) a molecule index, storing for each key the list of molecules where the key can be found.

5. CONCLUSION AND FUTURE SCOPE

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 untyped 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.

6. REFERENCES

[1] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt,
"GridVine: Building Internet-scale semantic overlay networks," in
Proc. 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 Internet
Comput., 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 massive
webs 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, pp274–288.

[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, Known. Grid, Nov. 2005,

[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.