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

**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. 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.

**EXISTING SYSTEM:**

* Private searching, which allows a user to retrieve files of interest from an untrusted server without leaking any information. Otherwise, the cloud will learn that certain files, without processing, are of no interest to the user.
* Commercial clouds follow a pay-as-you-go model, where the customer is billed for different operations such as bandwidth, CPU time, and so on. Solutions that incur excessive computation and communication costs are unacceptable to customers.
* To make private searching applicable in a cloud environment, our previous work designed a cooperate private searching protocol (COPS), where a proxy server, called the aggregation and distribution layer (ADL), is introduced.between the users and the cloud.
* The ADL deployed inside an organization has two main functionalities: aggregating user queries and distributing search results. Under the ADL, the computation cost incurred on the cloud can be largely reduced, since the cloud only needs to execute a combined query once, no matter how many users are executing queries.
* Furthermore, the communication cost incurred on the cloud will also be reduced, since files shared by the users need to be returned only once. Most importantly, by using a series of secure functions, COPS can protect user privacy from the ADL, the cloud, and other users.

**DISADVANTAGES OF EXISTING SYSTEM:**

1. In Cloud computing process has a high computational cost, since it requires the cloud to process the query on every file in a collection.

 2. It will quickly become a performance bottleneck when the cloud needs to process thousands of queries over a collection of hundreds of thousands of files. We argue that subsequently proposed improvements, like also have the same drawback.

3. The complexity of scaling out an application in the cloud (i.e., adding new computing files to accommodate the growth of some process) very much depends on the process to be scaled.

**PROPOSED SYSTEM:**

* In this paper, we propose DiploCloud, an efficient, distributed and scalable RDF data processing system for distributed and cloud environments.
* A new system architecture for handling fine-grained
RDF partitions in large-scale
* A novel data placement techniques to co-locate semantically related pieces of data
* A new data loading and query execution strategies taking advantage of our system’s data partitions and indices.
* In addition, we introduce a unique combination of physical structures to handle RDF data both horizontally (to flexibly co-locate entities or values related to a given instance) as well as vertically (to co-locate series of entities or values attached to similar instances)
* **ADVANTAGES OF PROPOSED SYSTEM:**

1. We introduce a novel scalable-based scheme for managing RDF data.RDF has the potential to support efficient graph-based queries, as well as advanced graph analytics, on RDF.

 2. The new query paradigm greatly reduces the volume of intermediate results, which in turn boosts query performance and system scalability.

 3. We introduce a new cost model, novel cardinality estimation techniques, and optimization algorithms for distributed query plan generation. These approaches ensure excellent performance on web scale RDF data.

**SYSTEM ARCHITECTURE:**



**MODULES:-**

* Cloud Servers
* Data Users Module
* Diplo Cloud
* User Registration

**MODULES DESCRIPTON:-**

**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:

* 1. **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.

* 1. **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.

* 1. **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.

* 1. **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, and iii) a molecule index, storing for each key
the list of molecules where the key can be found.

**User Registration**

* 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

**SYSTEM REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz.
* Hard Disk : 40 GB.
* Floppy Drive : 1.44 Mb.
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* Ram : 512 Mb.

**SOFTWARE REQUIREMENTS:**

* Operating system : Windows XP/7.
* Coding Language : JAVA/J2EE
* IDE : Netbeans 7.4
* Database : MYSQL

**REFERENCE:**

[1] S. Das, D. Agrawal, and A. El Abbadi, “G-store: A scalable data store for transactional multi key access in the cloud,” pp. 163–174,
2010.

[2] M. Brocheler, A. Pugliese, and V. Subrahmanian, “Dogma: A disk- €
oriented graph matching algorithm for RDF databases,” in Proc.
8th Int. Semantic Web Conf., 2009, pp. 97–113.

[3] 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.

[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.