International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 16
June 2018

# HDT Compression Mechanism for Efficiently Managing RDF datasets in Cloud environment

Shaik Shabbeer
(Shaik Shabbeer, Dept Of Computer Science And Engineering, Jntu College Of Engineering, Anantapur, India)

EMAIL ID: shabbeersrit587@gmail.com

## Abstract

*By using cloud technology we possess various uses like high storage space, shared resources and less management etc. Data storage is a major issue when we receive large amount of data from variety of sources. Performing any task with huge amounts of RDF data in the cloud is a major task. Basically it seems to be a simple data model but it involves encoded and complex graphs which are mined at both instance and schema-level data. Functioning or sharing this type of data using classical techniques or dividing the graph using traditional min-cut algorithms leads to ineffective distributed operations. Diplo cloud technique is developed to provide better a solution for this problem. It uses a non-relational storage format and it semantically relates data patterns which are mined from both the instance level and schema level data to produce less inter-node operations. This is a effective mechanism to parse and index the data but it takes more bandwidth and storage space. To avoid this and to Provide a solution for this problem, we introduce HDT compression mechanism. HDT mechanism is a efficient technique for querying and parsing. It can execute many queries per second and reduces the duplicate data by that it reduces storage space and bandwidth.*

**Keywords:** RDF(Resource Description Framework), HDT( Header Dictionary Triples), Parsing, Indexing, Qerying, Partitioning.

## 1. INTRODUCTION

The advent of cloud computing enables to easily and cheaply provision computing resources. Operating with large amounts of RDF data in the cloud is a major issue. Diplo cloud[1] technique is a effective mechanism to manage RDF[2] datasets and it uses a non-relational storage format[3] [4] and it mines both the instance level and schema level data to produce less number of operations. This is a effective mechanism to parse and index the data but it takes more bandwidth and storage space[5]. To avoid this and to Provide a solution for this problem, we introduce HDT compression mechanism. HDT compression mechanism is a efficient mechanism to manage these RDF datasets. HDT mechanism has three major components:

– A Header, mainly contains metadata and it describes about the RDF dataset. It act as a starting point to the information which is provided on the dataset.

– A dictionary, organizes all the identifiers provided in the RDF graph. It produces a catalogue of the RDF terms like URIs, blank nodes and literals that are mentioned in the graph along with compression.

– A triples component, composes the structure of RDF graph, i.e. it encodes the set of triples and avoids the noise which is produced by repetitions.

The HDT compression mechanism contains the following benefits:

(1) it provides the indexed access to the RDF graph, and
(2) it reduces the duplicate data, by that it uses less storage space and bandwidth.
(3) It can dispatch many queries in a second by using multiple threads at a time.

The rest of the paper is organized as follows: Section 2 presents the Related work. Section 3 discusses the proposed approach. Section 4 presents

the Performance Evaluation of the proposed approach. Finally Section 5 concludes the paper.

## 2. RELATED WORK

Our related work involves Diplo cloud for managing RDF data, triple prov to handle queries, bit map for representing huge number of RDF triples and also for query processing, grid vine for managing database effectively.

### 2.1. Diplo Cloud:

DiploCloud, is an effective, RDF data management system for both distributed and the cloud environments. Comparing to other distributed systems, this technique uses a non-relational storage format, which semantically relates the data patterns which are mapped based on the instance level and the schema level data reduce number of operations. The main features of this technique are:

_ it is a new hybrid storage model that efficiently and effectively partitions an RDF graph.

_ it is a new architecture for handling large amounts of RDF partitions.

_ is is a effective data management technique which gathers all the related data.

_ it contains new query execution techniques taking responsibility of data partitions and indices . The main drawback of this technique is, it is very expensive to parse and index the data and it takes more bandwidth.

### 2.2. Triple Prov:

Triple prov[6] is a new system to handle queries. It contains two storage models to gather lineage and instance data. Native storage model is used for ingesting new triples. Triple prov at first identifies RDF sub graphs. It analyzes all the receiving data and builds the templates. In this mechanism specific query execution techniques are used. It contains the complete record of how the results are processed. In this technique unions and joins are used to get the effective results. The main drawback of this technique is it takes more time to execute a query.

### 2.3. Bit Map:

Bit map[7] [8] is a three dimensional structure which contains subject, predicate and object. It is mainly used to avoid the network traffic for processing the join queries. It is used for representing huge number of RDF triples and for query processing. Bit map operations helps to process join queries faster. The drawback is it occupies more storage space.

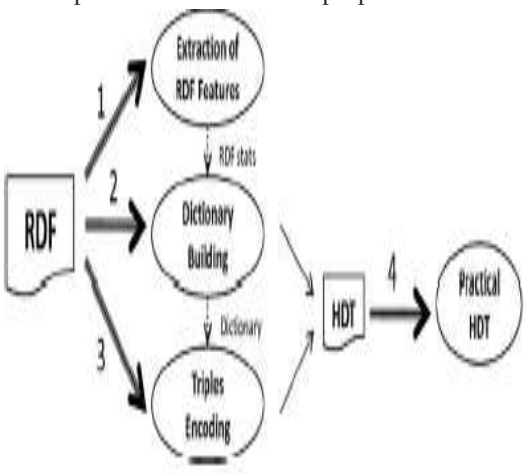
### 2.4. Grid Vine:

Grid vine[9] [10] is the first semantic overlay network. It is an scalable, effective technique and it provides decentralized access to create local schemas. This technique is mainly used for both managing as well as mapping data and metadata schemas. It is used for managing database effectively. By using different query processing[11] [12] [13] strategies, optimization of data takes place. The drawback of this technique is it occupies more storage space.

## 3. PROPOSED APPROACH:

### 3.1. HDT COMPRESSION MECHANISM:

To provide a solution for above problem, HDT compression mechanism is proposed.




. Fig.1. **A step by step construction of HDT from a set of triples. The last step covers practical decision to get a concrete implementation of HDT.**

### 3.2. Description Of HDT Components:

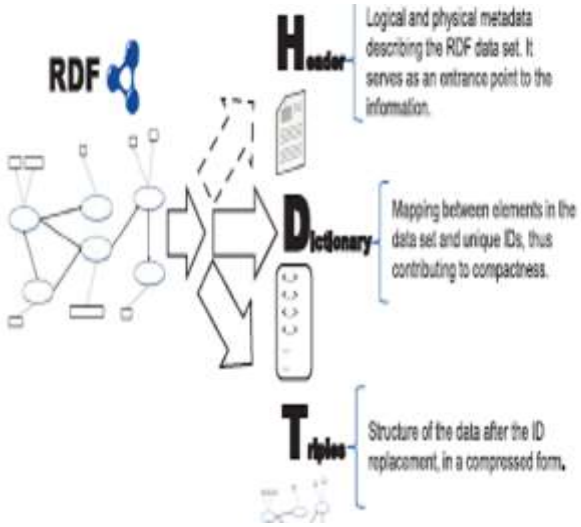



**Fig.2. Description of HDT components: Header-Dictionary-Triples.**

### 3.3. HEADER:

The responsibility of Header component is to provide metadata about an RDF dataset. It includes the following features.

(1) Publication metadata- It contains the metadata about the publication details such as publication site, creation date, modification and version etc. It also includes all kind of authority information about the source

(2) Statistical meta data-. When managing huge collections, one could consider including some pre computed statistics about what follows in the datasets.

(3) Format metadata- Collects the information about the concrete format of the RDF dataset, i.e., it specifies the concrete Dictionary and Triples implementations as well as their physical locations.

(4) Additional metadata- A provider can take into account other metadata for the understanding and management of the data..

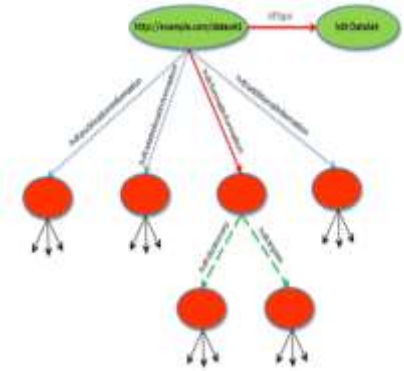


**Fig.4. Description of header component(contains metadata information).**

Header component responsibility is to provide metadata about an RDF dataset. Header component is considered as a first-class citizen. It is a flexible component in which the data provider may include a desired set of features.

### 3.4. DICTIONARY:

Data Dictionary acts as a centralized kind of storage of information about the received data such as meaning of data, relationships having with other data, origin of data, usage of data, and data format. Present RDF formats uses basic versions of dictionaries for handling both namespaces as well as prefixes. The major goal of the Dictionary component is to assign a unique ID to each element which are presented in the dataset. Dictionary Component mainly performs two important operations:

• locate(element): it is responsible for returning a unique identifier for the given element, if it appears in the dictionary.

• extract(id): it is responsible for displaying the element along with id , if it exists in the dictionary.

### 3.5. TRIPLES:

The Triples component plays a major role accessing and querying the RDF graph. The format for RDF Triples should be designed to optimize the common operations and uses of them. We distinguish here then fundamental ones:

-Exchange: At fundamental level, an RDF Triples component serves to compact the set of RDF statements, optimizing the objective of exchange. It might include functionalities to exchange only a part of the entire graph.

-Basic search: An important foundation for any search over RDF triples are triple patterns, which are templates of RDF triples where one or more element of the triple can be a variable.

-Join resolution: Joins are one of the most effective operations in RDF queries. They matches two or more triples patterns which share one or more variables.

-Complex querying: Ideally, the engine should be able to process and produces the answer efficiently to any query. This involves addressing many other operators and modifiers, such as UNION, OPTIONAL, as well as query evaluation optimization techniques.

The efficient indexing of the triples structure is one of the keys for good RDF query performance.

## 4. PERFORMANCE EVALUATION:

This section evaluates the size and performance.

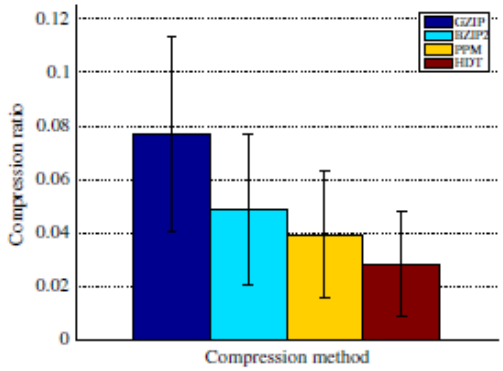



**Fig.4. HDT Compress results.**

HDT compression mechanism is the most effective technique in terms of compression. Compression ratios are calculated based on different compression methods. Fig.3. shows compared to GZIP, PPM, BZIP2, HDT compression mechanism[14] compresses the file better without any loss of information. one of the major feature of this technique is its size. HDT file can be directly loaded into the memory and easy to access data as it is a simple data structure. It avoids the expensive indexing operations. The data retrieval operations will be performed in a fastest way.

## 5. CONCLUSION

HDT compression mechanism is very effective and secure in manner. It efficiently carried out the querying and parsing of RDF datasets which are presented in the cloud environment. When it comes to manage large RDF datasets, the advantages of HDT compared to existing RDF serialization formats can be summarized as follows.

(1) HDT is a binary RDF serialization technique.
(2) The size of files is smaller than traditional RDF serialized formats.
(3) It can dispatch many queries in a second by using the multiple threads.
(4) It reduces the duplicate data.
(5) It provides high performance and consumes low storage space and bandwidth.
(6) HDT also provides efficient data querying and parsing. .

## REFERENCES:

[1] Marcin Wylot and Philippe Cudr´e-Mauroux, DiploCloud: Efficient and Scalable Management of RDF Data in the Cloud, Transactions on Knowledge and data engineering, 2015

[2] R.R. Lassila, O. Swick, Resource description framework (RDF) model and syntax specification, 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[3] K. Alexander, RDF in JSON: a specification for serialising RDF in JSON, in: SFSW 2008, 2008.

[4] D. Le-Phuoc, J.X. Parreira, V. Reynolds, M. Hauswirth, RDF on the go: an rdf storage and query processor for mobile devices, in: ISWC 2010, 2010. Available at:http://iswc2010.semanticweb.org/pdf/503.pdf.

[5] J.J. Carroll, Signing RDF graphs, in: ISWC 2003, 2003, pp. 369–384.
[6] M. Wylot, P. Cudre-Mauroux, and P. Groth, "TripleProv: Efficient Processing of Lineage Queries in a Native RDF Store," in Proceedings of the 23rd International Conference on World Wide Web, ser. WWW '14. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2014, pp. 455–466.
[7] Medha Atre, BitMat: A Main-memory Bit Matrix of RDF Triples for Conjunctive Triple Pattern Queries.
[8] M. Atre, V. Chaoji, M.J. Zaki, J.A. Hendler, Matrix Bit loaded: a scalable lightweight join query processor for RDF data, in: WWW 2010, ACM, 2010, pp. 41–50.
[9] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt, "GridVine: Building Internet-Scale Semantic Overlay Networks," in International Semantic Web Conference (ISWC), 2004.
[10] P. Cudr´e-Mauroux, S. Agarwal, and K. Aberer, "GridVine: An Infrastructurefor Peer Information Management," IEEE Internet Computing, vol. 11, no. 5, 2007.
[11] E. Prud'hommeaux, SPARQL query language for RDF. W3C Recommendation,
2008. http://www.w3.org/TR/rdf-sparql-query/.
[12] E.I. Chong, S. Das, G. Eadon, J. Srinivasan, An efficient sql-based RDF querying scheme, in: VLDB 2005, 2005, pp. 1216–1227..
[13] S. Álvarez, N. Brisaboa, J.D. Fernández, M.A. Martínez-Prieto, Compressed k2-triples for full-in-memory RDF engines, in: AMCIS 2011, Article 350, 2011.
[14] L. Sidirourgos, R. Goncalves, M. Kersten, N. Nes, S. Manegold, Column-store support for RDF data management: not all swans are white, VLDB J. 1 (2) (2008) 1553–1563.
.