Comparison of Efficient Parallel Index Algorithms used for RDF Data Store

Rajani Sharma
Graphic Era University (Dehradun, India)

Rajender Kumar Trivedi, Ph.D.
Graphic Era University (Dehradun, India)

ABSTRACT

GPU acceleration of compute-intensive applications has emerged as a new research frontier with phenomenal success- rates. Such applications are characterized by large data-sets being processed by singular functional units (FUs) often described as SIMD (Single Instruction Multiple Data) computing.

Moreover, with the proliferation of internet and its easy access on myriad devices, has resulted in huge amount of data generation. Initially, such data was considered disconnected and not related. But with the advent of semantic web, data has been found to be highly co- related and relevant. Organizing such huge amount of data and subsequently processing requires parallel processing framework that is both distributed and scalable. Graphical processing units (GPUs) are being actively probed in the domain of Big Data analysis, machine learning, and augmented reality since such applications are characterized by massive data spanned and generated over distributed network. GPUs provide a parallel programming framework using CUDA (Compute Unified Device Architecture) that can be utilized to efficiently collate and make inferences on these massive data-sets. Further, GPU multicores are available at commodity rates thus providing an option for cheap and low-power alternatives.

The exponential growth of semantic web and the resultant generation of large-scale RDF (Resource Description Framework) triples pose new challenges in the domain of RDF-storage and retrieval. RDF data consist of triples <Subject, Predicate, and Object> which need to be efficiently indexed. Following are some of the many challenges related to efficient indexing of RDF triples:

- As RDF-triples extensively contain recursive redundancies, self-joins so formed are inefficient.
- Self-joins also lead to large scale null values.

This paper presents the research initiatives of conducting literature survey of contemporary indices including those under active research, which matches the goals as outlined in above sections. Comparing the efficiency of different variety of indices that have been suggested for large data-sets (Map reduce, B+ tree Hashed Index, 3-level-cascade hash index, braided B+ tree index, etc.)

Keywords

RDF, Semantic Web, Graphical Processing Unit (GPU), Compute Unified Device Architecture (CUDA), Hashed B+ tree indexing.

1. INTRODUCTION

The presence of semantic web with RDF as one of the major component of same has increased tremendously to represent data over the web. Resource Description Framework presents several challenges to competent storage, indexing mechanisms and querying software. RDF being a graphical database does not bound to any specific structure poses big challenges for efficient indexing mechanisms. This makes storing and querying RDF and RDF schema a big challenge for applications on semantic Web. Many indexing algorithms are being proposed for organizing RDF with major stress upon their efficiency to handle large scale data which is due to exponential growth of semantic web over internet. The idea of increasing performance of systems for data analysis with help of GPUs itself isn't new. Various researches have been done with combination of distributed databases like (Hadoop or Map-Reduce) and capacities of a GPU. First Map-Reduce framework (Mars) for graphics processors achieved a 1.5 to 16 times increase in performance when analysing web data and processing web documents. Later based on Mars many more tools came into picture. With this paper the first step of comparison of already proposed methodologies and their efficiencies in indexing and querying RDF data is attempted.

2. COMPARISONS OF DIFFERENT PROPOSED METHODOLOGIES FOR RDF INDEXING AND QUERYING.

A. **Querying RDF Data from a Graph Database Perspective**

This paper presents the RDF model from a database perspective; contrast it with other conceptual database models, focusing on query languages and graph databases. This paper proposes that primitives of query language of graph database needs to be included in RDF query language.

The following process is used in the mentioned paper:

1 Compare the RDF representation with conventional conceptual database representations considering particular stress in graph database representations.

2 Learn recent RDF query languages with respect to their competencies to support graph-like queries and to evaluate theirshortcoming in performance and efficiencies.

3 Review the concept, procedures and structures developed in the field of graph database query languages, and their performance and efficiency when applied to the RDF model.

4 Recommend rules/procedures for RDF query languages based on the graph database understanding.

There are some results which are gathered from comparisons done on different perspective.

Table 1: Summary of comparison among different database models. .The parameters are: abstraction level, complexity of data items modeled, degree of connectivity among the data and support to get this information, and finally, flexibility to store different

PROPERTY	RQL	SeRQL	RDQL	TRIPLE	N3	VWRSA	RxPath
Adjacent nodes	Partial support	Partial support	Partial support	Partial support	Partial support	Partial support	No support
Adjacent edeges	Partial support	Partial support	Partial support	Partial support	No support	No support	No support
Degree of a node	Partial support	No support	No support	No support	No support	No support	No support
Path	No support	No support	No support	No support	No support	No support	Partial support
Fixed- Length- Path	Partial support	Partial support	Partial support	Partial support	Partial support	No support	Partial support
Distance between two nodes	No support	No support	No support	No support	No support	No support	No support
Diameter	No support	No support	No support	No support	No support	No support	No support

Table 2: Support of some current RDF query languages for some example graph property

MODEL	LEVEL	DATA COMPLEX	CONNECTIVITY	TYPES of DATA
Network	Physical	Simple	High	Homogeneous
Relational	Logical	Simple	Low	Homogeneous
Semantic	User	Simple/ Medium	High	Homogeneous
Object-O	Logical/ physical	Complex	Medium	Homogeneous
XML	Logical	Medium	Medium	Homogeneous
RDF	Logical	Medium	High	Homogeneous

B. Exchange and Consumption of Huge RDF Data

This paper shows how to improve the exchanged HDT (*Header, Dictionary, Triples*) with supplementary structures to hold some basic forms of SPARQL query decision without the need of "unpacking" the data.

Using an step to lightweight data exchange is a compressed (binary) RDF serialization format called HDT.(HDT (Header-Dictionary-Triples) is a binary serialization format which organizes RDF data in three logical components. The Header includes logical and physical metadata describing the RDF dataset and serves as an entry point to its information. The Dictionary provides a catalog of the terms used in the dataset and maps them to unique integer IDs. It enables terms to be replaced by their corresponding IDs and allows high levels of compression to be achieved. The Triples component represents the pure structure of theunderlying graph after the ID replacement.)

As the advantage the experiment shows that the exchanged

RDF data become direct and easily query able:

1 with an switching competence that performs better than common compression,

2 Post-processing now becomes a fast process which

3 Provides competitive query performance at consumption.

C. Storing and Indexing Massive RDF Data Sets

A general survey of the current state of the art in RDF storage and indexing. Identify three different perspectives on RDF:

(1) A relational erspective
(2) An entity erspective
(3) A graph-based erspective.

D. Effective and Efficient Entity Search in RDF Data

The paper proposed a method for useful and competent entity search over RDF data. It also described an variation of the BM25F grading function for RDF data, and shows that it performs better than other up to date procedures in ranking RDF resources. It also projected a set of new indexing data structures for competent recovery and grading of outcome which were implemented using the open-source MG4J framework.

The paper tried to evaluate the effective and efficiency of the index structures (vertical and horizontal).

This paper showed that each of these features contributes to effectiveness on its own and in combination with other features. In cross-validation, the combination of these features outperforms in effectiveness the baseline BM25 method that ignores RDF structure and semantics by 50% in MAP score. It also improves on other State-of-the-art methods on the ad-hoc object retrieval task by 42% in MAP and 52% in NDCG scores.

E. Cost Analysis of Joins in RDF Query Processing Using the TripleT Index

This work can be viewed as a continuation of the work on an RDF indexing technique called TripleT, developed in 2008 by Fletcher and Beck [2]. Using TripleT as the framework, the goal is to better understand the requirements for building an effective SPARQL query optimizer. Specifically, it studied the information necessary to facilitate good join ordering. This research develop a model for predicting the number of I/Os required for a join based on TripleT using statistics that are easily collected during the creation of the index. Experiments are conducted to validate the model.

Advantages

1 Synthetic benchmarks found that the nested-loop join predictably performed much worse than the other two in terms of CPU performance

2 Hash join was about an order of magnitude faster than the sort-merge join when the latter had to sort the lists.

3 Benchmarks performed on real datasets showed much smaller differences in CPU performance as there were much fewer triples considered for the joins.

4 In terms of I/O performance, the nested-loop join was about 10% worse than the hash join and sort-merge join, which performed equally.

F. Suffix Arrays based Indexing Scheme for RDF and RDF Schema

The paper proposed a scheme that first take out four types of DAGs (Directed Acyclic Graphs) from an RDF data, and also take out all path expressions from the directed acyclic graphs. Then, it produced four types of suffix arrays based on the path expressions. Using the indices, we can get resourceful processing of query retrievals on RDF data as well as graphic information defined by RDF Schema (for example, classes and/or properties).

G. An Efficient SQL-based RDF Querying Scheme

This paper explains the working of the RDF_MATCH table function for querying RDF data, which can optionally include user-defined rule bases, and discusses its implementation in Oracle RDBMS is introduced with the ability to perform pattern-based match against RDF data (graph) that can optionally include triples inferred by applying RDFS or user-defined rules. Users can do further processing (iterate over, constrain using filter conditions, limit the results, etc.) using standard SQL constructs.

The experiments are conducted using Oracle10g Release 1 (10.1.0.2.0) on a Red Hat Enterprise Linux AS 3 system with one 3.06GHz Pentium 4 CPU and 2048 MB of main memory. A database buffer cache of 256 MB, shared pool of 256 MB, and database block size of 8 KB is used.

The RDF_MATCH table function itself is implemented by generating a SQL query against tables holding RDF data. For resourceful query processing, generic and subject- property matrix materialized join views, and indexes (on RDF data and rulebases) are used. Furthermore, a kernel enhancement is implemented that eliminates RDF_MATCH table function run-time processing overheads.

Further work can be done with partial normalization, where only the namespaces are normalized. That is, URIs are represented by the (namespace identifier, URI suffix). Also, we plan to enhance RDBMS optimizer to improve its capabilities in optimizing the class of self-join queries that typically occur while querying RDF data.

H. Structure Index for RDF Data

To elaborate on a novel data partitioning strategy, this leverages the structure of the underlying data. This structure is represented inform of a parameterized structure index we propose for (RDF) data graphs called PIG. It is not only used for data partitioning but also has been designed to accelerate the matching of graph-structured queries against RDF data.

We have proposed techniques for RDF data partitioning and query processing that can exploit the underlying structure to improve the management of RDF data, based on a novel structure index call PIG. In an principled manner, we showed that this approach is faster than the state-of-the-art, especially for complex structured queries.

Methodology:

This approach gives an advantage by making process 7-8 times faster for a PIG that is parameterized according to the query workload. According to author future work can be done to elaborate on how existing work on RDF query optimization can be used for the proposed structure-based query processing technique. Further, strategies proposed for optimizing updates of XML structure indexes will be studied and adopted.

2.1 Efficient parallel Index for RDF Data Store using Graphical Processing Unit– Literature Review

Nguyen et al (2012) have proposed a novel type of index structure called a B+HASH TREE, which combines the strengths of traditional B+ Trees with the fast constant-time lookup of hash-based structures. This structure compares the two indexing data structures (RDF-3X, B+ Hash) for index lookups. This system enhances the B+ Tree with a Hash Map enabling constant lookup and retrieval time instead of the common logarithmic one. Although the proposed structure has certain advantages and disadvantages but the results of this technique / structure are scalable, updatable and lookup-

Table 3 Summary of support of some current RDF query languages for some example graph property

PROPERTY	G	G+	GraphLog	Gram	GraphDB	Lorel	F-G
Adjacent nodes	partial support	support	Support	support	partial support	partial support	support
Adjacent edeges	partial support	support	Support	support	partial support	partial support	support
Degree of a node	No support	support	Support	No support	No support	No info	No support
Path	support	support	Support	support	support	support	support
Fixed-Length-Path	support	support	Support	support	support	support	support
Distance between two nodes	No support	support	Support	No support	No support	No info	No support
Diameter	No support	support	Support	No support	No support	No info	No support

Optimized, on-disk index structure that is especially suitable for the large key-spaces of RDF datasets.

RDF data management schemes are constrained in terms of effectiveness and scalability; which means an competent RDF storage format should propose both scalability in its data management performance and simplification in its data storage, processing and illustration. Weiss et al. (2008), in order to achieve this double goal, proposed a novel approach to RDF data management framework which is based on the scheme of indexing the Resource Description Framework data in a multiple index structure. This study compared the performance of Hexastore approach with the representation of the column oriented vertical-partitioning (COVP) and experimentally documented the advantages of this approach on real-world and copied data sets with realistic queries.

Neumann and Weikum (2008) presented the RDF-3X engine, an execution of SPARQL that realizes outstanding performance by following RISC-style design with a streamlined design and carefully designed puristic data structures and procedures. The performances of RDF-3X, in contrast to the earlier best up to date structures has been calculated on a number of comprehensive datasets with more than 50 million Resource Description Framework triples and standardize queries that consist of pattern matching and extensive join paths in the original data graphs. This technique presents a complete system, coined RDF-3X (for RDF Triple eXpress), designed and implemented from scratch specifically for the management and querying of RDF data.

Matono (2009) proposed two approaches for RDF query processing such as paragraph table as an RDF storing scheme that is based on the structure of RDF documents and the bloom filter merge join is a merge join algorithm that is suitable when the join selectivity is low. The RDF storing scheme concept is based on that the structure of input data resembles that of queries and the paragraph table stores RDF paragraphs into their corresponding relational tables as they are without decomposing or connection. Further, the study evaluated the approaches through some experiments. In thesummation of the processing times of all queries compared between the bloom filter merge join (BFMJ) and sort merge join (SMJ).

3. CONCLUSION

We have presented the RDF model from a database perspective; contrast it with other conceptual database models, focusing on query languages and graph databases and described variation of the BM25F grading function for RDF data, and shows that it performs better than other up to date procedures in ranking RDF resources. Also we have shown that how to we get better performance using Storing and Indexing Massive RDF Data Sets and exchanged HDT (Header, Dictionary, Triples) with supplementary structures to hold some basic forms of SPARQL query decision without the need of "unpacking" the data. Proposes that primitives of query language of graph database needs to be included in RDF query language. Therefore in this paper we have presented comparison of different proposed methodologies for RDF querying and indexing.

4. REFERENCES

[1] M.K. Nguyen, C. Basca and A. Bernstein, Speeding up on-disk RDF index lookups using B+Hash Trees, 2012, IOS Press, Zurich, Switzerland

[2] C. Weiss, P. Karras and A. Bernstein Hexastore:Sextuple Indexing for Semantic Web Data Management, 2008, VLDB Endowment, Auckland, New Zealand

[3] T. Neumann and G. Weikum, RDF3X: a RISCstyle Engine for RDF, 2008, VLDB Endowment, Auckland, New Zealand

[4] A. Matono, A Storing Scheme and A Merge Join Algorithm for RDF Query Processing, Jaban

[5] H. Stuckenschmidt, R. Vdovjak, J. Broekstra, Index Structures and Algorithms for Querying Distributed RDF Repositories. 2004, New York, USA

[6] Daniel J. Abadi · Adam Marcus · Samuel R. Madden · Kate Hollenbach, SW-Store: a vertically partitioned DBMS for Semantic Web data management. 2009, Springer-Verlag.

[7] P. Bakkum and K. Skadron, Accelerating SQL Database Operations on a GPU with CUDA. 2010, Pittsburg, PA, USA

[8] M. A. Bornea, J. Dolby, A. Kementsietsidis, K. Srinivas, P. Dantressangle, O.Udrea, B.Bhattacharjee, Building an Efficient RDF Store Over a Relational Database. 2013, New York, USA.

[9] R. Angles and C. Gutierrez, Querying RDF Data from a Graph Database Perspective.

[10] M. A. Martínez-Prieto, Mario Arias, and Javier D. Fernández. Exchange and Consumption of Huge RDF Data

[11] Y. Luo, F. Picalausa, G. H.L. Fletcher, J. Hidders, and S.Vansummeren, Storing and Indexing Massive RDF Data Sets

[12] R. Blanco, P. Mika, and S. Vigna, Effective and Efficient Entity Search in RDF data

[13] K. Li. Cost Analysis of Joins in RDF Query Processing Using the TripleT Index. 2008. Master Thesis. Faculty of the Graduate School of Emory University.

[14] A. MATONO, T. AMAGASA, M. YOSHIKAWA, and S. UEMURA, An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays

[15] E. I. Chong , S. Das, G.Eadon and J. Srinivasan, An Efficient SQL-based RDF Querying SchemeProceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

[16] T. Tran and G.Ladwig, Structure Index for RDF Data. Workshop on Semantic Data Management (SemData@VLDB) 2010,September 17,2010, Singapore.

[17] YounHee Kim; ByungGon Kim; HaeChull Lim, "The index organizations for RDF and RDF schema," Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference , vol.3, no., pp.4 pp.,1874, 20-22 Feb. 2006 doi: 10.1109/ICACT.2006.206357 keywords: {database indexing; metadata;query processing;semantic Web;RDFschema;graph models;index organizations;indexing techniques;keyword index;keyword-based query;metadata;ontology;path-based query;semantic Web;Data models;Educational institutions;Image databases;Indexing;Information resources;Information retrieval;Ontologies;Resource description framework;Semantic Web;Web pages;Index schemes;Keyword-based query}, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1625962 & isnumber=34122