A Survey on Big RDF Graph by using Indexing Approaches in Big Data

Sunil Parmar[1] Prof. Darshan Upadhyay[2]
[1,2]Department of Information Technology
[1,2]V.V.P. Engineering College, Rajkot, India

Abstract— Many public knowledge bases are represented and stored into RDF graphs, where users can issue structured queries on such graphs using SPARQL. With massive queries over large and constantly growing RDF data, it is imperative that an RDF graph store should provide low latency and high throughput for concurrent query processing. We present, a distributed graph-based RDF store that leverages RDMA-based graph exploration to provide highly concurrent and low-latency queries over large data sets. This is follows in three ways. First, provides an RDMA-friendly distributed key/-value store that provides differentiated encoding and fine-grained partitioning of graph data to reduce RDMA transfers. Second, the cost of one-sided RDMA operations is largely oblivious to the payload size to a certain extent so avoid the cost of expensive final join operations. Third, countering conventional wisdom of preferring migration of execution over data, it is seamlessly combines data migration for low latency and execution distribution for high throughput by leveraging the low latency and high throughput of one-sided RDMA operations. An RDF stream is a sequence of RDF graphs, including associated metadata, as a flexible mechanism to add time-related metadata. RSP is to define a common model for producing, transmitting and continuously querying RDF Streams. It is continuous queries over streams of RDF data to improve the low latency and high throughput, it's better than prior system.

Keywords: Big Data, RDF Graph, Streaming Data, RDMA, SPARQL, RSP

I. INTRODUCTION

Many public knowledge bases are represented and stored as RDF graphs, where users can issue structured queries on such graphs using SPARQL. With massive queries over large and constantly growing RDF data, it is imperative that an RDF graph store should provide low latency and high throughput for concurrent query processing. We present, a distributed graph-based RDF store that leverages RDMA-based graph exploration to provide highly concurrent and low-latency queries over large data sets.[1]

Nowadays we all are surrounded by huge data. People upload/download videos, audios, images from variety of devices. Sending text messages, multimedia messages, updating their Facebook, WhatsApp, Twitter status, comments, online shopping, online advertising etc. generates huge data. As a result, machines have to generate and keep huge data too. Due to this exponential growth of data the analysis of that data become challenging and difficult.[2]

The term „Big Data" means huge volume, high velocity, variety and veracity i.e. uncertainty of data. This big data is increasing tremendously day by day. The Big data generated may be structured data, Semi Structured data or unstructured data. Existing databases and tools are not good enough to process, analyze, store and manage such a Big Data effectively and efficiently. Extremely large data sets, that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions.[2]

Many large datasets are continuously published using the Resource Description Framework (RDF) format, which represents a dataset as a set of <subject, predicate, object> triples that form a directed and labeled graph, eg. Google's knowledge graph. With the increasing scale of RDF datasets and the growing number of queries per second, it is highly demanding that an RDF store provides low latency and high throughput over highly concurrent queries. In response, much recent research has been devoted to developing scalable and high performance systems to index RDF data and process SPARQL queries.[1]

An RDF dataset is essentially a labeled, directed multigraph. Hence, it may be either stored as a set of triples as elements in relational tables (i.e., a triple store) or managed as a native graph (i.e., a graph store). They describe Wukong, a distributed in memory RDF store that provides low-latency, concurrent queries over large RDF datasets. To make it easy to scale out, They follows a graph-based design by storing RDF triples as a native graph and leverages graph exploration to handle queries. Unlike prior graph-based RDF stores that are only designed to handle one query at a time, It is also designed to provide high throughput such that it can handle hundreds of thousands of concurrent queries per second. The key techniques are centered around using one-sided RDMA to provide fast and concurrent graph exploration.[1]

An RDF dataset is a graph (RDF graph) composed by triples, where a triple is formed by <sub ject, predicate,ob ject>. A triple can be regarded as a directed edge (predicate) connecting two vertices (from subject to object). Thus, an RDF graph can be alternatively viewed as a directed graph G = (V,E), where V is the collection of all vertices (subjects and objects), and E is the collection of all edges, which are categorized by their labels (predicates).[1]

SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs. SPARQL is the standard query language for RDF datasets. The most common type of SPARQL queries is as follows: Q: = SELECT RD WHERE GP. where, GP is a set of triple patterns and RD is a result description. Each triple pattern is of the form <subject, predicate, object>, where each of the subject, predicate and object may denote either a variable or a constant.[1]

Remote Direct Memory Access (RDMA) is a cross-node memory access technique with low-latency and

All rights reserved by *www.ijsrd.com*

low CPU overhead, due to complete bypassing of target OS kernel and/or CPU. RDMA provides both two-sided message passing interfaces like SEND/RECV Verbs as well as one-sided operations such as READ, WRITE and two atomic operations (fetch-and-add and compare-and-swap). As in prior work, one-sided operations are usually less disruptive than its two-sided counterparts due to no CPU involvement to the target machine. An interesting feature is that the latency of RDMA is relatively insensitive to payload sizes, because small-sized requests cannot saturate the high-bandwidth network card.[1]

This designed to handle a massive number of RDF queries concurrently while trying to parallelize a single query to reduce the query latency. The difficulty is that, given the significantly varied query latencies, how to minimize inter-query interference while providing good utilization of resources, e.g., a lengthy query should not significantly extend the latency of a fast query.[1]

RSP (RDF Stream Processing) engines emerged in recent years, with the goal of extending RDF and SPARQL to process RDF streams. They can be broadly divided into two groups. The first is of those influenced by Complex Event Processing (CEP), e.g. EP-SPARQL, Spark wave and Instans. The second group is that of engines inspired by DSMS, which exploit sliding window mechanisms to capture a finite portion of the input data. Examples include CSPARQL, CQELS, and SPARQL stream.[1]

RSP queries should be able to access all knowledge explicitly expressed in the stream, including names of named graphs and triples containing such names. It is support combining multiple RDF streams as well as stored RDF (aka static RDF graphs or datasets)[3]

II. TECHNIQUES:

Jiaxin Shi.[1], authors have describe a model: Wukong, a distributed in memory RDF store that provides low-latency, concurrent queries over large RDF datasets. To make it easy to scale out, His model follows a graph-based design by storing RDF triples as a native graph and leverages graph exploration to handle queries. Unlike prior graph-based RDF stores that are only designed to handle one query at a time, His model is also designed to provide high throughput such that it can handle hundreds of thousands of concurrent queries per second. The key techniques of his model are centered on using one-sided RDMA to provide fast and concurrent graph exploration.

A distributed graph-based RDF store that leverages RDMA-based graph exploration to provide highly concurrent and low-latency queries over large data sets. They provides an RDMA-friendly distributed key/value store that provides differentiated encoding and fine-grained partitioning of graph data to reduce RDMA transfers, wukong leverage, the cost of one-sided RDMA operations is largely oblivious to the payload size to a certain extent so avoid the cost of expensive final join operations. His model seamlessly combines data migration for low latency and execution distribution for high throughput by leveraging the low latency and high throughput of one sided RDMA operations.

His model can achieve up to 269K queries per second on 6 machines with 0.80 milliseconds median latency. It targets SPARQL queries over a large volume of RDF data; it scales by partitioning an RDF graph into a large number of shards across multiple machines. His model is at least one order of magnitude faster than TriAD since the queries are quite selective and appropriate for graph exploration. It significantly outperforms state-of-the-art systems and can process a mixture of small and large queries at 269K queries/second on a 6-node RDMA-capable cluster.

Ruchita Shah[4]. Authors have describe the Query processing model which would be used for querying Streaming RDF Data. The data which is used in this model can be of RDF as well as non-RDF format. This Triple data is fed into Wave which gives RDF stream as an output. They focus on the Query Processing Model built by integrating C-SPARQL and Triple Wave and objective is to evaluate its performance in term of execution time and memory consumption of several queries having variable complexities.

RSP (RDF Stream Processing) engines emerged in recent years, with the goal of extending RDF and SPARQL to process RDF streams. Triple Wave is a reusable and generic tool that enables the publication of RDF streams on the Web. It is independent of any specific engine implementation, and it can be fed with existing Web streams (e.g. Twitter and Wikipedia streams) or time-annotated RDF datasets (e.g. the Linked Sensor Data dataset). Continuous SPARQL (CSPARQL) is a new language for continuous queries over streams of RDF data. C-SPARQL queries consider windows, i.e., the most recent triples of such streams, observed while data is continuously flowing. They compute the execution time and the memory consumption of queries with different complexities by varying the input parameters by Data Set, Query Set, Test Criteria, Hardware and Software Setup.

M. Tamer Ozsu [5]. Authors have been developing a set of graph-based RDF data management systems that follow this approach: gStore, gStore-D and gAnswer. The first two are designed to support efficient SPARQL query evaluation in a centralized and distributed environments, respectively, while the last one aims to provide an easy-to-use interface (natural language question/answering) for users to access a RDF repository.

They describe two typical approaches to designing RDF data management systems: relational approaches and graph-based approaches. The relational approaches map RDF data to a tabular representation in a number of ways and then execute SPARQL queries on them, sometimes mapping SPARQL queries to SQL. The second major approach of systems is graph-based, which model both RDF data and the SPARQL query as a graph and evaluate the query by sub-graph matching using homomorphism. The advantage of graph-based approach is that it maintains the original representation of the RDF data and enforce the intended semantics of SPARQL.

gStore's to reduce the search space and improve query performance, there are two key techniques in gStore: vertex encoding and indexing/querying techniques. The increasing size of RDF data requires a solution with good

All rights reserved by www.ijsrd.com

scale-out characteristics. Furthermore, The increasing amount of RDF data published on the Web requires distributed computing capability, they address this issue by developing a distributed version of gStore that we call gStore-D. gAnswer, Providing end users an easy-to-use interface to access RDF datasets in an effective way has been recognized as an important concern. This has lead to research for RDF question/answering (Q/A) systems. They have designed gAnswer to address the problem from the perspective of a graph database. The design philosophy behind our systems is to employ graph database techniques for RDF data management.

Avimanyu Mukhopadhyay[6]. Authors have shown a RAPIDFacet model, by Leveraging Apache Spark and the NTGA data model helped us develop RAPIDFacet which can perform faceted search significantly faster. Faceted search is an exploratory search paradigm where users can view data in terms of multiple orthogonal dimensions of data called facets. This search model is very common with e-commerce websites. They used Apache Spark platform, Spark's execution model is based on Resilient Distributed Datasets (RDDs) which get manipulated into other RDDs by actions and transformations in a distributed fashion. The RAPIDFacet have to approaches, apache spark and nested triplegroup data model and algebra(NTGA).

An Apache Spark-based query execution model for evaluating faceted search queries on knowledge graphs that aligns better with the iterative nature of faceted search and exploits Spark's lineage model for reuse of intermediate results and a conceptual query model for faceted search that is based on an earlier proposed Nested Triplegroup Data Model and Algebra (NTGA) and the compilation of NTGA algebra expressions representing faceted search into Spark workflows. A key advantage of this strategy is the minimized number of joins required in the execution workflows for faceted search on graphs.

Houda Khrouf[7]. Authors have been made to design a new generation of RSP engines based on modern big data frameworks. They propose, called WAVES, a new distributed RSP system built on top of popular big data frameworks and supporting continuous queries over event-based streaming data. They have presented the design of a distributed and production-ready RDF stream processing engine. Their generic architecture is based on the use of popular big data frameworks which have proven to be useful in countless scientific and industrial applications. The WAVES system is able to run continuous queries on event-based streaming data in distributed environment.

These systems processes dynamic and event-based data which are handled by distributed components, i.e., Redis as an in memory store, Kafka as a publish-subscribe distributed message broker and Storm as a distributed stream processing framework. They also propose an extension of RDSZ to improve compression and to adapt it for a distributed architecture, Their approach called D-RDSZ (Distributed RDSZ), uses gzip instead of zlib.

Hiba Arnaout[8]. Authors have been develop a general framework for effective searching of RDF knowledge graphs. Their framework allow users to search the underlying knowledge graph using both triple-pattern queries with keywords. This addresses the case when queries return too many results and ensures that the user is provided with a ranked list of result sub-graphs based on their relevance to the query.

Their framework supports result ranking by means of a novel ranking model based on statistical machine translation. In addition, these framework supports automatic query relaxation to improve query recall.

RDF knowledge graphs are typically searched using triple-pattern queries. They proposed three different methods to build triple pattern queries are formed as Knowledge- graph-based, Query-based and Text-based. Their framework provides various notions of result diversity specifically defined for the setting of RDF and that are achieved using the Maximal Marginal Relevance re-ranking approach.

Dongyan Zhao[9]. Authors have explore the workload for fragmentation and allocation, which aims to reduce the communication cost during SPARQL query processing. Specifically, they adaptively maintain some frequent access patterns (FAPs) to reflect the characteristics of the workload while ensuring the data integrity and approximation ratio. Based on these frequent access patterns, they propose three fragmentation strategies, namely vertical, horizontal and mixed fragmentation, to divide RDF graphs while meeting different types of query processing objectives.

They also discuss the technical issues, 1)FAP selection and maintenance: They build a fragment by collecting all its matches in the RDF graph to reduce the communication cost. 2) Vertical, horizontal, and mixed fragmentation: It through query evaluation and reduce the query response time for a single query. 3) Query decomposition and optimization: As there are some data replications in our fragmentation strategies for RDF graphs. The Vertical Fragmentation strategy utilizes the locality of SPARQL queries to improve both the query response time and throughput, Horizontal fragmentation strategy aims to utilize the parallelism of clusters of sites to reduce the query response time and Mixed strategy designed to reduce the query response time by parallelizing sub-queries.
They managing large RDF graphs in distributed environment so graph needs to be partitioned into smaller fragment then by applying allocation algorithm and graphs are distributed over different sites.

Abdessamad Belangour [10]. Authors have shown comparison between all types of NoSQL Databases and overview on the voluminous RDF data management systems using NoSQL databases. They studies and evaluates these databases RDF (called triplestore), and describes a comparison based on some criteria of database software such as the NoSQL database, and model of database, index structure, and database license.

They compared five proposed implementations for storing massive NoSQL-based RDF data that are: Jena + Hbase, Hive + Hbase, Couchbase, 4Store, and CumulusRDF, in both distributed and single-machine deployment modes. They presented an overview of huge RDF data management systems based on Big Data's NoSQL technology.

All rights reserved by www.ijsrd.com

III. CONCLUSION:

In the real-life application and systems are characterized by imprecise and uncertain data. RDF represent streaming data into graph format and then manage, storing and fast searching them through Amazon Neptune Database. From this we could get better result. It is also designed to provide high throughput such that it can handle the concurrent queries in microsecond. It handles the live streaming data by using RDMA to bypass transfer operation. In this model, we divides the performance of query in small and large query, so this model is more accurate and faster than existing model.

REFERENCE:

[1] Jiaxin Shi, Youyang Yao, Rong Chen, Haibo Chen,(Nov-2016), "Fast and Concurrent RDF Queries with RDMA-based Distributed Graph Exploration", (usenix), pp 317-332.
[2] Sachin Arun Thanekar, K. Subrahmanyam, Big Data and MapReduce Challenges, Opportunities and Trends http://ijece.iaescore.com/index.php/IJECE/article/view/5906
[3] RDF Stream Processing: Requirements and Design Principles, w3c-2016, http://streamreasoning.github.io/RSP-QL/RSP_Requirements_Design_Document/#time-in-Stream-data-elements
[4] Ruchita Shah, Ami Pandat, Minal Bhise, "Query Processing for Streaming RDF Data", 2018, pp 75-78, IEEE.
[5] Lei Zou, M. Tamer Ozsu, "Graph-Based RDF Data Management", 2017, vo-70, pp 56-70, Springer.
[6] Avimanyu Mukhopadhyay, HyeongSik Kim, Kemafor Anyanwu, "Scalable Exploratory Search on Knowledge Graphs Using Apache Spark", 2018, pp 154-159, IEEE.
[7] Hounda Khrouf, Badre Belanness, Laurent Bihanic, Gabriel Kepeklian, Olivier Cure, "WAVES: Big Data Platform for Real-time RDF Stream Processing", 2016, pp 1-12, wave-rsp.
[8] Hiba Arnaout, Shady Elbassuoni, "Effective searching of RDF knowledge graphs", 2018, pp 1107-2020, Elsevier.
[9] Peng Peng, Lei Zou, Lei Chen, and Dongyan Zhao, "Adaptive Distributed RDF Graph Fragmentation and Allocation based on Query Workload", 2018, vo-14, pp 1-14 IEEE.
[10] Mouad Banane, Abdessamad Belangour, El Houssine Labriji, "RDF Data Management Systems Based on NoSQL Databases: A Comparative Study", 2018, Vo-58, pp 98-102, IJCTT.
[11] Big data Architecture Definition, https://www.omnisci.com/technical-glossary/big-data-architecture

All rights reserved by *www.ijsrd.com*