International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 4 Issue: 4

ISSN: 2454-4248
505 – 508

Managing RDF Graphs using Mapreduce Algorithm with Indexing Solution for Future Direction

Hetal k. Makvana
M.E. Computer Engineering
Noble Group of Institutions
Junagadh, India
virat.jani@ngivbt.edu.in

Prof. Ashutosh A. Abhangi
Computer Engineering
Noble Group of Institutions
Junagadh, India
Ashutosh.abhangi@ngivbt.edu.in

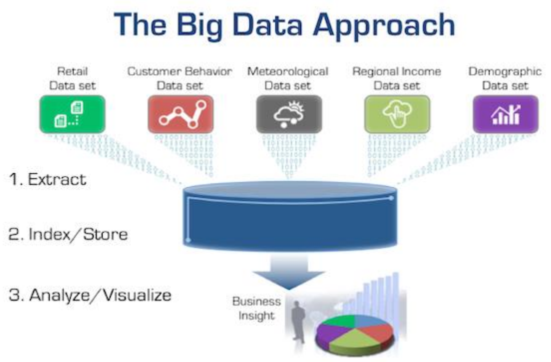
Abstract — *"Indexing solution" based on Big RDF (Resource Description Framework) graphs with improve processing which populate the semantic web, are the core data structure of the big web data, the natural transposition of big data on the web. Indexing data structure improve processing on the big RDF graph. it was present the "baseline operation" of fortunate web big data analytic. this require process, access and manage RDF graphs. It was dealing with severe temporal complexity. A solution to problem is represented by MapReduce model based algorithm for indexing solution which try to exploit the computation power offered by the MapReduce processing model in indexing order. this paper provide a survey on MapReduce based algorithm for state-of-the-art proposal using indexing solution.*

Keywords- *Big Data; Indexing; RDF Graph; Big data Management; Big Data Mapping*

*****

I. INTRODUCTION

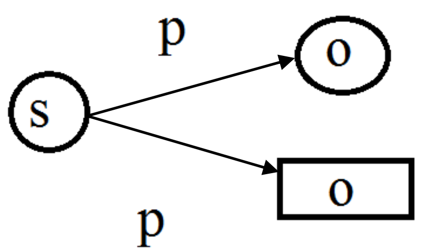
The term big data applies to information that can't be processed or analyzed using traditional processes or tools. Increasingly, organizations today are facing more and more Big data challenges. Challenges include capture, storage, analysis, data creation, search, sharing, transfer, visualization, querying, updating and information privacy.



<u>BIG DATA</u>

II. RDF

We can manage rdf items without destroy of items. After join and reduce some items from database we use it in future but instead of indexing items, using mapreduce algorithm we can use reuse those items in future if we need. By indexing items we can reduce the support or confidence of sensitive rules. Minimum number of graph that need to be modified to indexing a sensitive rule is derived. Try to achieve fewer side effects and indexing.



Vertices
Resources : URLs
Attribute value : Literal values
Edges
Relationship : URIs

III. LITERATURE SURVEY

A. *MapReduce-based Algorithms For Managing Big RDF Graphs: State-of-The-Art Analysis, Paradigms, And Future Direction .*

Big RDF (Resource Description Framework) graphs, which populate the emerging *Semantic Web*, are the core data structure of the so-called *Big Web Data*, the "natural" transposition of *Big Data* on the Web. *Managing big RDF graphs* is gaining momentum, essentially due to the fact that this task represents the "baseline operation" of fortunate *Web big data analytics*. Here, it is required to access, manage and process *large-scale*, *million-node* (big) RDF graphs, thus dealing with severe *spatio-temporal complexity challenges*. A possible solution to this problem is represented by the so-called *MapReduce- model-based algorithms for managing big RDF graphs*, which try to exploit the computational power offered by the MapReduce processing model in order to tame the complexity above. In this so-depicted scientific context, this paper provides a critical survey on MapReduce-based algorithms for managing big RDF graphs, with analysis of state-of-the-art proposals, paradigms and trends, along with a comprehensive overview of future research trends in the investigated scientific area.

B. rdf chain: chain centric for scalable join processing on rdf graph using mapreduce and hbase

RDFChain showed the best performance in large non-selective queries (Q2 and Q9). In particular, Q2 and Q9 have a complex structure, a low selectivity due to unbound objects, and a relationship of a chain pattern join. RDFChain greatly reduced the size of the intermediate results by limiting RDF triples to actual candidate rows which can satisfy a chain pattern join. RDFChain also shows smaller number of storage accesses than MAPSIN. Since Tcom is a common subset of Tspo and Tops, it scales down the scan space. RDFChain splits TPGs with compatible mappings and process the divided TPGs in a map task. So, the number of map jobs decreases in turn.

C. Scan-Sharing For Optimizing RDF Graph Pattern Matching on MapReduce

We extend our previous efforts on algebraic optimization of RDF graph pattern queries to enable efficient handling of graph pattern queries with multiple occurrences of a property type; a common scenario in RDF queries. Our approach formalizes and integrates the concept of “cloning” as part of appropriate operators of our NTGA algebra, avoiding the need for multiple scans of input relations required in relational algebra-based query plans. Our extensive experimental evaluations with various workloads have shown the effectiveness and scalability of our intra-query scan-sharing approach. Future directions will investigate additional work sharing opportunities across sub queries that may involve sharing across TG_Join operators.

D. Different Clustering algorithms for big data analytics: a review

We have considered several clustering methods which are presently and widely used for big data analysis. This work delivered an all-inclusive study of the clustering procedures projected in the literature. Analyzing the online streamed data can be considered in the future work. Still there is a huge gap in examining the big data.

E. The Memory Challenge in Reduce Phase of MapReduce Applications

Memory has an important role in performance of Reduce phase in many MapReduce applications. It not only can degrade the performance, but also can lead to job failure due to lack of memory. So, if an approach considers memory correctly in the process of decision making about Reduce slots configuration as well as number of Reduce tasks, it can achieve high performance. Our memory aware approach, Mnemonic, considers this fact and achieves high performance compared with Memory Oblivious and Fine Grain approaches. Our major contributions in this approach are 1) accentuating the impact of memory on intermediate data management, 2) investigating the slot configuration and configure memory size of each slot, and 3) setting the number of Reduce tasks as well as memory size of Reduce slots properly to eliminate job failure and increase the performance of applications.

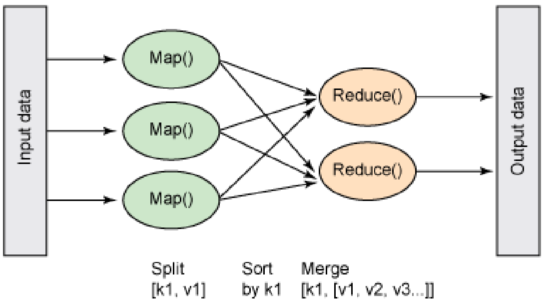
F. Scaling Unbound-Property Queries on Big RDF Data Warehouses using

RDF storage schema on HBase. We've also proposed a MapReduce join algorithm for SPA RQL BGP processing with evaluation results. As discussed in Section V, current implementation can be enhanced in many ways which we can adopt as future work. We hope we can implement a full featured RDF store with HBase and MapReduce finally.

IV. PROPOSED WORK

MapReduce :

The MapReduce algorithm contains two important tasks, namely Map and Reduce:



MapReduce process

The input data of the Map Phase Join comes from all joined triple queries formed in key/value pairs of the above format. Mappers read values contained in each pair and break them up to find the join variable. For each join variable binding, they produce a key-value pair with the binding as the key and the bindings for all other variables contained in the input pair as the value. The pattern id is also added in the value. Key-value pairs produced by mappers are sorted and grouped together based on their key.

The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).

The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples. The reduce task is always performed after the map job.

ISSN: 2454-4248
505 – 508

Let us now take at each of the phases :

Input Phase - Here we have a Record Reader that translates each record in an input file and sends the parsed data to the mapper in the form of key-value pairs.

Indexing phase - is an indexing all node in rdf graph

Map - Map is a user-defined function, which takes a series of key-value pairs and processes each one of them to generate zero or more key-value pairs.

Intermediate Keys - They key-value pairs generated by the mapper are known as intermediate keys.

Combiner - A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it is optional.

Shuffle and Sort - The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.

Reducer - The Reducer takes the grouped key-value paired data as input and runs a Reducer function on each one of them. Here, the data can be aggregated, filtered, and combined in a number of ways, and it requires a wide range of processing. Once the execution is over, it gives zero or more key-value pairs to the final step.

Output Phase − In the output phase, we have an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer. Let us try to understand the two tasks Map & Reduce .

MapReduce – Algorithm:

MapReduce implements various mathematical algorithms to divide a task into small parts and assign them to multiple systems. In technical terms, MapReduce algorithm helps in sending the Map & Reduce tasks to appropriate servers in a cluster. These mathematical algorithms may include the following .

- Sorting
- Searching
- Indexing

Sorting - Sorting is one of the basic MapReduce algorithms to process and analyze data. MapReduce implements sorting algorithm to automatically sort the output key-value pairs from the mapper by their keys. Sorting methods are implemented in the mapper class itself. In the Shuffle and Sort phase, after tokenizing the values in the mapper class, the Context class (user-defined class) collects the matching valued keys as a collection.

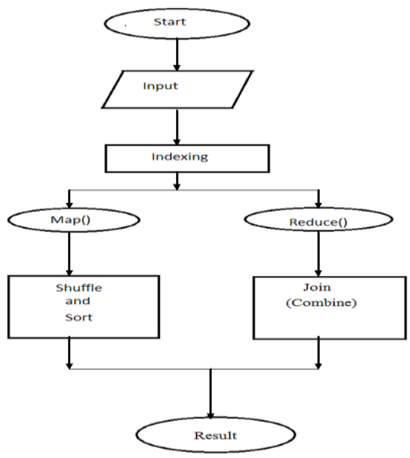
Searching - Searching plays an important role in MapReduce algorithm. It helps in the combiner phase (optional) and in the Reducer phase. Let us understand how Searching works with the help of an example.

Indexing - Normally indexing is used to point to a particular data and its address. It performs batch indexing on the input files for a particular Mapper.

Type of indexing

1) *Hash table indexing*
2) *Tree based table*
3) *Multidimentional indexing indexing*
4) *Bitmape indexing*

We have using tree based indexing. It was effective performance with MapReduce.



Flow Chart Of Proposed System

V. Result

Here we created rdf graph And we focused on indexing ratio and we what to check its performance so we first apply on to node.

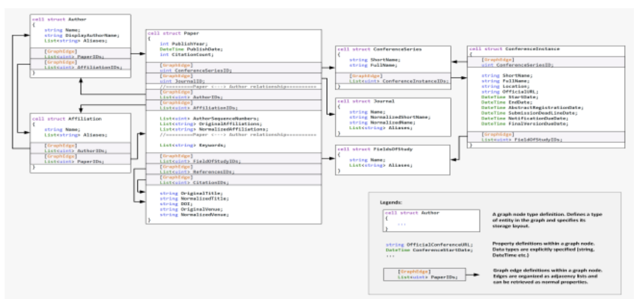


Chart-2: rdf graph input

ISSN: 2454-4248
505 – 508

Then we applied to proposed rdf graph and we get improved result shown below.

Teams	Hashtags #
australia	12609
bahamas	559
belgium	505
botswana	227
brazil	2190
canada	28719
caribbean	807
china	1177
croatia	431
egypt	1107
ethiopia	1651
france	1224
gb	401203
germany	532
ghana	426
grenada	945
haiti	740
holland	206
hungary	827
india	1434
indonesia	2646
iran	570
ireland	17798
israel	1048
italy	1052
jamaica	21033
japan	985
kenya	11152
korea	515
latvia	582
malaysia	9077
mexico	1355
mongolia	1673
nigeria	5716
pakistan	475
philippines	2159
poland	575
puertorico	753
qatar	1077
russia	2093
serbia	303
singapore	785
somalia	324
spain	550
sweden	206
turkey	391
uganda	728
usa	204269
zimbabwe	286

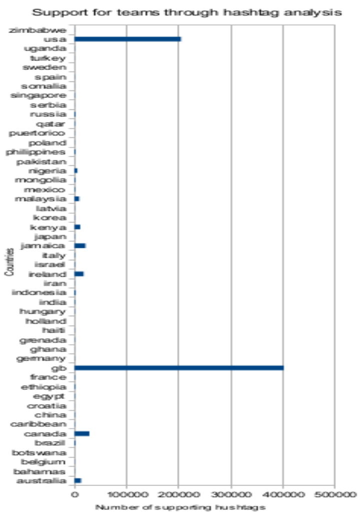


Chart-2: hashtag Analysis

VI. CONCLUSION

MapReduce Algorithm can help to manage big RDF graphs any items. It was achieving effective and efficient mapreduce based algorithms for supporting big RDF graph management. RDF multiple databases and multiple tables are join and reduce it referred to as a cluster. I will try to perform RDF graph indexing we need in future then perform manage on that then we can reuse it. Indexing data structure improve query processing on big RDF graph After map rules, it contains are selected for modification. So the side effects will be indexing RDF graph.

References :

I. Alfredo Cuzzeerea, Rajkumar Buyya, Vincenzo Passanisi, Giovanni Pilato, "MapReduce-based Algorithms For Managing Big RDF Graphs: State-Of-The-Art Analysis, Paradigms, and Future Direction", © 2017 17th IEEE/ACM Inernaional Symposium on cluster, cloud and grid computing.

II. Pilsik Choi1,2 *, Jooik Jung1 and Kyong-Ho Lee1," RDFChain: Chain Centric Storage for Scalable Join Processing of RDF Graphs using MapReduce and HBase",© ISWC-PD 2013 in International Semantic Web Conference.

III. HyeongSik Kim, Padmashree Ravindra, Kemafor Anyanwu" Scan-SharingforOptimizingRDFGraphPatternMatchingon MapReduce ",2012 IEEE Fifth International Conference on Cloud Computing 978-0-7695-4755-8/12 $26.00 © 2012 IEEE DOI 10.1109/CLOUD.2012.14.

IV. Dr. Meenu Davel and Remant Gianey2 "Different Clustering Algorithms for Big Data Analytics: A Review", SMART -2016, IEEE Conference ID: 39669 5th International Conference on System Modeling & Advancement in Research Trends, 25th_27'h November, 2016. Copyright © SMART -2016 ISBN: 978-1-5090-3543-4.

V. Seyed Morteza Nabavinejad, Maziar Goudarzi, "The Memory Challenge in Reduce Phase of MapReduce Applications", doi 10.1109/tbdata.2016.2607756, ieee transactions on big data journal of l atex class files, vol. 14, no. 8, august 2015.

VI. Padmashree Ravindra, Kemafor Anyanwu, 'Scaling Unbound-Property Queries on Big RDF Data Warehouses using MapReduce', (c) 2015, Copyright is with the authors. Published in 18th International Conference on Extending Database Technology (EDBT), March 23-27, 2015, Brussels, Belgium: ISBN 978-3-89318-067-7.