A Survey on an Efficient Query Processing and Analysis on Big Data (RDF) Using Map Reduce

Pravinsinh Mori  
PG Student  
Department of Computer Engineering  
Kalol Institute of Technology & Research Centre

A. R. Kazi  
Professor  
Department of Computer Engineering  
Kalol Institute of Technology & Research Centre

Sandip Chauhan  
Professor  
Department of Computer Engineering  
Kalol Institute of Technology & Research Centre

Abstract

In Big Data analysis Semantic Web Large data analysis is an important topic in cloud computing that are used particularly using RDF. This has positioned the issue of scalable data processing techniques for RDF as a vital issue in the Semantic Web research community. The RDF data model is a fine-grained model representing associations as binary relations. Thus, responding queries over RDF data requires several join operations to reassemble related data. While MapReduce based processing is emerging as the Map and Reduce paradigm for processing large scale data. In addition, most of the existing techniques for optimizing RDF data handing out do not transfer well to the MapReduce model and often require significant lead time for pre-processing. MapReduce is a programming structure in cloud computing to compute data analysis in parallel. We suggests three concepts. First data are filtered primary according to the query statements. Second, the filtered data are sent to its equal worker according to the join expression for advanced level parallelism. Each worker then performs the corresponding join operation after acceptance of the filtered data. Finally, we aggregate the result by using aggregate functions specific in the select clause.

Keywords: Map Reduce, MJR Framework, RDF, Query Optimization, RDF Graph Pattern Matching, Hadoop.

I. INTRODUCTION

Cloud Computing provide users with massive amount of Big data storage and analysis of High-level data. Big data is defined as the ability to wring business value from the large volume, variety, and velocity of information becoming available. As per the recent report on big data implementation shows that many organizations find the variety element of big data a much bigger challenge than volume or velocity. Many fields require organization to Manage and analysis of huge amounts of data. Such fields include social network to analysis, financial-risk management, threat finding in complex network systems, and medical and biomedical large databases. These are all examples of big data analytics, in which dataset sizes enlarge exponentially. These application fields create operational challenges not only in terms of sheer size but also in time to solution, because quickly answering queries is essential to obtaining market advantages, avoiding vital security issues, or preventing life-threatening health problems. Semantic graph databases seem to be a promising resolution for storing, managing and querying the large and different datasets of these application fields. Such datasets present an abundance of relations among many elements. Semantic graph databases organize the data in the form of subject-predicate-object triples following the Resource Description Framework (RDF) data model.

II. INTRODUCTION RDF AND MAP REDUCE

A. RDF:

In this section, we primary introduce RDF (Resource Description Framework) [7]. Semantic web technologies are being developed to current data in standardized way such that such data can be retrieved and understood by both human and machine. Historically, WebPages are published in simple html files which are not suitable for reasoning. Instead, the machine treats these html files large amounts of keywords. Researchers are developing Semantic web technologies that have been standardized to deal with such inadequacies. The most well-known standards are Resource Description Framework (RDF) , SPARQL Protocol and RDF Query Language (SPARQL). RDF is the model for storing and representing data and SPARQL is a query language to retrieve data from an RDF dataset.
A Survey on an Efficient Query Processing and Analysis on Big Data (RDF) Using Map Reduce

Resource Description Framework (RDF) [1] is a language for present the information or data about resources in the WWW. The resources are not limited to web pages but can also contain things that can be identified on the web. The design of metadata in the generic RDF format makes it appropriate for automatic consumption by a diverse set of applications. The RDF data represented as a set of <subject, property, object> triples.

As per above figure, consider RDF data about research paper checkers. The RDF classes and the triple instances are shown in Figure 1. Assuming the RDF data is stored in the database as the representation of 'checkers', user can issue the following query to find reviewers who are students with age less than 27:

```sql
SELECT t.r reviewer
FROM TABLE(RDF_MATCH(
  (?r ReviewerOf ?c)
  (?r rdf:type Student)
  (?r Age ?a),
  RDFModels('reviewers'),
  NULL, NULL)
WHERE t.a < 27;
```

The various arguments to RDF_MATCH are as follows: The first argument captures the graph pattern to search for. It uses SPARQL-like syntax [13] and variables are prefixed with a '?' character. The second argument specifies the model(s) to be queried.

The third argument specifies the rule bases (if any). Here the NULL argument indicates absence of rule bases. The fourth argument specifies user-defined namespace aliases (if any). Here the NULL argument indicates that no user-defined aliases are used, however default aliases such as rdf: are always available.

### B. MAP REDUCE:

Large amount of data processing frameworks, such as MapReduce supply capabilities of extensibility, fault tolerance, and parallel data processing. MapReduce framework is composed of a master and many number of workers that conduct MapReduce jobs. A MapReduce job includes Map segment and Reduce segment. At the time perform a MapReduce job, the master will divide the data to be processed into multiple same-sized blocks, and create M Map works and R Reduce works. MapReduce has emerged as accepted a way to tie together the power of large clusters of computers. MapReduce allows programmers to consider in a data-centric fashion: they focus on applying transformations on sets of data, allow the details of distributed execution, network communication and fault tolerance to be handled by the MapReduce. MapReduce is typically applied on big batch-oriented computations that are concerned primarily with time to job completion.
III. RDF VS. RELATIONAL DATABASE

In a usual Relational Database support structure database schema. In RD there are several databases table that are populating with information about their relation. In relational database the example as the Student information table, Employee data, Salary table that all are combine and join after we get School useful information. It required very long process and complicated. Now we talk about RDF. There have been a lot of attempts to shred RDF data into relational model. In approach involves a single triple-store relation with three subject, predicate and object. After that RDF triple becomes a single tuple, which for a popular dataset. SPARQL is popular query languages which is used to query RDF data. RDF data mainly supported Web based heavy data. RDF should be able of rapidly processing large amounts of data and should also generate more knowledge from the existing data. In RDF data that is gathered from sources in different geographical locations is naturally distributed. It mainly to work on compress the uncompressed data, also filtered data as per prediction, subject and object based.

IV. STATE OF THE ART ON MAP REDUCE TECHNIQUES

MapReduce is a programming model, which provides easy way of parallelizing complex tasks. MapReduce is encouraged from the functional programming language that provide map and reduce primitives. The framework surrounding MapReduce model manages the concerns related to work division, fault-tolerance, data area over a distributed file system and provides abstraction to implement the programming logic in Map and Reduce. This model is like to a split and combined model of parallelization, where a given task that needs to be performed on a set of data is managed by executing the task at a time on the chunk of data splits and later aggregating the results of all the tasks to provide the final result. The Map is analogous to the divide phase and the Reduce is analogous to the aggregation.

```
Map
(k1, v1) -> list(k2,v2)

Reduce
(k2, list(v2)) -> list(k3,v3)
```

The Map Reduce two methods are used:

A. MTJR: Two jobs in Map-Theta-Join-Reduce method (MTJR).
The first job is called the Filter-Theta-Join job and the second one is the Collection job. First steps that tables D1 and D2 are to be joined. In MTJR, the master will splits table tuples, stores them onto several workers after that invokes Mappers to deal with them. The master splits each table into blocks split0 and split1, and assigns them to four Mappers, M0 to M3. During the Map phase, a Mapper, for example M0, first retrieves the tuples in split0 of D1 and filters them by the condition issued in the WHERE clause of a query statement. The filtered tuples are attached a manipulated signature and later dispatched to their selected region.

Fig. 2: Hadoop Map Reduce Process [8]

Fig. 3: Map Reduce Concept[9]
files by our well-calculated partition function. Therefore, tuples of the same table will be allocated to the region files of the same signature. Once all Mappers are finished, Reducers will be invoked by the master and access their region files to execute joined computing. The master invokes four Reducers, each of which accesses their region files to perform join computing. For example, Reducer R0 retrieves the region files which are numbered 0.

The second job of MTJR is the Collection Job. The job is to collect and aggregate the result of Filter-Theta-Join job. The Map phase accesses the results of the prior job, and output them to the region files. The Reduce phase remotely accesses the region files to merges them to join all individual results. Last, it will aggregate the result of the reduce phase. At last aggregate results are output.

![Fig. 4: Filter-Theta-Join Job of MTJR](image)

**B. BFMTJR method : Bloom Filter Map-Theta-Join-Reduce**

MTJR may work weakly for star-join queries, especially for those without any filter condition on the fact table. This is because excessive data transmission of the fact table to each Reducer degrades the performance. Bloom Filter Map-Theta-Join-Reduce (BFMTJR) improves the shortcoming. It includes another filter process to reduce the data transmission time of the fact table. There are three jobs of BFMTJR. The first job is the Bloom-Filter job, which does the equal to as the map phase of MTJR. The second job is the Filter-Theta-Join job, which conducts filtering and performs Theta-Join. The third job is Collection-Aggregate job, which does the aggregate results.

![Fig. 5: BFMTJR Flows](image)

**C. PROPOSED METHOD :**

The main proposed method of MapReduce method is to try to increase the performance of query processing by using BFMTJR and MTJR. So, the basic idea is to build several filter using mapping-partition function on the RDF data and then summative their predictions. This plan follows the system easy and fast query processing that tends to get easy time optimization.

**V. CONCLUSION AND FUTURE WORK**

In this paper, method for Theta-Join, includes MTJR and BFMTJR. MTJR is useful for common theta-join queries, while BFMTJR is especially useful for filter out queries with theta-join operations. The idea of our proposed method is that the triples of RDF for joining will be allocated to a single Reducer. The idea is realized by manipulating data keys of MapReduce jobs and designing different partition functions. The investigational results show that our method outperforms MJR in the cases of star joins queries and I/O costs. In addition, our proposed methods support theta-join queries. We used two optimization techniques that allow efficient processing of join-intensive workloads such as RDF graph pattern matching. These two techniques help to small steps for the MR execution workflows, and reduce the intermediate tuples, resulting in less I/O, sorting, and communication overheads involved in traditional MapReduce based RDF graph processing.
ACKNOWLEDGMENT

The authors would like to be grateful the reviewers for their precious comments and suggestions that contributed to the expansion of this work.

REFERENCES


