A Survey of Mapreduce based Optimized Semantic Search in Hadoop Environment

Sejal Dave[1] Dr. Chirag S. Thaker[2]
[1]M.Tech. (Web Technology) [2]Assistant Professor (PhD (CSE) - ME (CE))
[1]Rollwala Computer Centre, Ahmedabad, India
[2]Shantilal Shah Engg. College, Bhavnagar, India

Abstract---Now-a-days the data has become bigger because of E-commerce and Social Networking sites. So, there is a need to handle such big data and it has to be analyzed in such a way that it can produce some useful information which can be used in some kind of decision making. MapReduce framework has emerged as one of the most widely used parallel computing platforms for processing data on terabyte and petabyte scales. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Today Map Reduce has been used daily at companies such as Yahoo!, Google, Amazon, and Facebook, and adopted more recently by several universities, it allows for easy parallelization of data intensive computations over many machines. There are many algorithms for retrieving the information using MapReduce like Sorting, Searching, TF-IDF, BFS, and Page-Rank etc Hadoop is a distributed file system where files can be saved with replication. It provides high fault tolerance and reliability. Moreover, it provides an implementation of MapReduce programming model. Hadoop can work directly with any distributed file system which can be mounted by the underlying OS. Hadoop-MapReduce can handle huge amount of data. Scalability is a matter of concern for a long time. Same is true for semantic web data. We can store huge amount of semantic web data in Hadoop clusters built mostly by cheap commodity class hardware and still can answer queries fast enough. The aim of this survey is improving the performance of parallel query processing using MapReduce.

I. Introduction

In this modern era, analysis of increasingly large amount of data is central to many enterprises' day-to-day operations and revenue generation. Today, even small enterprises are collecting terabytes of data. Analyzing that data in an effective and efficient manner can be key to their success. Various systems have been developed mainly by the industries to support Big Data analysis, including Google's MapReduce [1], Yahoo's PNUTS [4], Microsoft's SCOPE [5], LinkedIn's Kafka [8] and also, several companies, including Facebook [9] have contributed to Apache Hadoop (an open-source implementation of Map Reduce) and its eco-system.

Map Reduce has become the most popular framework for large-scale processing and analysis of vast data sets in clusters of machines, mainly because of its simplicity. With MapReduce, the developer gets various cumbersome tasks of distributed programming for free without the need to write any code; indicative examples include machine to machine communication, task scheduling to machines, scalability with cluster size, ensuring availability, handling failures, and partitioning of input data. Moreover, the open-source Apache Hadoop implementation of MapReduce has contributed to its widespread usage both in industry and academia [9].

Scope of this Survey: The aspect of this survey is huge coverage of existing work for analytical query processing using MapReduce and Hadoop. Also, to provide clear overview to new researchers who are unfamiliar with MapReduce and Hadoop. This survey also includes already alive knowledge for the veteran researchers in a meaningful way.

II. Map Reduce And Hadoop Basics

A. *MapReduce:*

MapReduce is a software framework for processing largedata sets in a distributed fashion over several machines.

There are two functions which are used in MapReduce.

1) Map Function
2) Reduce Function

Map function is handled by Mappers and Reduce function is handled by Reducers.

1) Work of Mappers and Reducers: The Mapper has a map method that transforms input (key, value) pairs into any number of intermediate (key, value) pairs.

The Reducer has a reduce method that transforms intermediate (key, value) aggregates into any number of output (key, value) pairs.

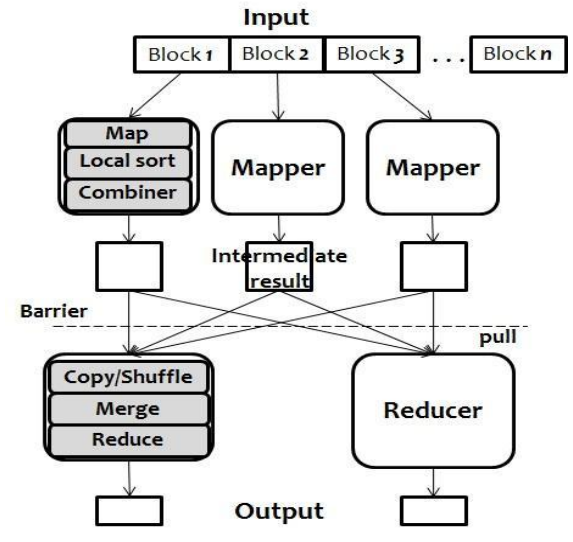

Fig. 1: MapReduce Architecture [10]

2) *Data flow beyond two key pieces (maps and reduces):* Input reader – divides input into appropriate size splits which get assigned to a Map function.

- Map function – maps file data to smaller, intermediate <key, value> pairs
- Partition function – finds the correct reducer: given the key and number of reducers, returns the desired Reduce node
- Compare function – input for Reduce is pulled from the Map intermediate output and sorted according to this compare function
- Reduce function – takes intermediate values and reduces to a smaller solution handed back to the framework
- Output writer – writes file output

B. Hadoop:

Hadoop is a software framework for distributed processing of large datasets across large clusters of computers. Hadoop is an open-source implementation for Google MapReduce. Hadoop is based on a simple data model; any data will fit on it.

Hadoop framework consists of two main layers:

- Distributed file system (HDFS)
- Execution engine (MapReduce) [6].

The Hadoop distributed file system (HDFS) is a distributed, scalable, and portable file-system written in java for the Hadoop framework.

There is a HDFS architecture for hadoop.

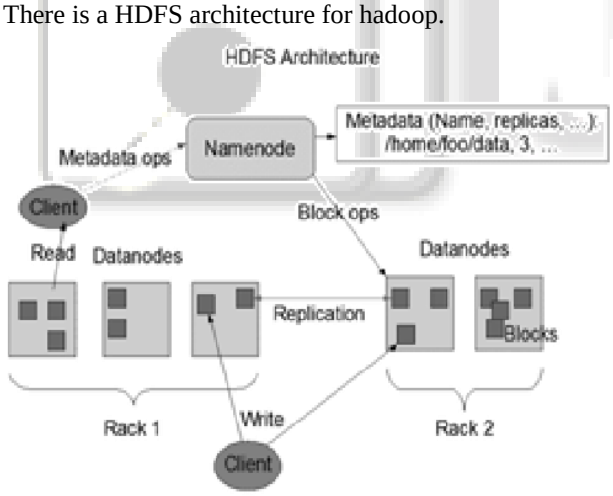

Fig. 3: HDFS-Data Placement and Replication [6].

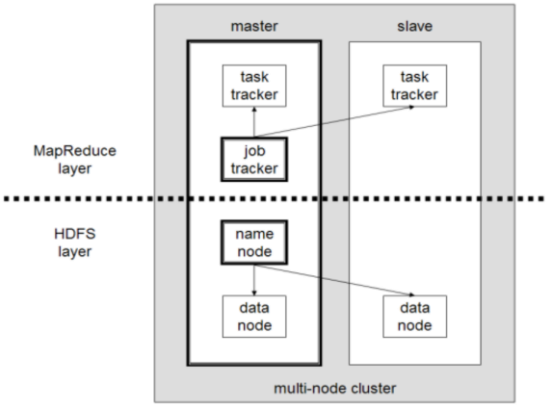

Fig. 4: Hadoop for Multi Node Cluster [7]

Hadoop is generally used for the single node and Multi Node cluster. In single node, Hadoop processes will run on single Machine and in Multi Node, Hadoop cluster includes a single master and multiple worker nodes. The master node consists of a Job Tracker, Task Tracker, Name Node and Data Node. A slave or *worker node* acts as both a Data Node and Task Tracker, though it is possible to have data-only worker nodes and compute-only worker nodes. Hadoop requires Java Runtime Environment (JRE).

C. Mapreduce: Hadoop Execution Layer:

- Job tracker knows everything about submitted jobs
- Divides jobs into tasks and decides where to run each task.
- Continuously communicating with task trackers
- Task trackers execute tasks (multiple per node)
- Monitors the execution of each task
- Continuously sending feedback to Job tracker

D. MapReduce is master-slave architecture:

- Master: Job Tracker
- Slaves: Task Trackers (100s or 1000s of task trackers)
- Every data node is running a task tracker

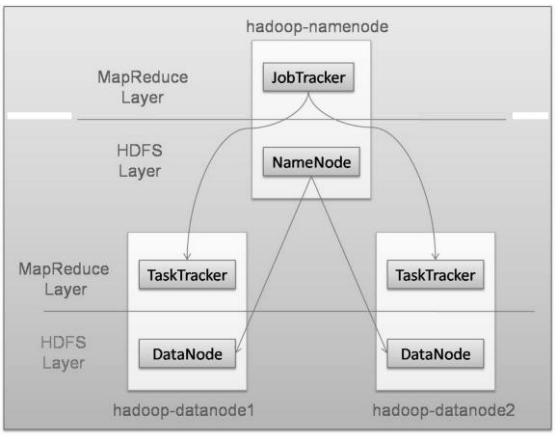

Fig. 5: Execution Layer of Hadoop MapReduce [6]

III. Different Algorithms For Mapreduce

There are many algorithms which are used in MapReduce for reducing some complex things in easy manner.

A. Sorting Algorithm:

Sort: Inputs

- A set of files, one value per line.
- Mapper key is file name, line number
- Mapper value is the contents of the line

How to Algorithm Works?

Takes advantage of reducer properties:

(Key, value) pairs are processed in order by key; reducers are themselves ordered:

- Mapper: Identity function for value (k, v) _ (v, _)
- Reducer: Identity function (k', _) -> (k', "")

Sort: The Trick

(Key, value) pairs from mappers are sent to a particular reducer based on hash (key).

• Must pick the hash function for your data such that k1 < K2 => hash (k1) < hash (K2).

Final Thoughts on Sort: Used as a test of Hadoop's raw speed.

B. Searching Algorithm:

Search: Inputs

- Mapper key is file name, line number.
- Mapper value is the contents of the line.
- Search patternsent as special parameter.

How to Algorithm Works?

• Mapper:

– Given (filename, some text) and "pattern", if "text" matches "pattern" output (filename, _)

• Reducer:

– Identity function.

Search: An Optimization

- Once a file is found to be interesting, we only need to mark it that way once.
- Use Combiner function to fold redundant (Filename, _) pairs into a single one.
- Reduces network I/O

C. Indexing Algorithm

A set of files containing lines of text

- Mapper key is file name, line number.
- Mapper value is the contents of the line.

Inverted Index Algorithm:

- Mapper: For each word in (file, words) Map to (word, file).
- Reducer: Identity function.

Index: MapReduce

```
Map (pageName, pageText):
    For each word w in pageText:
    emitIntermediate (w, pageName);
    Done
Reduce (word, values):
    For each page name in values:
    AddToOutputList (pageName);
    Done
    EmitFinal (FormattedPageListForWord);
```

IV. LIMITATIONS AND BENEFITS

A. Limitations:

- Cannot control the order in which the maps or reductions are run.
- For maximum parallelism, you need Maps and Reduces to not depend on data generated in the same Map Reduce job (i.e. stateless).
- A database with an index will always be faster than a Map Reduce job on non-indexed data.
- Reduce operations do not take place until all Maps are complete (or have failed then been skipped).
- General assumption that the output of Reduce is smaller than the input to Map; large data source used to generate smaller final value.

B. Benefits

1) Scalable:

Hadoop is a highly scalable storage platform, because it can store and distribute very large data sets across hundreds of inexpensive servers that operate in parallel. Unlike traditional relational database systems (RDBMS) that can't scale to process large amounts of data, Hadoop enables businesses to run applications on thousands of nodes involving thousands of terabytes of data.

2) Cost effective:

Hadoop also offers a cost effective storage solution for businesses' exploding data sets. The problem with traditional relational database management systems is that it is extremely cost prohibitive to scale to such a degree in order to process such massive volumes of data. In an effort to reduce costs, many companies in the past would have had to down sample data and classify it based on certain assumptions as to which data was the most valuable. The raw data would be deleted, as it would be too cost-prohibitive to keep.

3) Flexible:

Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data. This means businesses can use Hadoop to derive valuable business insights from data sources such as social media, email conversations or click stream data. In addition, Hadoop can be used for a wide variety of purposes, such as log processing, recommendation systems, data warehousing, and marketing campaign analysis and fraud detection.

4) Fast:

Hadoop's unique storage method is based on a distributed file system that basically 'maps' data wherever it is located on a cluster. The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing. If you're dealing with large volumes of unstructured data, Hadoop is able to efficiently process terabytes of data in just minutes, and petabytes in hours.

5) Resilient to failure:

A key advantage of using Hadoop is its fault tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster. The MapReduce distribution goes beyond that by eliminating the NameNode and replacing it with a distributed No NameNode architecture that provides true high availability. When it comes to handling large data sets in a safe and cost-effective manner, Hadoop has the advantage over relational database management systems, and its value for any size business will continue to increase as unstructured data continues to grow.

V. OPPORTUNITIES FOR FUTURE WORK

As a large parallel processing framework, Map Reduce has brought new enthusiasm. MapReduce is very well for its scalability, flexibility and fault tolerance. Most of this research is ongoing, and there will likely be significant improvements to the MapReduce framework as well as extensions to MapReduce paradigm for distributed and parallel computing. In this survey, we also discussed about the algorithms that will be helpful to implement the MapReduce algorithm as well as extend Map Reduce for big data concentrated application. This Survey provides the way of analysis to the skills and basis for further experiments and Research.

REFERENCES

[1] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.
[2] Estival, Dominique, et al. "Author profiling for English emails." Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics. 2007.
[3] Allahabadia, Amit, et al. "Age and Gender Predict the Outcome of Treatment for Graves' Hyperthyroidism 1." Journal of Clinical Endocrinology & Metabolism85.3 (2000): 1038-1042.
[4] Rabl, Tilmann, et al. "Solving big data challenges for enterprise application performance management." Proceedings of the VLDB Endowment 5.12 (2012): 1724-1735.
[5] Zhou, Jingren, et al. "SCOPE: parallel databases meet MapReduce." The VLDB Journal—the International Journal on Very Large Data Bases 21.5 (2012): 611-636.
[6] Ekanayake, Jaliya, Shrideep Pallickara, and Geoffrey Fox. "Mapreduce for data intensive scientific analyses." eScience, 2008. EScience'08 IEEE Fourth International Conference on. IEEE, 2008
[7] http://en.Wikipedia.org/wiki/Apache_Hadoop
[8] Tiwari, Mitul. "Large-scale social recommender systems: challenges and opportunities." Proceedings of the 22nd international conference on World Wide Web companion INternational World Wide Web Conferences Steering Committee, 2013.
[9] Doulkeridis, Christos, and Kjetil Nørvåg. "A survey of large-scale analytical query processing in MapReduce." The VLDB Journal (2013): 1-26.
[10] Lee, Kyong-Ha, et al. "Parallel data processing with MapReduce: a survey."AcM sIGMoD Record 40.4 (2012): 11-20.

.

