International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

A Survey Paper on Map Reduce in Big Data

P. Sudha[1], Dr. R. Gunavathi[2]

Assistant Professor, PG Department of Computer Applications, Sree Saraswathi Thyagaraja College, Pollachi – 642107, TN, India

Head, PG Department of Computer Applications, Sree Saraswathi Thyagaraja College, Pollachi – 642107, TN, India

Abstract: ***Big data is concern massive amount, complex, growing data set from multiple autonomous sources. It has to deal with large and complex dataset that can be structured, semi-structured or unstructured and will typically not fit into memory to be processed. MapReduce is a programming model for processing large datasets distributed on a large clusters.A rapid growth of data in recent time, Industries and academia required an intelligent data analysis tool that would be helpful to satisfy the need to analysis a large amount of data. MapReduce framework is basically designed to compute data demanding applications to support effective decision making. Since its introduction, remarkable research efforts have been put to make it more familiar to the users subsequently utilized to support the execution of enormous data intensive applications. This survey paper highlights and investigates various applications using recent MapReduce models.***

Keywords: MapReduce; Cloud Computing; Hadoop File System (HDFS); Key Value pairs; Hadoop; Big Data

1. Introduction

In the present days, a huge data handling is a prime concern topic for researchers. Many applications like data mining, Image processing, data analytic etc are required processing of massive amount of data. MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. [25] The real power of MapReduce is the capability to divide and conquer. Take a large problem and break into smaller, more manageable chunks, operate the chunk independedly, and then pull it all together at the end.

MapReduce is a processing paradigm of executing data with partitioning and aggregation of intermediate results. It works to process data in parallel in which splitting of data, distribution, synchronization and fault tolerance are handled automatically by the framework. MapReduce framework is famous for large scale data processing and analysis of voluminous datasets in clusters of machines.

[1]Popularity for the term „Cloud-Computing" has been increasing in recent years. There are many great companies such as Yahoo, Google etc. tried to provide related services to business community, even through public users. In addition to the SQL technique, MapReduce, a programming model that realizes implementing large-scale data processing, has been a hot topic that is widely discussed through many studies. Many real-world tasks such as data processing for search engines can be parallel implemented through a simple interface with two functions called Map and Reduce.

A MapReduce framework can be categorized into mainly two steps such as [10]:

A. Map Phase

Initially split the data into key value pair and fed into mapper which in turn process each key value pair and generate intermediate output.

B. Reduce Phase

- The Intermediate key value pair first collected sorted and grouped by key and generates values associated with each key.
- The receiver produces final output based on some calculation and stores it in an output file.

The Map and Reduce task run sequentially in a cluster; the output of the Map Phase becomes the input for the Reduce Phase.

2. Opening of Big Data

Recent developments in the Web, social media, sensors and mobile devices have resulted in the explosion of data set sizes. For example, Face book today has more than one billion users, with over 618 million active users generating more than 500 terabytes of new data each day [5].

Traditional data processing and storage approaches were designed in an era when available hardware, storage and processing requirements were very different than they are today. Thus, those approaches are facing many challenges in addressing Big Data demands.

The term "Big Data" refers to large and complex data Sets made up of a variety of structured and unstructured data which are too big, too fast, or too hard to be managed by traditional techniques. Big Data is characterized by the 4Vs [6]: volume, velocity, variety, and veracity. Volume refers to the quantity of data, variety refers to the diversity of data types, velocity refers both to how fast data are generated and how fast they must be processed, and veracity is the ability to trust the data to be accurate and reliable when making crucial decisions.

Enterprises are aware that Big Data has the potential to impact core business processes, provide competitive advantage, and increase revenues [6]. Thus, organizations are exploring ways to make better use of Big Data by analyzing them to find meaningful insights which would lead to better business decisions and add value to their business. Getting information about popular organizations that hold big data are face book, yahoo, twitter and EBay.

Licensed Under Creative Commons Attribution CC BY

Map/Reduce is a highly scalable programming paradigm capable of processing massive volumes of data by means of parallel execution on a large number of commodity computing nodes. It was recently popularized by Google [7], but today the Map/Reduce paradigm has been implemented in many open source projects, the most prominent being the Apache Hadoop [8]. The popularity of Map/Reduce can be qualified to its high scalability, fault-tolerance, simplicity and independence from the programming language or the data storage system.

3. Mapreduce Model

[2] Map-Reduce was created by one of the biggest search engine companies - Google for application development on data-centers with thousands of computing nodes. Nowadays, it has been used across a wide range of domains within Google, including [3]

- Large-scale machine learning problems
- clustering problems for the Google News and Froogle products
- extracting data to produce reports of popular queries(e.g. Google Zeitgeist and Google Trends)
- extracting properties of web pages for new experiments and products (e.g. extraction of geographical locations from a large corpus of Web Pages for localized search)
- processing of satellite imagery data
- Language model processing for statistical machine Translation
- Large-scale graph computations

The programming model can be summarized as follows [3].

The computation takes a set of input {*key, value*} pairs and produces a set of output {*key, value*} pairs. The use of the Map-Reduce library expresses the computation as two functions: Map and Reduce. First, Map characterizes the set of input pairs and produces a set of intermediate {*key, value*} pairs. Afterwards, the Map-Reduce library groups together all intermediate pairs associated with the same intermediate key, and passes them to the Reduce function. Then Reduce merges these tuples with a same key to form a possibly smaller data

Set. Typically, the result can be just zero or one output value 245 which is produced per Reduce invocation. The overall processing flow of Map-Reduce is schematized in Figure 1.

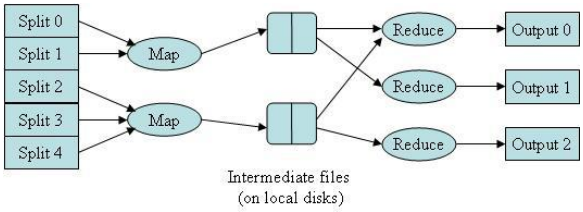

Figure 1: Processing flow overview of Map-Reduce

[26]Five common steps of parallel computing are as follows:

1) Preparing the Map() input: It will take the input row wise and emit key value pairs per rows.
   Map input: list(k1,v1)
2) Run the user-provided Map() code
   Map output: list(k2,v2)
3) Shuffle the map output to the reduce processors and shuffle the similar keys and input them to the same reducer.
4) Run the user-provided Reducer() code: In this phase will run the custom reducer code designed by the developer to run on the shuffled data and emit key and value.
   Reduce input: (k2, list (v2))
   Reduce output: (k3,v3)
5) Produce the final output: Finally, the node collects all reducer output and combines and writes them in a text file.

4. Example of Mapreduce

[18] Consider the problem of counting the number of occurrences of each word in a large collection of documents. The user would write code similar to the following pseudo-code:

```
Map (String key, String value):
// key: document name
// value: document contents
for each word w in value:
Emit Intermediate (w, "1");
Reduce (String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
result += ParseInt(v);

Emit (As String(result));
```

The map function emits each word plus an associated count of occurrences (just „1“ in this simple example).
The reduce function sums together all counts emitted for a particular word.
In addition, the user writes code to fill in a *mapreduce specification* object with the names of the input and output files, and optional tuning parameters. The user then invokes the *MapReduce* function, passing it the specification object. The user“s code is linked together with the MapReduce library (implemented in C++).

[18] Even though the previous pseudo-code is written in terms of string inputs and outputs, conceptually the map and reduce functions supplied by the user have associated types:
map (k1,v1) -> list(k2,v2)
reduce (k2,list(v2)) -> list(v2)
ie., the input keys and values are drawn from a different domain than the output keys and values. Furthermore, the intermediate keys and values are from the same domain as the output keys and values.
Our C++ implementation passes strings to and from the user-defined functions and leaves it to the user code to convert between strings and appropriate types.

5. Online Processing

The Velocity dimension, as one of the Vs used to define Big Data, brings many new challenges to traditional data processing approaches and especially to Map/Reduce. Handling Big Data velocity often requires applications with *online processing* capabilities, which can be broadly defined

Licensed Under Creative Commons Attribution CC BY

as real-time processing of fast and continuously generated data (also known as data *streams*).

From the business perspective, the goal is normally to obtain insights from these data streams, and to enable prompt reaction to them. This instantaneous reaction can bring business value and competitive advantage to organizations, and therefore has been generating research and commercial interest. Areas such as financial fraud detection and algorithmic trading have been highly interested in this type of solutions.

The Map/Reduce paradigm is not an appropriate solution for this kind of low-latency processing because:

- MapReduce computations are batch processes that start and finish, while computations over streams are continuous tasks that only finish upon user request.
- The inputs of MapReduce computations are snapshots of data stored on files, and the content of these files do not change during processing. Conversely, data streams are continuously generated and unbounded inputs [15].

In order to provide fault tolerance, most of MapReduce implementations, such as Google‟s [7] and Hadoop [8], write the results of the *Map* phase to local files before sending them to the reducers. In addition, these implementations store the output files in distributed and high-overhead file systems (Google File System [16] or HDFS [8], respectively). This extensive file manipulation adds significant latency to the processing pipelines.

- Not every computation can be efficiently expressed using the MapReduce programming paradigm, and the model does not natively support the composition of jobs.

Despite these limitations, the prevalence and success of MapReduce has motivated many researchers to work on systems that leverage some of its advantages, and at the same time try to overcome its limitations when applied to low latency processing.

One of the first projects in this direction was developed by Condie *et al.* [17]. In this work, the authors proposed an online MapReduce implementation with the goal of supporting online aggregation and continuous queries. In order to reduce the processing latency, the *Map* and *Reduce* phases are pipelined by having the *Map* tasks sending intermediate results to the *Reduce* tasks. The authors also introduced the idea of executing reducers on *snapshots* of the data received from the mappers.

6. Related Work

The most related work associated with the introduction of various MapReduce models and its relation with the database processing [19]. This tutorial provides the insight about how to improve the performance by increasing availability of the system in case of failure, reduced network communication overhead, process scheduling etc. [20] Authors performed a detailed study about its open source implementation-Hadoop and few factors such as 1) I/O mode, the way of a reader retrieving data from the storage system, 2) data parsing, the scheme of a reader parsing the format of records, and 3) indexing, which is used to speeding up data processing.

[27]Despite being featured such as scalability in clusters, ensuring availability, handling failures Google’s MapReduce has been unusable for certain kind of applications requires iterative computation, execution of high-level language such as SQL and work on an Internet desktop grid. Since the MapReduce introduced, numerous MapRduce frameworks have been developed by several companies including Google‟s Map/Reduce [9], Apache’s Hadoop MapReduce [8], AMPLab‟s spark [11], SASReduce [12], Disco [13] etc. A lot of research has been done to address the issues highlighted above and some recent Map/Reduce implementation helps to overcome the limitations of the prior framework. While we consider databases, an author described salient features of Map/Reduce implementation and its performance comparison with the parallel database. According to report, Map/Reduce works well in different storage systems and provide a good framework to fault tolerance for large jobs [14].

Google encountered the “Big Data” problem very early, because of the large amount of information they had to process and index. They solved this problem by using commodity hardware, and the architecture to tackle this problem was a shared-nothing architecture, popular because of its scalability. [21] [22]

As Google has demonstrated, a shared-nothing system scales faster, because adding new nodes will not create bottlenecks or slow down the system. This system also increases the availability of the system, since there is no single point of failure. This model is called *sharding*. A *shared-nothing* system typically partitions its data among many nodes on different databases (assigning different computers to deal with specific users or queries), or may require every node to maintain its own copy of the application's data, using some kind of coordination protocol. This is often referred to as database sharding.

In 2004, Google released the first paper about their technology, the map-reduce algorithm they used for indexing and processing data, and two years later, the paper about BigTable, their distributed storage system. [23] [24]

7. Tools for Big Data Analytics

R is open source software package to perform statistical analysis on data. R is a programming language and mainly used by data scientist statisticians. R is registered under GNU (General Public License). R has various built-in as well as extended functions such as Data extraction, Data Cleaning, Data Loading, Data Transformation, Statistical Analysis, Predictive Modeling, and Data Visualization.

Apache hadoop is an open source java framework for processing and querying vast amounts of data on large clusters of commodity hardware. Hadoop changes the economics and the dynamics of large-scale computing. It enables scalable, cost-effective, flexible and fault-tolerant

Licensed Under Creative Commons Attribution CC BY

solutions. Apache Hadoop has two main features-HDFS (Hadoop Distributed File System) and MapReduce.

[26] Using R with Hadoop will provide an flexible data analytics platform that will scale depending on the size of the dataset to be analyzed. Experienced programmers can be write MapReduce modules in R and run it using Hadoop"s parallel processing Map/Reduce mechanism to identify patterns in the dataset. Hadoop is a very popular framework that provides such parallel processing capabilities. So , we can use R algorithms or analysis processing over Hadoop clusters to get the workdone in Figure 2.

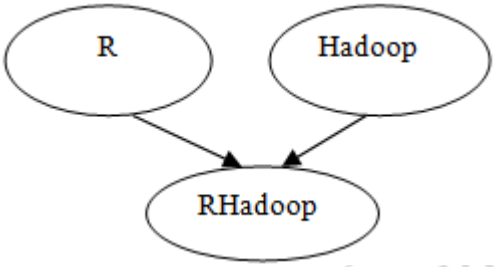

Figure 2: RHadoop

R is one of the most popular open source statistical analysis packages available on the market today. It is cross platform, has a very wide community support. R can connect with other data stores, such as MySQL, SQLite, MongoDB and Hadoop for data storage activities. Hadoop consists of lot of components like Mahout, Pig, Hive, HBase etc

8. Conclusion

The primary focus of this survey paper is to highlight some Map/Reduce implementation worked well to accomplish a specific purpose and compared with previously available frameworks. A remarkable performance improvement over the existing system seems after comparison.

The aim of this paper was giving an overview of the Map Reduce concepts in big data analytics. Map Reduce was developed by Google to handle big data analysis which is unstructured data such as web related data. But, it also can be used for structured data. We have discussed a number of Map Reduce models still researchers can develop a more efficient Map Reduce with improved functionalities.

References

[1] Jenq-shiou Leu et al" Comparison of Map-Reduce and SQL on Large-scale Data Processing" in International Symposium on Parallel and Distributed Processing with Applications on 2010 IEEE Computer Society, pp(244-248)
[2] Y.Hung-Chih, D. Ali, H. Ruey-Lung, and D. S. Parker, " Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters, " in proceedings of the 2007 ACM Sigmod International conference on management of data Beijing, China: ACM,2007
[3] D. Jeffrey and G. Sanjay, "MapReduce: Simplified Data Processing on Large Clusters," *COMMUN. ACM,* VOL. 51, PP. 107-113, 2008
[4] https://en.wikipedia.org/wiki/MapReduce
[5] P. Zadrozny and R. Kodali, Big Data Analytics using Splunk, Berkeley, CA, USA: Apress, 2013
[6] F. Ohlhorst, Big Data Analytics: Turning Big Data into Big Money, Hoboken, N.J, USA: Wiley, 2013
[7] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Commun ACM, 51(1), pp. 107-113, 2008
[8] Apache Hadoop, http://hadoop.apache.org
[9] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters" In ACM OSDI, 2004
[10] A. Elsayed, O. Ismail, and M. E. El-Sharkawi, MapReduce: State-of-the-Art and Research Directions, International Journal of Computer and Electrical Engineering, Vol. 6, No. 1, February 2014.
[11] M. Zaharia, M. Chowdhury, Michael J. Franklin, Scott Shenker and Ion Stoica, Spark: Cluster Computing with Working Sets University of California, Berkeley, May, 2010
[12] D. Moors, Whitehound Limited, UK, "SASReduce An implementation of MapReduce in BASE/SAS", Paper 1507-2014
[13] E. Bugnion, S. Devine, K. Govil, and M. Rosenblum, Disco: Running Commodity Operating Systems on Scalable Multiprocessors (1997)
[14] J. Dean and S. Ghemawat, MapReduce: A flexible data processing tool, Communications of the ACM, Vol. 53 No. 1, Pages 72-77
[15] W. Lam, L. Liu, S. Prasad, A. Rajaraman, Z. Vacheri and A. Doan, "Muppet: MapReduce-style processing of fast data," Proc.VLDB Endow., 5(12), pp. 1814-1825, 2012
[16] S. Ghemawat, H. Gobioff and S. Leung, "The Google file system," ACM SIGOPS Operating Systems Review, 2003
[17] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy and R. Sears, "MapReduce online," Proc. of the 7th USENIX Conference on Networked Systems Design and Implementation, 2010
[18] Jeffrey Dean et al, "MapReduce: Simlified Data Processing on Large Clusters" in USENIX Association OSDI '04: 6th Symposium on Operating Systems Design and Implementation, pp(137-149)
[19] J. Zhao, J. Pjesivac-Grbovic, MapReduce: The Programming Model And Practice, 2009
[20] D. Jiang, B. C. Ooi, L. Shi and S. Wu, The performance of mapreduce: An in-depth study. Proc. VLDB Endow., 3 pp. 472–483 (Sept 2010)
[21] M. Stonebraker, "The Case for Shared Nothing", University of California Berkeley, March 1986
[22] M. Hogan, "Shared-Disk vs. Shared-Nothing" - http://www.scaledb.com/pdfs/WP_SDvSN.pdf
[23] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Google, OSDI 2004
[24] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, R. Gruber, "Bigtable: A Distributed Storage System for Structured Data", Google, OSDI 2006
[25] Judith Hurwitz, Alan Nugent, Dr. Fern Halper, and Marcia Kaufman, "Big Data for Dummies", John Wiley & Sons, Inc., 2013
[26] Prajapati, Vignesh, "Big Data Analytics with R and Hadoop", Packt publishing Nov 2013

Licensed Under Creative Commons Attribution CC BY

[27] Shafali Agarwal, Zeba Khanam," Map Reduce: A survey paper on Recent Expansion " in International Journal of Advanced Computer Science and Applications, Vol.6, No.8, 2015

Author Profile

P. Sudha, Asssistant Professor in MCA Department, Sree Saraswathi Thyagaraja College, Pollachi, Tamil Nadu, India.

Dr. R. Gunavathi, HoD in MCA Department, Sree Saraswathi Thyagaraja College, Pollachi, Tamil Nadu, India.

www.ijsr.net
IJSR
ISSN (Online): 2319-7064

Licensed Under Creative Commons Attribution CC BY