ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

Hadoop on Big Data Analysis

[1]R.Shobana&[2]D.Saranya

[1&2] Assistant Professor, Department Of Computer Science and Applications,

D.K.M College for Women (Autonomous), Vellore, India.

shobanavasu.mca@gmail.com&saran154@gmail.com

Abstract

We live in on-demand, on-command Digital universe with data prolifering by Institutions, Individuals and Machines at a very high rate. This data is categories as "Big Data" due to its sheer Volume, Variety, Velocity and Veracity. Most of this data is unstructured, quasi structured or semi structured and it is heterogeneous in nature.

The volume and the heterogeneity of data with the speed it is generated, makes it difficult for the present computing infrastructure to manage Big Data. Traditional data management, warehousing and analysis systems fall short of tools to analyze this data.

Due to its specific nature of Big Data, it is stored in distributed file system architectures. Hadoop and HDFS by Apache is widely used for storing and managing Big Data. Analyzing Big Data is a challenging task as it involves large distributed file systems which should be fault tolerant, flexible and scalable. Map Reduce is widely been used for the efficient analysis of Big Data. Traditional DBMS techniques like Joins and Indexing and other techniques like graph search is used for classification and clustering of Big Data. These techniques are being adopted to be used in Map Reduce.

There are various methods for catering to the problems in hand through Map Reduce framework over Hadoop Distributed File System (HDFS). Map Reduce is a Minimization technique which makes use of file indexing with mapping, sorting, shuffling and finally reducing. Map Reduce techniques have been studied at in this paper which is implemented for Big Data analysis using HDFS.
***Keyword*-Big Data Analysis, Big Data Management, Map Reduce, HDFS**

***Keyword*-Big Data Analysis, Big Data Management, Map Reduce, HDFS**

I. INTRODUCTION

Big Data is a heterogeneous mix of data both structured (traditional datasets –in rows and columns like DBMS tables, CSV's and XLS's) and unstructured data like e-mail attachments, manuals, images, PDF documents, medical records such as x-rays, ECG and MRI images, forms, rich media like graphics, video and audio, contacts, forms and documents. Businesses are primarily concerned with managing unstructured data, because over 80 percent of enterprise data is unstructured and require significant storage space and effort to manage.“Big data” refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyse.
Big data analytics is the area where advanced analytic techniques operate on big data sets.

Map Reduce by itself is capable for analysing large distributed data sets; but due to the heterogeneity, velocity and volume of Big Data, it is a challenge for traditional data analysis and management tools. A problem with Big Data is that they use NoSQL and has no Data Description Language (DDL) and it supports transaction processing. Also, web-scale data is not universal and it is heterogeneous.
Map Reduce has following characteristics ; it supports Parallel and distributed processing, it is simple and its architecture is shared-nothing which has commodity diverse hardware (big cluster).Its functions are programmed in a high-level programming language (e.g. Java, Python) and it is flexible. Query processing is done through NoSQL integrated in HDFS as Hive tool.

Traditional experience in data warehousing, reporting, and online analytic processing (OLAP) is different for advanced forms of analytics. Organizations are implementing specific forms of analytics, particularly called advanced analytics. These are an collection of related techniques and tool types, usually including predictive analytics, data mining, statistical analysis, complex SQL, data visualization, artificial intelligence, natural language processing. Database analytics platforms such as Map Reduce, in-database analytics, in-memory databases, and columnar data stores are used for standardizing them.

Discovery analytics against big data can be enabled by different types of analytic tools, including those based on SQL queries, data mining, statistical analysis, fact clustering, data visualization, natural language processing, text analytics, artificial intelligence etc.

Map Reduce over HDFS gives Data Scientists the techniques through which analysis of Big Data can be done. HDFS is a distributed file system architecture which encompasses the original Google File System.Map Reduce jobs use efficient data processing techniques which can be applied in each of the phases of Map Reduce; namely Mapping, Combining, Shuffling, Indexing, Grouping and Reducing. All these techniques have been studied in this paper for implementation in Map Reduce tasks.

II. BIG DATA: OPPORTUNITIES AND CHALLENGES

In the distributed systems world, “Big Data” started to become a major issue in the late 1990‟s due to the impact of the world-

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

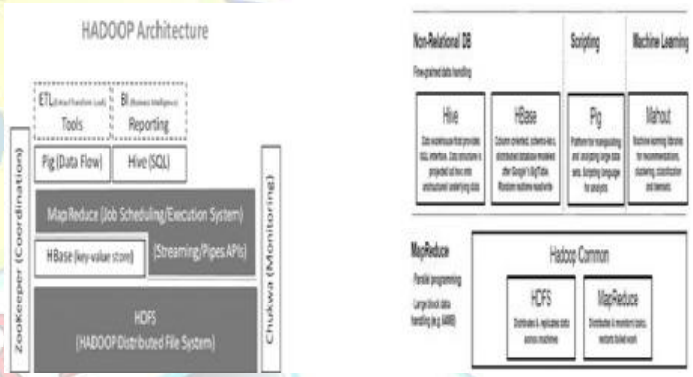
wide Web and a resulting need to index and query its rapidly mushrooming content.

Google's technical response to the challenges of Web-scale data management and analysis was simple, by database standards, but kicked off what has become the modern "Big Data" revolution in the systems world. To handle the challenge of Web-scale storage, the Google File System (GFS) was created. GFS provides clients with the familiar OS-level byte-stream abstraction, but it does so for extremely large files whose content can span hundreds of machines in shared-nothing clusters created using inexpensive commodity hardware. To handle the challenge of processing the data in such large files, Google pioneered its Map Reduce programming model and platform. This model, characterized by some as "parallel programming for dummies", enabled Google's developers to process large collections of data by writing two user-defined functions, map and reduce, that the Map Reduce framework applies to the instances (map) and sorted groups of instances that share a common key (reduce) – similar to the sort of partitioned parallelism utilized in shared-nothing parallel query processing.
Driven by very similar requirements, software developers at Yahoo!, Facebook, and other large Web companies followed suit. Taking Google's GFS and Map Reduce papers as rough technical specifications, open-source equivalents were developed, and the Apache Hadoop Map Reduce platform and its underlying file system (HDFS, the Hadoop Distributed File System) were born.
The Hadoop system has quickly gained traction, and it is now widely used for use cases including Web indexing, click stream and log analysis, and certain large-scale information extraction and machine learning tasks. Soon tired of the low-level nature of the Map Reduce programming model, the Hadoop community developed a set of higher-level declarative languages for writing queries and data analysis pipelines that are compiled into Map Reduce jobs and then executed on the Hadoop Map Reduce platform. Popular languages include Pig from Yahoo!, Jaql from IBM, and Hive from Face book. Pig is relational-algebra-like in nature, and is reportedly used for over 60% of Yahoo! "s Map Reduce use cases; Hive is SQL-inspired and reported to be used for over 90% of the Face book Map Reduce use cases. Microsoft's technologies include a parallel runtime system called Dryad and two higher-level programming models, Dryad LINQ and the SQLlike SCOPE language, which utilizes Dryad under the covers. Interestingly, Microsoft has also recently announced that its future "Big Data" strategy includes support for Hadoop.

III. HADOOP AND HDFS

Hadoop is a scalable, open source, fault-tolerant Virtual Grid operating system architecture for data storage and processing. It runs on commodity hardware, it uses HDFS which is fault-tolerant high-bandwidth clustered storage architecture. It runs Map Reduce for distributed data processing and is works with structured and unstructured data.

Figure1Illustrates the layers found in the software architecture of aHadoop stack. At the bottom of the Hadoop software stack is HDFS, a distributed file system in which each file appears as a (very large) contiguous and randomly addressable sequence of bytes. For batch analytics, the middle layer of the stack is the Hadoop Map Reduce system, which applies map operations to the data in partitions of an HDFS file, sorts and redistributes the results based on key values in the output data, and then performs reduce operations on the groups of output data items with matching keys from the map phase of the job. For applications just needing basic key-based record management operations, the HBase store (layered on top of HDFS) is available as a key-value layer in the Hadoop stack. As indicated in the figure, the contents of HBase can either be directly accessed and manipulated by a client application or accessed via Hadoop for analytical needs. Many users of the Hadoop stack prefer the use of a declarative language over the bare MapReduce programming model. High-level language compilers (Pig and Hive) are thus the topmost layer in the Hadoop software stack for such clients.



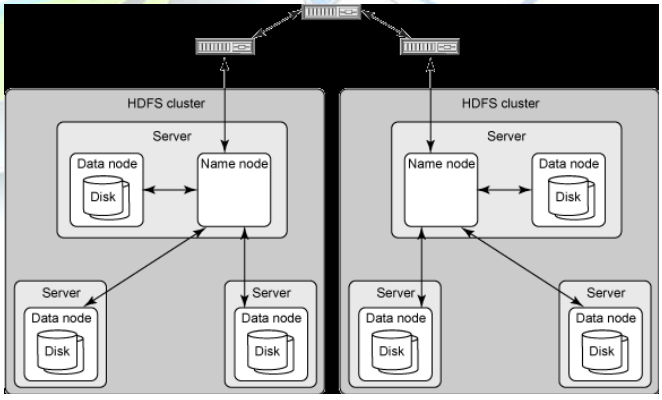


Figure 3. HDFS Clusters

Figure2 shows the relevancy between the traditional experience in data warehousing, reporting, and online analytic processing (OLAP) and advanced analytics with collection of related techniques like data mining with DBMS, artificial intelligence,

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

machine learning, and database analytics platforms such as Map Reduce and Hadoop over HDFS .

Figure 3 shows the architecture of HDFS clusters implementation with Hadoop. It can be seen that HDFS has distributed the task over two parallel clusters with one server and two slave nodes each. Data analysis tasks are distributed in these clusters.

IV. Big Data Analysis

Heterogeneity, scale, timeliness, complexity, and privacy problems with Big Data hamper the progress at all phases of the process that can create value from data. Much data today is not natively in structured format; for example, tweets and blogs are weakly structured pieces of text, while images and video are structured for storage and display, but not for semantic content and search: transforming such content into a structured format for later analysis is a major challenge. The value of data enhances when it can be linked with other data, thus data integration is a major creator of value. Since most data is directly generated in digital format today, we have the opportunity and the challenge both to influence the creation to facilitate later linkage and to automatically link previously created data. Data analysis, organization, retrieval, and modelling are other foundational challenges. Big Data analysis is a clear bottleneck in many applications, both due to lack of scalability of the underlying algorithms and due to the complexity of the data that needs to be analysed. Finally, presentation of the results and its interpretation by non-technical domain experts is crucial to extracting actionable knowledge as most of the BI related jobs are handled by statisticians and not software experts.

Figure 4, below gives a glimpse of the Big Data analysis tools which are used for efficient and precise data analysis and management jobs. The Big Data Analysis and management setup can be understood through the layered structured defined in the figure. The data storage part is dominated by the HDFS distributed file system architecture; other mentioned architectures available are Amazon Web Service (AWS) , Hbase and Cloud Store etc. The data processing tasks for all the tools is Map Reduce; we can comfortably say that it is the de-facto Data processing tool used in the Big Data paradigm.

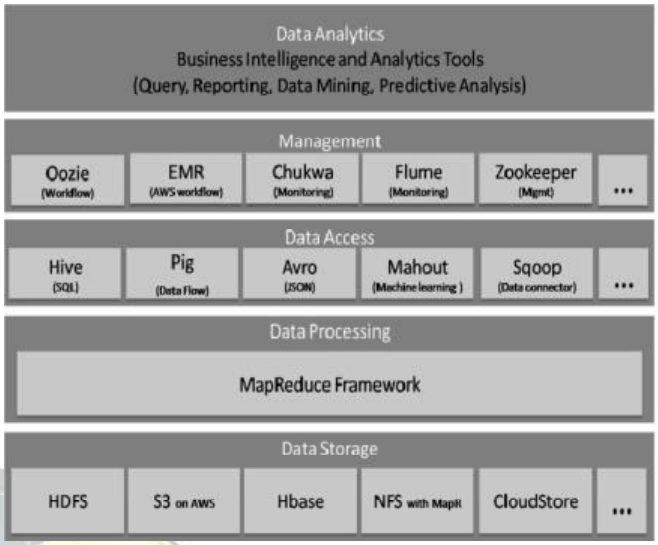


Figure 4. Big Data Analysis Tools

For handling the velocity and heterogeneity of data, tools like Hive, Pig and Mahout are used which are parts of Hadoop and HDFS framework. It is interesting to note that for all the tools used, Hadoop over HDFS is the underlying architecture. Oozie and EMR with Flume and Zookeeper are used for handling the volume and veracity of data, which are standard Big Data management tools. The layer with their specified tools forms the bedrock for Big Data management and analysis framework.

V. Map Reduce

Map Reduce is a programming model for processing large-scale datasets in computer clusters. The Map Reduce programming model consists of two functions, map () and reduce (). Users can implement their own processing logic by specifying a customized map () and reduce () function. The map () function takes an input key/value pair and produces a list of intermediate key/value pairs.

The Map Reduce runtime system groups together all intermediate pairs based on the intermediate keys and passes them to reduce () function for producing the final results.

Map (in_key, in value) --->list (out_key, intermediate_value) **Reduce** (out_key, list (intermediate value)) -- ->list (out value)

The signatures of map () and reduce() are as follows : **map (k1,v1) ! list (k2,v2)and reduce (k2,list(v2)) ! list (v2)**

A Map Reduce cluster employs a master-slave architecture where one master node manages a number of slave nodes. In the Hadoop, the master node is called Job Tracker and the slave node is called Task Tracker as shown in the figure 7. Hadoop launches a Map Reduce job by first splitting the input dataset into even-sized data blocks. Each data block is then scheduled to

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

one Task Tracker node and is processed by a map task. The Task Tracker node notifies the Job Tracker when it is idle. The scheduler then assigns new tasks to it. The scheduler takes data locality into account when it disseminates data blocks

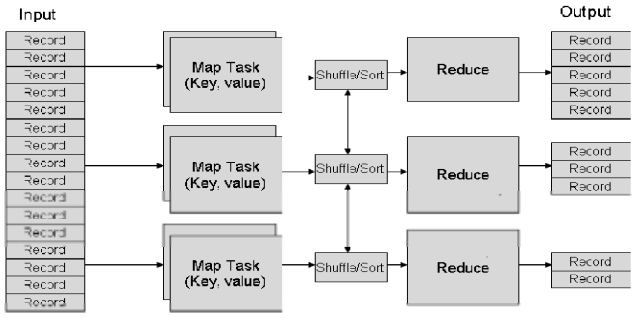


Figure 5. Map Reduce Architecture and Working

It always tries to assign a local data block to a Task Tracker. If the attempt fails, the scheduler will assign a rack-local or random data block to the Task Tracker instead. When map () functions complete, the runtime system groups all intermediate pairs and launches a set of reduce tasks to produce the final results. Large scale data processing is a difficult task, managing hundreds or thousands of processors and managing parallelization and distributed environments makes is more difficult. Map Reduce provides solution to the mentioned issues, as is supports distributed and parallel I/O scheduling, it is fault tolerant and supports scalability and monitoring of heterogeneous and large datasets as in Big Data

A. Map Reduce Components

1. **Name Node** – manages HDFS metadata, doesn't deal with files directly

2. **Data Node** – stores blocks of HDFS – default replication level for each block: 3

3. **Job Tracker** – schedules, allocates and monitors job execution on slaves – Task Trackers

4. **Task Tracker** – runs Map Reduce operations

B.Map Reduce Working

We implement the Mapper and Reducer interfaces to provide the map and reduce methods as shown in figure 6. These form the core of the job.

1) Mapper

Mapper maps input key/value pairs to a set of intermediate key/value pairs. Maps are the individual tasks that transform input records into intermediate records. The transformed intermediate records do not need to be of the same type as the input records. A given input pair may map to zero or many output pairs. The number of maps is usually driven by the total size of the inputs, that is, the total number of blocks of the input files. The right level of parallelism for maps seems to be around 10-100 maps per-node, although it has been set up to 300 maps for very cpu-light map tasks. Task setup takes awhile, so it is best if the maps take at least a minute to execute. For Example, if you expect 10TB of input data and have a block size of 128MB, you'll end up with 82,000 maps .

2) Reducer

Reducer reduces a set of intermediate values which share a key to a smaller set of values. Reducer has 3 primary phases: shuffle, sort and reduce.

2.1) Shuffle

Input to the Reducer is the sorted output of the mappers. In this phase the framework fetches the relevant partition of the output of all the mappers, via HTTP.

2.2) Sort

The framework groups Reducer inputs by keys (since different mappers may have output the same key) in this stage. The shuffle and sort phases occur simultaneously; while map-outputs are being fetched they are merged.

2.3) Secondary Sort

If equivalence rules for grouping the intermediate keys are required to be different from those for grouping keys before reduction, then one may specify a Comparator.

2.4) Reduce

In this phase the reduce method is called for each <key, (list of values)> pair in the grouped inputs. The output of the reduce task is typically written to the File System via Output Collector.

a) Partitioner

Partitioner partitions the key space. Partitioner controls the partitioning of the keys of the intermediate map-outputs. The key (or a subset of the key) is used to derive the partition, typically by a *hash function*. The total number of partitions is the same as the number of reduce tasks for the job.

b) Reporter

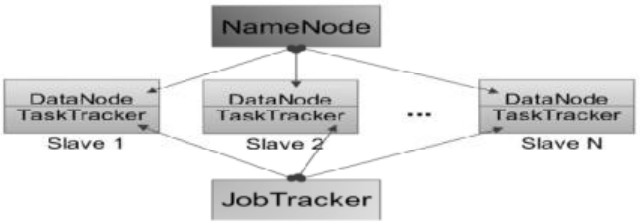
Reporter is a facility for Map Reduce applications to report progress, set application-level status messages and update Counters. Mapper and Reducer implementations can use the Reporter to report progress or just indicate that they are alive.

c) Output Collector

Output Collector is a generalization of the facility provided by the Map Reduce framework to collect data output by the Mapper

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

or the Reducer (either the intermediate outputs or the output of the job). HadoopMapReduce comes bundled with a library of generally useful mappers, reducers, and partitioners



C. Map Reduce techniques

a. Combining

Combiners provide a general mechanism within the MapReduce framework to reduce the amount of intermediate data generated by the mappers. They can be understood as "mini-reducers" that process the output of mappers. The combiner's aggregate term counts across the documents processed by each map task. This result in a reduction in the number of intermediate key-value pairs that need to be shuffled across the network, from the order of total number of terms in the collection to the order of the number of unique terms in the collection. They reduce the result size of map functions and perform reduce-like function in each machine which decreases the shuffling cost.

b. Inverse Indexing

Inverse indexing is a technique in which the keywords of the documents are mapped according to the document keys in which they are residing.
For example
Doc1: IMF, Financial Economics Crisis
Doc2: IMF, Financial Crisis
Doc3: Harry Economics
Doc4: Financial Harry Potter Film
Doc5: Harry Potter Crisis The following is the inverted index of the above data

IMF -> Doc1:1, Doc2:1
Financial -> Doc1:6, Doc2:6, Doc4:1
Economics -> Doc1:16, Doc3:7
Crisis -> Doc1:26, Doc2:16, Doc5:14
Harry -> Doc3:1, Doc4:11, Doc5:1
Potter -> Doc4:17, Doc5:7
Film -> Doc4:24

c.Shuffling

Shuffling is the procedure of mixing the indexes of the files and their keys, so that a heterogeneous mix of dataset can be obtained. If the dataset is shuffled, then there are better chances that the resultant query processing will yield near accurate results. We can relate the shuffling process with the population generating by crossover in the GA algorithms. The processes are different in nature, but their purpose is similar.

d. Sharding

It is a term used to distribute the Mappers in the HDFS architecture. Sharding refers to the groupings or documents which are done so that the Map Reduce jobs are done parallel in a distributed environment.

e. Joins

Join is a RDBMS term; it refers to combining two or more discrete datasets to get Cartesian product of data of all the possible combinations. Map Reduce does not have its own Join techniques, but RDBMS techniques are tweaked and used to get the maximum possible combinations. The join techniques which are adopted for Map Reduce are Equi Join, Self Join, Repartition Join and Theta Join.

f. Clustering &Classification

They are Data Analysis term, used mainly in Data Mining. In Map Reduce it is achieved through K means clustering. Here, iterative working improves partitioning of data into k clusters. After the clustering, the data sorted are grouped together based upon rules to be formed into classes.

The steps for clustering in Map Reduce are;

Step1: Do
Step2: Map
Step3: Input is a data point and k centres are broadcasted
Step4: Finds the closest centre among k centres for the input point
Step5: Reduce
Step6: Input is one of k centres and all data points having this centre as their closest centre
Step7: Calculates the new centre using data points
Step 8: Repeat 1-7, until all of new centres are not changed

VI. CONCLUSION

Big Data analysis tools like Map Reduce over Hadoop and HDFS, promises to help organizations better understand their customers and the marketplace, hopefully leading to better business decisions and competitive advantages. For engineers building information processing tools and applications, large and heterogeneous datasets which are generating continuous flow of data, lead to more effective algorithms for a wide range of tasks, from machine translationto spam detection.
Map Reduce can be exploited to solve a variety of problems related to text processing at scales that would have been unthinkable a few years ago.
No tool no matter how powerful or flexible can be perfectly adapted to every task. Implementing online learning algorithms in Map Reduce is problematic. The model parameters in a

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

learning algorithm can be viewed as shared global state, which must be updated as the model is evaluated against training data. All processes performing the evaluation (presumably the mappers) must have access to this state. In a batch learner, where updates occur in one or more reducers (or, alternatively, in the driver code), synchronization of this resource is enforced by the Map Reduce framework. However, with online learning, these updates must occur after processing smaller numbers of instances. This means that the framework must be altered to support faster processing of smaller datasets, which goes against the design choices of most existing Map Reduce implementations. Since Map Reduce was specifically optimized for batch operations over large amounts of data, such a style of computation would likely result in insufficient use of resources. In Hadoop, for example, map and reduce tasks have considerable start-up costs.

REFERENCES

[1] Jefry Dean and Sanjay Ghemwat, Map Reduce: A Flexible Data Processing Tool, Communications of the ACM, Volume 53, Issuse.1,January 2010, pp 72-77.

[2] Jefry Dean and Sanjay Ghemwat,.MapReduce: Simplified data processing on large clusters, Communications of the ACM, Volume 51 pp. 107–113, 2008

[3] Brad Brown, Michael Chui, and James Manyika, Are you ready for the era of „big data"?, McKinseyQuaterly,Mckinsey Global Institute, October 2011.

[4] DunrenChe, MejdlSafran, and ZhiyongPeng, From Big Data to Big Data Mining: Challenges, Issues, and Opportunities, DASFAA Workshops 2013, LNCS 7827, pp. 1–15, 2013.

[5] MarcinJedyk, MAKING BIG DATA, SMALL, Using distributed systems for processing, analysing and managing large huge data sets, Software Professional"s Network, Cheshire Data systems Ltd.

[6] OnurSavas, YalinSagduyu, Julia Deng, and Jason Li,Tactical Big Data Analytics: Challenges, Use Cases and Solutions, Big Data Analytics Workshop in conjunction with ACM Sigmetrics 2013,June 21, 2013.

[7] J. Dean and S. Ghemawat, "Map Reduce: Simplified data processing on large clusters," in USENIXSymposium on Operating Systems Design and Implementation, San Francisco, CA, Dec. 2004, pp. 137–150.

[8] S. Ghemawat, H. Gobioff, and S. Leung, "The Google File System." in ACM Symposium on Operating Systems Principles, Lake George, NY, Oct 2003, pp. 29 – 43.

[9] HADOOP-3759: Provide ability to run memory intensive jobs without affecting other running tasks on the nodes. https://issues.apache.org/jira/browse/HADOOP-3759

[10] VinayakBorkar, Michael J. Carey, Chen Li, Inside "Big Data Management": Ogres, Onions, or Parfaits?, EDBT/ICDT 2012 Joint Conference Berlin, Germany,2012 ACM 2012, pp 3-14.

[11] GrzegorzMalewicz, Matthew H. Austern, Aart J. C. Bik, James C.Dehnert, Ilan Horn, NatyLeiser, and GrzegorzCzajkowski,Pregel: A System for Large-Scale Graph Processing, SIGMOD"10, June 6–11, 2010, pp 135-145.

[12]Hadoop,"PoweredbyHadoop,"http://wiki.apache.org/hadoop/PoweredBy.

[13] Apache: Apache Hadoop, http://hadoop.apache.org

[14] Apache Hive, http://hive.apache.org/

[15] Apache Graph Project, http://giraph.apache.org/

[16] Mahout, http://lucene.apache.org/mahout/