Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 57 (2015) 563 – 571

MapReduce: Simplified Data Analysis of Big Data

Seema Maitrey[a*],C.K. Jha[b]

[a] Deptt. of Computer Science and Engg, Krishna Institute of Engineering and Technology,Ghaziabad,India
[b]Deptt. of Computer Science and Engg , Banasthali University, Rajasthan, India

Abstract

With the development of computer technology, there is a tremendous increase in the growth of data. Scientists are overwhelmed with this increasing amount of data processing needs which is getting arisen from every science field. A big problem has been encountered in various fields for making the full use of these large scale data which support decision making. Data mining is the technique that can discovers new patterns from large data sets. For many years it has been studied in all kinds of application area and thus many data mining methods have been developed and applied to practice. But there was a tremendous increase in the amount of data, their computation and analyses in recent years. In such situation most classical data mining methods became out of reach in practice to handle such big data. Efficient parallel/concurrent algorithms and implementation techniques are the key to meeting the scalability and performance requirements entailed in such large scale data mining analyses. Number of parallel algorithms has been implemented by making the use of different parallelization techniques which can be listed as: threads, MPI, MapReduce, and mash-up or workflow technologies that yields different performance and usability characteristics. MPI model is found to be efficient in computing the rigorous problems, especially in simulation. But it is not easy to be used in real. MapReduce is developed from the data analysis model of the information retrieval field and is a cloud technology. Till now, several MapReduce architectures has been developed for handling the big data. The most famous is the Google. The other one having such features is Hadoop which is the most popular open source MapReduce software adopted by many huge IT companies, such as Yahoo, Facebook, eBay and so on. In this paper, we focus specifically on Hadoop and its implementation of MapReduce for analytical processing.

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)

Keywords: Big Data, Data Mining, parallelization Techniques, HDFS, MapReduce, Hadoop.

1. Introduction

Organizations with large amounts of multi-structured data find it difficult to use traditional relational DBMS technology for processing and analyzing such data. This type of problem is especially faced by Web-based

* Corresponding author. Tel. 09720877848
E-mail address: seema.maitrey@gmail.com

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)
doi:10.1016/j.procs.2015.07.392

companies such as Google, Yahoo, Facebook, and LinkedIn which require to process huge voluminous data in a speedily and in cost-effective manner. A large number of such organizations have developed their own non-relational systems to overcome this issue. Google, for example, developed MapReduce and the Google File System. It also built a DBMS system known as BigTable. It becomes possible to search millions of pages and return the results in milliseconds or less with the help of the algorithms that drive both of these major-league search services originated with Google's MapReduce framework [1]. It is a very challenging problem of today to analyze the big data. Big data is big deal to work upon and so it is a big job to perform analytics on big data. Technologies for analyzing big data are evolving rapidly and there is significant interest in new analytic approaches such as MapReduce, Hadoop and Hive, and MapReduce extensions to existing relational DBMSs [2].

The use of MapReduce framework has been widely came into focus to handle such massive data effectively.
For the last few years, MapReduce has appeared as the most popular computing paradigm for parallel, batch-style and analysis of large amount of data [3].
MapReduce gained its popularity when used successfully by Google. In real, it is a scalable
and fault-tolerant data processing tool which provides the ability to process huge voluminous data in parallel with many low-end computing nodes [4]. By virtue of its simplicity, scalability, and fault-tolerance, MapReduce is becoming ubiquitous, gaining significant momentum from both industry and academic world. We can achieve high performance by breaking the processing into small units of work that can be run in parallel across several nodes in the cluster [5]. In the MapReduce framework, a distributed file system (DFS) initially partitions data in multiple machines and data is represented as (key, value) pairs. The MapReduce framework executes the main function on a single master machine where we may preprocess the input data before map functions are called or postprocess the output of reduce functions. A pair of map and reduce functions may be executed once or numerous times as it depends on the characteristics of an application [6]. Hadoop is a popular open-source implementation of MapReduce for the analysis of large datasets. It uses a distributed user-level filesystem to manage storage resources across the cluster [7]. Though, the system yields undesired speedup with less significant datasets, but produces a reasonable speed with a larger collection of data that complements the number of computing nodes and reduces the execution time by 30% as compared to normal data mining and other processing techniques [8].
Section 2 gives the overall demonstration of the evolution of map, reduce and Hadoop. Sections 3 give the detail description Big Data and programming model of MapReduce. Section 4 formulates the Hadoop architecture. Sections 5 provide the practical approach of MapReduce and Hadoop technology which is a powerful combination of map and reduce function with the advent of Hadoop.

2. Related Work

Big Data refers to various forms of large information sets that require special computational platforms in order to be analyzed. A lot of work is required for analyzing the big data. But, to analyze such big data is a very challenging problem today. The MapReduce framework has recently attracted a lot of attention for such application that works on extensive data. MapReduce is a programming model and an associated implementation for processing and generating large datasets that is responsive to a broad variety of real-world tasks [9]. The MapReduce paradigm acquires the feature of parallel programming that provides simplicity. At the same time along with these characteristics, it offers load balancing and fault tolerance capacity [10]. The Google File System (GFS) that typically underlies a MapReduce system provides an efficient and reliable distributed data storage which is needed for applications that works on large databases [11]. MapReduce is enthused by the map and reduces primitives present in functional languages [12]. Some currently available implementations are: shared-memory multi-core system [13], asymmetric multi-core processors, graphic processors, and cluster of networked machines [14]. The Google's MapReduce technique makes possible to develop the large-scale distributed applications in a simpler manner and with reduced cost. The main characteristic of MapReduce model is that it is capable of processing large data sets parallelly which are distributed across multiple nodes [15]. The novel Map-Reduce software is a proprietary system of Google, and therefore, not available for open use. Although the distributed computing is largely simplified with the notions of Map and Reduce primitives, the underlying infrastructure is non-trivial in order to achieve the desired performance [16]. A key infrastructure in Google's MapReduce is the underlying distributed file system to ensure data locality and availability [9]. Combining the MapReduce programming technique and an efficient distributed file system, one can easily achieve the goal of distributed computing with data parallelism over thousands of computing nodes; processing data on terabyte and petabyte scales with improved

system performance, optimization and reliability. It was observed that the MapReduce tool is much efficient in data optimization and very reliable since it reduces the time of data access or loading by more than 50% [16]. It was the Google which first popularized the MapReduce technique. [17]. The recently introduced MapReduce technique has gained a lot of attention from the scientific community for its applicability in large parallel data analyses [18]. Hadoop is an open source implementation of the MapReduce programming model which relies on its own Hadoop Distributed File System (HDFS). It does not depend on Google File System (GFS). HDFS replicates data blocks in a reliable manner, places them on different nodes and then later computation is performed by Hadoop on these nodes. HDFS is similar to other filesystems, but is designed to be highly fault tolerant. This distributed file system (DFS) does not require any high-end hardware and can run on commodity computers and software. It is also scalable, which is one of the primary design goals for the implementation. As it is found that HDFS is independent of any specific hardware or software platform, thus, it is easily portable across heterogeneous systems [19]. The grand achievement made by MapReduce has stimulated the construction of Hadoop, which is a popular open-source implementation. Hadoop is an open source framework that implements the MapReduce[20]. It is a parallel programming model which is composed of a MapReduce engine and a user-level filesystem that manages storage resources across the cluster [9]. For portability across a variety of platforms — Linux, FreeBSD, Mac OS/X, Solaris, and Windows — both components are written in Java and only require commodity hardware.

3. THE IMPORTANCE OF BIG DATA

Organizations need to build an investigative computing platform to realize the full value of big data. This enables business users to make use, structure and analyze big data to extract useful business information that is not easily discoverable in its actual original arrangement. The significance of Big Data can be characterized as[21]:

1) Big data is a valuable term despite the hype
2) It is gaining more popularity and interest from both business users and IT industry.
3) From an analytics perspective it still represents analytic workloads and data management solutions that could not previously be supported because of cost considerations and/or technology limitations.
4) The solutions provided enable smarter and faster decision making, and allow organizations to achieve faster time to value from their investments in analytical processing technology and products.
5) Analytics on multi-structured data enable smarter decisions. Up till now, these types of data have been difficult to process using traditional analytical processing technologies.
6) Rapid decisions are enabled because big data solutions support the rapid analysis of high volumes of detailed data.
7) Faster time to value is possible because organizations can now process and analyze data that is outside of the enterprise data warehouse.

The programmers use the programming model MapReduce to retrieve precious information from such big data. The main features and problems associated in handing different types of large data sets are summarized in the table below. It gives précises information how Big Data technologies can help solve them [22].

Table 1: Summarizes the main features, challenges and technology responses connected to handing different types of large data sets

Attribute	Features	Challenges and Skill responses
Volume	Amount of generated data has increased tremendously the past years. However, this is the less challenging aspect in practice.	Internet has created tremendous increment in the global data production. A response to this situation has been through the generalization of the cloud based solutions. The noSQL database approach is a response to store and query huge volumes of data heavily distributed.
Velocity	Production of data is growing with high speed and such produced data must be collected in shorter time frames.	Millions of connected devices (smartphones) are getting added daily which results in the increase of not only the volume but also velocity. To get a competitive edge, global companies considered the Real-time data processing platforms as a requirement.

Variety	There came the explosion of data formats that range from structured information to free text with the multiplication of data sources.	The current way to collect and analyse non-structured or semi-structured data is just opposite from the manner the traditional relational data model and query languages does. This reality has resulted in the evolution of new kinds of data stores that gives the ability to support flexible data models.
Value	Until recently, there was more focus on recording the large volumes of data but not bothered how to conquer them.	Big Data technologies are deeping their roots in creating, capturing and exploiting large volumes of data. In principle, the challenge comes while transforming underdone data into information that contains value and can be used in decision making or other business requirements.

3.1 MapReduce: A Programming Model

MapReduce is designed to be used by programmers, rather than business users. It is a programming model, not a programming language. It has gained popularity for its easiness, efficiency and ability to control "Big Data" in a timely manner. The steps involved in working of MapReduce can be shown in as:

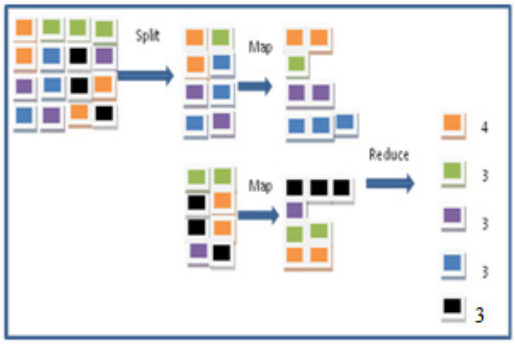

Fig. 1: Steps in MapReduce to process the database

The applications which include indexing and search, graph analysis, text analysis, machine learning, data transformation and many more, are not easy to implement by making the use of standard SQL which are employed by relational DBMSs. In such areas the procedural nature of MapReduce makes it easily understood by skilled programmers. It also has the advantage that developers do not have to be concerned with implementing parallel computing – this is handled transparently by the system. Although MapReduce is designed for programmers, non-programmers can exploit the value of prebuilt MapReduce applications and function libraries [3]. The architecture of MapReduce can be depicted as:

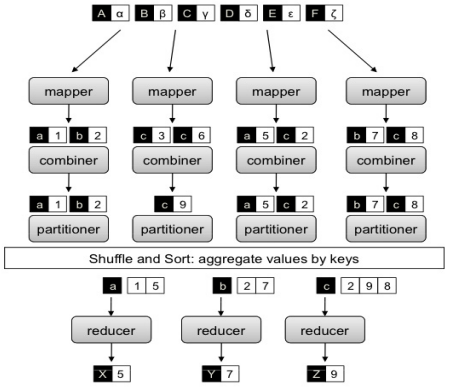

Fig. 2: MapReduce with combiners, partitioners

Table 2: Description of mappers, reducers, partitioners and combiners

Mappers	• Required to generate an arbitrary number of intermediate pairs.
Reducers	• Applied to all intermediate values associated with the same intermediate key.
Partitioners	• Its main job is to divide the intermediate key space, and then to assign the intermediate key-value pairs to reducers.
Combiners	• Combiners are an (optional) optimization. • Before performing the phase of shuffle and sort, it allows the local aggregation of data. • Essentially, combiners are used to save bandwidth, e.g.: word count program.

MapReduce programs are usually written in Java. They can also be coded in other languages such as C++, Python, Ruby, R, etc. These programs may process data stored in different file and database systems. At Google, for example, MapReduce was implemented on top of the Google File System (GFS).

4. Problem Formulation

Hadoop: Yahoo! became the primary contributor in 2006

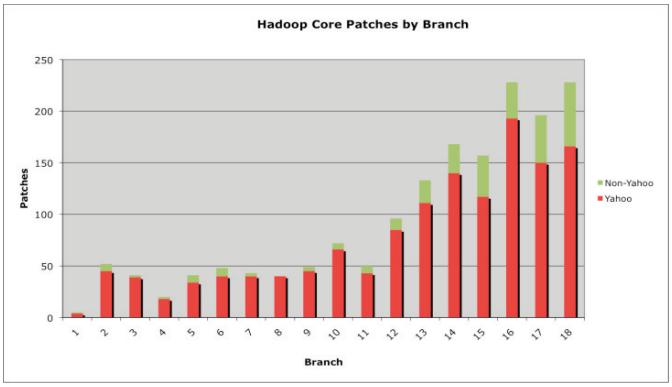

Fig. 3: Primary contribution of Hadoop

Apache Hadoop consists of several components. The ones that of are of interest from a database and analytical processing perspective are [23]:

Hadoop Distributed File System (HDFS), MapReduce, Pig, Hive, HBase, Sqoop

HDFS can be a source or target file system for MapReduce programs. It is best suited to a small number of very large files. Use of data replication make possible to achieve data availability in HDFS. But it results into the rise in storage required to cope the data. The Hadoop MapReduce framework helps in distributing the map program processing so that the required HDFS data is local to the program. To process all of the output files created by the mapping process, the Reduce program performs more movement and access to internode data. At the time of execution, both the map and reduce programs write the accomplished data to the local file system so as to reduce or even avoid the overhead of HDFS replication. HDFS supports multiple readers and one writer (MROW). The index mechanism is not available in HDFS, hence, it is best suited to read-only applications that need to scan and read the complete contents of a file. In HDFS, the actual location of the data is transparent to applications and external software.

HDFS architecture

The architecture of HDFS includes the master and the slave nodes, where the master is called a *NameNode* and the slaves are called *DataNodes*. HDFS contains only a single NameNode (master) and have many DataNodes (slaves) across the cluster, usually one per node. HDFS assigns a *namespace* (similar to a package in Java) to store the users data. A file might be split into one or more data blocks, and these data blocks are kept in a set of DataNodes. The NameNode will have the necessary metadata information on how the blocks are mapped to each other and which blocks are being stored in which of the NameNodes. The request made by the client to read and write the filesystem get processed directly by the DataNodes, whereas namespace operations like the opening, closing, and renaming of

directories are performed by NameNodes. The responsibilities of NameNode for DataNodes are to issue instructions regarding certain activities, such as, data block creation, replication, and deletion [20]. HDFS (Hadoop distributed file system) architecture is shown as [23]:

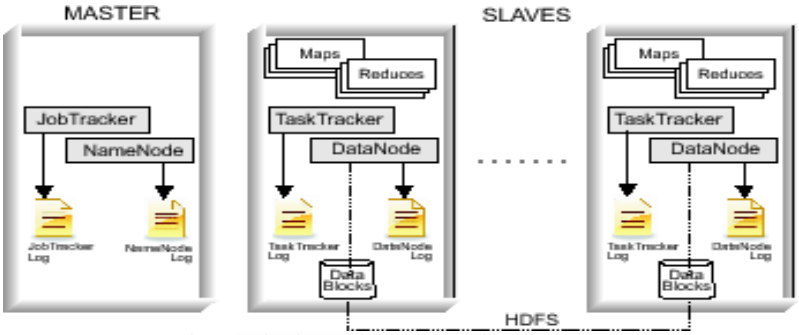

Fig. 4: A simple model of a multi-node Hadoop cluster

A typical deployment of HDFS has a dedicated machine that runs only the NameNode. Each of the other machines in the cluster typically runs one instance of the DataNode software, though the architecture does allow you to run multiple DataNodes on the same machine. The NameNode is concerned with metadata repository and control, but otherwise never handles user data. The NameNode uses a special kind of log, named *EditLog,* for the persistence of metadata.

Deploying Hadoop

Though Hadoop is a pure Java implementation, we can use it in two different ways. We can either take advantage of a streaming API provided with it or use Hadoop pipes. The latter option allows building Hadoop apps with C++. Here, we will focus on the former. Hadoop's main design goal is to provide storage and communication on lots of homogeneous commodity machines. The implementers selected Linux as their initial platform for development and testing; hence, if interested to work with Hadoop on Windows, it is required to install separate software to mimic the shell environment.

Hadoop can run in three different ways, depending on how the processes are distributed [24]:

- **Standalone mode**: This is the default mode provided with Hadoop. Everything is run as a single Java process.
- **Pseudo-distributed mode**: Here, Hadoop is configured to run on a single machine, with different Hadoop daemons run as different Java processes.
- **Fully distributed or cluster mode**: Here, one machine in the cluster is typically labelled as the NameNode and another machine is designated as the JobTracker. Only one NameNode is placed in each cluster, which manages the namespace, filesystem metadata, and access control. An optional SecondaryNameNode can also be placed for periodic handshaking with NameNode for fault tolerance. The rest of the machines within the cluster act as both DataNodes and TaskTrackers. The DataNode holds the system data; each data node manages its own locally scoped storage, or its local hard disk. The TaskTrackers carry out map and reduce operations.

5. Experiment

Writing a Hadoop MapReduce application

The best way to understand and get familiar with the working of Hadoop is to walk through the process of writing a Hadoop MapReduce application. We will be working with a simple MapReduce application that can reverse many strings. The example given below goes through a number of steps which first of all divides the data into different nodes, performs operation to reverse the data, associates the result strings, and then yield the results. This application provides an opportunity to examine all of the main concepts of Hadoop. First, we take a look at the

package declaration and imports in the steps below. The reverstringclass is in the com.javaworld.mapreduce package. It can be shown in set of two imports as given below:

First set of Imports

```
package com.javaworld.mapreduce;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.StringTokenizer;
import java.io.*;
import java.net.*;
import java.util.regex.MatchResult;
```

Second set of Imports

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapredu.JobClient;
import org.apache.hadoop.mapredu.JobConf;
import org.apache.hadoop.mapredu.MapredBase;
import org.apache.hadoop.mapredu.Mapper;
import org.apache.hadoop.mapredu.OutputCollector;
import org.apache.hadoop.mapredu.Reducer;
import org.apache.hadoop.mapredu.Reporter;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
```

The first set of imports is for the standard Java classes, and the second set is for the MapReduce implementation. The reverstringclass begins by extending org.apache.hadoop.conf.Configured and implementing the interface org.apache.hadoop.until.Tool,

Map and reduce

Now you can jump into the actual MapReduce implementation. The two inner classes are: Map: Includes functionality for processing input key- value pairs to generate output key-value pairs.

Map class

```
public static class Map extends MapredBase
      implements Mapper<LongWritable, Text, Text,
Text>
{
  private Text inpText   = new Text();
  private Text reverText = new Text();
  public void map(LongWritable key, Text inputs,
  OutputCollector<Text, Text> output,
  Reporter reporter) throws IOException {
  String inputString = inputs.toString();
  int length = inputString.length();
  StringBuffer reverse = new StringBuffer();
  for(int i=length-1; i>=0; i--)
      {
        reverse.append(inputString.charAt(i));     }
  inputText.set(inpString);
 reverseText.set(reverse.toString());
 output.collect(inpText,reverText);
 }     }
```

Now, it is required to combine all such outputs.
This job is done with the reduce()method of the Reduce class as shown in the steps below:

Reduc.redu()

```
public static class Reduc extends MapRedBase
      implements Reducer<Text, Text, Text, Text> {
      public void reduc(Text key, Iterator<Text> values,
                  OutputCollector<Text, Text> output,
                  Reporter reporter) throws IOException {
         while (values.hasNext()) {
            output.collect(key, values.next());
         }     }        }
```

Reduc: Includes functionality for collecting output from parallel map processing and outputting that collected data.

6. Our Contribution

Recently, in some experiments it has been discovered that applications using Hadoop performed poorly compared to similar programs using parallel databases. Our main objective is to optimize HDFS and provide significant impact on the overall performance of a MapReduce framework which will result in the boosting of overall efficiency of

MapReduce applications in Hadoop. There may be no change in the ultimate conclusions of the MapReduce versus parallel database debate but this new approach of Hadoop and MapReduce will certainly allow a fairer comparison of the actual programming models. Though Hadoop provides built-in functionality to profile Map and Reduce task execution but there are no built-in tools to contour the framework itself, that can allow performance hurdles to remain unexposed. This paper has retrieved the interactions between Hadoop and storage. Here, we explained how many performance blockages are not directly attributable to application code (or the MapReduce programming style), but rather are caused by the task scheduler and distributed filesystem underlying all Hadoop applications. HDFS performance under concurrent workloads can be significantly improved through the use of application-level I/O scheduling while preserving portability. Further improvements can be done by reducing fragmentation and cache overhead which are also possible at the expense of reducing portability. The portability in Hadoop support users by making the development simper and reduce installation complexity. This results in the widespread of this parallel computing paradigm.

7. Conclusion

Big data and the technologies associated with it can bring significant benefits to the business. But the tremendous uses of these technologies make difficult for an organization to strongly control these vast and heterogeneous collections of data to get further analysed and investigated. There are several impacts of using the Big Data. For facing the competitions and strong growth of individual companies, it supports by providing them a huge potential. Certain aspects are needed to be followed so that we can get timely and productive results from Big Data because the precise use of Big Data can give the proliferation to throughput, modernization, and effectiveness for entire divisions and economies. To be able to extract the benefits of Big Data, it is crucial to know how to ensure intelligent use, management and re-use of Data Sources, including public government data, in and across country to build useful applications and services. It is crucial to evaluate the best approach to use for filtering and/or analyzing the data. For the optimized analytic processing, Hadoop with MapReduce can be used. In this paper, we've presented the basics of MapReduce programming with the open source Hadoop framework. This outstanding framework of Hadoop speeds-up the processing of large amounts of data through distributed processes and thus, provides the responses very fast. It can be adopted and customized to meet various development requirements and can be scaled by increasing the number of nodes available for processing. The extensibility and simplicity of the framework are the key differentiators that make it a promising tool for data processing.

References

1. J R Swedlow, G Zanetti, C Best. Channeling the data deluge. Nature Methods, 2011, 8: 463-465
2. G C Fox, S H Bae, et al. Parallel Data Mining from Multicore to Cloudy Grids. High Performance Computing and Grids workshop, 2008
3. Maitrey S, Jha. An Integrated Approach for CURE Clustering using Map-Reduce Technique. In Proceedings of Elsevier, ISBN 978-81-910691-6-3,2nd August 2013].
4. D. DeWitt and M. Stonebraker. MapReduce: A major step backwards. The Database Column, 1, 2008.
5. Apache. Apache Hadoop. http://hadoop.apache.org, 2010.
6. Y. Kim and K. Shim. Parallel top-k similarity join algorithms using MapReduce. In ICDE, 2012.
7. Jeffrey Shafer, Scott Rixner, and Alan L. Cox. The Hadoop Distributed Filesystem: Balancing Portability and Performance. DOP is March 30,2010.
8. Moturi, Maiyo. Use of MapReduce for Data Mining and Data Optimization on a Web Portal.Published in International Journal of Computer Applications (0975 – 8887) Volume 56– No.7, October 2012].
9. Jeffrey Dean et al. Mapreduce: Simplified data processing on large clusters. In Proceedings of the 6th USENIX OSDI, pages 137–150, 2004.
10. S. Ghemawat et al. The google file system. ACM SIGOPS Operating Systems Review, 37(5):29–43, 2003.
11. C. Ranger et al. Evaluating mapreduce for multi-core and multiprocessor systems. In Proceedings of the 2007 IEEE HPCA, pages 13–24, 2007.
12. Yoo, R. M., Romano, A.K. and Kozyrakis, C. 2009. Phoenix Rebirth: "Scalable MapReduce on a Large-Scale Shared-Memory System". Proceedings of the 2009 IEEE International Symposium on Workload Characterization, pp. 198-207.
13. Rafique, Mustafa. M. 2009. "Supporting MapReduce on Large-Scale Asymmetric Multi-Core Clusters". ACM SIGOPS Operating Systems Review, Vol. 43, 2, pp. 25-34.
14. J. Dean et al. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1):107– 113, 2008.
15. Kyong, Lee, Choi, Chung, Moon. Parallel Data Processing with MapReduce: A Survey. Published in SIGMOD Record, December 2011 (Vol. 40, No. 4).
16. B. Panda, J. Herbach, S. Basu, and R. J. Bayardo, "Planet: Massively parallel learning of tree ensembles with mapreduce,"PVLDB, vol. 2, no. 2, pp. 1426–1437, 2009.
17. J. Dean et al. MapReduce: a flexible data processing tool. Communications of the ACM, 53(1):72–77, 2010.
18. Jaliya Ekanayake, Shrideep Pallickara, and Geoffrey Fox, MapReduce for Data Intensive Scientific Analyses. In Fourth IEEE International Conference on eScience (978-0-7695-3535-7/08) eScience, 2008.
19. "GFS, MapReduce, and Hadoop" (Geeking with Greg, June 2006).
20. http://hadoop.apache.org/common/docs/current/hdfs design.html, 2009.

21. MapReduce and the Data Scientist Colin White, BI Research January 2012.
22. Big Data: A New World of Opportunities. NESSI White Paper, December 2012
23. Tomasz Wiktor Wlodarczyk, Yi Han, Chunming Rong: Performance Analysis of Hadoop for Query Processing. AINA Workshops 2011:507-513.
24. W. Tantisiriroj, S. Patil, and G. Gibson. Data-intensive file systems for internet services: A rose by any other name. Technical report, Carnegie Mellon University, 2008.