Abstract— Cloud computing is joined with a new model for supplying of computing infrastructure. Big Data management has been specified as one of the momentous technologies for the next years. This paper shows a comprehensive survey of different approaches of data management applications using MapReduce. The open source framework implementing the MapReduce algorithm is Hadoop. We simulate the different design examples of the MapReduce which stored on the cloud. This paper proposes the application of MapReduce which runs on a huge cluster of machines, in Hadoop framework. The proposed implantation methodology is highly scalable and easy to use for non professional users. The main objective is to improve the performance of the MapReduce data management system in the basis of the Hadoop framework. Simulation result shows the effectiveness of the proposed implementation methodology for the MapReduce.

Keywords— Cloud Computing, Data Management, MapReduce, Hadoop framework, Massive data.

I. INTRODUCTION

Cloud computing is an improved implementation of grid and parallel computing [1]. Cloud computing provides secure, quick and convenient data storage by internet. MapReduce algorithm user for manages the large amount of data which stored in cloud computing. The best optimal solution for achieving the needful computational resources is cloud computing [2]. A common method is to have a massive data that needs to be converted, where the processing of each data item is essential. Hadoop soul, as well as a distributed file system, provides an open source framework for cloud computing.

In this time of excessive data, parallel processing needs a huge volume of data in a timely manner for processing [3]. To process a huge data in parallel with many low-end computing nodes, MapReduce is a scalable tool and fault tolerant tool that enables for processing these massive data [4,5].

The main objectives of the paper are to introduce the MapReduce studies and focusing on the real examples to improving and enhancing the MapReduce framework. Also, in the pi estimation example, we found by increasing the numbers of map and reduce functions the value will be more accurate. This paper is a good reference for using a single node Hadoop framework. The processing of the massive data will be so easy using MapReduce algorithm than traditional distributed systems.

The remainder of this paper is sorted as follows. Section II the architecture reviews and the concepts of cloud computing. Section III discusses the MapReduce Architecture and the advantages and disadvantages of MapReduce. Sections IV introduce the definition of Hadoop framework. In Section V and VI, we show the major application area where the MapReduce framework is observed and discuss open issues and challenges. Finally, Section VII conclusion.

II. CLOUD COMPUTING

In the past ten years, there has been a rapid development in the Internet [1]. The storage cost, the consumed power by computer and hardware is increasing. It is difficult to provide storage areas in data centers and the system and the original internet service can’t solve above questions, so we need new solutions. We need cloud computing to use the current resources of computer, increase the efficiency of economic by improving utilization rate and reduce the consumption of equipment energy. Cloud computing can used three kinds of systems, depending on the type of the possibilities available. There are three types where cloud is used [6]:

A. Infrastructure as a Service

Offering virtualized resources (computing, storage, and communication) [7]. A cloud infrastructure enables to
provisioning of servers running many choices of operating systems and the stack of customized software. The deep layer of cloud computing systems is the Infrastructure services.

B. Platform as a Service

Platform as a Service (PaaS) presents a high level of abstraction which makes a cloud easy programmable, infrastructure oriented clouds provides raw computing and storage services. An example of Platform as a Service, Google AppEngine, provides an environment for deploying and hosting Web applications, which should be written in specific programming languages such as Java or Python [7].

C. Software as a Service

Finally, there are services of potential concern to a wide diversity of hosted users in Cloud systems [6]. This is a local alternative to run applications. An example is the word processors, which is online alternatives of typical office applications. This type is called Software as a Service (SaaS).

III. MAPREDUCE ARCHITECTURE

MapReduce is a programming model which supports the framework model [3]. Hide details of parallel execution and only make users interested on data processing strategies this is the main concept of the MapReduce model. There are two preliminary functions consists on MapReduce model: Map and Reduce, as shown in figure 2. MapReduce input is a list of keys and values pair and function Map() is applied to each pair to compute intermediate key-value pairs. The intermediate key-value pairs are then collected together on the key basis equality, i.e. (key1, list(value1)). For each key1, function Combine() works on the list of all values, then produces aggregated results. The Map() and Reduce() functions defined by the users.

An open-source Java implementation of MapReduce is Apache Hadoop [8]. Same as MapReduce, Hadoop consists of two layers: storage of data layer called Hadoop DFS and a processing of data layer called Hadoop MapReduce framework.

A single MapReduce (MR) work will be done in two stages: Map and Reduce stages. Idle workers picks by the master and allocate each one a map or a reduce task according to the stage. An input file is loaded on the distributed file system before the Map task starting. when loading, the file is dividing into multiple data blocks with the same size, typically 64MB, and to guarantee fault-tolerance each block is triplicated. Each block is then implemented to a mapper, a worker which is specified a map task, and the mapper applies Map() to each record in the block of data. The mappers produced the intermediate output and then sorted locally for grouping key-value pairs sharing the same key. After local sort, function Combine() is applied optionally to perform pre-compilation on the grouped key-value pairs which makes the communication cost is minimized. Communication cost known as the time taken to transfer all the intermediate outputs to reducers.

The outputs of map function map() are stored in local disks of the mappers, divided into R, where R is the number of Reduce tasks in the MR job. This dividing is essential done by a hash function hash(key).

When all Map tasks are achieved, the Reduce tasks assigns by MapReduce scheduler to workers. By the HTTPS protocol intermediate results are shuffled and assigned to reducers. After the partitioning and stored of the output completed, each reducer performs the shuffling by simply dragging its partition of the mapped outputs from mappers. Basically, each mapped outputs are specified to only one reducer by one-to-one shuffling strategy. Note that this data transfer is performed by reducers’ pulling intermediate results. The intermediate results taken by the reducer and merges them by the intermediate keys, so that all values of the same key are grouped together. This grouping is done by external merge-sort. Then each reducer applies Reduce() to the intermediate values. The outputs of reducers are stored in Hadoop Distributed File System (HDFS).

Note that the number of Map tasks depend on the number of input not the number of nodes blocks [3]. Each block is appointment of a single Map task. However, all Map tasks do not need to be executed simultaneously and neither are Reduce tasks. For example, if an input is divided into 400 blocks and there are 40 mappers in a cluster, the numbers of map tasks are 400 and the map tasks are executed through 10 waves of run tasks. This pattern behavior is also reported in [9].
We summarize the defects in the MapReduce framework for data processing.

- **Simplicity and ease**: The model of MapReduce is simple but on the other hand expressive. With MapReduce, a programmer defines his job with only Map and Reduce functions, without knowing any physical distribution of his job across nodes.

- **Flexibility**: MapReduce does not have any dependency on the model of data and schema. With MapReduce a programmer can deal with rogue data or unstructured data more easily than they do with DBMS.

- **Independence in storage**: MapReduce is basically independent from underlying storage layers. Then, MapReduce works with different layers of storage such as BigTable and others [10].

- **Fault tolerance**: MapReduce is highly fault-tolerant. It is worth mentioning that the MapReduce can continue to work in spite of an average of 1.2 failures per analysis job at Google [4, 5].

- **High scalability**: The most important feature of using MapReduce is high scalability. Yahoo! reported that their Hadoop gear could increase the size more than 4,000 nodes in 2008 [5].

**B. MapReduce Disadvantages**

In spite of many advantages, MapReduce lacks many of the important features using in data analysis in DBMS [3]. As many researchers suggested, commercial DBMSs have a strategy “one size fits all” and are not suited for solving a massive large scale data in the processing tasks. There has been a need for special-purpose data processing tools to be commensurate with the solution of those problems [11, 12, 13]. Although the MapReduce is referred to as a modern way of processing large data in data-center computing [14], it is also defined as a “major step backwards” in parallel data processing in comparison with DBMS [15, 16].

We summarize the defects in the MapReduce framework below, compared with DBMS.

- **Not high-level language**: MapReduce itself does not stand up for any high-level language like SQL in DBMS and any query optimization technique. Users should code their operation works in Map and Reduce functions.

- **No index or schema**: MapReduce is schema-free and index-free. An MR job can work directly after its input is loaded into its storage. However, this impromptu processing delivered the advantages of data modeling. MapReduce needs parsing each item at reading input and switching it into data objects for data processing, which caused the deterioration of performance [16, 17].

- **A Single fixed dataflow**: MapReduce prepares the ease of use with a simple abstraction, but in a static dataflow. For that purpose, many complex algorithms are hard to design with Map and Reduce only in an MR job. Also, there are some algorithms require multiple inputs and there are not well supported since the dataflow of MapReduce is designed to read a single input and generate a single output.

- **Low efficiency**: MapReduce operations are not always optimized for I/O efficiency because the fault-tolerance and scalability as its key objectives. (Consider for sort-merge example based grouping, reflecting the results of the data caching triplication on the distributed file system). In addition, Map and Reduce are blocking operations. When all the tasks of the current stage are finished the transition to the next stage can be made. Therefore, parallelism pipeline may not be invested. Moreover, restart the block-level, a shuffling strategy one-to-one and also lower the efficiency per node can done by a simple runtime scheduling. MapReduce does not have certain plans execution and does not optimize plans like DBMS does to minimize data transfer across nodes. Therefore, MapReduce often shows lower performance than DBMS [16]. Moreover, the MapReduce framework has a hiding problem that comes from the nature of inherent batch processing. All of inputs for an MR job should be prepared in advance for processing.

- **Very young MapReduce**: has been generalized by Google since 2004. Compared to over 40 years of DBMS, codes are uncompleted yet and availability of third-party tools are still relatively few.

**IV. HADOOP**

Hadoop is an open source framework for coding and running distributed applications that process a massive amount of data [18]. Distributed computing is a loose and diverse field, but the distinctions key of Hadoop are that it is

- **Accessible**—Hadoop administers on large clusters of commodity machines or on cloud computing services such as Amazon’s Elastic Compute Cloud (EC2).

- **Robust**—Because it is rely on the administers on commodity hardware, Hadoop is architected with the assumption of hardware malfunctions frequent. That could allow you to deal with most of these failures.

- **Scalable**—Hadoop scales linearly to treat bigger data by adding more nodes to the cluster.

- **Simple**—Hadoop allows users to write efficient parallel code quickly.

The accessibility and simplicity of Hadoop give it an writing and running large distributed programs. Even students can create their own Hadoop cluster quickly and cheaply. And, on the other hand, durability and flexibility make it suitable to demanding jobs in Yahoo and Facebook. All these features make Hadoop commonly used in industrial and scientific circles.

Figure 3 illustrates how one interacts with a Hadoop cluster. Hadoop cluster is a set of commodity machines connected
together in one location by via a network. Data storage and data manipulation all occur within this “cloud” of machines. Many different users can suggest computing “jobs” to Hadoop from individual clients, which can be their own desktop machines in remote locations from the Hadoop cluster.

Fig. 3 A Hadoop cluster process and store a large data sets by many parallel machines.

V. APPLICATIONS

Our application separated into two parts: The first part is considered to the way for running and testing Hadoop framework as a single node cluster a two examples one for a wordcount and other for Pi estimation. In the wordcount example we run and test the example as it and after make some changes on the source java code. In these two examples we learn how to create your own code as you need in the application.

The second part is considered about the case study using a Java MapReduce on a real data for weather data set to mines weather in different locations.

A. Hadoop Examples

We running Hadoop MapReduce framework, we try to run Hadoop only on a desktop or laptop computer as a single node machine. The future work is to run Hadoop over a multi node cluster of machines. The more powerful development work is to run Hadoop on a single node[18].

To run Hadoop we need Java (version 1.6 or higher). We must install Hadoop as a single node cluster. You will find for example of dozens of programs with Hadoop and one of them is a wordcount and Pi estimation examples. The important classes of these programs are Java program which implements the map function and reduce function to the specific examples.

Running the WordCount Example

The input directory <input> is the only parameters of text documents to analyse the data and the output directory <output> where the program will case its output. Now we analyse its wordcount and see the sample result in figure 4.

The output will be stored on a separated file called output which shown in figure 5. The figure shows the counting of each word stored in the input file.

Fig. 4 The running sample for the WordCount example.

In the installation at src/examples /org /apache /Hadoop /examples /WordCount.java we can find Source code for WordCount[18]. Now we adjust it as per our requirements. Note: the output directory each time you run Hadoop command should be removed, because it is created directly [18].

Fig. 5 The output file for the wordcount example.

Running Pi Estimation Example

The pi example estimates pi using the Monte Carlo method [2]. The samples number is the number of random points set in the square. The calculation of pi will be more accurate when the value is large. For simplicity, we are going to make a good estimation of pi by using very large operations.

The program of pi takes two integer arguments: the number of maps and the number of samples per map. The number of maps times and the number of samples per map used to calculate the total number of samples. In a 1 × 1 area the map task generates a random point. For each sample, the point is inside if X^2+Y^2 <=1; otherwise, the point is outside. The key of map outputs is 1 or 0 if a value of 1 for a point that is inside or outside the circle, diameter 1. The reduce task sums the number of inside points and the number of outside points. The ratio between this is, in the limit, pi.

In this example, the job works quicker and with less output. For comparison we will choose 2 maps, with 10 samples each, for a total of 20 samples. After that we try to increase the number of samples to be 5 maps and 20 samples each. Then 10 maps and 100 sample each. Finally we increased maps to be 100 and 1000 sample for each.
To run the example, change the working directory of your shell to HADOOP_HOME (via cd ${HADOOP_HOME}) and enter the following:
tahani@ubuntu:~/src/hadoop-0.20.2$ hadoop jar hadoop-0.20.2-examples.jar pi 2 10. The bin/hadoop jar command submits jobs to the cluster. There are three steps to processes command-line arguments, with each step consuming some of the command-line arguments. The main class for the application found inside the hadoop-0.19.0-examples.jar file. The next three arguments are passed to this class. The output will be something like that shown in figure 6. And figure 7 and 8 and 9 shows a sample output with different maps numbers and samples.

When we analyze these figures we can simply notice that the estimation will be more accurate when the data samples increased. Time will increase slightly when the data sample increased. Figure 10 shows the relation between samples number and the estimation time.

---

VI. CASE STUDY

For our case study, we will write a program that weather data mines [8]. Sensors of weather collect data every hour from many locations. A large volume of weather data set is a good candidate for analysis with MapReduce. The data we will use is come from the National Climatic Data Center (NCDC, http://www.ncdc.noaa.gov/). The stored data using a line oriented ASCII format, in which each line is a record. The format supports a rich set of meteorological elements, many of which are optional or with variable data lengths. For simplicity, we will focus on the essential elements, such as temperature, which are always defined and are of fixed width. Data files are arranged by date and weather station. There is a guide for each year from 1901 to 2001. There are
tens of thousands of weather stations; all dataset is consists of a large number of relatively small files.

A. Analyzing Data

We need to express our query as a MapReduce job to use the advantage of the parallel processing that Hadoop provides. The map phase input is the raw NCDC data. The map function is simple. So these are the only fields we are interested, the year and the air temperature. The function reduce() finding the minimum temperature for each year. The final output is the minimum comprehensive temperature recorded on each year. The full flow of data is illustrated in figure 12.

B. A test run

After writing a code of a MapReduce job, it’s normal to try it out on a small dataset to show abroad any immediate problems with the code. Let’s test it on the input file for year 1901 and year 1902. Figure 13 and figure 14 shows the test run of the MapReduce job in year 1901 and 1902.

This result is the same as when we resolved by a regular way. We interpret this as saying that the minimum temperature recorded in 1901 was -5°C, and in 1902 it was 2°C.

VII. CONCLUSION

This bibliographical study focuses on the survey of the management of data Implementation and evaluation methodologies using MapReduce based on Hadoop framework. We state the general overview of cloud computing as well as its features, design and architecture. We also present several Data Management examples. This target is illustrated through running different case studies. The link between cloud computing and parallel data processing tools based on MapReduce discussed on this paper. From our Implementation methodology concludes that the Hadoop framework is efficient platform for data management based on MapReduce algorithm. The proposed Implementation framework considers the usability and the flexibility of implemented system. In the future we try to estimate the language models using Hadoop and Hbase.

REFERENCES