LARGE-SCALE DATA PROCESSING USING MAPREDUCE IN CLOUD COMPUTING ENVIRONMENT

Samira Daneshyar[1] and Majid Razmjoo[2]

[1,2]School of Computer Science,
Centre of Software Technology and Management (SOFTEM),
Faculty of Information Science and Technology,
Universiti Kebangsaan Malaysia (The National University of Malaysia),
43600 UKM Bangi, Selangor Darul Ehsan, Malaysia
[1]samira.daneshyar *at* gmail.com, [2]mrazmjoo*at* gmail.com

ABSTRACT

The computer industry is being challenged to develop methods and techniques for affordable data processing on large datasets at optimum response times. The technical challenges in dealing with the increasing demand to handle vast quantities of data is daunting and on the rise. One of the recent processing models with a more efficient and intuitive solution to rapidly process large amount of data in parallel is called MapReduce. It is a framework defining a template approach of programming to perform large-scale data computation on clusters of machines in a cloud computing environment. MapReduce provides automatic parallelization and distribution of computation based on several processors. It hides the complexity of writing parallel and distributed programming code. This paper provides a comprehensive systematic review and analysis of large-scale dataset processing and dataset handling challenges and requirements in a cloud computing environment by using the MapReduce framework and its open-source implementation Hadoop. We defined requirements for MapReduce systems to perform large-scale data processing. We also proposed the MapReduce framework and one implementation of this framework on Amazon Web Services. At the end of the paper, we presented an experimentation of running MapReduce system in a cloud environment. This paper outlines one of the best techniques to process large datasets is MapReduce; it also can help developers to do parallel and distributed computation in a cloud environment.

KEYWORDS

MapReduce, Hadoop, cloud computing, parallel and distributed processing

1. INTRODUCTION

Today, the need to process large amount of data has been enhanced in the area of Engineering, Science, Commerce and the Economics of the world. The ability to process huge data from multiple resources of data remains a critical challenge.

Many organizations face difficulties when dealing with a large amount of data. They are unable to manage, manipulate, process, share and retrieve large amounts of data by traditional software tools due to them being costly and time-consuming for data processing. The term large-scale processing is focused on how to handle the applications with massive datasets. Such applications devote the largest fraction of execution time to movement of data from data storage to the computing node in a computing environment [1]. The main challenges behind such applications

DOI : 10.5121/ijwsc.2012.3401

are data storage capacity and processor computing power constrains. Developers need hundreds or thousands of processing nodes and large volume of storage devices to process complex applications with large datasets, such applications process multi-terabyte to petabyte-sized datasets [2] and using traditional data processing methods like sequential processing and centralized data processing are not effective to solve these kinds of application's problems.

The question is how to process large amounts of distributed data quickly with good response times and replication at minimum cost? One of the best ways for huge data processing is to perform parallel and distributed computing in a cloud computing environment. Cloud computing as a distributed computing paradigm aims at large datasets to be processed on available computer nodes by using a MapReduce framework. MapReduce is a software framework introduced to the world by Google in 2004; it runs on a large cluster of machines and is highly scalable [3]. It is a high-performance processing technique to solve large-scale dataset problems. MapReduce computation processes petabyte to terabyte of unit data on thousands of processors. Google uses MapReduce for indexing web pages. Its main aim is to process large amount of data in parallel stored on a distributed cluster of computers. This study presents a way to solve large-scale dataset processing problems in parallel and distributed mode operating on a large cluster of machines by using MapReduce framework. It is a basis to take advantage of cloud computing paradigm as a new realistic computation industry standard.

The first contribution of this work is to propose a framework for running MapReduce system in a cloud environment based on the captured requirements and to present its implementation on Amazon Web Services. The second contribution is to present an experimentation of running the MapReduce system in a cloud environment to validate the proposed framework and to present the evaluation of the experiment based on the criteria such as speed of processing, data-storage usage, response time and cost efficiency.

The rest of the paper is organized as follows. Section II provides background information and definition of MapReduce and Hadoop. Section III describes workflow of MapReduce, the general introduction of Map and Reduce functions and it also describes Hadoop, an implementation of the MapReduce framework. Section IV we present MapReduce in cloud computing. Section V presents the related MapReduce systems. Section VI captures a set of requirements to develop the framework. Section VII shows the proposed framework and the implementation of the framework on Amazon Web Services for running a MapReduce system in a cloud environment it also presents an experimentation of running a MapReduce system in a cloud environment to validate the proposed framework and resulting outcomes with evaluation criteria. Section VIII concludes this paper.

2. BACKGROUND OF MAPREDUCE AND HADOOP

2.1.MapReduce Definition& History

MapReduce has been facilitated by Google as a programming framework to analyse massive amounts of data. It uses for distributed data processing on large datasets across a cluster of machines. Since the input data is too large, the computation needs to be distributed across thousands of machines within a cluster in order to finish each part of computation in a reasonable amount of time. This distributed concept implies to parallelize computations easily and using re-execution as the main technique for fault tolerance. J.Dean and S.Ghemawat [3] from Google Inc. published a paper in 2004 describing MapReduce. Google never released their implementation of MapReduce. Finally, the Apache Company made available a concrete implementation of MapReduce named Hadoop [4]. MapReduce allows developers to perform complex computations in a simple way while hiding the details of data distributation, parallelization, and fault tolerance.

The unique feature of MapReduce is that it can both interpret and analyse both structured and unstructured data across many nodes through using of a distributed share nothing architecture. Share nothing architecture is a distributed computing architecture consisting of multiple nodes. Each node is independent, has its own disks, memory, and I/O devices in the network. In this type of architecture, each node is self-sufficient and shares nothing over the network: therefore, there are no points of contention across the system.

A MapReduce programming model drives from three fundamental phases:

1. *Map phase*: partition into M Map function (Mapper); each Mapper runs in parallel. The outputs of Map phase are intermediate key and value pairs.
2. *Shuffle and Sort phase*: the output of each Mapper is partitioned by hashing the output key. In this phase, the number of partitions is equal to the number of reducers; all key and value pairs in shuffle phase share the same key that belongs to the same partition. After partitioning the Map output, each partition is stored by a key to merge all values for that key.
3. *Reduce phase*: partition into R Reduce function (Reducer); each Reducer also runs in parallel and processes different intermediate keys.

MapReduce libraries have been written in several programming languages,include, LISP, Java, C++, Python, Ruby and C. A presentation of the MapReduce workflow includes;a dataset is divided into several units of data and then each unit of data is processed in a Map phase.Finally, they are combined in Reduce phase to produce the final output.

Map function takes input pairs and produces a set of intermediate key and value pairs and passes them to the Reduce function in order to combine all the values associated with the same key. Reduce function accepts an intermediate key as a set of values for that key; it merges together these values to prepare a proper smaller set of values to produce the output file [5].

2.2. Hadoop Definition & History

Hadoop is an open-source, java based Apache Software foundation project. It is one of the most interesting third-party implementations of MapReduce for distributed computing. Hadoop provides a framework to support large dataset distributed computing on a cluster of servers. Hadoop was inspired in 2006 by Doug Cutting (named by his son's stuffed elephant) [4] now being used by major companies, including Amazon, IMB, Yahoo, Facebook and a growing number of other companies. Hadoop provides its own distributed file system called Hadoop Distributed File System (HDFS), HDFS is a distributed file system designed to store multiple copies of data block on a cluster of compute nodes, to enable reliable and rapid computations. Hadoop runs in a distributed environment. A simple Hadoop cluster consists of $n \geq 1$ machines running the Hadoop software [6]. Hadoop uses HDFS to store a dataset and applies the MapReduce's power to distribute and parallelize the processing of this dataset. Stream data is first fragmented into typical HDFS-sized block (64MB) and then smaller packets (64KB) by the user [7]. It means a file into HDFS is divided into 64 MB chunk and each chunk is resided on a different working machine.

3. MAPREDUCE FRAMEWORK

3.1.MapReduce Workflow Overview

When the user runs a MapReduce program, the following operations occur as shown in Figure1.

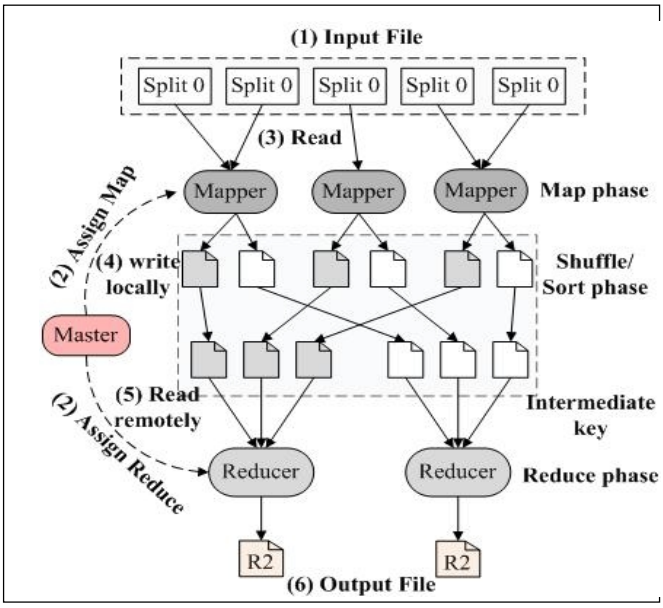

Figure1. MapReduce workflow

The numbered labels in the above figure relate to the numbers in the list below:

1. Execution begins by running the MapReduce program. User submits a job to the master node. Master node divides the input dataset into many pieces of data, typically 16MB to 64MB data per pieces into HDFS and then creates several copies of the user MapReduce program over working machines in the cluster. Each machine in the cluster has a separate instruction of the MapReduce program. HDFS makes multiple copies of data block for reliability, put them over working machines within the cluster. After that MapReduce automatically starts to process the blocks of data where they are located.
2. One copy of the program placed on a machine is specific and that machine is called the master node. The rest of the program is assigned to worker (slave) nodes by master node. The master node partitions a job into the Map and Reduce tasks. The Map tasks and Reduce tasks are performed in order by Mapper and Reducer functions. The master node then chooses the idle Mapper or Reducer and allocates each of them Map or Reduce task.
3. The Mapper function receives a Map task, read the content of a chunk of data and invoke the user defined Map function. Then the Map function produces the intermediate key and value pairs for each chunk of the input data. The outputs of a Map function (intermediate key and value pairs) are buffered in the memory.
4. The output data in the temporary buffers are stored on the local disk and the physical memory addresses of these buffered data are sent to the master node. The master finds the idle workers and forwards the location of these buffered data to them to perform Reduce tasks.
5. A Reducer function informs by master node about these physical memory addresses;it uses a remote procedure to access the buffered data from the Mappers on the local disks. When a reducer read all the intermediate key and value pairs, it sorts them by the intermediate keys so that all the data of the same intermediate key are classified together.
6. The Reducer sends each unique key and its consequent set of intermediate values to the Reduce function. The final output is available in the Reducer; then it is stored to the distributed file system (HDFS).

3.2. Map Function

This function accepts a chunk of input data and produces an intermediate key and value pairs. Map (in_key, in_value) →list (out_key, intermediate_value) All output intermediate key and value pairs from the Map function become integrated according to a common and known intermediate key K between them and then they are passed into the Reduce function. The Map function can be executed in parallel on non-overlapping portions of the input data [8].

3.3. Reduce Function

The Reduce function takes an intermediate key K and all the set of values for that intermediate key; after that it combines these values to produce a small set of values. Reduce (out_key, list (intermediate_value)) →list (out_value) the result of Reduce function is a zero or a single output value per function call. The Reduce function can be executed in parallel on each set of intermediate pairs with the same key.

3.4. Hadoop an Implementation of MapReduce

Hadoop, as stated above, is a java based software framework for running distributed large-scale applications on commodity of servers. Hadoop is a very useful and trustworthy tool for creating distributed programs that perform better in terms of computational efficiency [9]. Hadoop offers two main services: reliable data storage (HDFS) and high-performance parallel and distributed data processing using MapReduce. Hadoop runs on the commodity cluster of machines. Hadoop cluster configuration works under a master/slave concept. The master server controls all activities within a cluster and slaves.

Workers work for the master node. In the master/slave architecture of Hadoop as shown in Figure 2, there are two master nodes, including *namenode* and *jobtracker* and N number of slave nodes, including *datanodes* and *tasktrackers*.

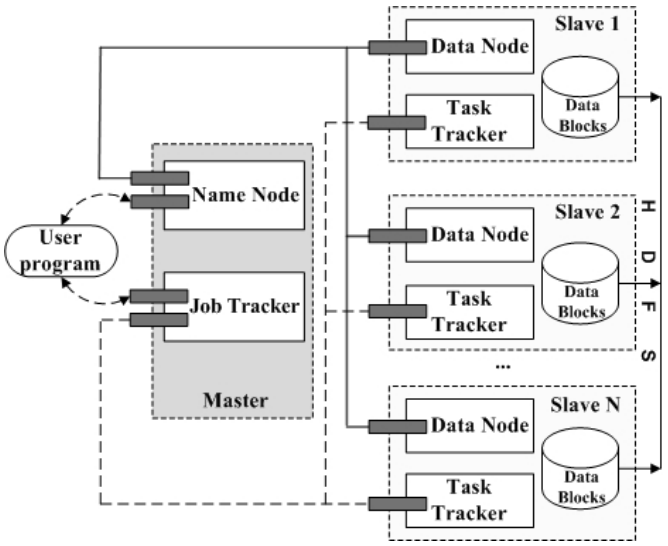

Figure 2.Hadoop architecture

Namenode is a single master server in a cluster environment that manages the file system data within the whole Hadoop's distributed file system and adjusts the read/write access to the data file

by the user. There are generous numbers of *datanodes* (one datanode per computer node in the cluster) which are used to store and retrieve the data, and they will be performing the read and write requests from the file system clients. *Datanodes* are also responsible for performing replication tasks and more importantly sorting the file system data [4]. *Namenode* and *datanodes* components play fundamental roles in a Hadoop file system. *Jobtracker* accepts and controls submitted jobs in a cluster environment, *and* is responsible for scheduling and monitoring all running tasks on the cluster. It divides the input data into many pieces of data to be processed by the Map and Reduce tasks. *Tasktrackers* execute Map tasks and Reduce tasks via received instructions from the *jobtracker*. They are responsible for partitioning Map outputs and sorting/grouping of reducer inputs. The *jobtracker* and *tasktrackers* make the backbone of running a MapReduce program.

4. MAPREDUCE IN CLOUD COMPUTING

Cloud Computing refers to both the applications delivered as services over the Internet, the hardware and the system software in the datacenters that provide those services [10]. Cloud platform, or platform as a service, refers to provide a computer platform or software stack as a service[11]. Developers by using the cloud computing paradigm are enabled to perform parallel data processing in a distributed environment with affordable cost and reasonable time. Thus, the advantage of data processing using cloud computing is the capability to easily do parallel and distributed computing on large clusters.

It is a fact that many cloud computing based computational processes will handle large datasets in a fully distributed environment in both a wired and wireless computer networking and communication environment. By using cloud computing, we enable to store, collect, share and transfer large amounts of data at very high speeds in a seamless and transparent manner that would out of sheer necessity classify all data to be “totally virtual”. Hence, all data in cloud computing captures the concept of data virtualization through a new programming paradigm or model which treats all data as a single entity through a process called MapReduce. MapReduce is a popular computing framework that is widely used for big data processing in cloud platforms. Cloud computing as a distributed computing paradigm, provides an environment to perform large-scale data processing. It enables massive data analytics on available computer nodes by using MapReduce platform. MapReduce and its open-source implementation Hadoop, allow developers to process terabytes of data that take hours to finish while hiding the complexity of parallel execution across hundreds of servers in a cloud environment. The main reason of using MapReduce with cloud computing is a key point of MapReduce that hides how parallel programming work away from the developer.

A Major web company, Amazon web services platform offers a service called Amazon Elastic MapReduce to store and process massive datasets by running MapReduce system on Amazon cloud. It utilizes a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3). EC2 is a web service platform that provides resizable compute capacity in a cloud [12]. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web [12].

5. MAPREDUCE REQUIREMENTS

5.1. Fundamental and Specific Requirements

MapReduce platform is a distributed runtime engine for managing, scheduling and running MapReduce systems in a cluster of servers across an open distributed file system. To develop a MapReduce system based on the proposed framework, fundamental and specific requirements have been captured. Table 1 lists the summary of fundamental requirements for MapReduce system. Table 2 lists the summary of specific requirements. Both requirements essentially must be met to make a MapReduce system for large-scale processing more efficient.

Table 1.Fundamental Requirements.

No	Fundamental Requirements	Description
1	Scalability	To scale petabytes of data on thousands of machines.
2	Parallelism	All tasks must run in parallel.
3	Distributed Data	MapReduce distributes a data file to all nodes across a cluster to execute the application.
4	Cost Efficiency	Afford to buy cheaper hardware and pay less for operation, especially if the size of dataset is too big.

Table 2.Specific Requirements.

No	Specific Requirements	Description
1	Availability	Many machine nodes and servers should be available in a computing cluster in failure mode.
2	Reliability	Multiple copies of data should be automatically stored in case of failure.
3	Flexibility	The system should enable to analyse and process various kinds of structured and unstructured data.
4	Security	Before running the system, user authentication is required.
5	Usability	The system should be developed as a service for running arbitrary code.
6	Locality	The system should divide tasks based on location of input file; each part is 64 MB same size of Google File System.
7	Data Consistency	The system should support coordination of data changes and it helps to provide consistency of data to ensure correctness of the execution and result.
8	Trust	When all nodes faithfully execute their tasks, the result is accurate and can trust the result.

6. PROPOSED FRAMEWORK

A framework is proposed based on the MapReduce requirements; it is shown in Figure 3. This framework is designed to operate in a cloud computing environment.

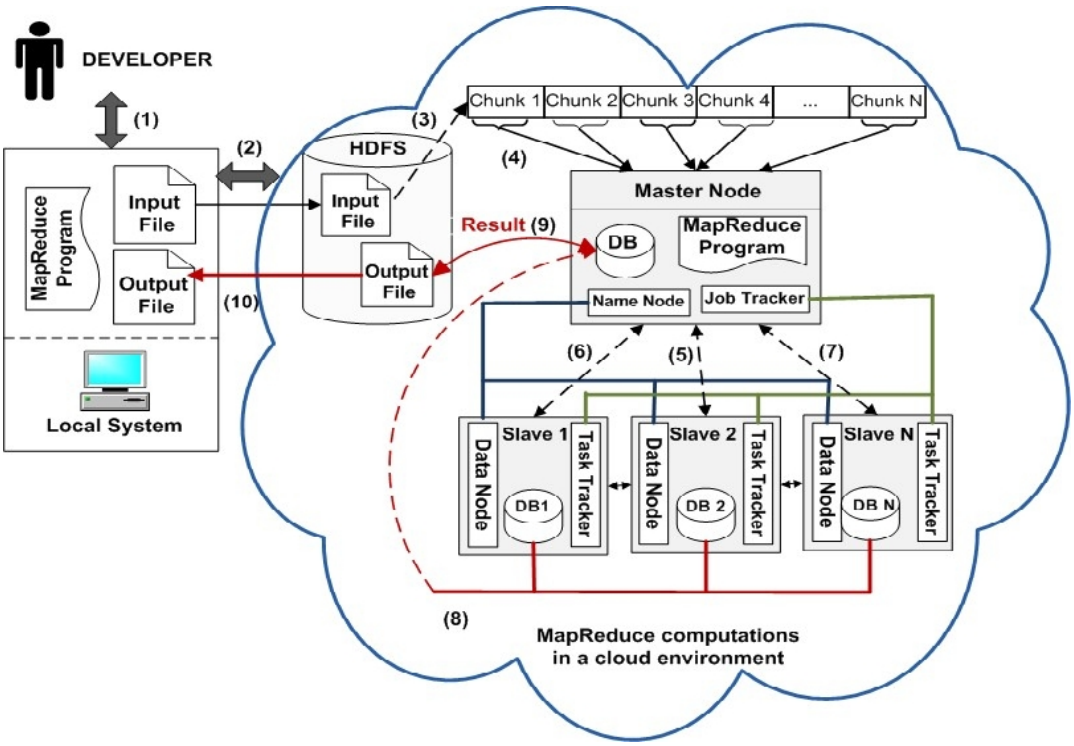

Figure3. MapReduce framework and its components in cloud computing at the platform level

Legends:

(1) Develop a MapReduce program to process a large dataset.
(2) Input file is transmitted into HDFS.
(3) Input file is divided into many chunks.
(4) Master access to chunks and send chunks to slave nodes.
(5) Master and slaves frequently have interaction with each other.
(6) Name node sends chunks to data nodes for processing.
(7) Job tracker submits a job to task trackers.
(8) The results from each slave are combined and backed to master node.
(9) Master node transmits the result to the HDFS.
(10) Developer accesses the output file.

The framework consists of three main components:

- Master: Master node schedules the job component tasks to process on the workers, monitors them and re-executes him failed tasks. It takes the input file and split it up into smaller chunks, then distributes those chunks to the worker nodes. Master takes the output of performing tasks by individual workers, combine them and store the output file into the HDFS to be accessible by the user.

- Slaves: Process those smaller tasks as directed by the master on a determined part of dataset and pass the result back to the master node.
- HDFS: As a distributed, scalable file system stores virtually input and output files. User of the MapReduce system enables to access the final result from this virtual storage.

6.1. The framework implementation on Amazon Web Services

Amazon Web Services are a collection of remote infrastructure services that make a cloud computing environment.

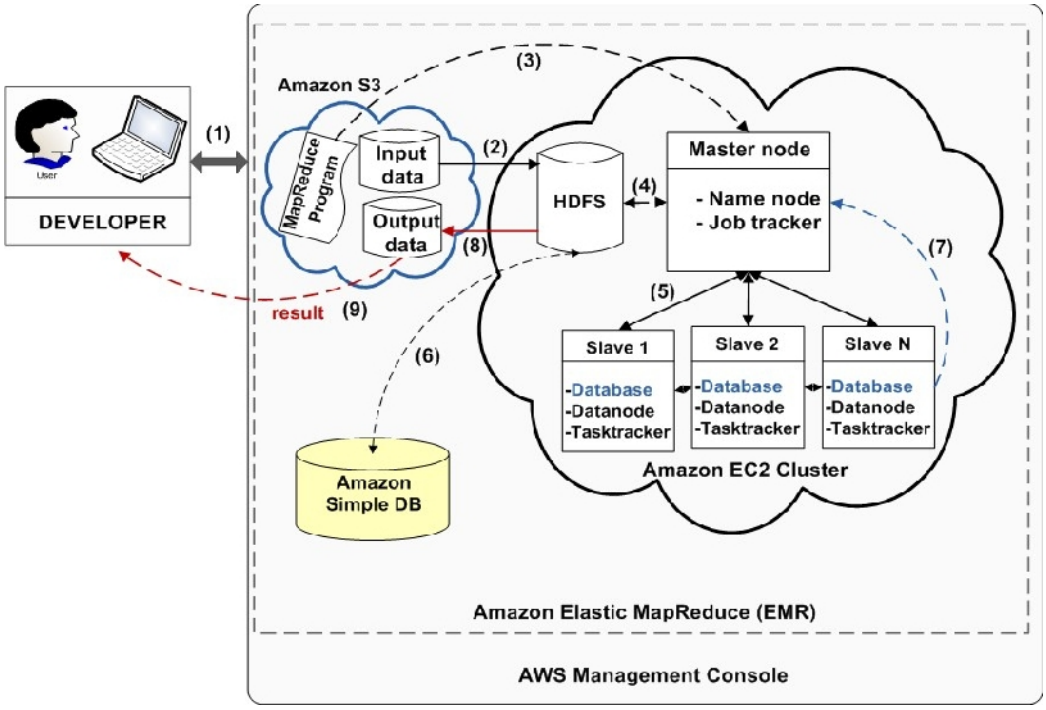

Figure 5. Amazon Web Services running a MapReduce program

Legends:

(1) Connect to AWS console, upload MapReduce program and input dataset on Amazon S3 service.
(2) Input file is uploaded into HDFS
(3) MapReduce program is located at the master node.
(4) Master node gets the input file from HDFS.
(5) Master node distributes data blocks to slaves for processing.
(6) Master node checks and updates the state of the running jobs.
(7) Results from slaves are backed to master node and stored on HDFS.
(8) Final output file is transmitted from HDFS to Amazon S3.
(9) Final output file is transmitted from Amazon S3 to the local machine.

Figure5 shows a working model of implementing the proposed framework in Amazon Web Services, for massive data processing. It shows how the Amazon Web Services uses cloud infrastructure to work with MapReduce for running a MapReduce program for large-scale data processing. It includes a number of Amazon Web Services listed below:

1. AWS management console: it manages and accesses a collection of remote services offered over the internet by amazon.com.
2. Amazon Elastic MapReduce: it introduces the Hadoop framework as a service.
3. Amazon S3: the Simple Storage Service (S3) stores the input and output dataset.
4. Amazon EC2 Cluster: the Elastic Compute Cloud (EC2) is used for running a large distributed processing in parallel.
5. Amazon Simple DB: it determines the state of a given component or process.

To run the system from AWS management console, user who is developed a MapReduce program, sign up for AWS Console to get AWS services, then uploads MapReduce program and input dataset on Amazon S3 service. Input dataset is transmitted into HDFS to be used by EC2 instances.A job flow is started on Amazon EC2 cluster.In this service, one machine works as master node and the others work as slave nodes to distribute the MapReduce process. During the running processes Simple DB shows the status of the running job and all information about EC2 instances. All machines are terminated once the MapReduce tasks complete running. The final result is stored in an output file and the output can be retrieved from Amazon S3.

7. EXPERIMENTATION

For our experiment, a MapReduce program was written in Java to process a large dataset. The program determines the number of occurrences of each word in the given dataset. We executed the program based on our proposed framework in a cloud computing environment hosted byAmazon Web Services[1]. The proposed framework, which is one contribution of this work, was tested by running the MapReduce program on Amazon Elastic Compute Cloud (EC2) cluster. In the experiment, a dataset which is a text file with the size of 1 GB and 200,000,000 words, was used and stored on Amazon Simple Storage Service (S3) for being processed.

The dataset was transmitted from Amazon S3 into Hadoop HDFS to be processed in a cluster of machines from Amazon EC2. In Hadoop HDFS, the dataset was initially divided into 16MB to 64MB chunks of data, each chunk of data residing on one machine in a cluster, one copy of the MapReduce program was created on the master node of the Amazon EC2 cluster and then master node distributed this copy of the program with one chunk of data on all working machines in the cluster as a MapReduce job. Each working machine in the Amazon EC2 cluster executes a MapReduce job based on the instructions of the MapReduce program to process one chunk of data. All auctions were performed in the cloud environment offered by Amazon Web Services at the price of $0.16 per hour for a medium working machine. The output was stored in Amazon S3 to be accessible by the user. It represents the word and the number of times it has appeared in the input dataset with a short response time and minimum cost. In the discussion part, the experiment is evaluated base on the criteria such as data-storage usage, speed of processing, response time and cost efficiency.

Processing details of 1GB dataset which contains 200,000,000 words, are presented in Figure 6, it is based on the completion percentage of Map tasks and Reduce tasks in cloud computing mode with eight numbers of instances.

[1] Amazon cloud computing Web services hosted at: http://aws.amazon.com/

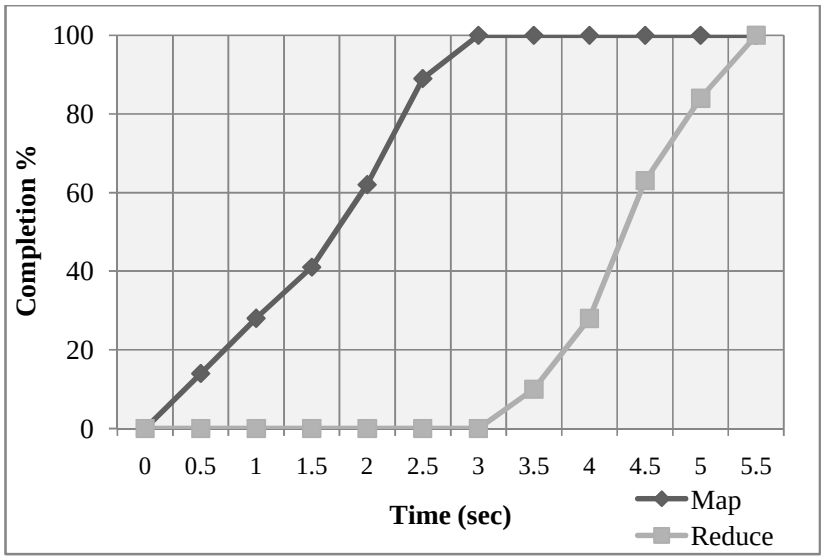

Figure 6. MapReduce progress

As shown in Figure 6 by completion of Map tasks on 3 seconds, the Reduce tasks are started and finished on 5.2 seconds. Map tasks take 3 seconds for 100% completion and Reduce tasks take 2.2 seconds for 100% completion.

7.1. Speed of processing

The speed of processing depends on the how big is the scale of the Amazon EC2 cluster in a cloud environment? The 1GB dataset is used to specify the speed of data processing with a different number of medium instances in the cluster. It is observed from Table 3 that the running speed is improving by increasing the number of instances in the cluster. By using scaling property of Amazon EC2 cluster, we take advantage of high speed processing in a distributed environment.

Table 3. Speed of processing in Amazon ec2 cluster.

No. of instances	Time Taken (sec)
2	65.3
4	44.8
6	26.6
8	5.2

7.2. Data Storage Usage

Amazon EC2 cluster has its own data storage, Hadoop HDFS that stores data in the cluster. HDFS is based on the Google File System. The input file is stored into HDFS and it divided into 64 MB data blocks and HDFS replicates blocks of data to the alternate datanodes through the namenode which is located in the master node; therefore each block of the system which processes a datasets is compatible with it. This system writes its data only once but reads it one or more times when required. With HDFS, when a datanode crashes, the cluster does not crash, but performance decreases in proportion to the amount of lost storage and processing capacity.

7.3. Response time

Before executing MapReduce system a Hadoop cluster needs to be run in standalone or cloud computing environment to submit the MapReduce job in the running cluster. For response time evaluation, the time between sending a request to start the Hadoop cluster and starting the cluster, is calculated as the response time to the Hadoop cluster in a cloud environment. Response time depends on how big is the scale of the Amazon EC2 cluster. Table 4 shows the response time at the different sizes of the cluster.

Table 4. Response time in amazon ec2 cluster

No. of Instances	Time Taken (sec)
2	15.7
4	25.4
6	36.2
8	47

7.4. Cost Efficiency

In Amazon EC2 cluster, we pay only for what we use by the hour. It allows users to rent resources in this pay-as-you-go environment. This feature provides a cost-effective service to process large amounts of distributed data quickly at minimum and affordable cost, especially if the size of dataset is too big. In terms of cost, we run a cost-efficient Hadoop cluster in AWS cloud, which can be dynamically extended by renting more instances. Because we used medium instances for our experiment, Table 5 presents pricing model of the Amazon EC2 cluster for four numbers of medium instances. As it observed by increasing the number of instances, the price is slightly rinsing.

Table 5. Amazon ec2 pricing

No. of Instances	Price per Hour
2	$0.32
4	$0.64
6	$0.96
8	$1.28

As discussed above, MapReduce performs data processing with the available working machine at Amazon EC2 clouds with an extremely low budget. The combination of MapReduce frameworks and cloud computing is an attractive propositioning for large-scale data processing so that MapReduce systems can take advantage of massive parallelization in a computer cluster. From our analysis using cloud computing and MapReduce together improve the speed of processing and decrease consuming cost.

8. CONCLUSION

In conclusion, this paper conducted a comprehensive review and analysis of a MapReduce framework as a new programming model for massive data analytics and its open source implementation, Hadoop. MapReduce is an easy, effective and flexible tool for large-scale fault tolerant data analysis. It has proven to be a useful abstraction that allows programmers to develop easily high performance system for running on cloud platforms and to distribute the processing over as many processors as possible. This paper described Hadoop, an open source

implementation of MapReduce to process large-scale datasets. We also determined the fundamental and specific requirements to develop a framework. We proposed a framework to process a large dataset in a cloud environment in parallel and distributed fashion, as well proposed Amazon Web Services as one instance of the using MapReduce framework. At the end we presented an experimentation of running a MapReduce system in a cloud environment to validate the proposed framework's working schema and the main requirements of the MapReduce framework.

REFERENCES

[1] R. W. Moore, C. Baru, R. Marciano, A. Rajasekar and M. Wan, (1999) "Data-Intensive Computing",Morgan Kaufmann Publishers Inc. San Francisco, USA,ISBN:1-55860-475-8, pp105-129.

[2] I. Gorton, P. Greenfield, A. Szalay, R. Williams, (2008) "Data Intensive Computing in the 21 Century", IEEE Computer Society,ISSN: 0018-9162, pp30-32.

[3] J. Dean and S.Ghemawat,(2004) "MapReduce: simplified data processing on large clusters", Google Inc. In OSDI'04: Proceeding of the 6th conference on Symposium on Operating Systems Design & Implementation, San Francisco, CA.

[4] Hadoop MapReduce, (accessed February 2012), Available online at http://wiki.apache.org/hadoop/MapReduce

[5] R. Lammel, Data Programmability Team, Microsoft Corp, Redmond, (2007), "Google's MapReduce programming model – Revisited", WA, USA, Available online at http://www.sciencedirect.com

[6] S.N.Srirama, P.Jakovits, E.Vainikko, (2011) "Adapting scientific computing problems to clouds using MapReduce", Future Generation Computer Systems, Vol. 28, No. 1, pp184-192.

[7] J.Shafer, S. Rixner, and A.L. Cox, (2010) "The Hadoop Distributed File system: Balancing Portability and Performance", IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS), ISBN: 978-1-4244-6023-6, pp122-133.

[8] C.Ranger, R.Raghuraman, A.Penmetsa, G.Bradski, CH. Kozyrakis, (2007) "Evaluating MapReduce for Multi-ore and Multiprocessor Systems", IEEE 13th International Symposium on High Performance Computer Architecture (HPCA), ISBN: 1-4244-0805-9 , pp13-24.

[9] K.Talattinis, A Sidiropoulou, K.Chalkias, and G.Stephanides, (2010) "Parallel Collection of Live Data Using Hadoop", IEEE 14th PanHellenic Conference on Informatics (PCI), ISBN: 978-1-4244-7838-5, pp66-71.

[10] M. Armbrust , A.Fox, R. Griffith , A.D. Joseph , R.H. Katz , A. Konwinski , G.Lee, D.A. Patterson , A. Rabkin , L. Stoica , M.Zaharia,(2009)"Above the Clouds: A Berkeley View of Cloud Computing", Electrical Engineering and Computer Sciences, University of California at Berkeley, Technical Report No. UCB/EECS-2009-28, pp1-25.

[11] P.T.Jaeger, J.Lin, J.M.Grimes and S.N.Simmons, (2009), "Where is the cloud? Geography, economics, environment, and jurisdiction in cloud computing", First Monday, Vol. 14, No. 5.

[12] Amazon Web Services, (accessed February 2012), Available online at http://aws.amazon.com