EVALUATION OF DATA PROCESSING USING MAPREDUCE FRAMEWORK IN CLOUD AND STAND-ALONE COMPUTING

Samira Daneshyar[1] and Ahmed Patel[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]whinchat2010 *at* gmail.com

ABSTRACT

An effective technique to process and analyse large amounts of data is achieved through using the MapReduce framework. It is a programming model which is used to rapidly process vast amount of data in parallel and distributed mode operating on a large cluster of machines. Hadoop, an open-source implementation, is an example of MapReduce for writing and running MapReduce applications. The problem is to specify, which computing environment improves the performance of MapReduce to process large amounts of data? A standalone and cloud computing implementation are used for the experiment to evaluate whether the performance of running MapReduce system in cloud computing mode is better than in stand-alone mode or not, with respect to the speed of processing, response time and cost efficiency. This comparison uses different sizes of dataset to show the functionality of MapReduce to process large datasets in both modes. The finding is, running a MapReduce program to process and analysis of large datasets in a cloud computing environment is more efficient than running in a stand-alone mode.

KEYWORDS

MapReduce, Hadoop, Cloud Computing, Data Processing, Parallel and Distributed Processing

1. INTRODUCTION

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 vast amount of data while hiding the complexity of parallel execution across hundreds of servers in a cloud environment. The main reason of using MapReduce in cloud computing and standalone mode is a key point of MapReduce that hides how parallel programming work away from the developer.

Amazon web services platform provides a facility to execute a MapReduce system by offering its services to store, process and analyse large-scale datasets. Amazon services include, Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). Amazon EC2 is a web service platform that provides resizable compute capacity in a cloud [1]. 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 [2].

DOI : 10.5121/ijdps.2012.3605

MapReduce and cloud computing have made it easier than ever for academic, government, and industrial organizations to step into the world of big data analytic [3]. Cloud computing provides massive data processing and data-storage services by using MapReduce as a computational platform in Amazon cluster. In a cloud computing environment offered by Amazon, developers enable to write their MapReduce systems and run them in a fully parallel and distributed platform. They will be charged only for the consumed time used on each working machine in the cluster. Therefor they are enabled to perform parallel data processing in a distributed environment with affordable cost and reasonable time.

This study presents data processing by taking advantage of MapReduce framework in both stand-alone and cloud computing mode, as well as it shows the cloud computing mode as a distributed computing environment aims at large datasets to be processed in parallel more effectively on available computer nodes in a distributed cluster.

The first contribution of this work is to provide a comprehensive review of the MapReduce framework and its open-source implementation Hadoop and the second contribution is to present the evaluation of running a MapReduce program in both standalone and cloud computing mode based on the criteria of speed of processing, response time and cost efficiency.

The rest of the paper is organized as follows. Section II provides background information and definitions of the MapReduce programming model, its architecture and its open-source implementation Hadoop. Section III presents an experimentation of running a MapReduce program in both standalone and cloud computing mode. Section IV presents the criteria evaluation such as speed of processing, response time and cost efficiency for running the experiment in both standalone and cloud computing mode. Finally, section V concludes the paper.

2. MAPREDUCE PROGRAMMING MODEL

MapReduce has been produced by Google as a new programming model to process and analyse massive amounts of data. It uses for processing of large datasets across a cluster of machines in a distributed way. In these datasets, since the input data is too large, the computation units have to be distributed across thousands of machines within a running cluster. J.Dean and S.Ghemawat [4] from Google Inc. published a paper in 2004 describing MapReduce. Google never released their implementation of MapReduce, but the Apache Company made available a concrete implementation of MapReduce named Hadoop [5]. MapReduce by hiding the details of data distribution, and parallelization allows developers to perform complex computations in a simple way. The main feature of MapReduce is that it can both handle and process both structured and unstructured data across many working machines 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 framework consists of three major phases:

1. *Map phase*: this phase takes input file and partitions it into M Map functions, each Map function run in parallel. This phase produces the output of intermediate (key, value) pairs.
2. Shuffle and Sort phase: this phase groups and partitions the intermediate (key, value) pairs from previous phase by common values of the key. The number of partitions is equal to the number of reducers. After partitioning, each partition is stored by a key to combine all values for that key.
3. *Reduce phase*: finally, this phase takes all pairs for a given key and produces a new value for the same key which is the same output of the MapReduce system.

MapReduce is implemented in different types of programming languages, including, LISP, Java, C++, Python, Ruby and C [6]. A working flow of MapReduce system includes, firstly an input dataset is partitioned into several units of data then each unit of data is processed in Map functions in the Map phase and finally combined in Reduce functions in the Reduce phase to produce the final result. A Map function takes input pairs and produces a set of intermediate (key, value) pairs and passes those pairs to a Reduce function in order to combine all the values associated with the same key. A Reduce function takes 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 [7]. The main advantage of MapReduce is that it allows large computations to be easily parallelized and using re-execution of failed tasks as the primary mechanism for fault tolerance [8]. The other advantages of using MapReduce for data processing include:

- It hides the complexity of running distributed data processing systems across many computing nodes in the cluster.
- It handles the gathering and combining of the results from across the cluster to produce and return a single result.
- It allows developers to create their own MapReduce system with no need to have specific knowledge of distributed programming [9].
- It uses an independent data storage system.

2.1. Hadoop: An Implementation of MapReduce

Hadoop is an open-source, Java based implementation of MapReduce, which widely used for parallel and distributed computing on a cluster of servers. Hadoop was introduced in 2006 by Doug Cutting [5]. Hadoop has its own distributed file storage system called Hadoop Distributed File System (HDFS), HDFS as a distributed file system is used to store many blocks of data on a cluster of compute nodes for reliable and rapid computations. A simple Hadoop cluster consists of $n \geq 1$ machines running the Hadoop software [10]. Hadoop by utilizing HDFS as a shared file system provides data locality optimization within a cluster of working machines and presents each machine as a single machine to perform data processing. Input data firstly is fragmented into typical HDFS-sized block (64MB) and then smaller packets (64KB) by the user [11]. It means an input file into HDFS is divided into 64 MB blocks and each block is resided on a different working machine. HDFS supports write once, read many times concept, thus it is suitable for MapReduce systems to process large datasets. These systems write their data once, but they read it one or more times when required.

2.2. MapReduce Architecture

The MapReduce architecture representing the workflow framework to process large amounts of data by seven main components: Job Tracker, Task Tracker, Mapper, Combiner, Partitioner, Sort and Reducer are shown in Figure 1.

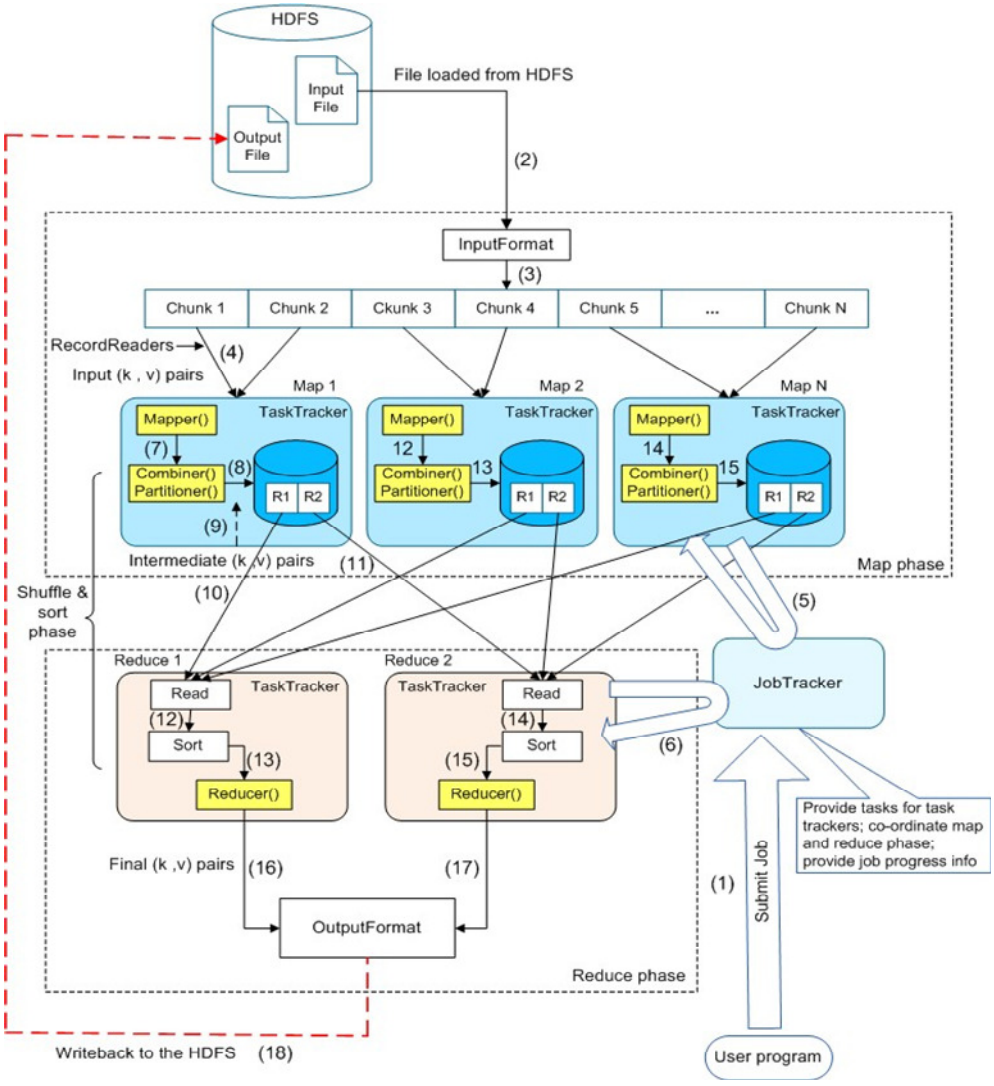

Figure 1. MapReduce system architecture for large-scale data processing

The legend for Figure 1 shows the workflow of the system:

(1) Job execution is started when a user uploads a MapReduce program jar file, after that jobtracker is notified about new submission of job.

(2) Input dataset is transmitted into the HDFS by master node then it loads from HDFS into Input Format component for processing.

(3) The Input Format located in Map phase splits the dataset into several chunks.

(4) Each chunk of dataset assigns to a Map task; each word in each chunk is one pair with two parameters (k, v) which k is the word and v is the value of the key.

(5) Job tracker notifies task trackers to perform a task in Map phase.

(6) Job tracker notifies task tracker to perform a task in Reduce phase.

(7) Record reader in Map phase reads idle chunks of data and sends them to the written Mapper function.

(8) The output of Mapper sends to Combiner and then Partitioner functions. The results of these two functions are stored into a database.
(9) The output of the Combiner and Partitioner functions are in the format of the intermediate (k, v) pair.
(10) The output of each Map task is read by task tracker to continue processing in Reduce phase.
(11) The output of each Map task is read by task tracker to continue processing operations in Reduce phase.
(12) The output of the first Map task is read in Reduce phase then it is sent to a Sort function in order to be sorted in a sequential format.
(13) The output of the sort function goes to the first Reducer function to produce the final result.
(14) The output of the second Map task is read in Reduce phase then it is sent to the Sort function in order to be sorted in a sequential format.
(15) The output of the sort function enters to the second Reducer function to produce the result.
(16) Reducer function in Reduce Task 1 produces the Final (k, v) pairs of words and sends them to the standard output format.
(17) Reducer function in Reduce Task 2 produces the Final (k, v) pairs of words and sends them to the standard output format.
(18) The obtained result from Reduce phase is stored in HDFS

Components of the architecture:

- Mapper: Mapper takes a chunk of data as input and produces zero or more results as output. The results of Mapper are key/value pairs of each word in the dataset.
- Job Tracker: It allocates a Map task or a Reduce task to the idle task trackers. The job tracker specifies proper jobs for task trackers based on how busy they are. When the job tracker does not receive anything from any task tracker nodes for a period of time, it automatically assumes the task tracker node has been failed and re-executes the failed tasks to other task tracker nodes.
- Task Tracker: It performs assigned jobs by job tracker, they continually inform job tracker about their status for.
- Combiner: all key/value pairs from Mapper are combined together and passed to the Partitioner.
- Partitioner: put same words in each part and produce intermediate key/value pairs of those words.
- Sort: the outputs from Map phase are sent to this component for being sorted in a sequential format.
- Reducer: the purpose of a Reducer is to combine the output of many Mappers into one result.

3. EXPERIMENTATION

Linux environment as a distributed operating system is suitable to develop the MapReduce program, it is the only supported production platform to run Hadoop applications; MapReduce system could not be completely implemented on Windows due to the problem of the connection to a distributed cluster. Therefor Ubuntu 10.04 LTS is selected as a development environment. LTS stands for Long-Term Support. The program is developed by using Java's programming language. For running and testing the MapReduce program in both standalone and cloud

computing mode, Hadoop needs to be installed on Ubuntu 10.04 LTS. In addition, Elastic Compute Cloud (EC2) command-line tools need to be installed on Ubuntu for interacting with instances directly in a cloud environment; this tool is an interface between the developer and cloud environment. For the experiment, a MapReduce program is written in Java to count the number of occurrences of each word in the given input dataset. To run this MapReduce system in standalone and cloud computing mode, an installation of Hadoop on Ubuntu Linux is required. Hadoop is supported on Linux as a development and production platform especially for distributed operations. Hadoop is configured to run in two supported mode:

- Standalone: in this mode Hadoop makes a single node cluster on the local machine.
- Cloud computing mode: in this mode Hadoop makes a cluster of machines in a fully distributed environment.

The MapReduce program will be run via Hadoop single node cluster for standalone mode and Hadoop cluster of machines for cloud computing mode; these two modes are described in following sections.

3.1. Standalone Mode

In standalone mode, a local machine or laptop is used, which at least has CPU1.6 GHz 1.2G RAM of memory and 40G of hard disk space. In order to run the MapReduce program in standalone mode, after installing the Ubuntu 10.04 LTS (Long-Term Support), a single node Hadoop cluster must be initiated. This system processes a dataset and generates a set of key and value pair of each word in the given input dataset. In the experiment, the system was tested with three different sizes of dataset, 2,000,000, 20,000,000 and 200,000,000 words respectively, and their execution time of processing data is shown in Table 1.

Table 1. Execution time of processing data in standalone mode.

Dataset	Size of Dataset	No. of Words	Time Taken (sec)
1	10 MB	2,000,000	56.5
2	100 MB	20,000,000	141.3
3	1 GB	200,000,000	1272.8

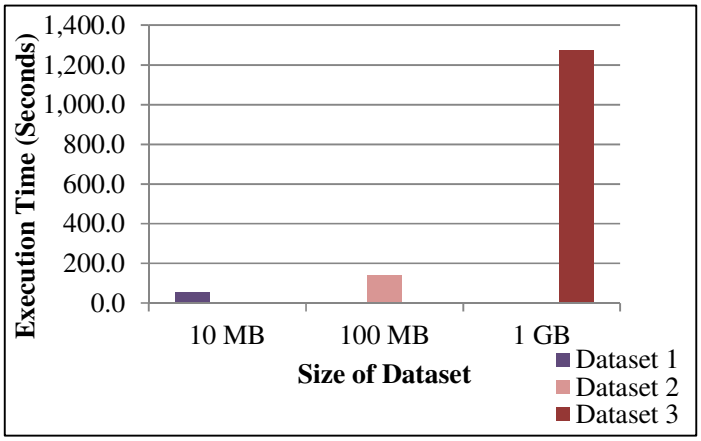

Figure. 1 Test results of processing data in standalone mode

In standalone mode, there is only one running machine that everything runs in a single java processor with storage using a local machine's standard file system. By comparing the processing of these three datasets, the time taken increases in each test by increasing the size of dataset or the number of words as shown graphically in Figure 2. It shows that the size of dataset or the number of words is a key parameter for time taken to process the datasets. The time taken of processing these three datasets is high in standalone mode because in this mode, Hadoop provides a non-distributed environment to process large datasets and it has the limitations of the single processor in this environment. In the next section, these three datasets are processed and tested in Amazon cluster in a cloud environment.

3.2. Cloud Computing Mode

In cloud computing mode, we need to set up a Hadoop cluster in order to interact with Amazon Web Services[1] such as Amazon S3, Amazon EC2 for running the MapReduce program in a fully distributed environment.

The Elastic Compute Cloud (EC2) and Simple Storage Service (S3) are the two main services for running Hadoop in the cloud computing mode. The Amazon EC2 service provides virtual computing machines for large dataset processing and the S3 service is a storage service which stores large datasets in clouds.

To run the MapReduce system in cloud computing mode requires a Hadoop cluster in Amazon EC2 to be launched. The cluster has one master instance and more than one worker instances. After starting the cluster, the MapReduce program is run in the cluster and counts the number of occurrence of each word in the given input dataset. In this cloud computing mode, the system was tested with three different sizes of dataset, 2,000,000, 20,000,000 and 200,000,000 words respectively, and their execution time of processing data as *instances* is shown in Table 2.

All three datasets are same as tested datasets in standalone mode. The MapReduce systems consist of two components: code and data. First, the code is moved to the master instance of the Hadoop cluster then the data from the local machine is moved into the cluster's HDFS through Amazon S3 after that the MapReduce system is run by submitting the MapReduce job in Hadoop cluster in Amazon EC2 cloud. It processes the dataset located in the input directory of HDFS and produce the final result in the output directory of HDFS.

Dataset 1 and dataset2 were tested with 2 running medium instances but dataset3 due to a large size was tested with two, four, six and eight numbers of running medium instances. Note that each medium instance consists of 2 EC2 Compute Unit with 3.75 GB memory and 410 GB storage.

Table 2. Execution time of processing data in cloud computing mode.

Time Taken (Sec)	No. of Instances	Dataset 1	Dataset 2	Dataset 3
	2	31.4	39.8	65.3
	4	-	-	44.8
	6	-	-	26.6
	8	-	-	5.2

According to the results listed in Table 2, Figure 3 illustrates graphically the test results of processing these datasets with two medium running instances in cloud computing mode.

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

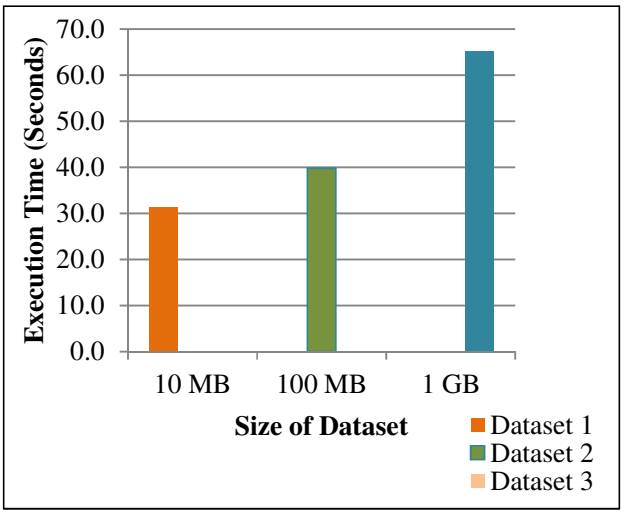

Figure. 3 Test results of processing data in cloud computing mode

In cloud computing mode, the MapReduce program is run in a fully distributed environment. There is more than one running instances in the cluster. One instance is designated as the master node and other instances as the worker nodes. In the experiment, the MapReduce system is run with two running medium instances for dataset1 and dataset2 and four numbers of running medium instances for dataset3, each instance in the cluster runs on a single machine, with storage using HDFS by default. The time taken is increasing slightly in each test by increasing the size of dataset. By comparing the processing of three datasets in Table 2 and Figure 3 shows, the time taken to process these datasets in cloud computing mode is smaller than the time taken to process these datasets in standalone mode. In cloud computing mode Hadoop provides a distributed environment to run the MapReduce system in parallel over the cluster of machines. It means the processing of large datasets in cloud computing mode is faster than the processing of large datasets in standalone mode.

4. EVALUATION

The evaluation of criteria for the implemented MapReduce system in both standalone and cloud computing mode, is resented in this section. It includes the response time, speed of processing and cost efficiency.

4.1. Response Time

Response time is one of the most important criteria in the MapReduce system evaluation. 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 both standalone and cloud computing mode. This response time in standalone mode is a constant value which may be changed in different local machines. In this experiment, the response time in standalone mode for three sizes of dataset is the same amount of time 5.5 seconds. The response time in cloud computing mode depends on how big is the scale of the Amazon EC2 cluster in a cloud environment. The scale of the Amazon EC2 cluster is specified with the number of running instances. Table 3 shows the response time in the different sizes of cluster in cloud computing mode.

Table 3. Response time in cloud computing mode.

No. of Instances in the Cluster	Response Time (Seconds)
2	15.7
4	25.4
6	36.2
8	47

By increasing the number of instances in the cluster the response time is respectively going up.

4.2. Speed of Processing

The speed of processing depends on the response time and the time taken to process datasets in both standalone and cloud computing mode. The formula of the speed of processing calculation is:

$$\text{Speed of Processing (Sec)} = \text{Response Time (Sec)} + \text{Time Taken (Sec)}$$

Above formula is written in short:

$$S_t\,(\text{sec}) = R_t\,(\text{sec}) + T_t\,(\text{sec})$$

Where St stands for the speed of processing, R_t stands for response time of the cluster and T_t stands for time taken to process datasets. Table 4 presents the results of the speed of processing in standalone mode.

Table 4. Results of the speed of processing calculation in standalone mode.

Dataset	S_t (Speed of Processing)
1	62 sec
2	146.8 sec
3	1278.3 sec

Based on the Table 4, Figure 4 illustrates graphically the speed of processing in standalone mode.

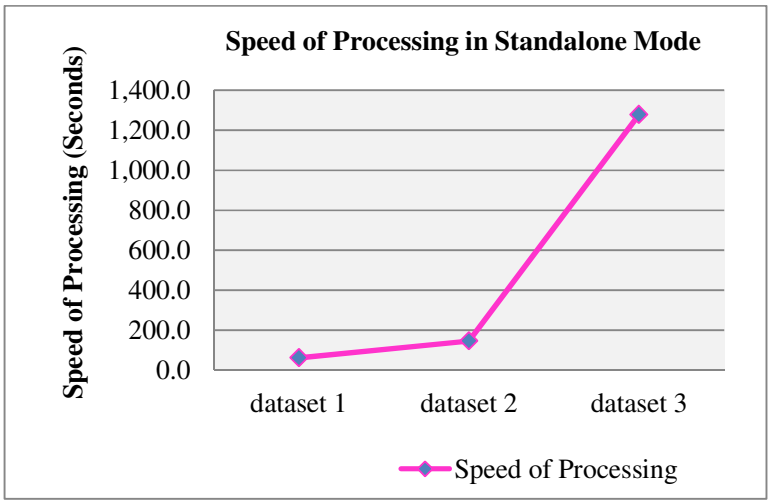

Figure. 4 Speed of processing in standalone mode

It is observed from Table 4 and Figure 4 that the running speed is going up by increasing the size of datasets. Table 5 presents the results of the speed of processing for three datasets in cloud computing mode.

Table 5. Results of the speed of processing calculation in cloud computing mode.

No. of Instances	S_t (Speed of Processing)		
	Datase1	Datase2	Dataset3
2	46.1 sec	54.5 sec	81 sec
4	-	-	70.2 sec
6	-	-	62.8 sec
8	-	-	52.2 sec

Based on this table, Figure 5 illustrates graphically the speed of processing for dataset 3 with different number of instances in cloud computing mode.

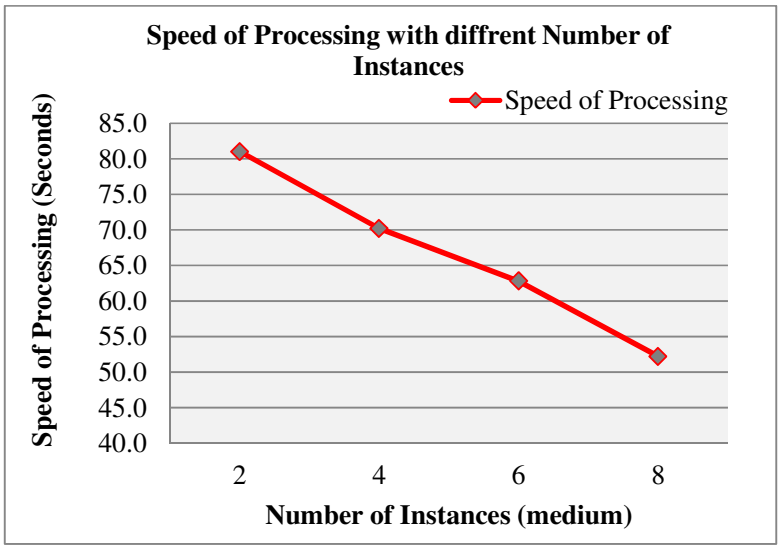

Figure 5. Speed of processing in cloud computing

It is observed from Table 5 and Figure 5 that the running speed is improving by increasing the number of instances in the cluster. By using scalable property of Amazon EC2 cluster, we take advantage of high speed processing in a fully distributed environment.

4.3. Cost Efficiency

Hadoop is a free software framework therefor, there is no need to pay for Hadoop installation in standalone mode. In this mode, for the experiment we run the Hadoop cluster in a local machine which all the specifications were described above. In cloud computing mode for 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 6 presents pricing model of the Amazon EC2 cluster for four numbers of medium instances.

Table 6. Pricing model of Amazon EC2 in cloud environment.

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

Based on the results of Table 6, Figure 6 illustrates the pricing model for medium instances.

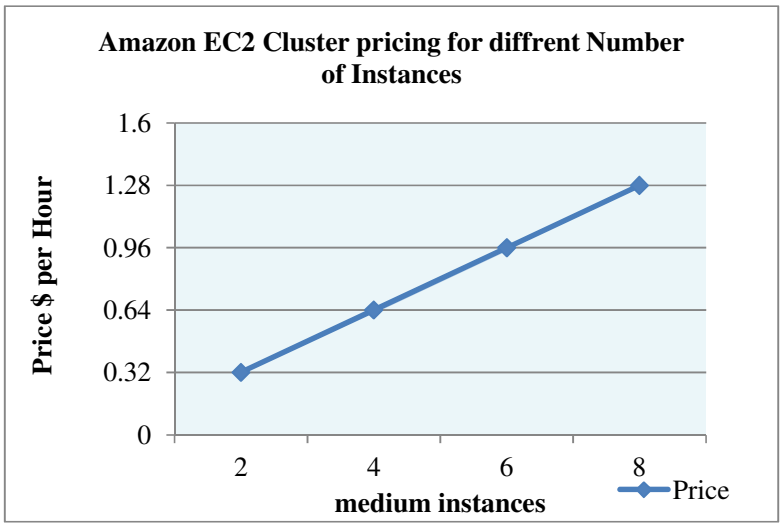

Figure. 6 Amazon EC2 cluster pricing for different number of instances

As it observed by increasing the number of instances the price is slightly rinsing.

5. CONCLUSION

In conclusion, this paper introduced MapReduce as a new programming framework to process large datasets; MapReduce provides automatic parallelism and distribution and a cost-effective mechanism for data processing, today most of the leading technologies for managing and processing their datasets are developed on MapReduce. We described MapReduce and its open-source implementation, Hadoop to perform automatic parallelism and distribution over a cluster of machines. This paper comprehensively discussed the evaluations of an implemented MapReduce system in both standalone and cloud computing mode. The system was implemented by using java's programming language and MapReduce framework, which enable to perform parallel and distributed data processing. The architecture is suitable for running MapReduce in both standalone and cloud computing mode. However, the implemented system performs data processing very fast with the highest end supercomputers available on Amazon Web Services with an extremely low budget in a cluster. The MapReduce system can be massively parallelized in a cluster of machines. It also utilizes a combination of MapReduce framework and cloud computing as an attractive propositioning for huge data processing. From our analysis using cloud computing and MapReduce together improves the speed of processing and decreases the response time and cost for processing of large datasets.

REFERENCES

[1] Amazon, EC2, (accessed January 2012), Available online at http://aws.amazon.com/ec2

[2] Amazon, Elastic MapReduce, (accessed January 2012), Available online at http://aws.amazon.com/elasticmapreduce

[3] A Babu, (2010) "Towards Automatic Optimization of MapReduce Programs", *ACM symposium on Cloud computing* Vol. 10, pp137-142.

[4] A 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.

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

[6] 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, pp 105-129.

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

[8] S.Sakr, A.Liu, D.M. Batista and M.Alomari, (2011) "A Survey of Large Scale Data Management approaches in Cloud Environment", *IEEE Communication surveys and Tutorials*, ISSN: 1553-877X, pp1-26.

[9] J.Dean, (accessed April 2011), "Experience with MapReduce, An Abstraction for Large-Scale Computation", Google Inc. Available online at www.slideshare.net/rantav/introduction-to-map-reduce

[10] 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.

[11] 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.

[12] K. Talattinis, A. Sidiropoulou., K. Chalkias, (2010) "Parallel Collection of Live Data Using Hadoop", *Informatics (PCI). IEEE Transaction on* Vol. 3, No. 11, pp66-71.