วารสารวิชาการ วิศวกรรมศาสตร์ ม.อบ.
UBU Engineering Journal

บทความวิจัย

A Comparison of Hadoop Distributions – Cluster Installation and Management Aspects

Araya Florence[1]* Thanisa Numnonda[2]

[1] Department of Electrical and Electronic Engineering, Faculty of Engineering, Ubon Ratchathani University

[2] Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang

* Corresponding author.

E-mail: araya.f@ubu.ac.th; Telephone: 0 4535 3331

Received 19 October 2017; Revised #1 8 February 2018; Revised #2 5 May 2018; Accepted 31 May 2018

Abstract

Big data is one of most promising technology which works with Cloud computing and Internet of Everything. Every second, data generated from billions of devices are sending to the cloud to be analysed and probably used for prediction or prevention in various applications. Big data platform is a foundation of its implementation to provide an ecosystem that data can be imported, processed and exported. This article reports a comparison of three different platforms; Apache Hadoop, Cloudera (Express), and Hortonworks in the aspects of stability, installation and cluster management. Apache Spark was chosen to test processing of all three distributions since it is ten times faster than Hive and 100 times faster than MapReduce. In addition, HiBench was chosen to be used as a testing benchmark, results were previously reported in [1]. In the aspects of cluster management, commercial based distributions are more likely to offer a better tool for installation and cluster management while Apache Hadoop is robust but lacking manageability.

Keywords

Apache Hadoop; big data; cluster installation; cluster management

1. Introduction

In the digital era, data and information are generated, collected, and analysed intensely. The amount of data created in total for each and every devices is enormous. These data are produced in high velocity and they can be structured or unstructured coming from various sources. The three Vs defined volume, velocity, and variety are referring to characteristics of big data. Analysing this data can be useful for any organizations to better target clients or improve their services. To be able to analyse this kind of data, big data platform should be implemented as an information infrastructure to serve big data requirements. The big data platform is used for gathering, preparing, and managing this kind of data for further analysis, including machine learning implementation for both prediction and possibly prevention. Choosing the platform is selecting a foundation for the system and defining the ecosystem of big data analysis including tools for data analysis such as MapReduce, Hive, Mahout, Impala or Spark.

In this research, three distributions were set up in similar requirements using Amazon Web Services cloud platform. Each cluster consisted of 5 nodes, which are one manager, one namenode and 3 datanodes. The namenode was m3.2xlarge while the other nodes of the cluster were m3.xlarge EC2s. The benchmark test was executed using HiBench [2,10], a realistic and comprehensive big data benchmark suite for Hadoop originally developed by IBM. HiBench consists of three sets of Hadoop programs, including Micro-Benchmarks, Web Search, and Machine Learning. Three categories of test conducted in this research were MapReduce, Hive, and Spark. For Micro-Benchmarks groups, this research used HadoopJoin (Hive), JavaSparkJoin, ScalaSparkJoin, and PythonSparkJoin programs. Then, for Web Search groups, HadoopPageRank (MapReduce), JavaSparkPageRank, ScalaSparkPageRank, and PythonSparkPageRank were used. Finally, for Machine Learning groups, HadoopKmeans (Mahout), JavaSparkKmeans, ScalaSparkKmeans, and PythonSparkKmeans were used. All of these programs were executed to evaluate the system performance in term of processing speed, throughput, resource utilization, and data patterns.

2. Related Theories

Hadoop [3] is one of Apache’s opensource projects for store and manage big data. Hadoop is written in Java and has ability to support fault tolerance for storing data in more than one place. Hadoop system normally runs on many commodity servers in horizontal scale mode. Hadoop project was originally started by Doug Cutting and Mike Cafarella [4] when they were working in Yahoo. Later on, many companies started to use Hadoop including eBay, Facebook, and Amazon. At the time of conducting this research, Hadoop 2.5 was used, however, the latest stable version is 3.1.0 [5]. Normally Hadoop technology is run on a group of servers consisted of at least one Master node and many Worker nodes which will be used for processing and storing data. These worker nodes can be used as Data Node or node management (Node Manager). There are a few popular Hadoop distributions in the market such as Cloudera, MapR, and Hortonworks. This article reports a comparison of three different big data platforms; Apache Hadoop, Cloudera (Express), and Hortonworks (HDP) in the aspects of installation, cluster management, cost, and stability.

In any distribution, Hadoop ecosystem mainly comprised of HDFS as a filesystem, MapReduce as a software framework and YARN as a resource negotiator. These are not enough for further data analysis which required SQL queries, random access read/write data and data management including transferring data to and from RDBMS or retrieving streaming data. Additional tools such described are Hive, Spark, Pig, and Impala. In this research, we focused on HDFS, YARN, Hive, and Spark.

HDFS [6], Hadoop Distributed File System is the most popular technology for unstructured data storage because Hadoop stores data in commodity server's storage in distributed fashion. Being distributed means that its data in small blocks are duplicated at least 3 copied and are stored in different nodes which is auto-fault tolerance. This characteristic makes HDFS a secured storage system.

MapReduce [7] is the main processing framework for older Hadoop version which processes data in batch model. Java programs need to be developed to use MapReduce for processing data in HDFS. In a newer version of Hadoop, Spark was introduced and predicted to replace MapReduce as data processing nowadays can be interactive, real-time processing and even machine learning, not only batch processing anymore.

YARN, Yet Another Resource Negotiator is a resource manager on Master node in Hadoop system, the main duty is distributing jobs to worker nodes via Node manager. The processing mode can be MapReduce in batch, Tez in interactive mode or Spark in real-time.

Hive is a simple SQL like language tool for query data in HDFS to bypass Java program development for using MapReduce. However, Hive actually translate SQL like queries to be MapReduce which is then processed in batch mode.

Mahout is a predictive analytic tool for data scientist using Java which offering various predictive algorithms such as recommender, classification, and clustering.

Spark [8] is a big data real-time data processing technology using in-Memory processing mode. Spark is the most prominent framework in big data processing tools as its processing time is a lot faster than MapReduce and it is less complicated to work with because of its rich APIs. Spark offers importing data from various sources including HDFS, Cloud storage, and NoSQL in different languages those are Java, Scala, R, and Python.

Pig is very similar to Hive, Pig Latin script can be used to query data on HDFS without Java program to use MapReduce. However, Pig is suitable for ETL for dealing with JSON.

Impala is an SQL like language which similar to Hive but work faster than Hive as Impala works in interactive mode using its Statestore at Master node then distributing jobs to Worker nodes through Impala daemon (Impalad). Impala itself is developed by C++ and only available on Cloudera distribution.

One important requirement of the big data platform is the ability of processing data from/to other sources that is not HDFS. Processing data on HDFS can be classified as four types those are interactive analysis, batch analysis, real-time analysis and machine learning. When choosing big data platform, the main concern should be both using Hadoop with data on HDFS and processing data from other sources using Spark.

2.1 Processing data with Hadoop

When analysing data on Hadoop, MapReduce is used and completed in batch processing fashion which required Java programming or some other languages. Recently, the popularity of MapReduce has been declined as being replaced by Spark. However, some Hadoop distributions offer analytic tools with SQL like language/script, these tools are Hive, Pig and Impala.

2.2 Processing data with Apache Spark

Spark can process data on HDFS ten times faster than MapReduce. It can be used as a tool to import data from other types/sources such as cloud storage, NoSQL, RDBMS. Spark has ability to be operated as standalone or distributed mode over Hadoop cluster on YARN as interactive scheme. Four main components of Spark are Spark Core, Spark Streaming, Spark SQL, and MLib. Spark Core provided processing platform API for various languages including Java, Scala, Python, and R. Spark Streaming offered real-time processing which is suitable for importing real-time feeds of data. Spark SQL is an SQL like processing platform while MLib is a machine learning processing tools.

3. Methodology

Three main comparison aspects in research methodology being discussed here are cluster installation, benchmark test using HiBench and cluster management. In this paper, installation and cluster management are focused. The benchmark test results are reported in previous publication [1].

3.1 Cluster installation

All three distributions being investigated in this research were installed in AWS cloud service using four EC2s [9], each distribution consisted of one as a master node with other three as worker nodes. Each EC2 was Ubuntu server 14.04 LTS, the master was m3.2xlarge (8 vCPU, 30 GB RAM, 160 GB SSD, 500 GB Storage) and the worker nodes were m3.xlarge (4 vCPU, 15 GB RAM, 80 GB SSD, 500 GB Storage). This configuration is a minimum of Hadoop Cluster as Hadoop normally create at least 3 replications of data and distributed to each node [5].

3.2 HiBench tests

A simple test of importing data into HDFS and exporting data out were tested using both command line and over HUE. Data processing was tested on MapReduce, Hive, Pig, and Spark. HiBench was then used as a benchmark test, HiBench consisted of three groups, those are: -

- Micro Benchmark, SQL query was tested using Join for Hive and Spark
- Web Search, PageRank algorithm was tested on both MapReduce and Spark
- Machine Learning, K-Means algorithm was tested on Mahout and Spark.

3.3 Cluster management

In term of cluster management, we tested how difficult for each distribution to add or delete a node to and from a cluster. Another EC2 was created and added into the cluster then one node was removed from the cluster. After adding a new node to the cluster, configurations for services on the new node were observed. Before removing any node, services must be stopped completely for that node to be allowed to be removed. Adding and removing nodes can be challenging in the real situation when a node need a replacement.

4. Results

According to three comparison aspects discussed in methodology section, results can be divided into three parts those are cluster installation, HiBench testing results and cluster management.

4.1 Cluster installation

Hadoop distribution was the most difficult on in all three distributions tested. It required a knowledgeable staff to conduct a cluster installation using pure command line. The challenging work of installing each node for the cluster was node and services configuration. The installer needed to know all related services available for the version of installation as some services are not compatible with others in specific versions. The order of services installation is also crucial which make the process of installing the pure Hadoop distribution is an exhausting job. Apache Hadoop ecosystem installed is shown in Fig. 1. While the installation and configuration on Cloudera (Express) and Hortonworks were quite simple with a well-designed graphic user interface (GUI).

Cloudera or CDH offered a set of services called parcels so the installer can choose from Cloudera Manager to install any related services using parcels without having to know which chaining required services. Cloudera Manager installation interface in Fig.2 showing an option for installing CDH as a package meaning that users can choose related services by themselves without using available parcels provided by Cloudera. Once the parcel was chosen, big data engineers can also choose to add additional services using Cloudera Manager to simplify the cluster management. However, Hortonworks installation was different from the other two, the installation was conducted using CloudBreak and at least 9 nodes using EC2s on AWS. The user interface of CloudBreak which is made available by Hortonworks for installing

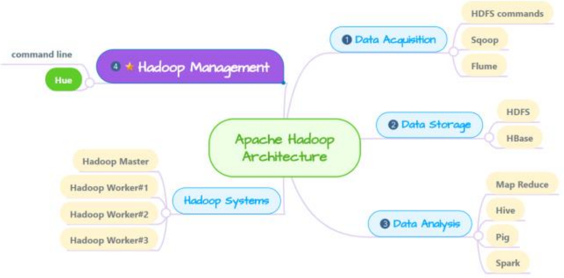
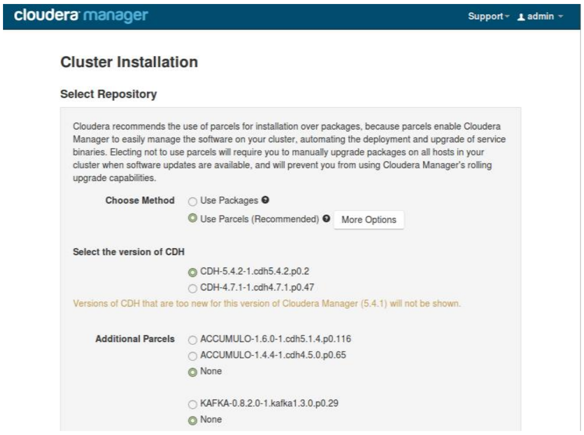


Fig. 1 Apache Hadoop installed components



Cluster Installation

Select Repository

Cloudera recommends the use of parcels for installation over packages, because parcels enable Cloudera Manager to easily manage the software on your cluster, automating the deployment and upgrade of service binaries. Electing not to use parcels will require you to manually upgrade packages on all hosts in your cluster when software updates are available, and will prevent you from using Cloudera Manager's rolling upgrade capabilities.

Choose Method ○ Use Packages
● Use Parcels (Recommended) More Options

Select the version of CDH

● CDH-5.4.2-1.cdh5.4.2.p0.2
○ CDH-4.7.1-1.cdh4.7.1.p0.47

Versions of CDH that are too new for this version of Cloudera Manager (5.4.1) will not be shown.

Additional Parcels ○ ACCUMULO-1.6.0-1.cdh5.1.4.p0.116
○ ACCUMULO-1.4.4-1.cdh4.5.0.p0.65
● None

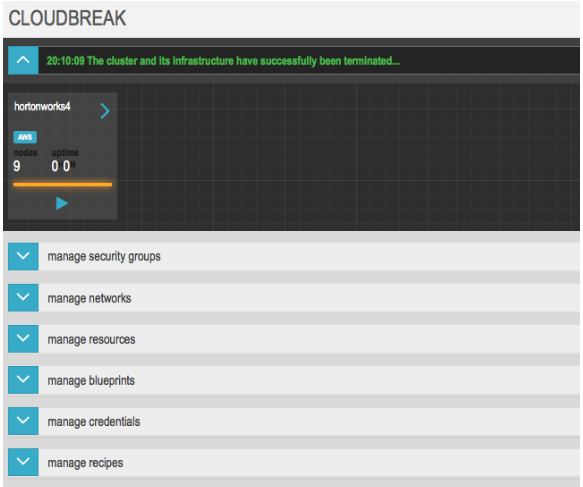
○ KAFKA-0.8.2.0-1.kafka1.3.0.p0.29
● None

Fig. 2 Choosing parcels for CDH installation

HDP on public cloud is shown in Fig. 3. Like CDH, the installation process is straightforward on its GUI.

4.2 HiBench testing results

According to the installation of Hortonworks that required 9 nodes in total and some difficulty of YARN configuration, therefore, the results from Hortonworks distribution cannot be used in comparison here. Expected results are small processing time with high throughput.


CLOUDBREAK

20:10:09 The cluster and its infrastructure have successfully been terminated...

Fig. 3 Hortonworks's CloudBreak UI

A comparison of processing time and processing throughput between Hadoop 2.6 and CDH is shown in Fig. 4 and 5 accordingly. Interestingly, microbenchmark test results, only Hive on CDH can be processed faster than Hadoop. At machine learning groups, CDH actually gave a better throughput over Hadoop on Mahout. Processing time of each testing program using CDH, when processed by Spark, Hadoop is faster. However, both CDH and Hadoop, Spark confirmed a better speed over traditional tools offering by Hadoop technology. Similarity shown in throughput, Hadoop offered a better throughput in most run results. Spark was found to be ten times faster than Hive and 100 times faster than MapReduce [9].

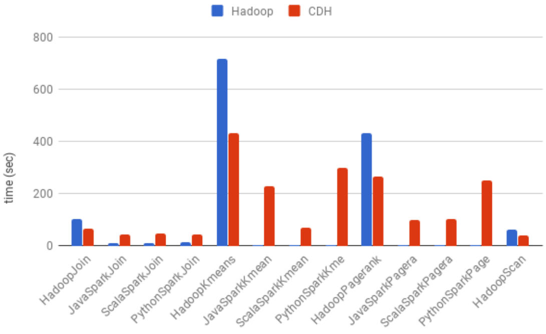


Fig. 4 Processing time using Hadoop versus CDH

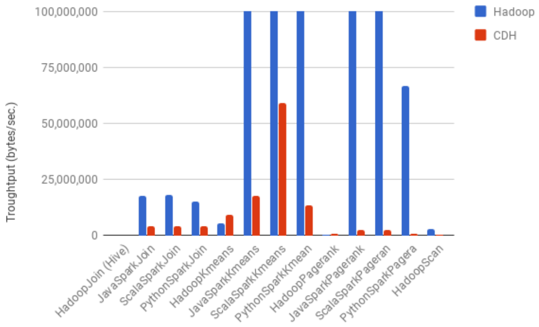


Fig. 5 Processing throughput using Hadoop versus CDH

4.3 Cluster management

Managing Hadoop cluster can be achieved with a simple GUI in some Hadoop distributions, however, extra configurations sometimes needed for additional services adding to the cluster therefore cluster management is a significant factor when choosing distribution to deploy as a foundation of big data. For Apache Hadoop, there is no GUI for cluster management, hence, it is the main lacking ability to manage clusters easily. Only a simple HUE for using

HDFS on GUI was available in Apache Hadoop. On the other hand, CDH offered Cloudera Manager shown in Fig. 6 for cluster management and monitoring while Ambari shown in Fig. 7 was available on Hortonworks as a dedicated cluster manager application.

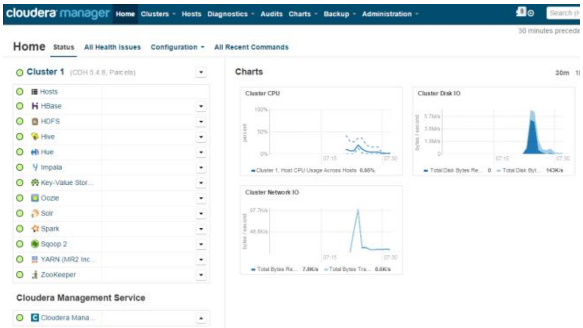


Fig. 6 Cloudera manager for cluster and node management

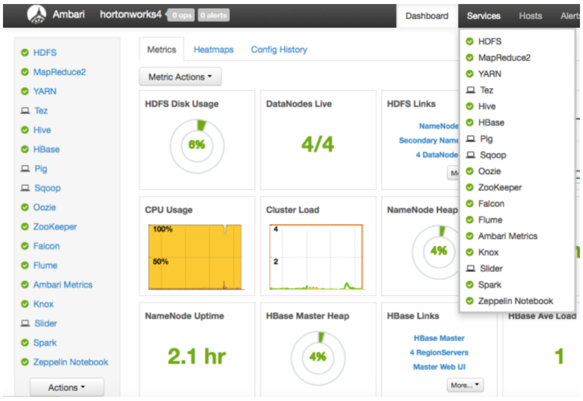
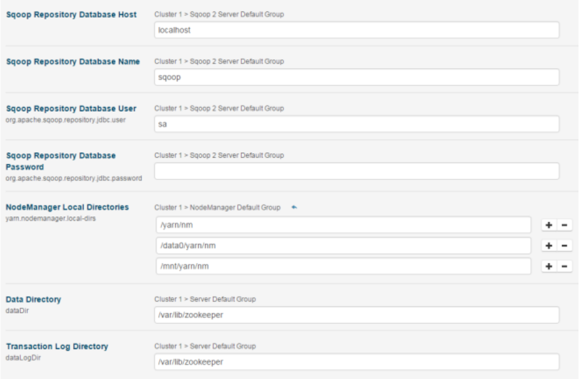


Fig. 7 Ambari—Hortonworks's cluster manager

Cluster management is a vital factor when implement a platform, especially when several nodes in the cluster is growing. GUI cluster manager can be helpful when merging cluster, adding/deleting nodes and very useful for monitoring purpose.

Adding node to an existing cluster for Apache Hadoop was complex and complicated as related services required to be stopped before putting in an additional node to be used on the service. The new node need to be preconfigured before restarting a series of services and the cluster. Adding nodes in CDH and Hortonworks were easily done over GUI using Cloudera Manager for CDH and Ambari on Hortonworks.



Sqoop Repository Database Host — Cluster 1 > Sqoop 2 Server Default Group: localhost

Sqoop Repository Database Name — Cluster 1 > Sqoop 2 Server Default Group: sqoop

Sqoop Repository Database User (org.apache.sqoop.repository.jdbc.user) — Cluster 1 > Sqoop 2 Server Default Group: sa

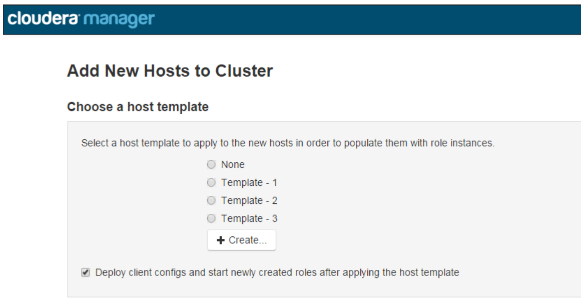
Sqoop Repository Database Password (org.apache.sqoop.repository.jdbc.password) — Cluster 1 > Sqoop 2 Server Default Group:

NodeManager Local Directories (yarn.nodemanager.local-dirs) — Cluster 1 > NodeManager Default Group: /yarn/nm; /data0/yarn/nm; /mnt/yarn/nm

Data Directory (dataDir) — Cluster 1 > Server Default Group: /var/lib/zookeeper

Transaction Log Directory (dataLogDir) — Cluster 1 > Server Default Group: /var/lib/zookeeper

Fig. 8 Changing configuration of services on GUI



Add New Hosts to Cluster

Choose a host template

Select a host template to apply to the new hosts in order to populate them with role instances.

○ None
○ Template - 1
○ Template - 2
○ Template - 3
+ Create...

☑ Deploy client configs and start newly created roles after applying the host template

Fig. 9 Adding new node to a cluster using template

When it comes to configuration of services, on Apache Hadoop only can be achieved on command line while commercial distributions like CDH or HDP, the configuration of newly installed cluster or modification of available services can be done using GUI as shown in Fig. 8.

New nodes can be EC2 and pre-installed operating system before adding to the cluster. Fig.9 shows a new node being added into existing cluster on Cloudera Manager. A new node can be added using node template or just a clean node that service roles can be decided later.

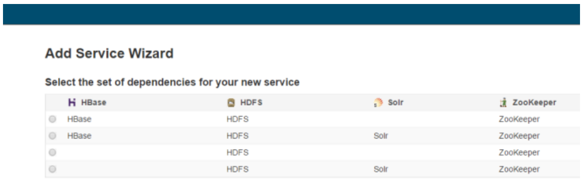
Once new node installed, node's role can be chosen. In Apache Hadoop, each service must be configured individually and the cluster administrator needs a deep understanding of related services on the cluster. In Cloudera Manager as an example of GUI cluster manager, setting up a role for a new node is simple, related services for each role available to be called as shown in Fig. 10.

Deleting nodes from the cluster should be conducted carefully under circumstances that the related services already stopped. Thus, on cluster management aspect, CDH and Hortonworks are suggested for less-skilled administrators while Apache Hadoop required a well Unix knowledgeable staff.

In a heterogeneous Hadoop cluster in Cloud service, using Cloudera Manager can be useful in monitoring node status and setting a template for different nodes when adding a new one. When using heterogeneous cluster, parameters to be adjusted and considered are number of map reduce tasks, data locality, replication, block reports and heartbeats [10]. Since the information can be gathered and calculated, most parameters can be configured easily with the Cloudera Manager GUI.

5. Conclusion

The results showed that Cloudera Express was the easiest in installation and cluster management aspects. However, Hortonworks also provided with graphical user interface for cluster maintenance using open source called Ambari. Being light and flexible, Apache Hadoop was recommended with some tradeoffs such as complexity of command line uses. In all three distributions, Spark was highly shown enormous potential to be chosen as a processing platform for big data. The results of Micro-Benchmark tests, both speed and throughput of using Spark were 10 times better than Hive. The PageRank testing results confirmed Apache Spark (any languages) were 100 times faster than MapReduce. Equivalent results for K-Means of Machine Learning testing group, Apache Spark were exceptional superior over Mahout, especially Scala Spark.



Add Service Wizard

Select the set of dependencies for your new service

	HBase	HDFS	Solr	ZooKeeper
○	HBase	HDFS		ZooKeeper
○	HBase	HDFS	Solr	ZooKeeper
○		HDFS		ZooKeeper
○		HDFS	Solr	ZooKeeper

Fig. 10 Adding a service using wizard

References

[1] Florence A, Numnonda T. A comparison of apache hadoop distributions using HiBench. In: *22nd International Symposium on Artificial Life and Robotics*. Japan; 2017. p. 218–222.

[2] Huang S, Huang J, Dai J, Xie T, Huang B. The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis. In: *2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*. 2013. p. 41–51.

[3] Holmes A. *Hadoop in Practice, 1st Edition*. Manning Publications; 2012.

[4] Julio P. Big data Analytics with Hadoop, Available from: http://www.slideshare.net/PhilippeJulio/hadoop-architecture [Accessed August 2017].

[5] Apache, Apache Hadoop Releases. Available from: http://hadoop.apache.org/releases.html [Accessed August 2017].

[6] Apache, HDFS Available from: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html [Accessed August 2017].

[7] Wikipedia, MapReduce Available from: https://en.wikipedia.org/wiki/MapReduce [Accessed August 2016].

[8] Apache, Spark Available from: http://spark.apache.org/ [Accessed August 2016].

[9] Holoman J, O'Dell K. *How-to: Deploy Apache Hadoop Clusters Like a Boss*. 2015.

[10] Thirumala Rao B, et al. Performance Issues of Heterogeneous Hadoop Clusters in Cloud Computing. *Global Journal of Computer Science and Technology*. 2012; Volume XI Issue VIII May 2011.

[11] Gu R, et al. SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters. *J. Parallel Distrib. Comput*. 2014; 74 (2014): 2166–2179.