TOWARDS PROVIDING HADOOP STORAGE AND COMPUTING AS SERVICES

BY

SHAWQI MOHAMMED AL-MALIKI

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

DECEMBER 2017

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Shawqi Mohammed Al-Mulaiki** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE.**

[illegible]

Dr. Farag Azzedin
(Advisor)

[illegible]

Dr. Khalid Al-Jasser
Department Chairman

[illegible] 8/5/2018

Dr. Mohammad Alshayeb
(Member)

[illegible]

Dr. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

[illegible]

Dr. Sami Zhioua
(Member)

15/5/18

Date

Dedication

I dedicate this work to my parents, my wife, and my kids: Ahmed, Areej, and Abdulrahman.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Allah Almighty who granted me the patience, strength, and motivation to accomplish this work.

I wish to express my appreciation to my advisor Dr. Farag Azzedin for his persistent support. His suggestions, comments, and encouragement will never be forgotten.

Thanks are due to my thesis committee members Dr. Mohammad AlShayeb and Dr. Sami Zhioua for their cooperation and support.

Thanks to my colleagues and friends, especially Mustafa Ghaleb for his consistent help and support.

Last but not least, I would like to thank my parents for providing all kinds of help, and my wife for her encouragement and endless support.

TABLE OF CONTENTS

ACKNOWLEDGMENTS III

TABLE OF CONTENTS IV

LIST OF TABLES VI

LIST OF FIGURES VII

LIST OF ABBREVIATIONS VIII

ABSTRACT IX

ملخص الرسالة X

CHAPTER 1 INTRODUCTION 1

1.1 CLOUD COMPUTING 2
1.2 CURRENT HADOOP ARCHITECTURE 3
1.3 PROBLEM STATEMENT AND MOTIVATION 4
1.4 RESEARCH OBJECTIVES AND CONTRIBUTIONS 5

CHAPTER 2 RELATED WORK 7

2.1 HADOOP 7
2.1.1 MAPREDUCE 10
2.1.2 HDFS 11
2.2 COMPARING DIFFERENT HADOOP CLUSTERS 13
2.2.1 SAME DEPLOYMENT ENVIRONMENT 13
2.2.2 DIFFERENT DEPLOYMENT ENVIRONMENTS 14
2.3 SOA-BASED HADOOP SEPARATED MODELS 15
2.4 LITERATURE REVIEW SUMMARY 16

CHAPTER 3 CLOUD-BASED PROPOSED MODEL 18

3.1 LAYERED ARCHITECTURE FOR BIG DATA SOLUTIONS 19
3.1.1 PHYSICAL LAYER 19
3.1.2 VISUALIZATION LAYER 19
3.1.3 CORE COMPONENTS LAYER 20
3.1.4 APPLICATION LAYER 23
3.2 HADOOP PROPOSED MODEL 24
3.2.1 BIG DATA SOLUTION 26
3.2.2 VIRTUAL MACHINES PREPARATION 27
3.2.3 MAPPING THE BIG DATA SOLUTION WORKERS INTO THE PREPARED VMS 28

CHAPTER 4 PERFORMANCE EVALUATION 29

4.1 EXPERIMENTAL ENVIRONMENT 29
4.1.1 PHYSICAL COMPONENTS 30
4.1.2 VIRTUALIZATION COMPONENTS 31
4.1.3 DEPLOYING HADOOP CLUSTER CORE COMPONENTS 33

4.1.4 BENCHMARK SUITE 37
4.2 EXPERIMENTAL METHODOLOGY 38
4.3 RESOURCE DISTRIBUTION 40
4.4 HADOOP EXPERIMENTAL SCENARIOS 41
4.5 RESULTS AND EVALUATION 45
4.5.1 PERFORMANCE EVALUATION METRICS 46
4.5.2 WORDCOUNT 47
4.5.3 ENHANCED DFSIO 53
4.6 CONCLUDING REMARK 68
4.7 PROPOSED MODEL LIMITATIONS AND TRADE-OFFS 68

CHAPTER 5 CONCLUSION AND FUTURE WORK 70

5.1 CONCLUSION 74
5.2 FUTURE WORK 74

REFERENCES 76

VITAE 84

LIST OF TABLES

TABLE 1 SUMMARIZED LIMITATIONS OF RELATED WORK 17
TABLE 2 PHYSICAL HOST CHARACTERISTICS 31
TABLE 3 ALLOCATED RESOURCES FOR HYPERVISOR' HOST MACHINE 43
TABLE 4 ALLOCATED RESOURCES FOR THE HADOOP MASTER VMS 43
TABLE 5 ASSIGNED RESOURCES FOR HADOOP WORKERS IN DIFFERENT EXPERIMENTAL SCENARIO 45
TABLE 6 BENCHMARK SELECTED WORKLOAD CHARACTERISTICS 46
TABLE 7 METRICS USED IN EVALUATING THE PERFORMANCE OF HADOOP TWO MODELS 47
TABLE 8 WORDCOUNT COMPLETION TIME (SECONDS) 47
TABLE 9 WORDCOUNT NORMALIZED COMPLETION TIME 48
TABLE 10 COMPARISON OF NORMALIZED COMPLETION TIME FOR WORDCOUNT (%)....... 49
TABLE 11 COMPLETION TIME VALUES WHEN SCALING THE DATA-SIZE OF THE WORKLOAD (WORDCOUNT).. 51
TABLE 12 COMPLETION TIME OF DFSIO-READ- FIXED NUMBER OF FILES (512) & VARIATION OF FILE-SIZE 55
TABLE 13 COMPLETION TIME OF DFSIO-READ- FIXED NUMBER OF FILES (1024) & VARIATION OF FILE-SIZE 56
TABLE 14 COMPLETION TIME OF DFSIO-READ- FIXED FILE-SIZES (400MB) & VARIATION OF NUMBER OF FILES 56
TABLE 15 DFSIOE-READ COMPLETION TIME (SECONDS).. 57
TABLE 16 DFSIOE-READ NORMALIZED COMPLETION TIME (SECONDS) 57
TABLE 17 COMPARISON OF NORMALIZED COMPLETION TIME FOR DFSIOE-READ (%).... 58
TABLE 18 COMPLETION TIME VALUES WHEN SCALING THE DATA-SIZE OF THE WORKLOAD (DFSIOE-READ) 59
TABLE 19 DFSIOE-WRITE COMPLETION TIME (SECONDS) 62
TABLE 20 DFSIOE-WRITE NORMALIZED COMPLETION TIME 62
TABLE 21 COMPARISON OF NORMALIZED COMPLETION TIME FOR DFSIOE-WRITE (%) .. 63
TABLE 22 COMPLETION TIME VALUES WHEN SCALING THE DATA-SIZE OF THE WORKLOAD (DFSIOE-WRITE) 65
TABLE 23 CORRELATION OF NATIVE AND PROPOSED MODES WITH DATA-SIZE 67
TABLE 24 CORRELATION OF DATA-SIZE WITH NATIVE AND PROPOSED MODELS 67
TABLE 25 SUITABILITY RECOMMENDATIONS FOR PROPOSED MODEL USAGE 68
TABLE 26 THE NATIVE MODEL VS THE PROPOSED MODEL 71

LIST OF FIGURES

FIGURE 1 HADOOP CORE COMPONENTS 10
FIGURE 2 MAPREDUCE DATAFLOW [23] 12
FIGURE 3 LAYERED ARCHITECTURE FOR BIG DATA SOLUTIONS 18
FIGURE 4 CLOUD-BASED PROPOSED BIG DATA SOLUTION 26
FIGURE 5 CLOUD-BASED NATIVE BIG DATA SOLUTION 28
FIGURE 6 EXPERIMENTAL PLATFORM DIAGRAM 30
FIGURE 7 HADOOP PROPOSED WORKERS 35
FIGURE 8 HADOOP NATIVE WORKER 35
FIGURE 9 HADOOP CLOUD-BASED NATIVE MODEL 36
FIGURE 10 HADOOP CLOUD-BASED PROPOSED MODEL 37
FIGURE 11 EXPERIMENTAL SETUP APPROACH 39
FIGURE 12 HADOOP CLUSTER EXPERIMENTAL SCENARIOS 42
FIGURE 13 COMPLETION TIME IN SECONDS RELATED TO WORDCOUNT WORKLOADS 50
FIGURE 14 IMPACT OF SCALING DATA-SIZE ON COMPLETION TIME (WORDCOUNT) 52
FIGURE 15 THROUGHPUT IN MB PER SECOND RELEVANT TO WORDCOUNT 53
FIGURE 16 COMPLETION TIME IN SECONDS FOR DFSIOE-READ WORKLOAD 59
FIGURE 17 IMPACT OF SCALING DATA-SIZE ON COMPLETION TIME (DFSIOE-READ) 60
FIGURE 18 THROUGHPUT FOR DFSIOE-READ WORKLOAD 61
FIGURE 19 COMPLETION TIME IN SECONDS FOR DFSIOE-WRITE WORKLOAD 64
FIGURE 20 IMPACT OF SCALING DATA-SIZE ON COMPLETION TIME (DFSIOE-WRITE)..... 65
FIGURE 21 THROUGHPUT FOR DFSIOE-WRITE WORKLOAD 66

LIST OF ABBREVIATIONS

VM	:	Virtual Machine
CWs	:	Computation Workers
SWs	:	Storage Workers
HDFS	:	Hadoop Distribution File System
YARN	:	Yet Another Resource Allocator
HDP	:	Hortonworks Data Platform
WVM_C	:	A worker VM having a Compute Slave
WVM_S	:	A worker VM having a Storage Slave

ABSTRACT

Full Name	:	Shawqi Mohammed Mohammed Al-Maliki
Thesis Title	:	Towards Providing Hadoop Storage and Computing as Services
Major Field	:	Computer Science
Date of Degree	:	December 2017

Big data is huge, unstructured, and rapidly generated. Handling big data requires a special type of database solution to deal with such characteristics. The Hadoop framework is the prominent solution to big data. In the default architecture of Hadoop (also known as native model), the storage and computing modules are colocated. This condition makes Hadoop rigid, inelastic, and inefficient in resource utilization. An elastic solution that can respond to different demands in real time is a prerequisite for any cloud service. In this work, we propose another architectural model in which storage and computing modules are decoupled. Such decoupling makes the proposed architecture flexible, elastic, and efficient in terms of resource utilization. To evaluate the performance of the proposed model, we compared it with the native model. Based on the evaluation experiments, the proposed model performed better for I/O- and CPU-bound workloads. In addition to the features gained, we also evaluated the overhead of the proposed model.

ملخص الرسالة

الاسم الكامل: شوقي محمد محمد المليكي

عنوان الرسالة: بإتجاه توفير حل معالجة البيانات الضخمة (Hadoop) كخدمات تخزين وحوسبة منفصلة

التخصص: علوم الحاسب الآلي

تاريخ الدرجة العلمية: ديسمبر 2017

تُوصف البيانات الضخمة (Big Data) بأنها بيانات كبيرة جداً في الحجم ومخزنة بشكل غير منظم و يستمر حجمها بالتزايد بشكل سريع جداً. يتطلب معالجة هذا النوع من البيانات نوعاً خاصاً من حلول قواعد البيانات والتي تستطيع التعامل مع الخصائص المُمَيزة لهذه البيانات. الحل الأشهر على الإطلاق لمعالجة البيانات الضخمة هو هادوب (Hadoop). البنية الهيكلية لهذا الحل تقوم على فكرة دمج وحدتي التخزين والحوسبة معاً بحيث يرتبطان ويتواجدان في كل مكون من مكونات الحل، وهو بهذه التركيبة يُعرف بنموذج هادوب الأساسي. لكن ما يعيب هذا النموذج هو أن الترابط بين وحدتي التخزين والحوسبة يجعل هادوب حلاً جامدًا وغير مرن وغير فعال في استخدام الموارد. نحن الآن في عصر الخدمات السحابية والتي تتطلب حلاً مرناً يستجيب للطلبات والأوامر المختلفة في نفس الوقت، لذا فإننا في هذا البحث نقترح نموذجًا بنيوياً مختلفاً لهادوب يتم فيه فصل الترابط بين وحدات التخزين والحوسبة. هذا الفصل يجعل النموذج المقترح مرناً وذو كفاءة أكبر في استخدام موارد التخزين والحوسبة. لتقييم أداء النموذج المقترح ، قمنا بمقارنته بالنموذج الأساسي لهادوب، واستنادًا إلى نتائج تجارب التقييم ، كان النموذج المقترح أفضل وبالأخص في معالجة نوع التطبيقات التي تستنزف مورد وحدة المعالجة المركزية أو تلك التي تتطلب عمليات قراءة وكتابة وبشكل مركز على وحدة التخزين. بالإضافة إلى الميزات العديدة لتبني نموذج هادوب المقترح ، قمنا أيضًا بتقييم الكُلفة العامة في أداء هادوب والتي يسببها تطبيق هذا النموذج.

CHAPTER 1

INTRODUCTION

Simply defined, big data is massive data that is generated very quickly with an unstructured or semi-structured format. For such kinds of data, traditional database solutions like Relational Database Management Systems fail to work properly, as they mainly handle data that is structured, centralized, and limited in size [1]. In a traditional database solution, when data size exceeds a certain threshold, performance starts suffering. Such a situation requires adding further resources to recover system performance. The only way to add further resources to traditional database systems is to upgrade the existing system vertically (scaling up). However, this type of scaling is both limited and costly. Therefore, there is a need for a solution that can handle massive and unstructured data in a smooth and cost-effective way. Big data solutions, such as Hadoop, distributed search, in-memory, Spark [2], Storm [3], and NoSQL, have been developed to meet that need [4].

A big data solution can be deployed either in a bare-metal infrastructure or on the cloud. Deploying it as bare-metal requires expensive hardware and expert staff. However, deploying it on the cloud is feasible and is the most-common choice, as many interested parties cannot afford the expense of physical big data infrastructure [5]. Therefore, the recent direction is to provide big data solutions as a cloud-based service [4]. Some companies have taken the lead in providing Hadoop as a service, such as Amazon EMR [6], IBM Info Sphere Big Insights [7], and Microsoft HDInsight [8].

1.1 Cloud Computing

Currently, the cloud has become an integral part of all business. The recent revolution in cloud computing technologies has attracted the attention of most companies due to its amazing impact on reducing costs and effort. Cloud computing has quickly transformed computing into a model of offering sophisticated services known as cloud-based services. The techniques that play the most important roles in cloud computing are Service Oriented Architecture (SOA) and virtualization [9]. SOA addresses flexibility, reusability, componentization, and extensibility. It was formally adopted in cloud computing in 2009 when a model called Cloud Computing Open Architecture was proposed to bridge the power of virtualization with the power of SOA to form reusable and extensible cloud computing [9]. The other key technique, virtualization, efficiently manages the way operating systems and applications allocate shared physical resources. It is considered the core-computing layer that helps with a smooth configuration, deployment, scheduling, and efficient resource utilization. This can be implemented using either hypervisor solutions, which are dominant, or containers which are called lighter virtualization; in some situations outside of cloud solutions, they outperform hypervisors [10], which explains their recent popularity.

The Hadoop distribution of Hortonworks Data Platform (HDP) that runs on Microsoft Hyper-V is an example of a cloud-based big data solution. MapReduce and HDFS are the core components of Hadoop1. The computation module is represented by MapReduce, while the storage module is represented by HDFS. However, in Hadoop2, the core components are Yarn and HDFS, and MapReduce is an application of Yarn that works in the user-space but not at the core of Hadoop [11].

1.2 Current Hadoop Architecture

In Hadoop’s default configuration, the computation and storage modules are coupled. This is called Native Hadoop, and if Hadoop is deployed on the cloud, as in our work, this model is called Cloud-Based Native Hadoop. For the rest of this thesis, we refer to this model as the Native Model.

A key characteristic of the Native Model is data locality, which means that data is processed locally in each virtual machine and is not transferred into a centralized location to avoid network data transfer overhead. This was the core assumption for Hadoop in its early days. For the Native Model to be well-performing, elastic, and flexible, there are many challenges. One of these challenges is overcoming the tight coupling of the storage and computing modules.

The coupling of storage and computing roles makes this model inflexible, so to scale out, we added a new VM with both layers, even if the running workloads were more I/O or more CPU-bound, i.e., when the computation-to-storage ratio is unknown or is subject to change. This led to the inefficient utilization of resources assigned to those VMs. Another limitation occurs when we need to highly securing stored data. In this model, and because the data is scattered through the whole cluster, we are forced to add the security layer on top of the whole VMs, which causes a lot of overhead. One more issue in the native model is the necessity of removing both layers at once, even in cases where we only needed to remove computing or storage.

1.3 Problem Statement and Motivation

An essential requirement for any cloud service is having an elastic architecture, which can be achieved by tightly coupling storage and computation. However, this can be restricting in situations where the computation-to-storage ratio is not known in advance, or if that ratio changes over time. Thus, attention should be paid to tight coupling to overcome this limitation.

Hadoop consumers pay more attention to its performance in terms of throughput and completion time. Hence, segregating HDFS from MapReduce may give system administrators and developers better chances to modify model configurations in a way that helps maximize cluster utilization and, consequently, performance. This practice could increase consumer’s satisfaction due to gained performance quality. Another gained capability when implementing the Hadoop Proposed Model is the elasticity of the computation module. Tightly coupling computing and storage will get relaxed so each layer can be shrunk or expanded independently. However, layer segregation brings extra overhead caused by data transferring between different nodes because of the lack of data locality. Depending on the Hadoop placement policy, this overhead might vary based on data node location: whether it is in a separate rack or within the rack itself where the computing node exists. Evaluating two comparable cloud-based Hadoop models—the Native Model and the Proposed Model—can provide a solid answer for the performance of each model in terms of throughput and completion time. We can know the feasibility of implementing the Proposed Model when we know the paid cost in exchange for gaining all SOA capabilities, such as elasticity, security, and flexibility.

Our proposed methodology was motivated by the work done on the Hadoop decoupled model. Specifically, our work is relevant to [12] because by shaping or configuring the Hadoop cluster as a Proposed Model, we gain most SOA features, such as flexibility, security, and elasticity. In addition, resources are utilized efficiently because scaling out in a Proposed Model is achieved by adding an optimally-configured and tuned virtual machine for computing or storage. In [12], the authors tested the performance of Hadoop 1, which is rarely used and was replaced, in 2013, by Hadoop 2. Also, the performed experiments were done on one physical host with restricted resources and no consideration for physical networking overhead. Therefore, the results obtained by [12] cannot be generalized to Hadoop 2 or an environment with more than one physical host. Therefore, we propose our methodology to fill these gaps.

1.4 Research Objectives and Contributions

This work explores providing Hadoop storage and computing as independent services. This is achieved by the following objectives:

1) Understand how current Hadoop architecture provides computation and storage services.
2) Propose a new architecture for Hadoop to provides independent computation and storage services.

3) Evaluate the new proposed Hadoop architecture.

Recently, the idea of providing Hadoop as a service is gaining ground [6][8][13]. Industrial organizations such as VMware [14] are channeling their efforts to support Hadoop storage and computing separation. Unfortunately, research is limited. To the best of our knowledge, Frankfurt Big Data Lab [15] attempted to evaluate the performance of two Hadoop models. However, that work [16] did not fully utilize the SOA features. Research that was done in [16] tested Hadoop native and decoupled Models, but only based on one physical host environment, which restricts the notion of SOA in terms of elasticity, security, and flexibility. In addition, this negates a Hadoop key feature: a massive data processing solution. Furthermore, the experiments in [16] were done on Hadoop 1 and HiBench1.0, which are considered obsolete frameworks. In our research, we complement the work done in [16] as follows:

1) Propose a service-oriented Hadoop environment.
2) Propose an elastic computation Hadoop environment.
3) Provide a prototype of cloud-based Hadoop 2.0 clusters with various physical hosts and different Hadoop distribution namely, Hortonworks.
4) Evaluate our prototype using a different hypervisor, Windows Hyper-V, and enhanced benchmarking suite, HiBench-6.0.
5) Compile suitability recommendations for prototype usage.

CHAPTER 2

RELATED WORK

Research on the performance evaluation of Hadoop clusters can be classified into two categories. The first category compares different Hadoop clusters from a deployment perspective, i.e., in the same deployment environment or in a different deployment environment. In contrast, the second category compares different Hadoop SOA-based models for a single Hadoop cluster deployed on premise or on the cloud.

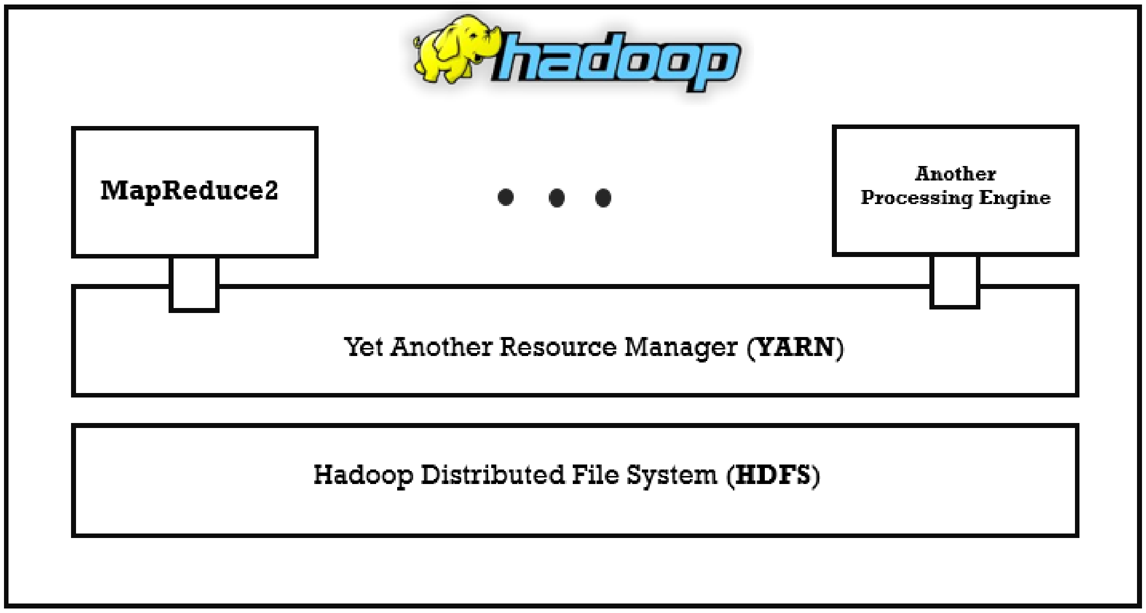
2.1 Hadoop

Hadoop is an open source framework that handles big data in a distributed and cost-effective manner. It is a highly scalable cluster that comprises commodity machines [17]. It has become widely adopted in both industry and academia. In industry, it is used in various applications, such as web search, spam filtering, and social network recommendations. In academia, much academic research is built on Hadoop [18]. Hadoop is considered the most popular, cost-effective, and scalable distributed computational framework for big data; it is the dominant and de-facto standard. It integrates a storage module, a computation module, and other modules to form a powerful distributed processing and storage solution [1]. It is an ecosystem with many components. Figure 1 illustrates two of the core components: Hadoop Distributed File System (HDFS), which is used for handling the storage module; and MapReduce, which is used as a computation framework. These two components are built from work on Google GFS [19] and Google Bigtable [20], both published as white papers. Other Hadoop components are Pig, Hive, Storm, Mahout, Spark, Tez, Zookeeper, and Hbase [17]. Hadoop has two generations: Hadoop 1, which is a batch-oriented MapReduce model; and Hadoop 2, which is an interactive and specialized processing model with other capabilities that turn Hadoop upside down. The most important advances in Hadoop 2 are HDFS federation, YARN, and HDFS high-availability. HDFS federation gives a cluster the capability of scaling by adding more NameNodes, which are storage master processes that handle portions of the file system namespaces. YARN, in contrast, is a resource manager that was created by separating the processing engine and resource management capabilities of MapReduce. So, MapReduce became an application of YARN [11].

The most fundamental advantage of using YARN is to no longer be restricted to work only on I/O intensive and high latency MapReduce frameworks [1]. For example, in Hadoop 1, we had no option other than processing big data using a batch-oriented framework, MapReduce, which is I/O intensive. However, with YARN capability in Hadoop 2, things are more flexible in terms of choosing processing framework, i.e., in addition to batch-processing, we could also choose interactive or real-time processing frameworks, such as Storm [3] and Spark [2], that are more in-memory rather than I/O operations.

Outside the revolutionary change caused by adding YARN, opening-up the Hadoop framework is the most important development [11]. That is, we can now run multiple applications on a single Hadoop cluster, each of which has a different processing type. This enables Hadoop to efficiently share data among applications. The introduction of YARN caused significant changes to how MapReduce works. For example, scheduling is no longer part of MapReduce. Instead, it became part of YARN's jobs. YARN's responsibilities are resource scheduling and monitoring. Monitoring is significant, as it ensures two functions, reallocating resources using freed-up capability and terminating containers that exceed the agreed-upon and allocated resources. The simplicity of YARN comes from not keeping the history of executed applications and for not knowing the type of the running application. YARN components are ResourceManager and NodeManagers. ResourceManager is the master process of YARN, while NodeManager is a slave process that runs on every node in a cluster. Its responsibilities are: creating monitoring and killing containers. For creating containers, it receives requests from the ResourceManager and Application Master. It also reports the container status to the ResourceManager. Application Master is the master process of a YARN application. It is the first container that is created by NodeManager on behalf of ResourceManager.

Here, it is worth mentioning that we still need batch-processing, as there are situations where we only can use batch-processing, such as when a dataset cannot fit into a memory to be handled using real-time processing. Concerning Hadoop 2's high-availability implementation, there are a couple of NameNodes in an active-standby configuration. So, in case of the failure of the active NameNode, the standby takes over its duties to continue servicing client requests without significant interruptions [1].


hadoop
MapReduce2
Another Processing Engine
Yet Another Resource Manager (YARN)
Hadoop Distributed File System (HDFS)
Figure 1 Hadoop Core Components

2.1.1 MapReduce

Our work is based on Hadoop 2, to which YARN was introduced and MapReduce was re-written as a YARN-based application called MapReduce2 that lacks scheduling [11]. In our work, we refer to MapReduce2 as MapReduce. MapReduce is a distributed computing component of Hadoop that works based on Mapping and Reducing algorithms. It is a programming model for batch processing, so it does not suit real-time analysis in which results are expected to be seen instantly. Instead, queries take minutes or even hours to be finished. Despite the emergence of new processing frameworks, MapReduce is still needed because it is useful in understanding how batch processing works and how a dataset is divided into smaller pieces. Furthermore, in some situations, getting a real-time result is not required.

MapReduce works by splitting the processing into maps, then reducing them. The output of the former is used as an input for the latter, as illustrated in Figure 1. The unit of work to be processed is called a MapReduce job. This job runs in Hadoop by separating it into two different types of tasks: map tasks and reduce tasks. Using YARN, map tasks are scheduled to be run on cluster nodes, and in case a task fails, it will be rescheduled to run on another node. The output of map tasks is written to the local disk, not the HDFS, because it is considered a temporal output that will be consumed by a reduce task to produce the final output. For a reduce task, input data comes to reduce task nodes across the network from different map tasks to be merged and then, for reliability, the output is saved in the HDFS.

2.1.2 HDFS

Hadoop is based on an abstract notion of filesystems. HDFS is just one implementation that is designed to work efficiently in conjunction with MapReduce. Other examples of file systems that Hadoop supports are Microsoft Azure and Amazon S3. HDFS is a file system that is designed for storing huge files on clusters consisting of commodity hardware. A Hadoop core component, HDFS, stands for Hadoop Distributed File System; it is the default distributed file system for Hadoop. However, Hadoop has a general-purpose file system abstraction that enables it to integrate with other DFS, such as Amazon S3, IBM GPFS [21], and Azure Blob [22].

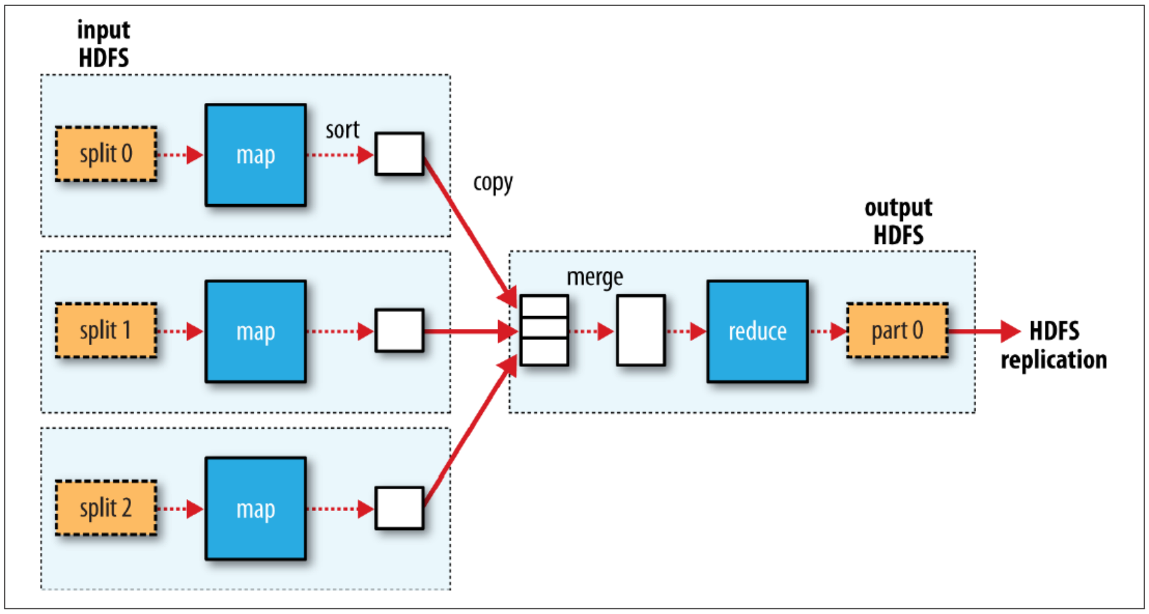


Figure 2 MapReduce DataFlow [23]

HDFS is built around the idea that the most efficient data processing pattern which is write-once and read many times. In the typical situation, a dataset is copied or generated from a source, then several analyses are done on that dataset. Each analysis includes a large amount of the dataset. Therefore, the time needed to read the complete dataset is more significant than the delay in reading the first record. However, there are areas where HDFS is not recommended, such as for low-latency data access, lots of small files, and numerous writers and random file changes. HDFS will not efficiently serve programs in which low-latency access to data is essential. It is optimized for delivering a high throughput of data. However, this comes at the expense of latency. The limit to the number of files in a filesystem is determined by the amount of NameNode available memory, because the NameNode keeps file system metadata in memory. Files in HDFS are written by a single writer. Writes are made in an append way. Multiple writers are not supported [1].

2.2 Comparing Different Hadoop Clusters

2.2.1 Same Deployment Environment

The authors in [24] compared two different Hadoop distributions: DataStax [25] and Cloudera [26]. The main purpose of their work was to investigate the workload types that better suit one Hadoop distribution or the other, taking into consideration data size. Three workload types were investigated: CPU-bound, I/O-bound, and mixed. Based on conducted experiments, they found that CPU-intensive workloads were almost linear in both Hadoop distributions. For I/O intensive, in read-intensive workloads, DataStax Distribution performed up to 32% slower than Cloudera. However, in write-intensive workloads, DataStax performed up to 81% faster than Cloudera.

Another work that fits into this category was done by [27], which compared Hadoop enterprise distributions based on metrics such as Hadoop solution's architectural and operational functionality, modeling, storage, and low latency. The researchers found leaders and strong performers in providing Hadoop solutions. Based on that, IBM [7], Amazon [6], Hortonworks [28], MapR [29], and Cloudera [26] are leading enterprises, while Datameter [30], DataStax [25], and Pentaho [31] are strong performers.

The authors in [32] studied the impact of scaling out and scaling up on the performance of two Hadoop clusters. Scaling out means extending the cluster horizontally, while scaling up means extending the cluster vertically by adding more resources. The results showed that scaling up outperforms scaling out in CPU-bound operations, whereas scaling out outperforms scaling up in I/O-bound operations. The researchers noticed that other components like network I/O, which were not investigated fully, affect the performance of the cluster.

2.2.2 Different Deployment Environments

The authors in [33] showed that a bare metal Hadoop cluster is 4% better than the simplest virtual Hadoop cluster, meaning that gaining all virtualization advantages with that small overhead cost is considered a very good achievement. The results also showed that running more than one virtual machine, up to a specific limit, brings better performance than corresponding physical machines. Another relevant work [34] compared an optimized separated Hadoop virtual cluster against separated physical Hadoop cluster and showed that virtualization overhead cost can be compensated by fine-tuning the configuration of the virtual environment.

Other researchers have compared MapReduce computation speed in a virtualized environment against MapReduce in a physical environment, such as in [35], where the MapReduce service was deployed into a Hadoop virtual cluster to compare its performance with on-premise deployment. It was found that many virtualized Hadoop issues need to be addressed and optimizing the MapReduce virtualized service was a suggested solution to avoid such issues. A similar MapReduce performance evaluation was done in [36], which showed that increasing MapReduce relevant computation speed is possible under complete virtualization conditions. Similarly, other researchers [37] found that from a resource utilization perspective, partitioning a physical host into multiple VMs can result in a similar or even better performance than the physical platform with regards to MapReduce jobs. In contrast, the researchers in [38] found that the performance of the I/O-bound workload was more sensitive to virtualization overhead than was the CPU-bound workload. Hence, for I/O-bound workloads, they recommended adding more VMs rather than adding more VCPUs to a VM.

2.3 SOA-Based Hadoop Separated Models

This category of literature work compares different models for a specific Hadoop cluster. Decoupling Hadoop's computing layer from its storage layer within the same Hadoop distribution was discussed in [39] and [40] as a way to achieve elasticity in physical deployments. However, there is some work in industrial organizations comparing virtualized Hadoop clusters, but little work in academia. An industry-related work was done by VMware [14] explaining the Hadoop Native Model and the Proposed Model in terms of capability and the support given by the VMware hypervisor, while also showing how those models can be deployed when a relevant hypervisor solution named vSphere.

The researchers in [16] conducted performance comparisons on two cloud-based Hadoop cluster models: the standard model and the data-compute model. The results showed that CPU-bound workloads are suitable for the data-compute model. However, read-bound workloads are suitable for the standard model, and adding more data nodes in the data-compute model improved read performance. They also showed that write I/O workloads are suitable for the data-computer model. However, a lower number of data nodes results in better write performance.

2.4 Literature Review Summary

In this section, we elaborate on how our proposed model fills the gap of the existing related work. We first start by showing the limitations of the research papers we discussed in this chapter. Table 1 shows the limitations of the existing work. For example, only one technical report from VMware explains the Hadoop Native Model and Separated Model in terms of capability and the support given by the VMware hypervisor and shows how those models can be deployed when using their own hypervisor solution named vSphere. Also, one paper tested the performance of Hadoop 1, which is rarely used and was replaced in 2013 by Hadoop 2. Furthermore, the performed experiments were done in one physical host with restricted resources and no consideration for physical networking overhead. Therefore, the obtained results cannot be generalized to Hadoop 2 or environments with more than one physical host.

To fill in these gaps, we developed a methodology to overcome the limitations summarized in Table 1: Limitations (1, 2) were addressed by comparing the performance of one model of Hadoop with another model (the Native Model vs. the Proposed Model). Limitation (3) was addressed by not supporting this academic research by a specific industrial organization. Limitation (4) was addressed by my thesis contributions (3-5). That is, we proposed a service-oriented and an elastic computation Hadoop environment. In addition, we provided a prototype of cloud-based Hadoop 2.0 clusters with various physical hosts and different Hadoop distribution, namely Hortonworks. Moreover, we evaluated our prototype using a different hypervisor, Windows Hyper-V, and enhanced benchmarking suite, HiBench-6.0.

Table 1 Summarized Limitations of Related Work

Paper#	Limitation	Limitation#
[24] [27] [32]	- Comparing different Hadoop clusters (all have the same architectural model, Native) deployed on the same environment. That is, either both were deployed on the cloud or both were deployed in a bare-metal environment	1
[33] [34] [35] [36] [37] [38]	- Comparing different Hadoop clusters (all have the same architectural model, Native) deployed on different environments. That is, one was in the cloud and the other in bare metal.	2
[14]	Industry work (a technical report from VMware): - Explains the Hadoop Native Model and Separated Model in terms of capability and the support given by VMware's hypervisor. - Shows how these models can be deployed when using the hypervisor solution named vSphere	3
[16]	- The performed experiments were done on one physical host with restricted resources and no consideration for physical networking overhead. This negates Hadoop's key feature of being a massive data processing solution - The obtained results cannot be generalized to Hadoop 2 or environments with more than one physical host. - The experiments in this paper were evaluated by benchmark suite HiBench1.0, which is considered an obsolete framework as it works only with Hadoop 1. - Tested the performance of Hadoop 1, which is rarely used and was replaced, in 2013, by Hadoop 2.	4

CHAPTER 3

CLOUD-BASED PROPOSED MODEL

In this chapter, Section 3.1 explains the general architecture of the cloud-based big data solutions, while Section 3.2 discusses the gained capabilities that result from adopting the big data Proposed Model.

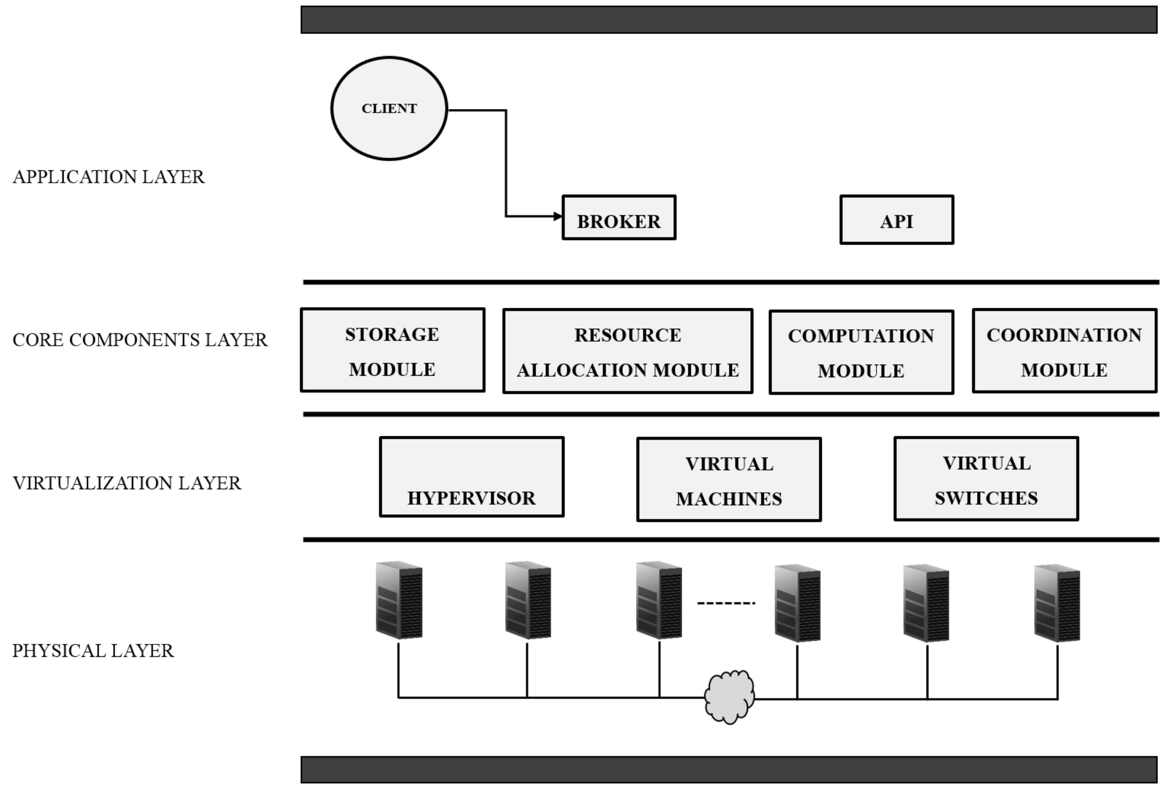


Figure 3 Layered Architecture for Big Data Solutions

3.1 Layered Architecture for Big Data Solutions

A typical architecture of a big data solution includes four key layers that are illustrated in Figure 3 and namely, physical layer, virtualization layer, core components layer, and application layer. The layers provide an approach for arranging components that belong to the same category. They are just a logical representation that doesn't indicate that they are working independently.

3.1.1 Physical Layer

Tangible resources such as physical servers, switches, network connectivity, and data center are the units that make up the physical layer. The physical layer is the basis for all above subsequent layers. Therefore, while we construct this layer, we need to consider the requirements of the remaining layers. Scalability is the most significant requirement that needs to be considered while building this layer. It is the foundation for a cloud-based big data solution. Hence, the potential of infrastructure continuous growth is very high.

3.1.2 Visualization Layer

Virtualization technologies are leveraged to hide the complexity of the physical resources, and to enable attractive capabilities such as sharing resources, dynamic provisioning of resources [41], flexibility and efficient utilization of the resources. Virtualization tools and technologies such as hypervisors are used for transforming the environment into a cloud-based by enabling the capability of creating virtual entities such as machines and switches which are the component of this layer.

3.1.3 Core Components Layer

Core components layer comprises storage module, resource allocation manager module, a computation module, and coordination module. Each has a specific role and perform set of related functions. These modules can be provisioned and managed manually or using dedicated big data management solutions such as Ambari [42] and Cloudera Manager [43]

3.1.3.1 Storage Module

Big data storage can be classified into two categories, file systems, and database technologies. A distributed file system is the base of the big data storage. For that, it highly attracted the attention of academy and industry. Recently, big data distributed file systems have become bit matured as a result of a long journey of large-scale commercial operation [18]. Researchers and leading providers have created their own solutions to meet distinct big data storage requirements. For example, Google File System GFS was designed and implemented as a scalable distributed file system for massive distributed data. HDFS [44] and Kosmosfs [45] emerged as derivatives of GFS. On the other hand, Facebook developed Haystack [46] to store a huge amount of small-file photos, and Amazon Simple Storage Service (S3) [6]was implemented to store and retrieve any volume of data from anywhere: business applications, IoT sensors, mobile applications, and websites. One more example is Windows Azure Binary Large Object (Blob) [47] storage which provides object-store functionalities.

With regards to database technologies as the other category of big data storage, many database systems have been proposed for handling huge datasets as the traditional relational database systems fail in addressing the complexity and the massive size of big data. NoSQL is positioned on top of the proposed solutions that emerged to cope with big data problems. It is viewed as the standard due to its key characteristics such as schema-free, replication support, having a simple API, and consistency and working smoothly with a huge amount of data. key-value stores, column-oriented databases, and document databases are the main three types of NoSQL databases from the data model point of view [18]. Examples of popular NoSQL products are MongoDB, HBase, Cassandra.

The storage can support two types of processing, synchronous and asynchronous. In the former, data is processed in real-time or near real-time, so the storage should be enhanced for low latency. However, in the latter, data is captured, recorded and processed in batch and for that storage low latency is not required.

3.1.3.2 Resource Manager

Cluster resource manager is the architectural center of big data solution that allows multiple data processing engines such as real-time streaming and batch processing to handle data stored in a single platform which unlock an entirely innovative approach to analytics. This type of foundation modules is considered a new generation of resource management and is enabling organizations everywhere to realize a modern data architecture. The main goal of the cluster resource manager is to enable sharing the resources of a large cluster of machines between different computation frameworks due to the inefficiency of creating separate infrastructures to accommodate applications. The resource manager is a per-cluster level Component and it has two main functionalities: Scheduling and application management. The resource manager scheduler is responsible to schedule required resources to applications and it does care about monitoring or tracking of those applications. On the other hand, application Master is a per-application level component which is responsible for interacting with both resource manager scheduler. Examples of cluster resource managers are Moses [48], Kubernetes [49], and Yarn [17]. The Mesos and YARN cluster managers are superior to other managers because they consider the resource needs of other applications running on the cluster and impose a scheduling policy through all of them.

3.1.3.3 Computation Module

There are many big data computation frameworks, namely batch processing, real-time processing, and hybrid. In the real-time processing paradigm, data analysis is done as soon as possible to be able to gain instant insights. In this paradigm, data comes in a stream, and while it is arriving continuously, only a small portion of the stream is stored in limited memory [18]. Few passes over the stream are used in finding approximation results. The real-time processing paradigm is used for online applications at the level of second or millisecond. Representative open source big data real-time modules include Storm [3], Impala [50], Spark [2], and Tez [51].

Batch Processing, on the other hand, is a paradigm where data is first stored and then processed.ma Popular examples are MapReduce [52], Hive [53], and Pig [54]. MapReduce is the dominant batch-processing model. It is a powerful programming model that adds the paralleling and distribution of massive computation applications on clusters of commodity machines [18]. The main idea of MapReduce is that data are initially separated into smaller portions. Then, these portions are processed in parallel in a distributed manner to generate intermediate results. The last result is derived by combining all the intermediate results. This model typically utilizes computation resources near to data location to avoid the network overhead of data transferring [18]. The MapReduce model is widely implemented in web mining and machine learning.

Hybrid Processing is a combination of both batch and real-time processing needs. A key factor in providing performance for big data applications is the data locality in which the computation is moved into the location of data. This is the preferred option in typical high-performance computing systems [55].

3.1.3.4 Coordination Module

It is a centralized service for providing distributed synchronization, maintaining configuration information, and naming [56].For coordinating the actions of independent applications or computing entities that are involved in big data solution, a dedicated big data coordination module is needed to enable a highly reliable distributed coordination. It helps application developers to focus on the core business logic and rely completely on this dedicated module. Tasks such as naming service, distributed synchronization, such as locks and barriers, and configuration management can be accomplished by coordination dedicated module to avoid the potential bugs resulted from manual implementation by developers [57].

3.1.4 Application Layer

Big data cluster core components are reached through a specific cluster-API. A user can interact freely and directly with these core components by exploiting the interface provided by the programming models [18] to perform different data analysis functions, clustering, and classification. Another way for interaction with big data cluster is an indirect way through utilizing a broker, an intermediate tool that has the needed capabilities to facilitate the interaction between the user and the cluster core components. A benchmark suite is an example of an intermediate tool that is used for heavily testing the big data solution using synthetic or real-world workloads.

3.2 Hadoop Proposed Model

To overcome the Native Model's limitations, we explored providing Hadoop cloud-based computation or storage modules as independent services, as opposed to providing both Hadoop modules at once as one service. Therefore, decoupling the computation and storage modules was required. In doing this, we came up with a new model called the Cloud-Based Proposed Hadoop. For the rest of this thesis, we refer to this model as the Proposed Model. Scalability, elasticity, efficient resource utilization, and security are the fundamental advantages of the Proposed Model, which is based on the notion of SOA [4].

Scaling a Hadoop cluster in or out means extending that cluster horizontally by adding or removing nodes [58]. In contrast, the capability of Hadoop cluster to adapt to workload changes by allocating and reallocating resources in an autonomic way is called elasticity [59]. So, for Hadoop to scale out and scale in the storage and compute modules independently, these modules must be decoupled so each individual module can be handled separately [34].

However, gaining elasticity can be achieved by the computing module, but not by the storage module. The elasticity of the computing module can be achieved seamlessly because it is easy to shrink or expand in real-time. This contrasts with the storage module, where adding a new storage node affects the whole cluster. That is because the rebalancing is a network-bandwidth intensive and time-consuming process [4]. In addition to the elasticity and scaling in and out that are gained by the decoupling of the computing and storage modules, the Proposed Model has further benefits, such as efficient resource utilization, inheriting all SOA capabilities, and others, as explained below [4]:

- Flexibility and Agility: scaling out by adding an optimally-configured and tuned VM for the computing or the storage module. So, we can modify the assigned resources at any time and scale in or out strictly and independently based on the workload requirement. For that, we can directly begin projects instead of spending a long time building infrastructure that meets the expected workload. In other words, we are not required to know the needed computing and storage capacities in advance. That frees us from having to guess the exact need for resources, which can result in either over- or under-provisioning.
- Improve Data Protection and Security: When storage VMs are decoupled, we can put data-relevant security rules only on them, without disturbing the computing module with an avoidable security overhead [60].

However, the Proposed Model does have a performance penalty as a result of the extra overhead caused by data transferring between VMs due to the loss of data locality [14] [16]. This model was previously considered infeasible and expensive to achieve because of the low speed of storage mediums and networking. In this work, due to the increased speed of networks, we could investigate the Proposed Model to gain all virtualization and SOA advantages, specifically: agility, cost effectivity, flexibility, and efficient resource utilization, all without sacrificing much performance. If we could overcome the networking overhead by utilizing very high-speed networking technologies, data-locality would no longer be needed. Thus, we could gain all benefits of the decoupled model with no penalty to cluster performance.

3.2.1 Big Data Solution

In this thesis, we propose a big data solution model that is shown in Figure 4. In this model, we separated the computation module from the storage module which means that each individual worker VM will contain either a compute slave or a storage slave but not both.

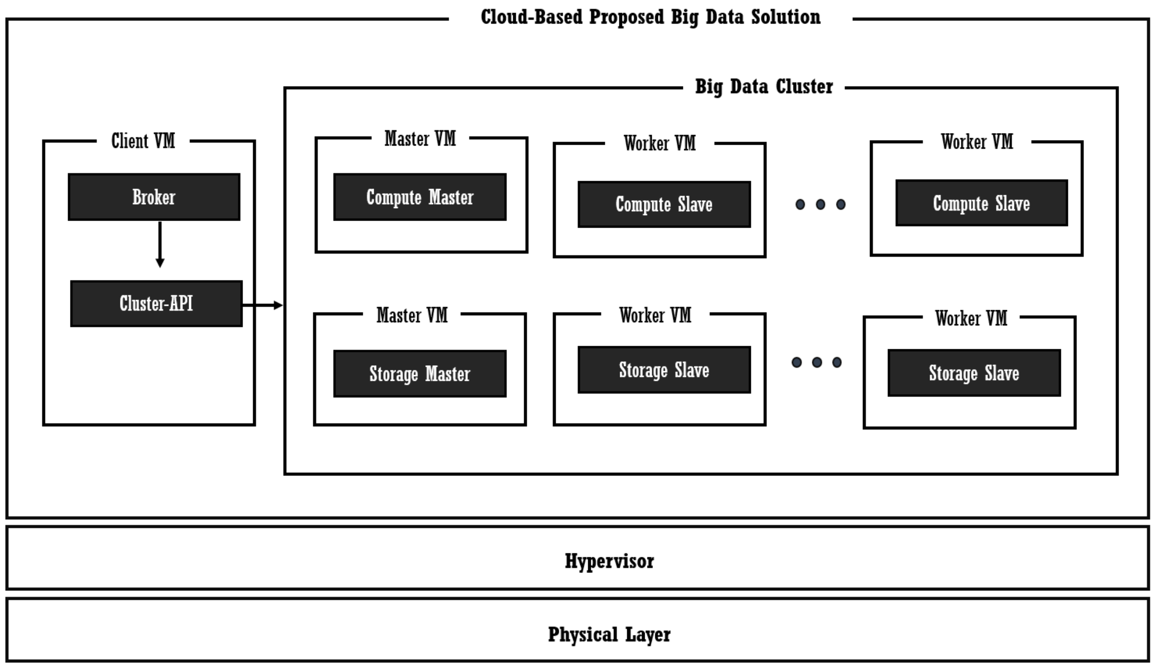


Figure 4 Cloud-Based Proposed Big Data Solution

In the Native architecture of big data solution as shown in Figure 9, and because of its coupling nature, there is no option but configuring it one way following the default settings in which the computation workers and the storage worker are hosted in one VM. So, each VM has one computation worker and one storage worker. However, for the Proposed Model and due to the decoupling of computation and storage module, we have the flexibility to come up with many different configurations. For example, in one configuration, we can distribute the computation workers and the storage workers equally. In another example, we can configure it to have the computation module bigger than the storage module. Another way of configuring the Proposed Model is by having the computation module less than the storage module. The typical scenario for building a big data solution utilizing the Proposed Model is detailed in the following subsections:

3.2.2 Virtual Machines Preparation

A big data workload can be one of three common workload patterns. First workload pattern is the CPU-Intensive. In this pattern, the solution heavily consumes CPU cycles and memory. The second workload pattern is the I/O-Intensive. That is, solution spends much time on reading and writing operations. The last pattern is the balanced one. In the workload of balanced pattern, the solution consumes memory, CPU, and I/O all in a fair way. For leveraging the Proposed Model, the assumption is that the big data administer has an idea of the nature of the workload pattern that will be running on the solution. Given that, building a big data solution will be different based on the workloads pattern. For example, for CPU-Intensive application, following considerations should be taken into account:

1) The vast majority of the cluster VMs will be configured to be part of the computation module.
2) Those VMs will be assigned with more CPU cores, more memory, and less storage.

However, for I/O-Intensive application:

1) Most of the cluster VMs should be configured to be part of the storage module.
2) Those VMs will be allocated more storage, medium memory space, and fewer CPU cores.

For the balanced workloads, all VMs should be provisioned with same resources. That is, each VM will be part of computation and storage module. Therefore, it will utilize all types of resources in a fair way.

3.2.3 Mapping the big data solution workers into the Prepared VMs

The process of deploying a big data solution utilizing the Proposed Model needs some attention to be paid for the mapping of the big data workers and the prepared VMs. The VMs that are prepared and provisioned with better CPU cores and better memory are supposed to be mapped into the big data computation workers. Similarly, the VMs that are prepared with more storage are supposed to be mapped in the storage workers.

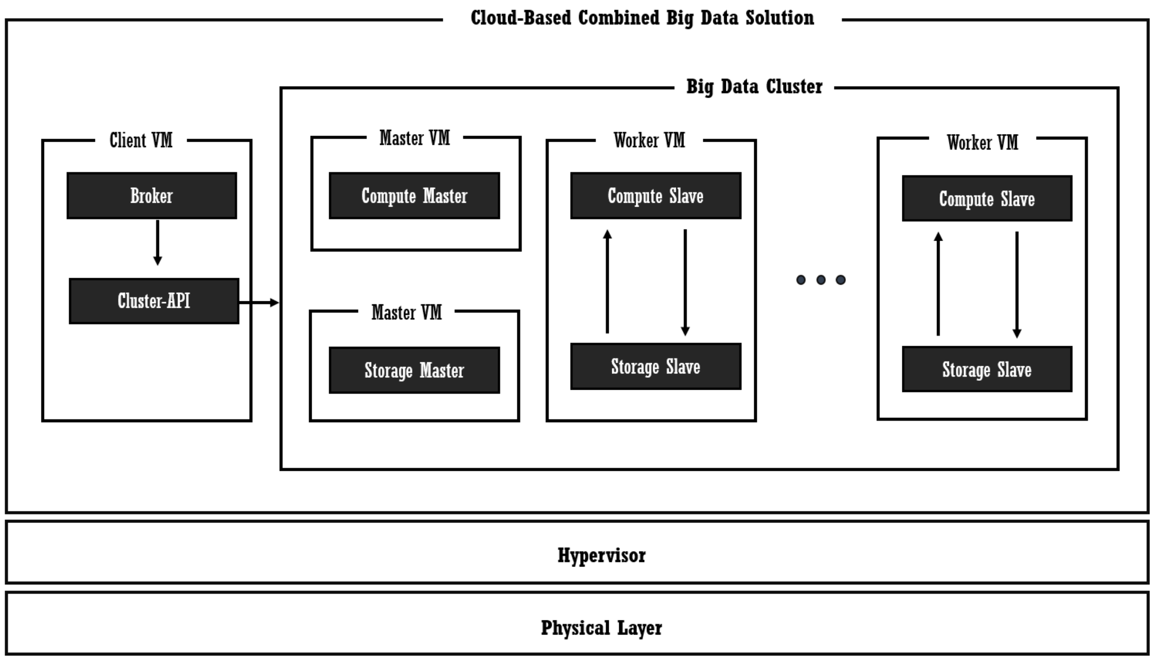


Figure 5 Cloud-Based Native Big Data Solution

CHAPTER 4

PERFORMANCE EVALUATION

We evaluate the performance of two comparable big data cloud-based architectures, the Native-Model and the Proposed Model. For that, we built a platform with a specific configuration that meets the requirement of the model under investigation. Then, we run a set of experiments, collect the results, and perform the analysis. Section 4.1 shows the details of building the experimental platform, while Section 4.2 shows the approach we followed while conducting the experiments in different experimental scenarios. Section 4.3, explains the different experimental scenarios of the Native and the Proposed Models. Section 4.5 discusses and analyzes the results obtained from the experiments.

4.1 Experimental Environment

As shown in Figure 6, we setup the cloud-based environment that is required for running our experiments as follows: First, we prepared the physical components by providing hosts and switches with the required capabilities to handle datasets of hundreds of Gigabytes. Then, we moved on configuring the virtualization part of the environment. In each physical host, we installed Microsoft Windows Server 2012 R2 as a cloud operating system to help us in building a private cloud environment through enabling its virtualization capability by activating Hyper-V which is the hypervisor feature that is built-in Windows Server 2012 R2. Hyper-V was utilized for creating several VMs and several virtual switches based on the requirement of the intended cluster to be deployed. Next, Hadoop clusters were deployed. Some of them were configured as Native Models and others were configured as Proposed Models. Finally, we installed HiBench as a benchmark suite that was utilized for evaluating the performance of the deployed clusters. By doing this, our platform is ready for running the experiments. The following sub-sections show the details of the components that were used for building the experimental platform.

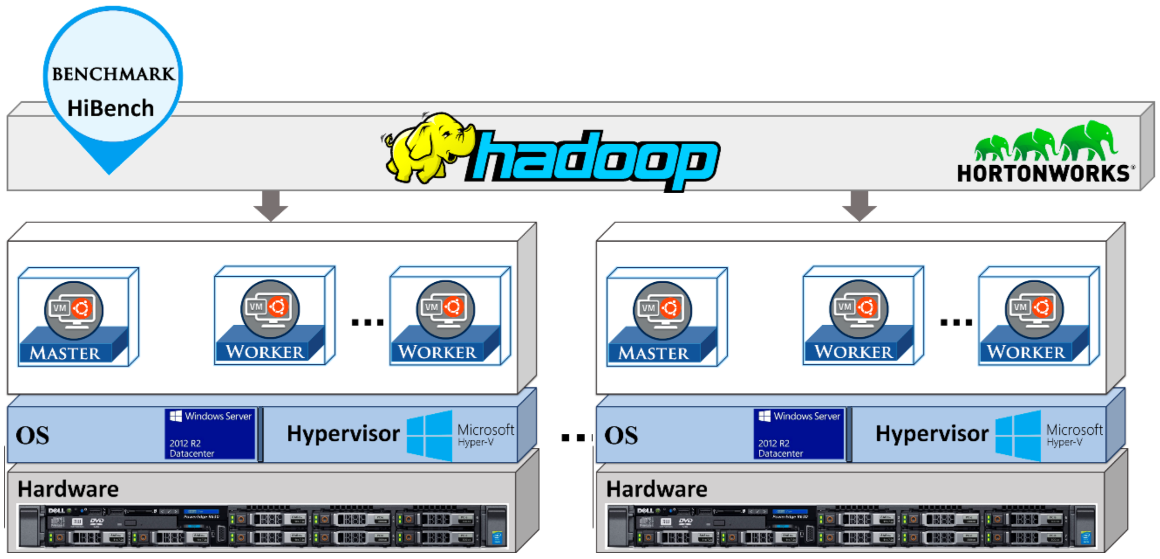


Figure 6 Experimental Platform Diagram

4.1.1 Physical Components

The physical components of our platform are two hosts, and one switch that connects the hosts to each other. The specifications of each physical host are shown in Table 2 with the following details: Two network adapters each with a maximum capacity of 1Gbps.The storage is eight SAS Hard-Disks, 300 GB each, that were merged into a RAID-0 native disk array and mounted as one logical volume coming up with a total of 2400 GB. The processors are two each has 6 Cores and 12 Logical processors. Physical memory is 14 chips, and each one has the capacity of eight GB, so the whole capacity is 112 GB in each individual physical host.

Table 2 Physical Host Characteristics

System Model	Dell PowerEdge R620
Physical Memory (RAM)	112 GB
Processors	2x [Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz, 2000 Mhz, 6 Core(s), 12 Logical Processor(s)]
Storage	2.4 TB Eight SAS (15K PM) each 280GB
Network Adapters	2x Broadcom NetXtreme Gigabit Ethernet

4.1.2 Virtualization Components

In our environment, Microsoft Windows Server 2012 R2 Datacenter was chosen as a host operating system due to its various key features [61] such as highly-virtualized private cloud environments, stability, supporting up to 1024 of active virtual machines, while other visualization solutions such as VMware supports only 512. In addition, there is another set of features that encouraged us on selecting Windows Server 2012 Datacenter as a host OS such as dynamic memory, distributed file system replication, and automatic virtual machine activation[62].Within this OS, we enabled the built-in hypervisor to create the virtual machines and virtual switches.

4.1.2.1 Hypervisor

Because of its appealing features, Hyper-V has been selected among many available hypervisors. For instance, it is a free hypervisor -licensing cost is included with the license cost of Windows Server- that offers an enterprise-grade virtualization, also it is considered a type-1 hypervisor that outperforms another type of hypervisors, scale-in, and scale-out can be done with much ease, and comprehensive support. In addition to that, it can be run in the private, public, or hybrid cloud. Hyper-V has a proven track record as the key websites of Microsoft, TechNet and MSDN, are hosted in a Hyper-V environment [63]. It is a Microsoft product that is offered in two forms, as embedded into Windows Server, or configured as a standalone server known as Hyper-V Server [64].

4.1.2.2 Virtual Machines and Guest OS

Utilizing the appealing functionalities of Hyper-V, we smoothly created and configured several VMs in each physical host. That converts each host into a virtual rack in which we assigned resources to VMs quickly and efficiently. Hence, the base of our work is the two virtualized racks which were used for building virtualized Hadoop clusters. In our work, the two investigated models were deployed into Linux OS as they are the typical operating systems that have a full compatibility with Hadoop and has a wide community of support. In our work, we created VMs with Ubuntu 14.04 OS. Linux-based OS is supported by the key players of Hadoop distributions such as Hortonworks Data Platform (HDP) and Cloudera Distribution including Apache Hadoop (CDH).

4.1.2.3 Networking

The physical switch and the two network adapters in each physical host were utilized in establishing virtual as well as physical connections among all machines in our environment as follows: In each physical host and using Hyper-V and the two-physical adapters, two virtual switches were created. The purpose of the former is to link all the virtual machines in each host to each other, and to the physical host itself using private IPs, while the purpose of the latter is to provide internet connection to the pool of virtual machines by assigning

IPs utilizing DHCP server. By linking the two physical hosts together utilizing the physical switch, we came up with one pool of virtual machines which is considered a prerequisite for deploying a Hadoop cluster.

4.1.3 Deploying Hadoop Cluster Core Components

Utilizing the hypervisor solution (Hyper-V), the two powerful physical hosts and physical switch, we ended up with one pool of virtual machines which are distributed in two independent virtual racks. Hence, the environment became ready to deploy a big data solution as a Native Model or as a Proposed Model. In our environment, the selected big data solution was Hadoop, and specifically the distribution of HDP 2.4.3 [65]. It is an enterprise-ready open source Apache Hadoop distribution based on YARN as a centralized architecture.

In any cloud-based big data solution, including Hadoop, there are three types of Virtual Machines VMs namely, masters, workers, and clients. A master VM hosts a master process which is considered the key process. For that, it is usually assigned more resources and dedicated VM. The other type of VMs is called workers. They can be configured in two ways, Native Workers NW or Proposed Workers PW, based on the placement of the compute and the storage slaves. In the Native Worker, the compute and the storage slaves are installed in the same VM, and the relevant cluster is called Native Model, while in the Proposed Worker, a VM contains either a compute slave or a storage slave but not both, and the relevant cluster is called the Proposed Model. The last type of VMs are called clients in which we can install third-party applications.

As part of Hadoop cluster deployment, Hadoop core components were provisioned, managed, and monitored using Ambari 2.2.2.18 [42] which is a Hadoop cluster management tool. One of Hadoop core components is the storage module where the storage master is called NameNode and the storage slaves are called DataNodes. Other Hadoop core component is the resource allocation module. It is called Yet Another Resource Negotiator, YARN, and it consists of ResourceManager as a resource allocation master and NodeManagers as resource allocation slaves. The third core component of Hadoop is the compute module. It has many compute modules. However, in our work, we considered the MapReduce which is a batch processing module. It works in conjunction with the resource allocation module, Yarn. Last Hadoop core component is the coordination module that allows Hadoop distributed processes to get updated. It usually consists of an odd number of Zookeeper servers.

The difference between Hadoop Proposed Model and Hadoop Native Model is the type of workers they are contained in the cluster. For example, to deploy Hadoop cluster as a Native Model, we use the native workers that are shown in Figure 8 as the building blocks. Similarly, if we want to deploy Hadoop cluster as a Proposed Model, we use the proposed workers that are shown in Figure 7 as the building blocks.

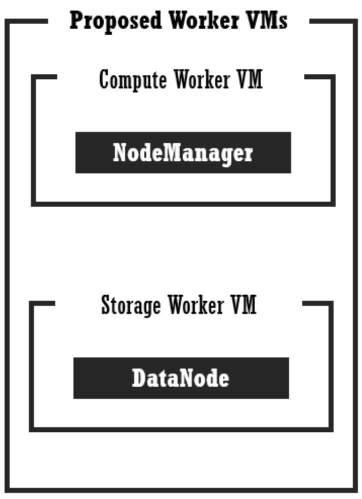


Figure 7 Hadoop Proposed Workers

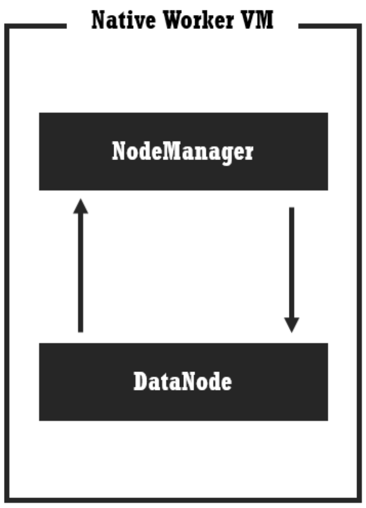


Figure 8 Hadoop Native Worker

4.1.3.1 Deploying Hadoop as a Native Model

In situations where we choose to run the experiments on the Native Model, as shown in Figure 9, we move on building a cloud-based Hadoop cluster using native workers, i.e., by assigning a NodeManager and a DataNode into each worker VM. In this model, data locality is maintained as a key part of the architecture.

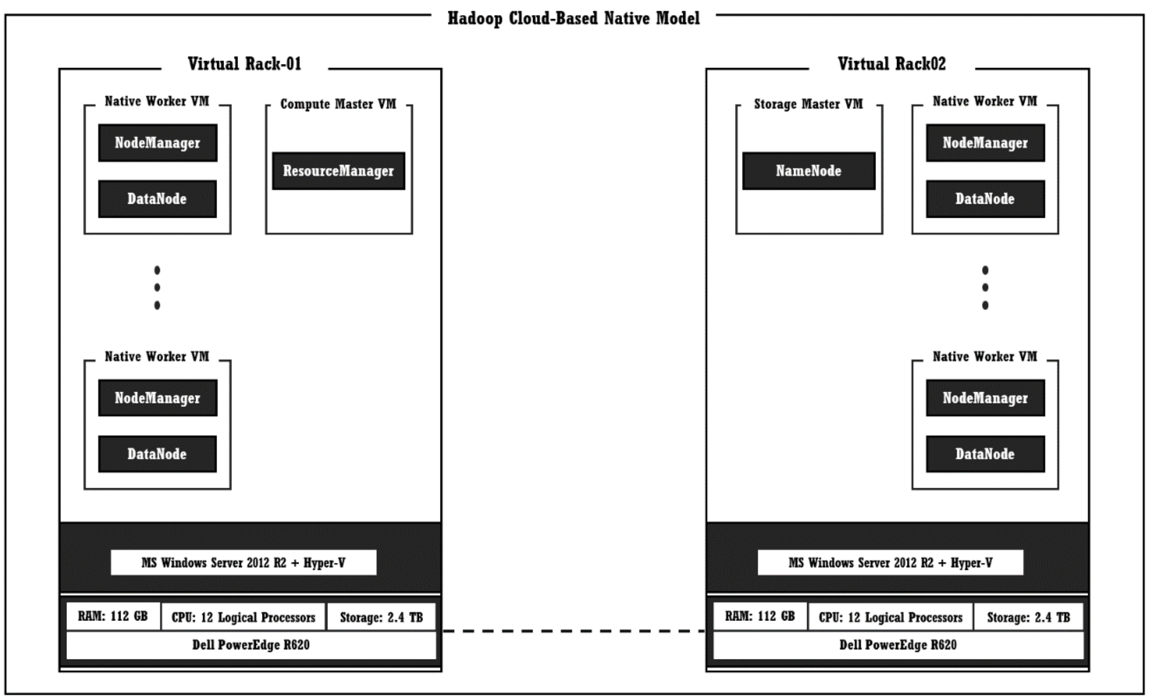


Figure 9 Hadoop Cloud-Based Native Model

4.1.3.2 Deploying Hadoop as a Proposed Model

If we choose to conduct the experiments on the Proposed Model, as shown in Figure 10, we build a cloud-based Hadoop cluster using proposed workers in which a VM has either a DataNode worker or NodeManager worker.

Physical networking has a key effect on the performance of the decoupled model (Proposed Model). There are two types of networking overhead. One is representing in the Native model which is between the storage and computing within the same VM and this type has a minimal networking overhead as it a virtual networking. However, measuring the exact effect is not an easy job as Hadoop cluster depends on complicated algorithms that specify the location of the storage worker that has the replicated blocks that need to be transferred.

That is, the source of the transferred replicated blocks is changeable.

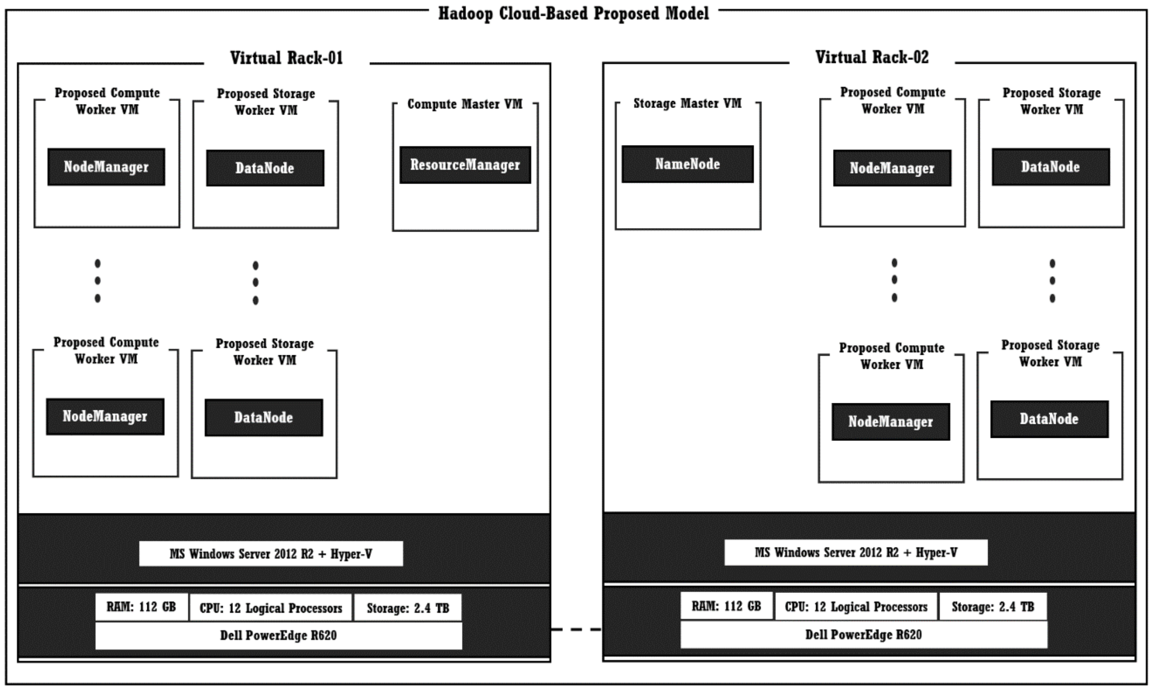

Hadoop Cloud-Based Proposed Model
Virtual Rack-01
Proposed Compute Worker VM
NodeManager
Proposed Storage Worker VM
DataNode
Compute Master VM
ResourceManager
Proposed Compute Worker VM
NodeManager
Proposed Storage Worker VM
DataNode
MS Windows Server 2012 R2 + Hyper-V
RAM: 112 GB
CPU: 12 Logical Processors
Storage: 2.4 TB
Dell PowerEdge R620
Virtual Rack-02
Storage Master VM
NameNode
Proposed Compute Worker VM
NodeManager
Proposed Storage Worker VM
DataNode
Proposed Compute Worker VM
NodeManager
Proposed Storage Worker VM
DataNode
MS Windows Server 2012 R2 + Hyper-V
RAM: 112 GB
CPU: 12 Logical Processors
Storage: 2.4 TB
Dell PowerEdge R620

Figure 10 Hadoop Cloud-Based Proposed Model

4.1.4 Benchmark suite

There are many Hadoop benchmarking suits that are invaluable in assessing cluster performance. We used Hibench-6.0 [66] as a benchmark suite to evaluate the performance of the Native Model as well as the Proposed Model. It is a comprehensive and very popular Hadoop benchmark tool that was developed by Intel to test Hadoop clusters. It consists of a set of Hadoop workloads that contain synthetic micro-benchmarks as well as real-world Hadoop applications [67]. It helps in evaluating the performance of Hadoop cluster by identifying the completion time, throughput, and resource utilization.

4.2 Experimental Methodology

For a deployed model, native or proposed, we run a set of experiments that all follow the same series of phases. In this section, we show the approach we followed and used throughout all the conducted experiments to ensure their comparability. Figure 11 shows the phases we were repeating while performing each individual experiment. When the platform is ready with an intended Hadoop cloud-based model, the phases of running experiments start by utilizing HiBench benchmark suite as follows:

For each deployed Hadoop Model, we were supposed to select a workload type to start doing its relevant experiment. We were performing two sets of experiments. First set of experiments concerning the first workload type (WordCount) which is a CPU-Intensive workload, and the other set of experiments with regards to second workload type (DFSIOE) which is an I/O-Intensive workload.

After selecting a workload type, we configure the experiment's dataset input size parameter with the values of 100 GB, 200GB, or 400GB, consecutively. Then, dataset generation for a corresponding data size parameter starts immediately.

We run the experiment relevant to the latest prepared workload, and then repeat it for three consecutive times to gain more confidence in the collected results. After that, we take the average value as a representative result. The next step could be one of the following three options:

- Go back to the step of workload preparation to configure another dataset size and generate the corresponding synthetic dataset. Then, continue doing the subsequent steps.

- Go back to the first step to select another workload. We go with this option in case we finish repeating one experiment for three times.
- Go to the phase of deploying another Hadoop Model to start conducting another set of experiments. This option is selected in the case where all experiments for the two different workloads are finished.

After conducting each experiment, a basic analysis was done through comparing the latest result with the previous results to start accumulating the whole picture about the findings. On the other hand, at the end of running a set of experiments relevant to a specific workload, we were performing a comprehensive investigation and evaluation on collected results to be able to see the correlation of a workload results on different models.

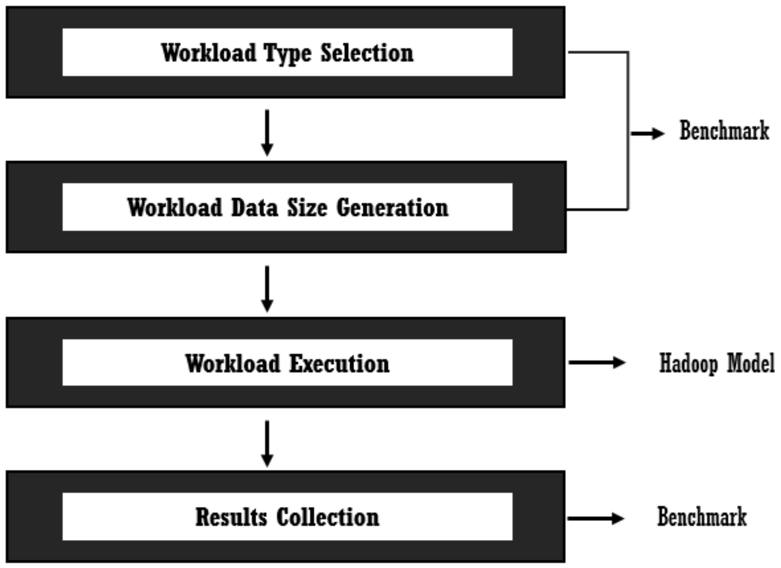


Figure 11 Experimental Setup Approach

4.3 Resource Distribution

Having comparable Hadoop clusters is a key to getting right results and valid conclusions. Therefore, we thought about the factors that should be taken into consideration to assure comparison validity. A total number of hardware resources and the number of running workers in each cluster are the factors that we considered while conducting experiments.

We have three kinds of resources, namely RAM, CPU, and Storage. The allocated resources for master VMs and for the machine that hosts the hypervisor are the same in the Native and in the Proposed Models, i.e., they are considered a fixed parameter which will not be changing in all experimental scenarios. Therefore, the available resources are denoted by RAM_T to denote the total number of available RAM. Similarly, CPU_T and $Storage_T$ denote the total CPU and Storage available for the Hadoop environment. That means that worker VMs will be allocated all available resources as shown in Equation 1.

$$\text{Number of Worker VMs} = \text{RAMT} + \text{CPUT} + \text{StorageT} \qquad \textbf{Equation 1}$$

For the Native Model, we divided the number of available resources equally between the VMs. Hence, each VM will be allocated resources according to:

$$\text{Allocated\_Resources} = \frac{\text{Total\_Resources}}{\text{VMs\_Num}} \qquad \textbf{Equation 2}$$

With regards to the Proposed Model, resources are distributed among worker VMs. A worker VM having Compute Slave is denoted by $WVM_{C.}$ Likewise, a worker VM having a Storage Slave is denoted by $WVM_{S.}$ Therefore, resources are distributed among these workers following:

$$(\text{Allocated-Resources-WVM}_S) + (\text{Allocated-Resources-WVM}_C) = \text{Total-Resources}$$ **Equation 3**

4.4 Hadoop Experimental Scenarios

As depicted in Figure 12, we have four different experimental scenarios. The first one was configured based on the Native Model, and the remaining three were configured based on the Proposed Model. For the Native Model and because of its coupling nature, we had no option but to configure one experimental scenario with the default Hadoop settings in which the computation workers and the storage workers are the same. So, each virtual machine has one computation worker called NodeManager, and one storage worker called DataNode. As for the Proposed Model and due to the decoupling of computation and storage module, we had the flexibility to configure three different experimental scenarios. In the first experimental scenario, called Proposed1, we distributed the computation workers and the storage workers equally. So, we configured 8VMs with 8 NodeManagers, and 8 VMs with 8 DataNodes. That is, a single worker is mapped into one VM. In the second experimental scenario of the Proposed Model, we configured it to have the computation module bigger than the storage module. Therefore, we configured it with 12 NodeMangers and 4 DataNodes, one in each VM. The last experimental scenario, the computation module is less than the storage module: 4 NodeManagers and 12 DataNodes.

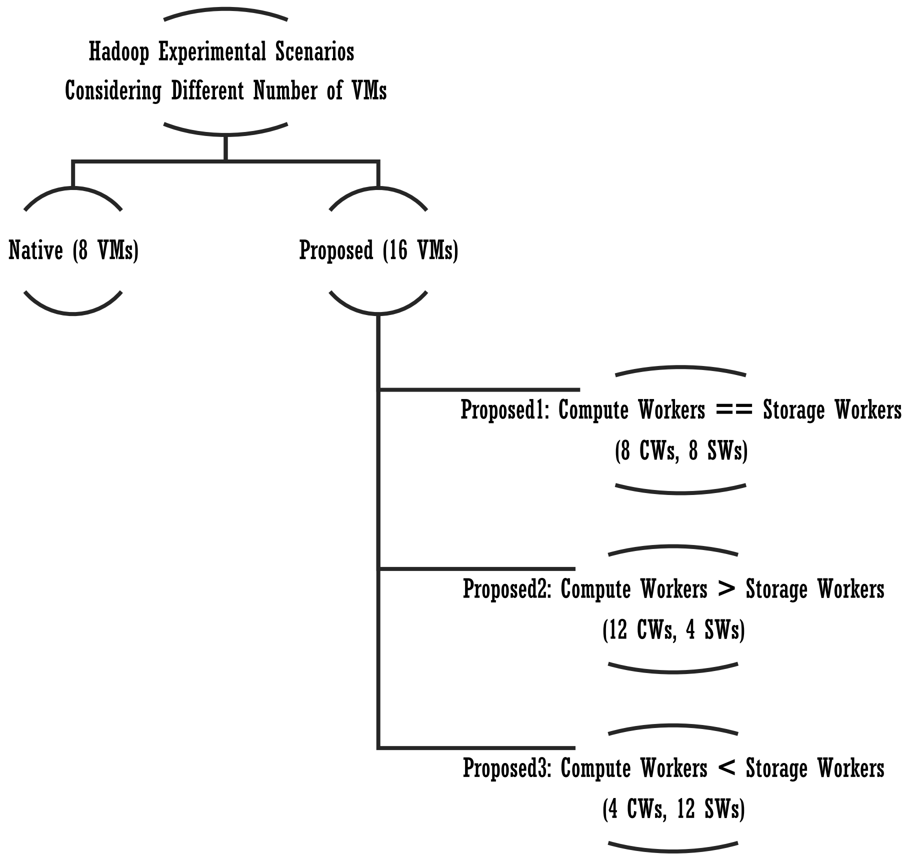

Hadoop Experimental Scenarios
Considering Different Number of VMs
Native (8 VMs)
Proposed (16 VMs)
Proposed1: Compute Workers == Storage Workers
(8 CWs, 8 SWs)
Proposed2: Compute Workers > Storage Workers
(12 CWs, 4 SWs)
Proposed3: Compute Workers < Storage Workers
(4 CWs, 12 SWs)

Figure 12 Hadoop Cluster Experimental Scenarios

A worker in Hadoop might be a DataNode or a NodeManager. DataNode is the storage worker, while NodeManager is a computing worker. Due to the decoupled nature of the Proposed Model, a VM contains only one worker. On the contrary, a VM in the Native Model contains two workers. Hence, if we intend to compare the performance of the two models taken into consideration the same name of workers, a VM in the Native Model should correspond to two VMs in the Proposed Model. The extra number of VMs in the Proposed Model are expected to cause more overhead on the hypervisor. Therefore, we configured the machine hosting the hypervisor with enough resources to minimize the impact of this overhead on the performance of the cluster. As shown in Table 3, the machine that hosts the hypervisor was allocated 24 GB RAM and 5 CPU cores with 250 GB storage. Similarly, we allocated many resources to the master servers as the expected overhead resulted from the communication between Masters and a bigger number of workers will be more. As shown in Table 4, each Master VM was allocated 24 RAM and 5 CPU cores with 250 GB storage.

Table 3 Allocated Resources for Hypervisor' Host Machine

Hadoop Model	Hypervisor	Ram	CPU	Storage
Native/Proposed	Hyper-V	24 GB	5 cores	250 GB

Table 4 Allocated Resources for the Hadoop Master VMs

Hadoop Model	Master Server VM	Ram	CPU	Storage
Native/Proposed	Storage Master VM (NameNode)	24 GB	5 cores	250 GB
Native/Proposed	Compute Master VM (ResourceManager)	24 GB	5 cores	250 GB

Table 5 shows the details of the physical resource distribution for the workers that comprise each experimental scenario. For the experimental scenario of the Native Model. We used Equation 2 to distribute the resources between its worker VMs. Each VM was allocated 16 GB RAM, 4 CPU cores, and 400 GB for storage. So, all VMs have the identical resources, and this is the typical distribution of the resources because the same VM participates in computing and storage, so it cannot be tuned differently. With respect to the Proposed Model, on the other hand, we have two groups of VMs. One group has a specific number of VMs dedicated for storage, and the other group has a specific number of VMs dedicated to computing. For that, a VM in one group can be allocated resources different than the allocated resources for a VM in the other group. Because of such flexibility, using the same resources used by the Native Model, we were able to shape the Proposed Model into three different forms or experimental scenarios:

A Proposed Model in which the number of NodeManager VMs same as the number of the DataNode VMs. Hence, the computation and storage capabilities are the same. This model will be referred to as Proposed1.

A Proposed Model that has a larger number of NodeManager VMs than the number of the DataNode VMs. Hence, the computation capability is stronger than the storage capability. This model will be referred to as Proposed2.

A Proposed Model that has a larger number of DataNode VMs than the number of the NodeManager VMs. Hence, the computation capability is weaker than the storage capability. This model will be referred to as Proposed3.

We used Equation 3 to distribute the resources between the worker VMs of three forms of Proposed Model. For the Proposed1, a NodeManager VM was assigned 10 GB RAM, 3 CPU Cores, and 100 GB storage. The total number of NodeManager VMs is 8. A DataNode, on the other hand, was assigned 6 GB RAM, 1 CPU Core, and 300 GB storage. The total number of DataNode VMs is 8. For the Proposed2, a NodeManager VM was assigned 9 GB RAM, 3 CPU Cores, and 50 GB storage. The total number of NodeManager VMs is 12. A DataNode was assigned 5 GB RAM, 1 CPU Core, and 650 GB storage. The total number of DataNode VMs is 4. For the Proposed3, a NodeManager VM was assigned 14 GB RAM, 5 CPU Cores, and 50 GB storage. The total number of NodeManager VMs is 4. A DataNode was assigned 6 GB RAM, 1 CPU Core, and 250 GB storage. The total number of DataNode VMs is 12.

In the Proposed Model experimental scenarios, due to the flexibility of resources distribution between NodeMangers and DataNodes, each experimental scenario is expected to fit more in a specific workload pattern.

Table 5 Assigned Resources for Hadoop Workers in Different Experimental Scenario

Hadoop Model	**Experimental Scenarios**	**Worker Name**	**Worker Type**	**No. of VMs**	**Worker VM Characteristics**		
					Ram	**CPU**	**Storage**
Native Model	Exp. Scen.#1 (Native)	Native-Workers	Compute /Storage	8	16 GB	4 Cores	400 GB
Proposed Model	Exp. Scen.#2 (Proposed1)	Proposed-Workers (CWs==SWs)	Compute -Worker	8	10 GB	3 Cores	100 GB
			Storage-Worker	8	6 GB	1 Core	300 GB
	Exp. Scen.#3 (Proposed2)	Proposed-Workers (CWs > SWs)	Compute -Worker	12	9 GB	2 Cores	50 GB
			Storage-Worker	4	6 GB	2 Cores	650 GB
	Exp. Scen.#4 (Proposed3)	Proposed-Workers (CWs < SWs)	Compute -Worker	4	14 GB	5 Cores	50 GB
			Storage-Worker	12	6 GB	1 Core	250 GB

4.5 Results and Evaluation

Our goal is to evaluate the performance of two comparable Hadoop models (the Native Model and the Proposed Model) using Hadoop MapReduce framework as a computation module. In our work, we aimed to test two different workload patterns, CPU-Intensive as well as I/O-Intensive workloads. For that, utilizing the benchmark tool (HiBench), we chose specific workloads that are relevant to our objectives. Those workloads are found under the micro-benchmark category and namely, the WordCount which is a CPU-Intensive workload, and the Distributed File System Input Output Enhanced (DFSIOE) which is an I/O-Intensive workload. Table 6 describes the characteristics of these two workloads.

Table 6 Benchmark Selected Workload Characteristics

Workload Name	Data Type	CPU Usage	I/O Usage (Read)	I/O Usage (Write)	Workload Type
WordCount	Synthetic Big Data	High	Low	Low	MapReduce
Enhanced DFSIO	Synthetic Big Data	Low	High	High	MapReduce

Both workloads are synthetic MapReduce representative applications. Synthetic workloads are preferable due to their portability and scaling comparing with big challenges in real-world workloads [68].

Section 4.5.1 explains the metrics used in evaluation experiments. Section 4.5.2 discusses and analyzes the results obtained from experiments relevant to CPU-Intensive workload (WordCount), while Section 4.5.3 discusses and analyzes the results obtained from experiments relevant to I/O-Intensive workload (DFSIOE).

4.5.1 Performance Evaluation Metrics

The metrics we used for evaluating the performance of the two models are: The completion time and cluster throughput as shown in Table 7. Completion time is the workload running time in seconds. The shorter time means better performance and respectively the longer time means worse performance. Throughput is the number of tasks completed per time unit, taking into consideration that workload composed of a set of tasks. For throughput, the higher means better performance and respectively the lower means worse performance. We chose those metrics because they are standard two metrics used by many related papers, such as the work done in [16] and [69]. Also, we are interested in knowing which Hadoop model is faster in executing a batch-oriented application with a higher throughput. These are the most important metrics for measuring the speed and the productivity of Hadoop model that process a batch-oriented application. Therefore, the best model should run the tested batch-oriented application in faster excution time and produce higher throughput.

Table 7 Metrics Used in Evaluating the Performance of Hadoop Two Models

Completion Time	Workload Execution Duration Time in Seconds (Shorter is Better)
Throughput	Completed Tasks Per Minute Overall the Cluster (Longer is Better)

4.5.2 Wordcount

This workload counts the occurrence of each word in the input data. Data is generated using the Hadoop RandomTextWriter program that is contained in the Hadoop cluster [67]. It is considered a representative of another typical category of real-world MapReduce jobs that extracts a small amount of interesting data from large dataset [66].

Following our experimental methodology that is shown in Figure 11, the WordCount workload was executed with three different datasets (100GB, 200GB, 400GB) in a consecutive way in each experimental scenario that is described in Figure 12.The obtained results (completion time in seconds) are shown in Table 8.

Table 8 WordCount Completion Time (Seconds)

Data-Size (GB)	**Native** (CWs == SWs)	**Proposed1** (CWs == SWs)	**Proposed2** (CWs > SWs)	**Proposed3** (CWs < SWs)
100	1141	1452	1360	1963
200	2313	2944	2782	3922
400	4588	5700	5420	7601

For the sake of simplicity and readability, these obtained values were initially normalized and then used for calculating the difference between the values obtained from the Native Model (the baseline) and the corresponding values obtained from the Proposed Model

(Proposed1, Proposed2, and Proposed3). Below is the explanation of the used calculations in simplifying the completion time results related to Wordcount application (the same calculations were used and applied in the subsequent sections which are relevant to DFSIOE-Read & DFSIOE-Write) :

With respect to values normalization, we selected the values of the Native Model experimental scenario as a baseline. Then, a value in an experimental scenario belongs to the Proposed Model (Proposed1, Proposed2, Proposed3) was being divided on the corresponding value of the baseline as shown in the below equation:

$$\text{Normalized Value} = \frac{\text{a Value in the Proposed Model}}{\text{The Corresponding Value in the Native Model}} \qquad \textbf{Equation 4}$$

For that, the normalized values of Proposed1, as an example, were calculated as follows: (1452 / 1141), (2944 / 2313), (5700 / 4588), which resulted in 1.27, 1.27, and 1.24, respectively. The complete normalized values are shown in Table 9.

Table 9 WordCount Normalized Completion Time

Data-Size (GB)	**Native** (CWs == SWs)	**Proposed1** (CWs == SWs)	**Proposed2** (CWs > SWs)	**Proposed3** (CWs < SWs)
100	1	1.27	1.19	1.72
200	1	1.27	1.20	1.70
400	1	1.24	1.18	1.66

Concerning the calculation of the difference between the corresponding values in the two models (based on the normalized values in Table 9), below is Equation 5 which was used for populating the normalized values comparison table (Table 10), which shows the time difference in (%) between the completion time of the experimental scenario of the native model, as a baseline, and the experimental scenarios of the proposed model.

$$\text{The difference} = \left(\frac{\text{a Value in the Proposed Model} - \text{the Corresponding Value in the Baseline}}{\text{The Value in the Proposed Model}}\right) X\ 100$$

Equation 5

Table 10 Comparison of Normalized Completion Time for WordCount (%)

Data Size (GB)	**Proposed1 (**CWs == SWs**)**	**Proposed2 (**CWs > SWs**)**	**Proposed3 (**CWs < SWs**)**
100	21.26	15.97	41.86
200	21.26	16.67	41.18
400	19.35	15.25	39.76

We investigated the results obtained from the experiments relevant to WordCount workloads to check the correlations amongst Hadoop Model, data-size, and the number of storage/compute workers. Moreover, to see the experimental scenario that fits more with CPU-Intensive applications such as Wordcount. The obtained results were evaluated from two perspectives. In one perspective, we compared the experimental scenarios to each other in general (to check the correlation between the completion time and Hadoop Model on all experimental scenarios). In the other perspective, we compared the obtained results in different data size to see the impact of increasing data-size on the performance of the clusters (to check the correlation of data-size and completion time on all experimental scenarios of Hadoop Models). For the former perspective, we started with evaluating the performance of the experimental scenario related to the Native Model with the set of the experimental scenarios that belong to the Proposed Model. Then, we compared the experimental scenarios of the Proposed Model to each other. For the sake of the latter perspective, Figure 14 was added to clearly show the correlation between the data-size and the model type.

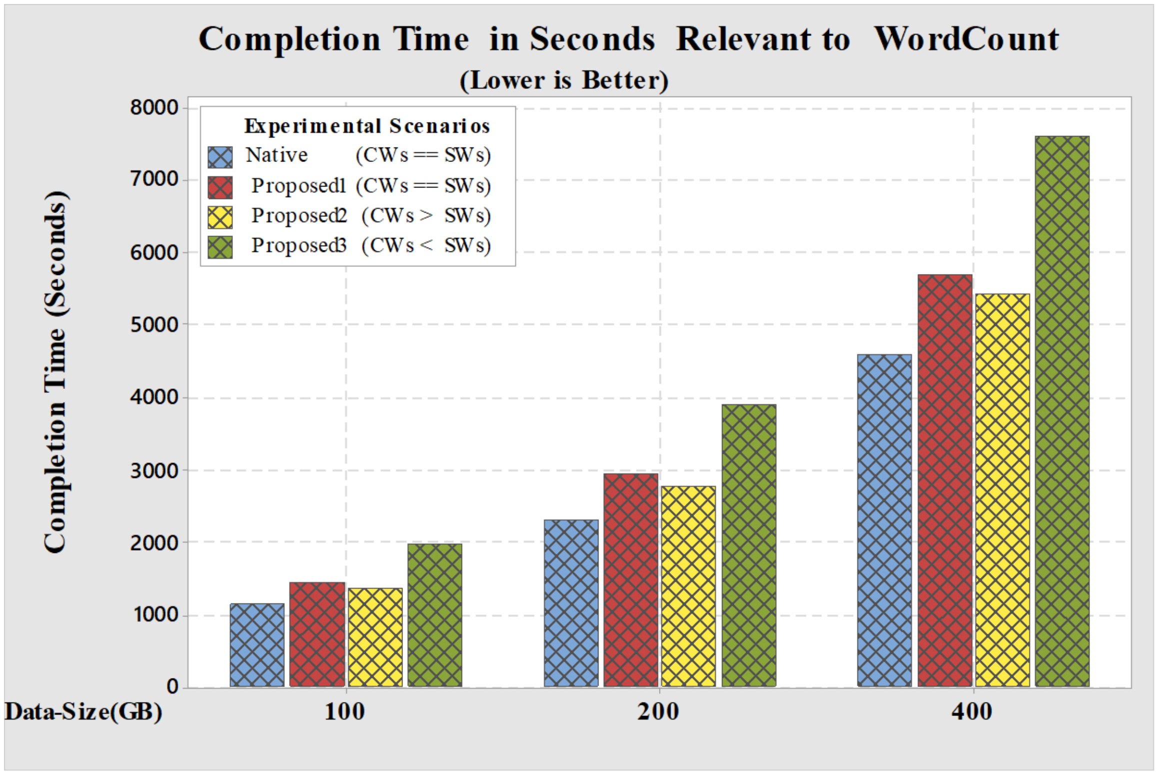


Figure 13 Completion Time in Seconds related to WordCount Workloads

As illustrated in Figure 13, the set of the experiments conducted on the Native Model perform better than the same set of the experiments when they were conducted on the experimental scenarios of the Proposed Model. However, the three experimental scenarios of the Proposed Model (Proposed1, Proposed2, Proposed3), each of which has a different performance than the other. That is, based on the values in Table 10 and taking Native Model results as the baseline, we observed the following:

For the Proposed Model (Proposed1) where the computing workers and the storage workers are the same, the performance is lower than the baseline with an average of 20%, while Proposed Model (Proposed2) where the computing workers are more than the storage workers, the performance is lower with an average of 15%. Proposed Model (Proposed3), on the other hand, performs slower with an average of 40%. That means that by comparing the experimental scenarios of the Proposed Model with each other, Proposed2 is the best, then the Proposed1 and after that the Proposed3. That makes sense as the investigated workload (Wordcount) is a CPU-Intensive that needs more computing resources, which is the case of (Proposed2) where the assigned compute workers are more than the corresponding ones on the other experimental scenarios (Proposed1, Proposed3). That means that the more computing workers, the better completion time.

Concerning the impact of increasing input data size on the results, it is observed that results are scaling almost linearly with the increase of the data-size. For Example, based on Figure 14 that was illustrated using the values in Table 11, it is shown that when an input data-size of the (Native, Proposed1, Proposed2, or Proposed3) gets doubled, the completion time also gets almost doubled. The same impact happened when the input data size gets increased 400%, the completion time increased around 400%.

Table 11 Completion Time Values When Scaling the Data-Size of the Workload (WordCount)

Data-Size	**Data-Size Scale**	**Native** (CWs == SWs) **Time-Scale**		**Proposed1** (CWs == SWs) **Time-Scale**		**Proposed2** (CWs > SWs) **Time-Scale**		**Proposed3** (CWs < SWs) **Time Scale**	
100GB	100%	1140.83	100%	1452.13	100%	1360.40	100%	1962.88	100%
200GB	200%	2312.78	202%	2943.62	203%	2782.32	182%	3922.15	200%
400GB	400%	4588.25	402%	5699.95	393%	5419.64	389%	7601.38	387%

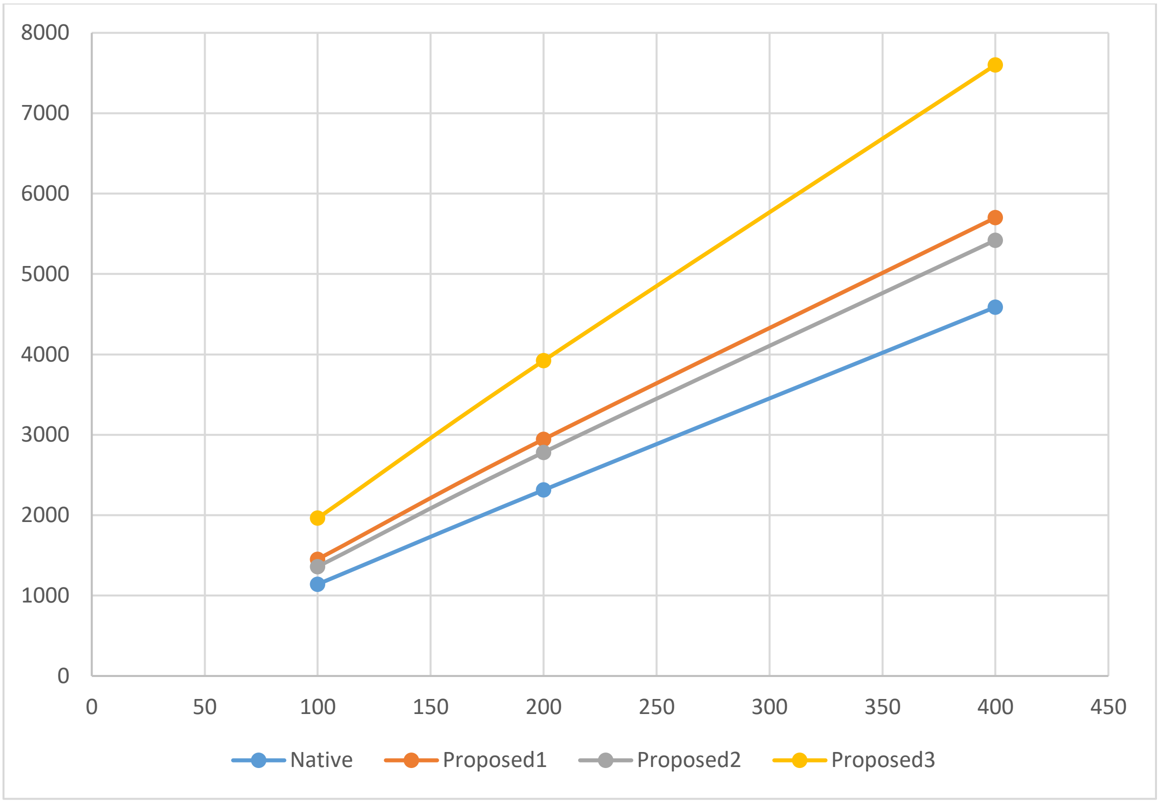


Figure 14 Impact of Scaling Data-Size on Completion Time (WordCount)

With regards to another metric we investigated (the throughput), the higher value means better performance. Related results are collected and illustrated in Figure 15, which depicts the throughput in each different experimental scenario. It is obvious that there is a correlation between Hadoop cluster throughput and the completion time, i.e., the less completion time the more throughput. For example, amongst the experimental scenarios of the Proposed Model (Proposed1, Proposed2, Proposed3), Proposed2 have the lower completion time and the higher throughput. We can conclude that for WordCount workload, Native Model gives the best completion time, and then the Proposed Model (Proposed2), which complies with the conclusion reported in [12]. So, we can gain all the appealing features of the Proposed Model by utilizing the experimental scenario (Proposed2) with as an expense of 15% lower performance.

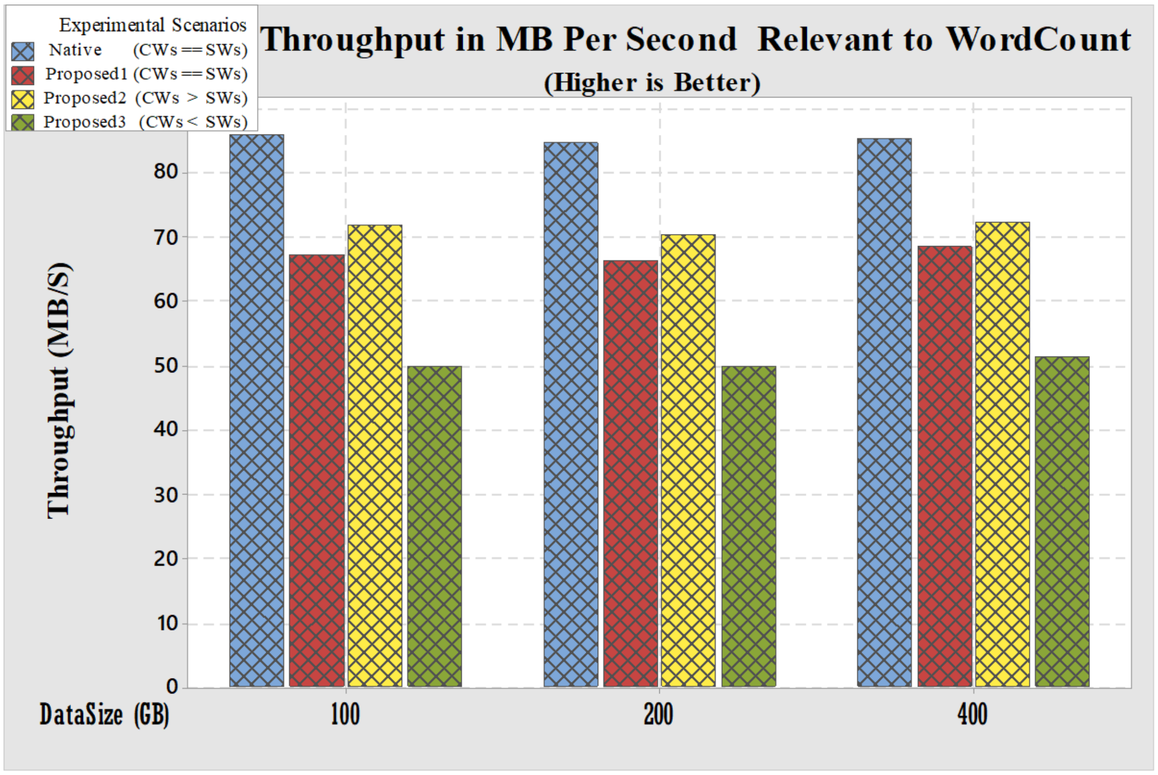


Figure 15 Throughput in MB Per Second Relevant to Wordcount Workloads

4.5.3 Enhanced DFSIO

Enhanced DFSIO tests the HDFS throughput of the Hadoop cluster by generating an enormous number of tasks doing writes and reads at once [66]. It is an extension of the TestDFSIO benchmark which was developed specifically for HiBench benchmark suite. The Enhanced DFSIO workload, which is part of HiBench, calculates the aggregated bandwidth by sampling the number of bytes read/written at fixed time intervals in each map task. As a result, when a map task is finished, a set of samples is brought [67]. During the reduce phase, the samples of each map task are linear included and re-sampled at a fixed plot rate. Then, we calculated the total of all the re-sampled points of all map tasks. The Enhanced DFSIO workload calculates the aggregated HDFS throughput by taking the average of the throughput value of each time slot in the steady period. In Enhanced DFSIO, when the number of concurrent map tasks at a time slot is above a threshold, say 50%, of the total map task slots, the slot is considered in the steady period. It consists of two parts: DFSIOE-Write and DFSIOE-Read.

4.5.3.1 DFSIOE Dataset Preparation

The Enhanced DFSIO workload takes four input configuration parameters. The first two define the number of files to be read or written. The other two parameters define the size of each file. As shown in Equation 6, the generation of Enhanced DFSIOE data size is the product of multiplying two parameters (the file size) and (the number of files to be read or written). Hence, to generate a dataset, there are two ways. In one way, we fix the first parameter (file size) and keep changing the number of files. In the other way, we can fix the parameter of (number of files) and keep changing the (file size).

$$\text{Enhanced DFSIOE Dataset-Size} = (\text{File-Size}) \text{ X } (\text{Number of Files}) \qquad \textbf{Equation 6}$$

To choose the proper calculation way that fits the nature of our work, we run three sets of DFSIO-Read experiments utilizing the Proposed Model with the experimental scenario (Proposed3). In each set of experiments, we were generating the same data size but using a different combination of the parameters in Equation 6 . That is, in some of these experiments we fixed the file size. And in the others, we fixed the number of files. Table 12 and Table 13 show the completion time in seconds for running experiments relevant to datasets 100GB, 200GB, 400GB when we fixed the number of files and kept changing the file size. Table 14 on the other hand, shows the completion time in seconds when we fixed the file size and kept changing the number of files. It is clearly shown that in case of fixing the number of files (Table 12, Table 13), there is no proportional difference in completion time when we move from 100GB to 200GB and 400GB. That is because of the way in which HDFS handles the process of reading a file [1]. For HDFS to read a file, it needs initially to open and locate the file controller block from the meta data of the storage master (NameNode). Then, read the blocks in parallel from DataNodes. After that, the file is closed. So, in case of having the same file but with bigger size, the difference in reading cost is only the cost of reading the extra blocks which is a matter of retrieval the blocks with a known location, i.e., the cost of opening, locating the block locations, and closing is the same for small or big file. Whenever the file is opened, and file block locations are known, the cost of reading extra blocks (bigger file size) is not considered big, especially because HDFS adopts a sequential block storing approach. For that, the more number of files, the more overhead on Hadoop cluster which is caused by more open, controller block locating, and closing for each file. Hence, for the rest of the work, we will be choosing the way of generating the dataset by fixing the number of file size and keep changing the number of files as this makes more sense for testing the overhead of I/O-Intensive workloads. This conclusion is verified by the work in [16], [24],[69] .

Table 12 Completion Time of DFSIO-Read- Fixed number of files (512) & Variation of file-size

Set of Exp.	**File-Size**	**Fixed No. of Files**	**Total Data-Size**	**Completion Time (Second)**	**Scaling**
Exp#1	200MB	512	100GB	725	100%
Exp#2	400MB	512	200GB	1000	137%
Exp#3	800MB	512	400GB	1979	273%

Table 13 Completion Time of DFSIO-Read- Fixed number of files (1024) & Variation of file-size

Set of Exp.	File-Size	Fixed No. of Files	Total Data-Size	Completion Time (Second)	Scaling
Exp#1	100MB	1024	100GB	1246	100%
Exp#2	200MB	1024	200GB	1459	117%
Exp#3	400MB	1024	400GB	1962	157%

Table 14 Completion Time of DFSIO-Read- Fixed file-sizes (400MB) & Variation of number of Files

Set of Exp.	Fixed File-Size	No. of Files	Total Data-Size	Completion Time (Second)	Scaling
Exp#1	400MB	256	100GB	479	100%
Exp#2	400MB	512	200GB	924	193%
Exp#3	400MB	1024	400GB	1961	410%

4.5.3.2 DFSIOE-Read

Following our experimental methodology that is shown in Figure 11, the DFSIOE-Read workload was executed with three different datasets (100GB, 200GB, 400GB) in a consecutive way in each experimental scenario that is described in Figure 12. The obtained results (completion time in seconds) are shown in Table 15, which demonstrates the DFSIOE-Read related results, which were obtained from the experiments conducted on the different experimental scenarios relevant to the Native and Proposed Models (Native, Proposed1, Proposed2, Proposed3). For the sake of simplicity, by utilizing Equation 4, we normalized those values by dividing each of which by the corresponding value of the baseline (Native) coming up with the values shown in

Table ***16***. In both tables, lower completion time value indicates experiment faster completion time. For that, a negative value means better performance, while a positive value means lower performance.

Table 15 DFSIOE-Read Completion Time (Seconds)

Data-Size (GB)	**Native (CWs == SWs)**	**Proposed1 (CWs == SWs)**	**Proposed2 (CWs > SWs)**	**Proposed3 (CWs < SWs)**
100	406.812	**453.1235**	**471**	**478.601**
200	760.386	**872.975**	**907**	**923.757**
400	1687.376	1851.5065	1919	1961

Table 16 DFSIOE-Read Normalized Completion Time (Seconds)

Data-Size	**Native** (CWs == SWs)	**Proposed1** (CWs == SWs)	**Proposed2** (CWs > SWs)	**Proposed3** (CWs < SWs)
100	1.00	1.11	1.16	1.18
200	1.00	1.15	1.19	1.21
400	1.00	1.10	1.14	1.16

By analyzing the values shown in Table 17 (populated utilizing Equation 5) and illustrated in Figure 16, it is noticed that the completion time correlates with Hadoop model type and the number of storage/computing workers. For example, taking the Native as a baseline, the experimental scenario (Proposed1) pays a completion time price of 11% (on average), while (Proposed2) & (Proposed3) pay the price of 14% and 15%, respectively. That means that Reading-Intensive workloads performs betters (faster completion time) when utilizing Hadoop Proposed Model with equal number of storage and computing workers because the reading process in Hadoop (HDFS) work in optimal way when utilizing the parallelism to the maximum, which is achieved by using the same number of storage workers

(DataNodes) and the computing workers (NodeManagers), so each computing worker can read block from the corresponding storage worker avoiding any delay caused by queuing. For the correlation between completion time and the data-size, as shown the values of Table 18 and illustrated in Figure 17, it is almost a linear correlation. The reason beyond that is, when we were changing the data size from 100GB to 200GB and to 400GB, we were doubling the number of files, which means that the overhead of opening the file, locating the controller block, and closing the file, is doubled.

Table 17 Comparison of Normalized Completion Time for DFSIOE-Read (%)

Data Size (GB)	Time Difference in (%) Between the Experimental Scenario of the Native, as a Baseline, and the Experimental Scenarios of the Proposed Model		
	Proposed1 (CWs == SWs**)**	**Proposed2 (**CWs > SWs**)**	**Proposed3 (**CWs < SWs**)**
100	9.91	13.79	15.25
200	13.04	15.97	17.36
400	9.09	12.28	13.79

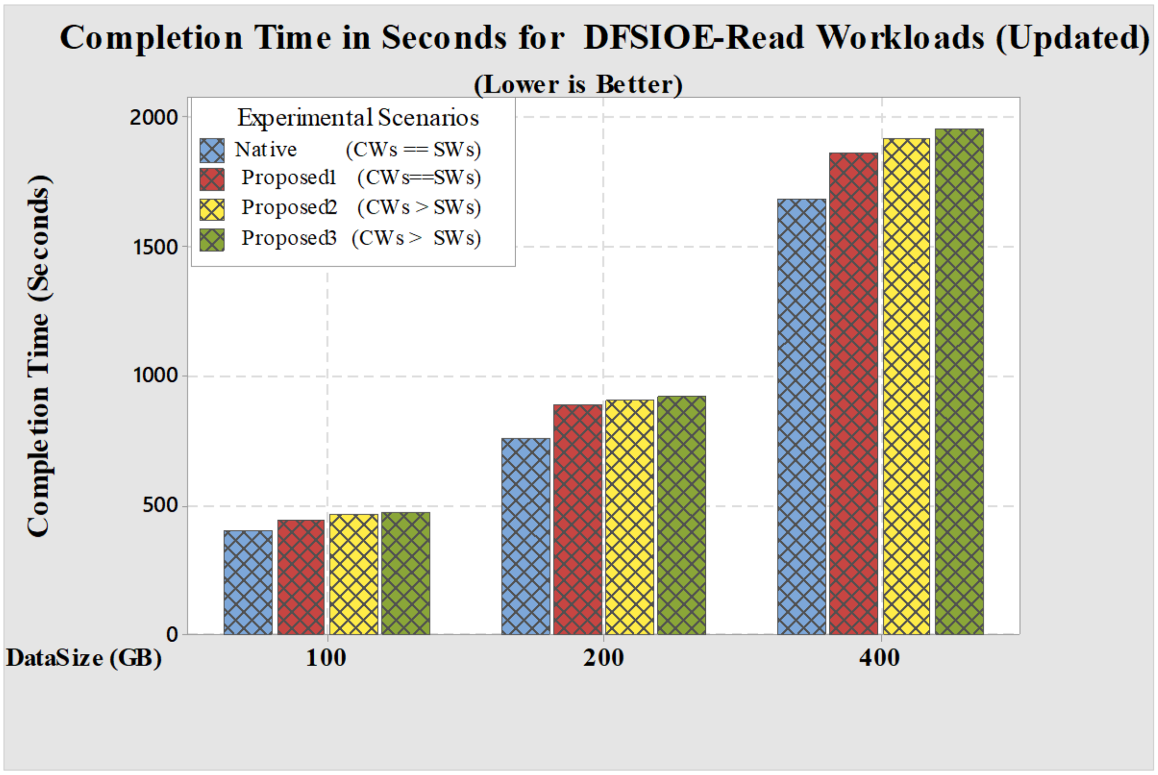


Figure 16 Completion Time in Seconds for DFSIOE-Read Workload

Table 18 Completion Time Values When Scaling the Data-Size of the Workload (DFSIOE-Read)

Data-Size (GB)	**Data-Size Scale**	**Native** (CWs == SWs) **Time-Scale**		**Proposed1** (CWs == SWs) **Time-Scale**		**Proposed2** (CWs > SWs) **Time-Scale**		**Proposed3** (CWs < SWs) **Time Scale**	
100	100%	407	100%	453	100%	471	100%	479	100%
200	200%	760	187%	873	**193%**	907	**193%**	924	**193%**
400	400%	1687	415%	1852	409%	1919	407%	1961	410%

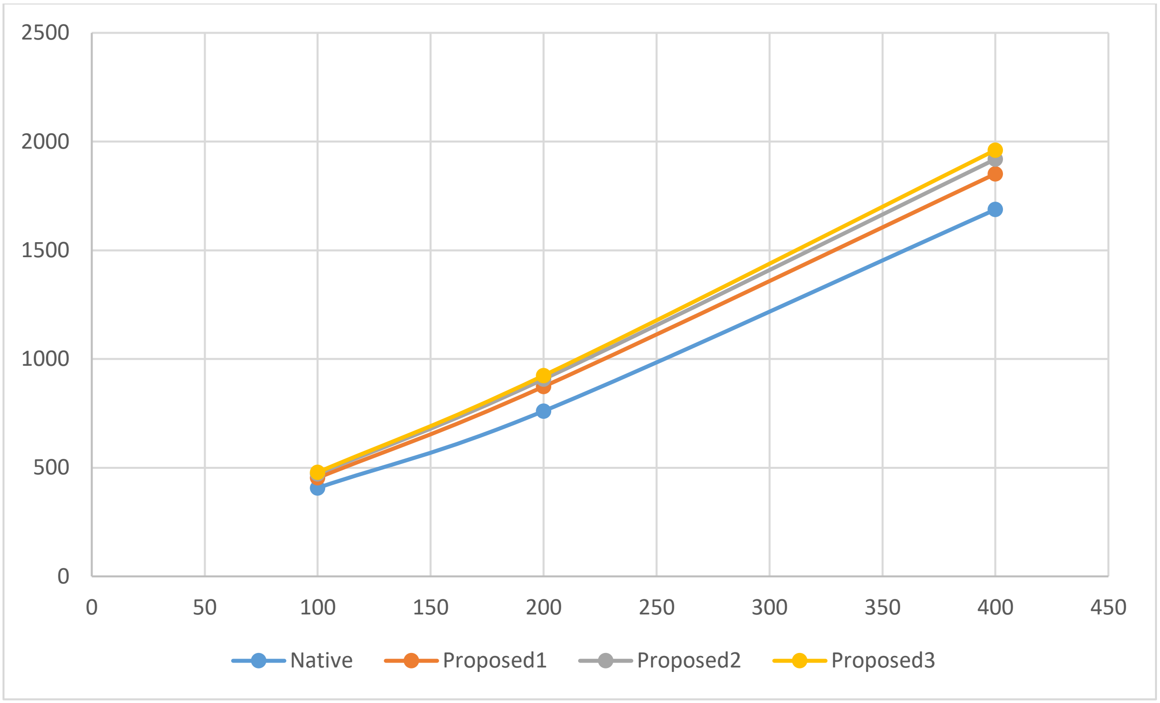


Figure 17 Impact of scaling data-size on completion time (DFSIOE-Read)

In general, the completion time of Read-Intensive workloads gets improved by using more DataNodes as this avoid the potential of network conflict and overhead that happens when more than one NodeManager try to read simultaneously from one DataNode. However, based our experiments, it seems that the threshold is to have the same number of storage/compute workers. That is because adding more storage workers will not be utilized as no more corresponding computing worker will be reading from it.

Figure 18 depicts the throughput relevant to each experimental scenario of Native and Proposed Model. The higher value the better performance. Based on that, we can see a simple and direct relationship between the completion time and throughput of every single experimental scenario, i.e., the lower completion time, the higher throughput value. That is consistent for all data-sizes and all experimental scenarios of Hadoop cluster Models.

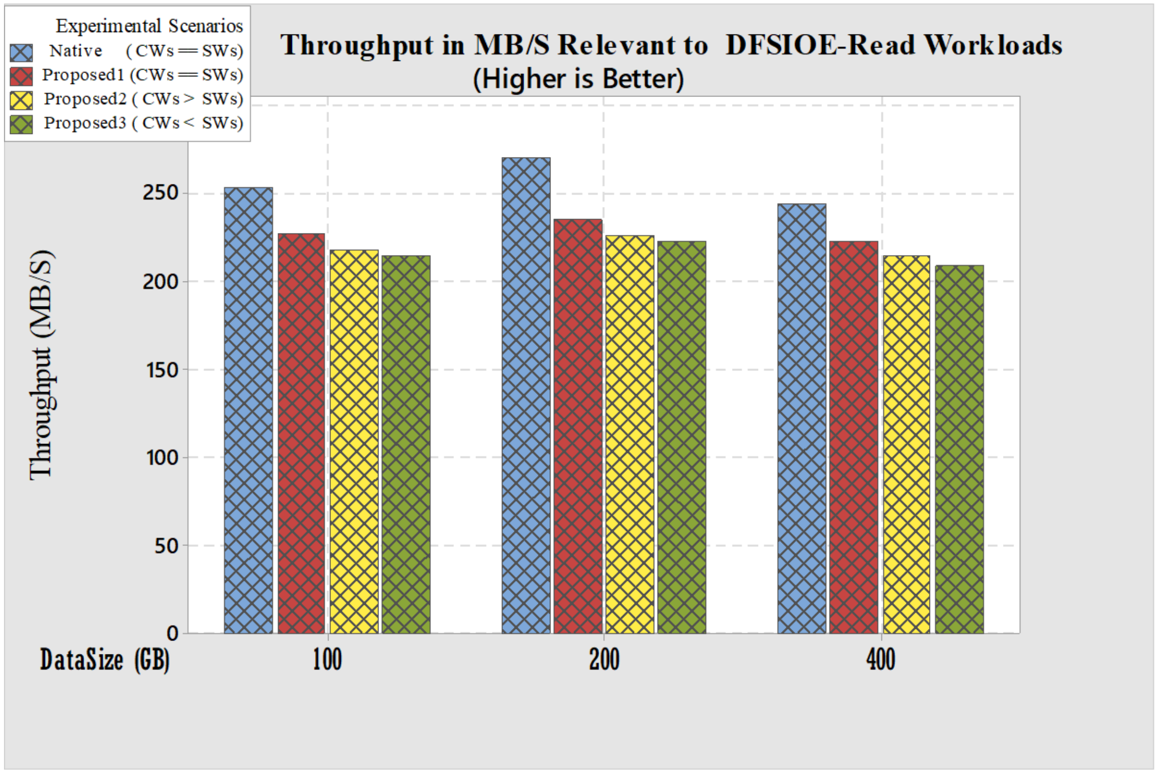


Figure 18 Throughput for DFSIOE-Read Workload

In brief, we can conclude that for Read-Intensive applications, it is recommended to utilize the Proposed Model (Proposed1) to get higher throughput and better completion time with a penalty price of 11%.

4.5.3.3 DFSIOE-Write

Following our experimental methodology that is shown in Figure 11, the DFSIOE-Write workload was executed with three different datasets (100GB, 200GB, 400GB) in a consecutive way in each experimental scenario that is described is described in Figure 12. Table 19 shows the collected results of all DFSIOE-Write related experiments that were conducted on all experimental scenarios relevant to the Native and Proposed Models. For the sake of simplicity and utilizing Equation 4, we normalized these values by dividing all by the corresponding value for a selected experimental scenario (Native) as a baseline, which results in the normalized values that are shown in Table 20. Lower completion time value indicates experiment faster completion time. Therefore, a negative value means better performance, while a positive value indicates less performance.

Table 19 DFSIOE-Write Completion Time (Seconds)

Data-Size	**Native** (CWs == SWs)	**Proposed1** (CWs == SWs)	**Proposed2** (CWs > SWs)	**Proposed3** (CWs < SWs)
100	915.128	1221.9175	1144.7	1568
200	1803.45	2409	2339.8	3429
400	4217.12	5419.3545	4901	7419

Table 20 DFSIOE-Write Normalized Completion Time

Data-Size (GB)	**Proposed1** (CWs == SWs)	**Proposed2** (CWs > SWs)	**Proposed3** (CWs < SWs)
100	1.34	1.25	1.71
200	1.34	1.30	1.90
400	1.29	1.16	1.76

By analyzing the values shown in Table 21 (populated utilizing Equation 5) and illustrated in Figure 19, we observed that the experimental scenario of the Native Model outperforms the experimental scenarios of the Proposed Model. However, the degree of difference in performance depends on the number of computing/storage workers in each experimental scenario of the Proposed Model. For example, in Proposed2 (CWs > SWs) the performance is better than Proposed1 (CWs == SWs), which is better than Proposed3 (CWs < SWs). The averages of their degraded performances are (18%,24%,45%), respectively. This observation explains that, for the workloads that are Write-Intensive, the more storage workers (DataNodes) the less completion time. For that, the experimental scenario

(Proposed2) is the best amongst the other experimental scenarios of the Proposed Model. Having that makes sense as for more DataNodes, the replication of the blocks will be on different DataNodes which causes more overhead. This overhead comes from the way the Hadoop handles the process of a block replication. As per the Hadoop placement policy [17], each DataNode takes care of block replication process into the subsequent DataNode in a pipelining way. Therefore, the more DataNodes cause more networking traffic between DataNodes. On the contrary, in case of fewer DataNodes, the process of replication will be done within the limited DataNodes avoiding further networking overhead.

Table 21 Comparison of Normalized Completion Time for DFSIOE-Write (%)

Data Size(GB)	Time Difference in (%) Between the Experimental Scenario of the Native, as a Baseline, and the Experimental Scenarios of the Proposed Model		
	Proposed1 (CWs == SWs**)**	**Proposed2 (**CWs > SWs**)**	**Proposed3 (**CWs < SWs**)**
100	25.37	20.00	41.52
200	25.37	23.08	47.37
400	22.48	13.79	43.18

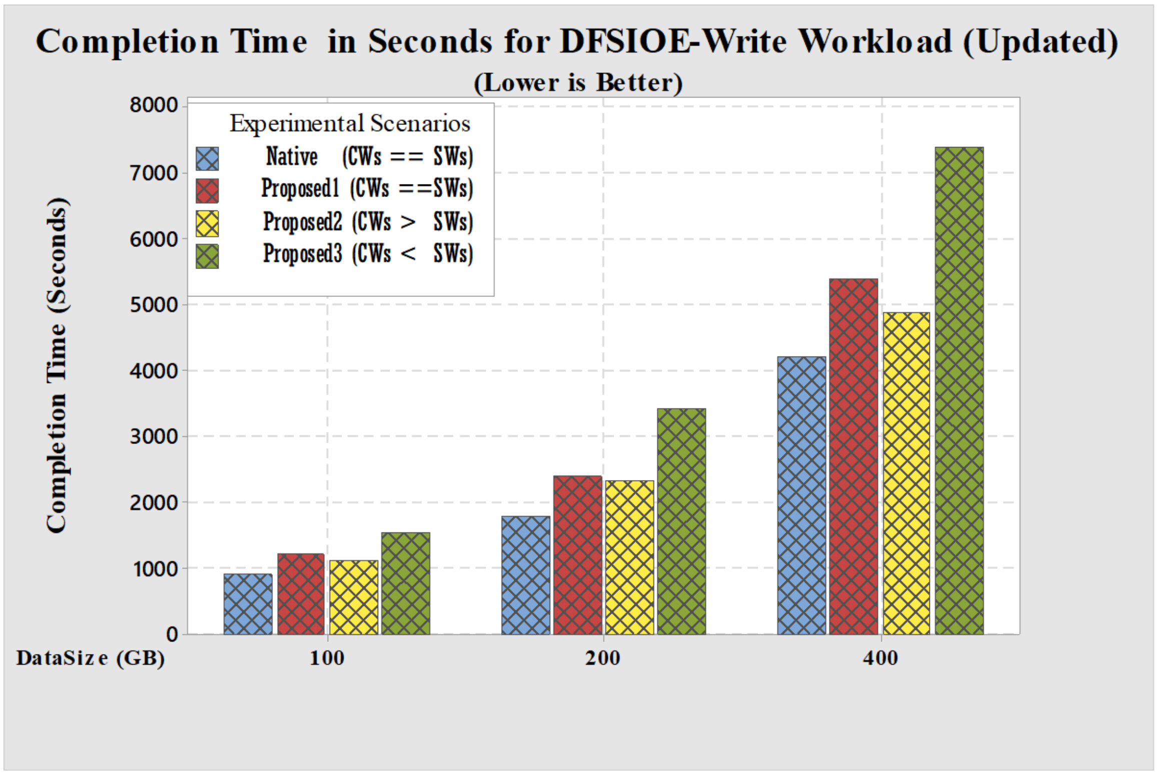


Figure 19 Completion Time in Seconds for DFSIOE-Write Workload

The values of Table 22 and illustrated in Figure 20 show the impact of increasing input data size on the results of DFSIOE-Write, it is observed that for Native, Proposed1, and Proposed2, results are scaling almost linearly with the increase of the data-size. However, for the Proposed3, the degree of scaling is not same as the other. This can be explained by the extra overhead that caused by the increased number of DataNodes in this experimental scenario. By investigating Figure 21, we remarked that throughput of the write jobs is correlated to the completion time of the write jobs in a consistent manner. In each write-intensive job, a high throughput relates to a lower completion time with the same ratio in all experimental scenarios belongs to the two investigated Hadoop Models.

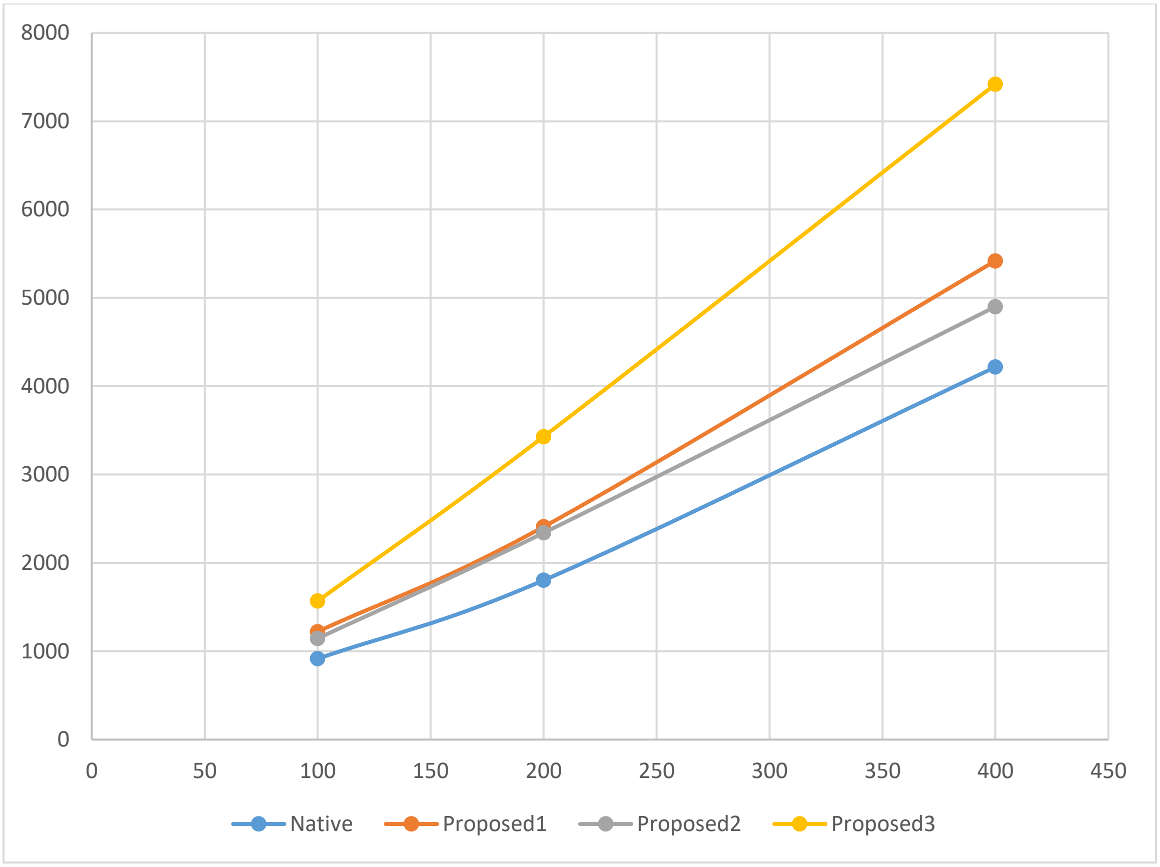


Figure 20 Impact of Scaling Data-Size on Completion Time (DFSIOE-Write)

Table 22 Completion Time Values When Scaling the Data-Size of the Workload (DFSIOE-Write)

Data-Size	**Data-Size Scale**	**Native** (CWs == SWs) **Time-Scale**		**Proposed1** (CWs == SWs) **Time-Scale**		**Proposed2** (CWs > SWs) **Time-Scale**		**Proposed3** (CWs < SWs) **Time Scale**	
100GB (baseline)	100%	915	100%	1222	100%	1145	100%	1568	100%
200GB	200%	1803	197%	2409	197%	2340	204%	3429	219%
400GB	400%	4217	461%	5419	444%	4901	428%	7419	473%

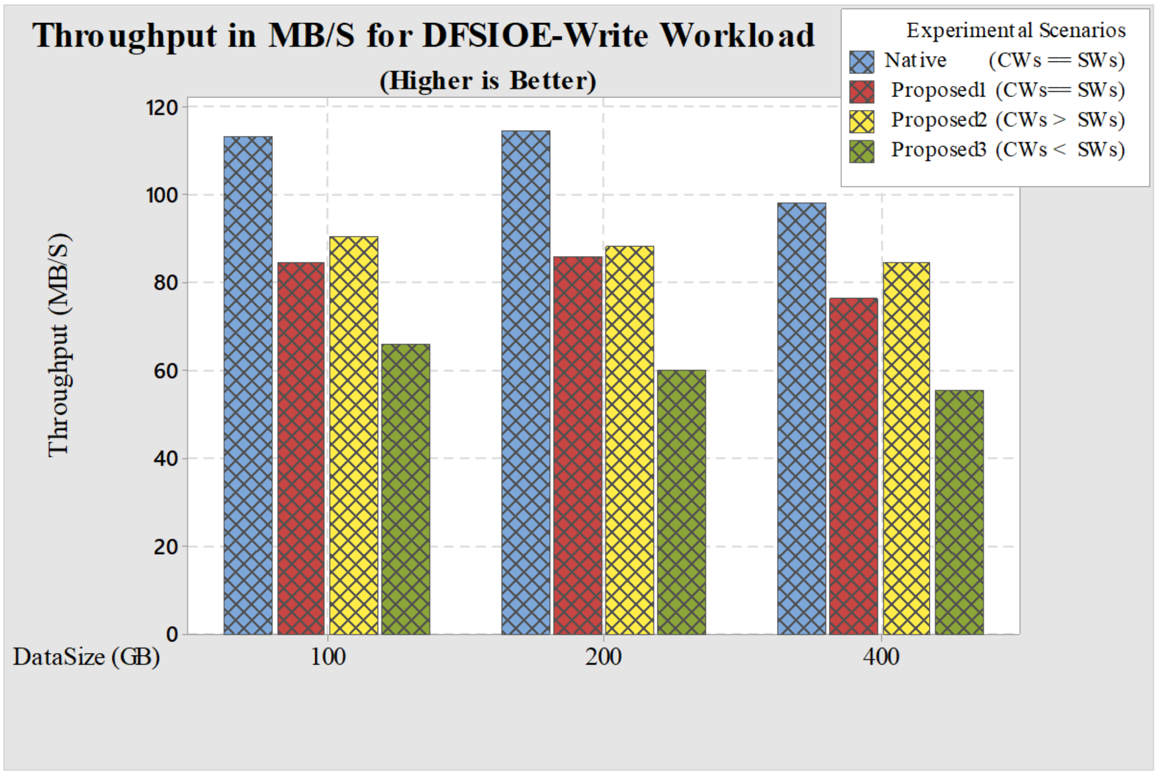


Figure 21 Throughput for DFSIOE-Write Workload

As shown in Table 23, the throughput of DFSIO-Write running in the Native and Proposed Models correlates with the data-size, the Hadoop Model, and the number of workers (various clusters of the Proposed Model). That is, if we take 100GB as a baseline, throughput is almost the same in 200GB for all Hadoop experimental scenarios except Proposed3 in which it is worse than the baseline by 9%. That gives us an insight that the number of storage worker causes a negative impact on the cluster performance and comes from the extra overhead of writing replicated blocks in distinct workers that are located in different physical hosts. For 400GB, on the other hand, the throughput of all Hadoop experimental scenarios (Native, Proposed1, Proposed2, Proposed3) is worse than the baseline by (13%, 9%,7%, 15%), respectively. Given that the replication factor of the stored blocks is three, having a dataset of 400GB means that there are another two replicas each of which is 400GB, so the total is 1200GB. Handling such huge dataset with only four computing workers with limited computing resources makes the performance starts degrading when the cluster storage capacity exceeds 40% of the total storage.

It is also noticed (as shown in Table 24) that the experimental scenario (Proposed2) has the best throughput amongst the remaining experimental scenarios of the Proposed Model, i.e., Proposed1 and Proposed3. That gives us an insight that having Hadoop cluster with more computing workers and fewer storage workers positively affect the throughput of the workloads that are write-intensive.

Table 23 Correlation of Native and Proposed Modes with Data-Size

	Native	**%**	**Proposed1**	**%**	**Proposed2**	**%**	**Proposed3**	**%**
100 GB	113	100	85	100	91	100	66	100
200 GB	115	102	86	101	89	98	60	91
400 GB	98	87	77	91	85	93	56	85

We can conclude that by utilizing (Proposed2) as an experimental scenario in the Proposed Model, we can get all the Proposed Model benefits at the expense of 18% lower performance.

Table 24 Correlation of Data-Size with Native and Proposed Models

	100 GB		**200 GB**		**400GB**	
Native	113	100%	115	100%	98	100%
Proposed1	85	75%	86	75%	77	78%
Proposed2	91	80%	89	77%	85	86%
proposed3	66	58%	60	53%	56	57%

4.6 Concluding Remark

To obtain all the appealing SOA features that are integral to the Proposed Model, an acceptable performance price can be paid compared with the corresponding performance for the Native Model. For CPU-Intensive applications, the sacrificed performance is around 15% by utilizing the Proposed Model (Proposed2), whereas for Read-Intensive applications, the Proposed Model (Proposed1) can outperform the Native Model with an average of 11%. The Write-Intensive applications, on the other hand, can utilize the Hadoop Proposed Model (Proposed2) at the expense of 20% lower performance.

Table 25 Suitability Recommendations for Proposed Model Usage

Workload pattern	**Proposed Model Experimental Scenario**	**Performance Penalty**
CPU-Bound	Proposed 2	15%
Read-Intensive	Proposed 1	11%
Write-Intensive	Proposed 2	20%

4.7 Proposed Model Limitations and Trade-offs

Limitations of this work are as follows: No changes have been made for evaluation experiment to accommodate security. Another limitation, measuring network traffic overhead has not been considered. Moreover, this work is limited to the specifically used cloud technologies (such as Hyper-V), the used Hadoop Distribution (HDP), and limited to the experimental environment which consists of two physical servers and one physical switch.

In the approach of (Proposed Model), we gained flexibility and elasticity in exchange for acceptable performance penalty. Importance of flexibility appears in situations where the workload pattern is known. It is very common nowadays for workloads to be pre-defined. Therefore, if a workload is CPU-Intensive or Write-Intensive, then the Proposed Model (Proposed2), where computing module has more workers is the best option. However, in case of workload pattern is Read-Intensive, then the Proposed Model (Proposed1), where computing and storage modules have the same number of workers is the best choice. The importance of Proposed Model elasticity features appears in situations where the workloads running on top of the clusters need to continue running while scaling out the cluster.

CHAPTER 5

CONCLUSION AND FUTURE WORK

The research work that has been done on big data is very huge. However, with respect to the area of framework architecture that handles big data, it is still in its infancy compared with other matured big data research areas. This motivated us to explore the current big data solutions and specifically looking at the way the framework components interact with each other. We found that the most popular and the de facto standard solution is Hadoop. For that, we realized it is worthy to spend all our efforts on such common framework. Exploring Hadoop's core components and how they interact with each other, we found that the core concept of Hadoop default architecture is data locality. It means that instead of moving the data into central computation entity, the computation is moved into the data entities in a distributed manner. Therefore, the data transfer networking overhead will be minimal. That was a valid consideration as the networking and storage mediums were the bottleneck of the performance. However, in the current days, due to the major advancement in networking and storage speed and efficiency, the data locality is not a bottleneck. We were motivated to challenge the default configuration of Hadoop by proposing a model that breaks the linkage of computing and storage to open the door for many enhancement features that can be gained consequently.

Decoupling Hadoop computation and storage modules is opening opportunities for gaining all Service-Oriented-Architecture attracting features such as flexibility, resource utilization efficiency, and security. So, our research goal has been set to explore the notion of providing Hadoop computing and storage as independent services. Below is a table that summarizes the difference between the Hadoop Native Model and the Hadoop Proposed Model.

Table 26 The Native Model vs The Proposed Model

	The Native Model	**The Proposed Model**
1	Rigid: we are forced to add or remove the computing and the storage workers at once.	Flexible: we can add or remove a computing or a storage worker independently.
2	inefficient in resource utilization	Utilize resources efficiently
3	Adding specific security rules for storage workers will affect the performance of the whole cluster.	Adding specific security rules for storage will affect only the machines that host the storage workers.
4	No way for supporting storage nor computing elasticity.	Support elasticity for computation. It can’t be for storage because of the overhead and time consuming of storage rebalancing process.
5	Virtual environment is not utilized efficiently.	Virtual environment is utilized.
6	Mapping of workers and VMs is general.	Mapping of workers and VMs is specific.

A big data solution can be deployed either in a bare-metal infrastructure or on the cloud. Deploying it as a bare-metal needs pretty expensive in terms of human and hardware requirements. On the contrary, deploying it on the cloud is cost-effective, most popular, and recent direction that is adopted by big corporations. Furthermore, many of the interested parties in Hadoop deployment cannot afford the expense of physical big data infrastructure. Consequently, we preferred deploying Hadoop cluster using cloud technologies as the end goal of our work is to provide big data solutions as a cloud-based service.

In this work, we explored the notion of providing cloud-based Hadoop computing or storage modules as independent services as opposed to the native Hadoop architecture in which computing and storage are coupled. For that, we evaluated the performance of two cloud-based models namely, the Native Model and the Proposed Model. By this, we investigate the feasibility of adopting the Proposed Model.

A large set of experiments were executed on four clusters (each cluster is a 2-host server with 18 virtual machines) containing the Hortonwork Distribution for Apache Hadoop running on MS Hyper-V. In addition, we built these four cloud-based Hadoop clusters with aim of looking for the correlations between HiBench selected workloads (Wordcount, DFSIOE), Data-Size (100,200,400), and Hadoop model type (Native, Proposed) so we can compile suitability recommendations for Proposed Model usage. Moreover, to see the expense of decoupling the compute and storage services. Specific evaluation environment setup is as follows:

1) Built a cloud-based Native Hadoop cluster (Prototype for Hadoop Native Model).
2) Built cloud-based Proposed Hadoop clusters (Prototype for the Proposed Model):
    a. Proposed1 (Computation Module equal to the Storage Module)
    b. Proposed2 (Computation Module larger than the Storage Module)
    c. Proposed3 (Computation Module smaller than the Storage Module)
3) Run set of experiments on each cluster using synthetic big data workloads types: CPU-bound and I/O-bound.

Evaluation of two comparable cloud-based Hadoop models namely, the Native Model and the Proposed Model gave us a solid answer for the performance of each model in terms of throughput and completion time. We knew the feasibility of implementing the Proposed Model because we knew the paid cost in exchange for gaining all SOA capabilities such as elasticity, security, and flexibility. The expected impact of decoupling was acceptable for all the experiments executed on various experimental scenarios of the Proposed Model (Proposed1, Proposed2, Proposed3).

Using the approach of (Proposed Model), we gained flexibility and elasticity in exchange for acceptable performance penalty. Importance of flexibility appears in situations where the workload pattern is known. It is very common nowadays for workloads to be pre-defined. Therefore, if a workload is CPU-Intensive or Write-Intensive, then the Proposed Model (Proposed2), where computing module has more workers is the best option. However, in case of workload pattern is Read-Intensive, then the Proposed Model (Proposed1), where computing and storage modules have the same number of workers is the best choice. The importance of Proposed Model elasticity features appears in situations where the workloads running on top of the clusters need to continue running while scaling out the cluster.

5.1 Conclusion

In our exploratory journey of providing Hadoop as storage and computing services, we have found that the Proposed Model is opening the door for independent services. To obtain all the appealing SOA features that are integral to the Proposed Model, an acceptable performance price can be paid compared with the corresponding performance for the Native Model. For CPU-Intensive applications, the sacrificed performance is around 15% by utilizing the Proposed Model (Proposed2), whereas for Read-Intensive applications, the Proposed Model (Proposed1) can outperform the Native Model with an average of 11%. The Write-Intensive applications, on the other hand, can utilize the Hadoop Proposed Model (Proposed2) at the expense of 20% lower performance.

Based on the analysis of the experiments on the Native and Proposed Models, the Proposed Model is a better option, specifically for workloads that are either I/O- or CPU-Intensive. It is obvious that in exchange for the appealing features that are gained by using the Proposed Model, the performance overhead is considered minor.

5.2 Future Work

In our future work, we plan to perform further experiments after building a prototype using state-of-the-art technologies and tools. Using the latest networking technologies such as Optical Fiber and Switch Network adapters that support a speed of 10 Gbps will minimize the networking overhead to the minimum. Consequently, the performance of the Proposed Model will increase. We also plan to repeat the performance evaluation work on another cloud solution such as the OpenStack [70] framework. OpenStack is an open-source and free solution that is deployed as infrastructure-as-a-service (IaaS) and composed pools of networking, storage, and processing that are supported by provided by different vendors [71]. Another future work is performing a real-time processing for a real data generated by the Internet of Things (IoT) sources which are the trend (Big Data and IoT) by utilizing another processing modules (Spark) of Hadoop. To conduct such experiments, we need a larger environment such as two physical racks instead of the two physical servers that were used in this work. Also, we intend to utilize another benchmark suite (Big Data Benchmark for BigBench) [72]. It is very popular big data benchmark that is supported and adopted by Transaction Processing Performance Council (TPC) as TPCx-BB [73]. Moreover, we intend to design experimental scenarios that consider (in addition to the total number of hardware resources) the number of VMs. Unlike the work we did, where a total number of hardware resources and the number of running workers in each cluster were the factors that we considered while conducting experiments.

References

[1] T. White, *Hadoop: The definitive guide*. 2015.

[2] "APACHE SPARK," *SPARK*. [Online]. Available: http://spark.apache.org/. [Accessed: 03-Aug-2017].

[3] "Storm Distributed Procesing Framework," *[1] "Storm Distributed Procesing Framework." [Online]. Available: http://storm.apache.org/.* [Online]. Available: http://storm.apache.org/. [Accessed: 02-Aug-2017].

[4] R. George Trujillo, Charles Kim, Steve Jones, *Virtualizing Hadoop: How to Install, Deploy, and Optimize Hadoop in a Virtualized.* .

[5] C. Paper, "SERVICE DELIVERY MODEL FOR BIG DATA AS A SERVICE DELIVERY MODEL FOR BIG," no. July, 2016.

[6] "Amazon." [Online]. Available: https://aws.amazon.com/emr/. [Accessed: 01-Aug-2017].

[7] "IBM InfoSphere BigInsights." [Online]. Available: http://www.ibm.com/analytics/us/en/technology/biginsights/. [Accessed: 01-Aug-2017].

[8] "Mircrosoft HDInsight." [Online]. Available: https://azure.microsoft.com/en-us/services/hdinsight/.

[9] L. J. Zhang and Q. Zhou, "CCOA: Cloud Computing Open Architecture," *2009 IEEE Int. Conf. Web Serv. ICWS 2009*, pp. 607–616, 2009.

[10] R. Morabito, J. Kjällman, and M. Komu, "Hypervisors vs. lightweight

virtualization: A performance comparison," *Proc. - 2015 IEEE Int. Conf. Cloud Eng. IC2E 2015*, pp. 386–393, 2015.

[11] A. Holmes, *Hadoop in Practice, Second Edition*. Manning, 2015.

[12] T. Ivanov, R. V. Zicari, and A. Buchmann, "Benchmarking virtualized hadoop clusters-Lecture Notes," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8991, pp. 87–98, 2015.

[13] "Google Cloud." [Online]. Available: https://cloud.google.com/. [Accessed: 01-Aug-2017].

[14] J. Li, X and Murray, "Deploying Virtualized Hadoop Systems with VMware vSphere ® Big Data Extensions TM," *Tech. White Pap. VMware Inc*, 2014.

[15] "Frankfurt Big Data Lab." [Online]. Available: http://www.bigdata.uni-frankfurt.de/. [Accessed: 02-Nov-2017].

[16] T. Ivanov, R. V. Zicari, S. Izberovic, and K. Tolle, "Performance Evaluation of Virtualized Hadoop Clusters," 2014.

[17] "Apache Hadoop." [Online]. Available: http://hadoop.apache.org/. [Accessed: 03-Aug-2017].

[18] H. Hu, Y. Wen, T. S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE Access*, vol. 2, pp. 652–687, 2014.

[19] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, p. 29, 2003.

[20] F. Chang *et al.*, "Bigtable," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 1–26,

2008.

[21] “Hadoop support for IBM GPFS.” [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/STXKQY_4.1.1/com.ibm.spectrum.scale.v4r11.adv.doc/bl1adv_hadoop.htm. [Accessed: 04-Nov-2017].

[22] “Hadoop Integaration with Microsoft Azure Storage Blob.” [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-azure/index.html. [Accessed: 04-Oct-2017].

[23] “Hortonworks Data Platform Hortonworks Data Platform : Ambari Reference Guide,” 2015.

[24] T. Ivanov, R. Niemann, S. Izberovic, M. Rosselli, K. Tolle, and R. V. Zicari, “Performance Evaluation of Enterprise Big Data Platforms with HiBench,” *Proc. - 14th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2015*, vol. 2, pp. 120–127, 2015.

[25] “DataStax http://www.datastax.com/.” [Online]. Available: http://www.datastax.com/nosql-databases/nosql-cassandra-and-hadoop. [Accessed: 02-Jul-2017].

[26] “Cloudera.” [Online]. Available: http://www.cloudera.com/. [Accessed: 01-Jul-2017].

[27] N. Yuhanna, “The Forrester Wave^{TM}: Enterprise ETL, Q1 2012,” *Forrester Res.*, vol. 59511, no. 27 February 2012, p. 17, 2012.

[28] “Hortonworks.” [Online]. Available: http://hortonworks.com/. [Accessed: 01-Jul-

2017].

[29] "MapR." [Online]. Available: https://www.mapr.com/. [Accessed: 01-Aug-2017].

[30] "Datameter." [Online]. Available: https://www.datameer.com/. [Accessed: 02-Jul-2017].

[31] "Pentaho." [Online]. Available: http://www.pentaho.com/solutions/hadoop. [Accessed: 01-Aug-2017].

[32] Y. He, X. Jiang, Z. Wu, K. Ye, and Z. Chen, "Scalability analysis and improvement of hadoop virtual cluster with cost consideration," *IEEE Int. Conf. Cloud Comput. CLOUD*, pp. 594–601, 2014.

[33] J. Buell, "A Benchmarking Case Study of Virtualized Hadoop Performance on VMware vSphere 5," *Tech. white Pap. VMware, Inc*, 2011.

[34] R. Magdon-Ismail, Tariq and Nelson, M and Cheveresan, R and Scales, D and King, A and Vandrovec, P and McDougall, "Toward an elastic elephant enabling hadoop for the Cloud," *VMware Tech. J*, vol. 3, no. 1, 2014.

[35] S. Ibrahim, H. Jin, L. Lu, L. Qi, S. Wu, and X. Shi, "Evaluating mapreduce on virtual machines: The hadoop case," *Springer*, no. 2009, pp. 519–528.

[36] H. González-Vélez and M. Kontagora, "Performance evaluation of MapReduce using full virtualisation on a departmental cloud," *Int. J. Appl. Math. Comput. Sci.*, vol. 21, no. 2, pp. 275–284, 2011.

[37] J. Fan, X. Li, C. H. Liu, J. Buell, G. Lu, and L. Lu, "Diagnosing virtualized hadoop performance from benchmark results: An exploratory study," *Proc. - 2014*

IEEE Int. Congr. Big Data, BigData Congr. 2014, no. 2, pp. 578–585, 2014.

[38] M. Ishii, J. Han, and H. Makino, “Design and performance evaluation for Hadoop clusters on virtualized environment,” *Int. Conf. Inf. Netw.*, pp. 244–249, 2013.

[39] G. Porter, “Decoupling storage and computation in Hadoop with SuperDataNodes,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, p. 41, 2010.

[40] J. Shafer, “A Storage Architecture for Data-Intensive Computing,” *Rice Univ.*, no. May 2010.

[41] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: State-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010.

[42] “Apache Ambari.” [Online]. Available: https://ambari.apache.org/. [Accessed: 03-Aug-2017].

[43] “Cloudera Manager.” [Online]. Available: https://www.cloudera.com/products/product-components/cloudera-manager.html. [Accessed: 09-Aug-2017].

[44] “Hadoop Distributed File System.” [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. [Accessed: 04-May-2017].

[45] “kosmosfs Distributed File System.” [Online]. Available: https://code.google.com/archive/p/kosmosfs.

[46] D. Beaver, S. Kumar, H. C. Li, J. Sobel, and P. Vajgel, “Finding a needle in Haystack : Facebook ’ s photo storage,” *Design*, no. October, pp. 1–8, 2010.

[47] “Microsoft Binary Large Object (BLOB).” [Online]. Available: https://azure.microsoft.com/en-us/services/storage/blobs/. [Accessed: 02-Aug-2017].

[48] “MESOS, a Resource Management System.” [Online]. Available: http://mesos.apache.org/.

[49] “kubernetes, a Resource Manager.” [Online]. Available: https://kubernetes.io/. [Accessed: 04-Jul-2017].

[50] “Apache Impala.” [Online]. Available: https://impala.apache.org/. [Accessed: 02-Jul-2017].

[51] “Apache Tez.” [Online]. Available: https://tez.apache.org/. [Accessed: 01-Jul-2017].

[52] “MapReduce Processing Model.” [Online]. Available: https://en.wikipedia.org/wiki/MapReduce.

[53] A. Hive, “Apache Hive.” [Online]. Available: https://hive.apache.org/.

[54] “Apache Pig.” [Online]. Available: https://pig.apache.org/. [Accessed: 03-Aug-2017].

[55] K. Bakshi, “Considerations for big data: Architecture and approach,” *IEEE Aerosp. Conf. Proc.*, pp. 1–7, 2012.

[56] “Coordination and Synchronization, ZooKeeper.” [Online]. Available: https://zookeeper.apache.org/. [Accessed: 01-Aug-2017].

[57] *Hadoop Essentials*. Packt Publishing Ltd, 2015.

[58] "Scalability." [Online]. Available: https://en.wikipedia.org/wiki/Scalability.

[59] "Elasiticity in Cloud Computing." [Online]. Available: https://en.wikipedia.org/wiki/Elasticity_(cloud_computing). [Accessed: 05-Jun-2017].

[60] "Deploying Virtualized Hadoop Systems with VMware vSphere ® Big Data Extensions TM."

[61] "Windows 2012, the Cloud OS Becomes Reality." [Online]. Available: https://blogs.technet.microsoft.com/serverandtools/2012/09/04/the-cloud-os-becomes-reality-windows-server-2012-now-available/.

[62] Microsoft, "Windows Server 2012 R2 Products and Editions Comparison," 2012.

[63] Microsoft, "Hyper-V Features Overview." .

[64] S. Dandu, "Data Center Server Virtualization Solution Using Microsoft Hyper-V," 2017.

[65] "Hortonworks Data Platform HDP." [Online]. Available: https://hortonworks.com/products/data-platforms/hdp/. [Accessed: 09-Aug-2017].

[66] "HiBench Benchmark Suite." [Online]. Available: https://github.com/intel-hadoop/HiBench. [Accessed: 03-Sep-2017].

[67] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis," *Lect. Notes Bus. Inf. Process.*, vol. 74 LNBIP, pp. 209–228, 2011.

[68] C. Baru, M. Bhandarkar, R. Nambiar, M. Poess, and T. Rabl, "Setting the direction

for big data benchmark standards," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7755 LNCS, pp. 197–208, 2013.

[69] T. Rabl, K. Sachs, M. Poess, C. Baru, and H. A. Jacobson, "Benchmarking Virtualized Hadoop_Clusters_2014_Springer_book_TODO," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8991, pp. 87–98, 2015.

[70] "OpenStack." [Online]. Available: https://www.openstack.org/. [Accessed: 09-Oct-2017].

[71] "OpenStack WikiPedia." [Online]. Available: https://en.wikipedia.org/wiki/OpenStack. [Accessed: 03-Sep-2017].

[72] "Big Data Benchmark for BigBench." [Online]. Available: https://github.com/intel-hadoop/Big-Data-Benchmark-for-Big-Bench. [Accessed: 04-Nov-2017].

[73] "TPCx-BB (Big Bench)." [Online]. Available: http://www.tpc.org/tpcx-bb/default.asp. [Accessed: 01-Aug-2017].

VITAE

Name	:Shawqi Mohammed Al-Maliki
Nationality	:Yemeni
Date of Birth	:1/1/1982
Email	:almulaiki@gmail.com
Address	:Taiz, Yemen

Education, Research, and Experience:

Academic Background :Bachelor of Science in Computer Science from King Faisal University in 2006 with GPA 4.70 out of 5

Research Interests :Big Data, Hadoop, Cloud Computing, Web Security, and Virtualization

Experience :Senior Full-Stack Web Developer & Web Project Manager, Alyaum Media House, Dammam, Saudi Arabia (October 2010 – May 2018)