A COMPARATIVE STUDY OF HADOOP AND SPARK FRAMEWORKS

A MASTER'S THESIS

IN

SOFTWARE ENGINEERING

ATILIM UNIVERSITY

BY

ARSAN MOHAMMED ALI ALI

JULY 2016

A COMPARATIVE STUDY OF HADOOP AND SPARK FRAMEWORKS

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

ARSAN MOHAMMED ALI ALI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF SOFTWARE ENGINEERING

JULY 2016

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof. Dr. Ibrahim Akman

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Ali Yazıcı

Head of Department

This is to certify that we have read the thesis A Comparative Study of Hadoop and Spark Frameworks submitted by Arsan Mohammed Ali ALI and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
Asst. Prof. Dr. Ziya Karakaya	Prof. Dr. Ali Yazıcı
Co-Supervisor	Supervisor

Examining Committee Members

Prof. Dr. Ali Yazıcı _____

Asst. Prof. Dr. Çiğdem Turhan _____

Prof. Dr. Erdoğan Doğdu _____

Date: 15.7.2016

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Arsan Mohammed Ali ALI

Signature:

ABSTRACT

A COMPARATIVE STUDY OF HADOOP AND SPARK FRAMEWORKS

ALI, Arsan Mohammed Ali

M.S., Software Engineering Department

Supervisor: Prof. Dr. Ali YAZICI

Co-Supervisor: Asst. Prof. Dr. Ziya KARAKAYA

July 2016, 63 pages

In this thesis, Apache Hadoop and Apache Spark are compared with respect to performance, resource usage, and speedups of four different Big Data workloads. The comparison is done by tuning the parameters with various combinations for the optimum performance of each framework. The evaluations show that, Spark outperforms Hadoop for WordCount, and Sort workloads. However, Hadoop outperforms Spark in Naive-Bayes workload, and there is no a remarkable difference in the performances for Grep workload.

Keywords: Big Data, MapReduce, Hadoop, Spark.

ÖZ

HADOOP VE SPARK SİSTEMLERİN KARŞILASTIRMALI ÇALIŞMASI

ALI, Arsan Mohammed Ali

Yüksek Lisans, Yazılım Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Ali YAZICI

Ortak Tez Yöneticisi: Y. Doç. Dr. Ziya KARAKAYA

Temmuz 2016, 63 Sayfa

Bu çalışma, büyük veri sistemlerden ikisi olan Hadoop ve Spark'ı karşılaştırarak incelemektedir. Bu çalışmaya dahil edilen sistemler ideal performanslarını gösterebilecek şekilde çeşitli parametrelerle ayarlanmış; performans, donanım kullanımları ve kullanılan donanım sayısının arttırılmasıyla oluşan hızlanma oranları dört farklı uygulamada değerlendirilmiştir. Değerlendirme sonuçları incelendiğinde Spark'ın WordCount ve Sort uygulamaları için daha iyi performans gösterdiği, Hadoop'un Naive-Bayes uygulamasında Spark'tan daha başarılı olduğu gözlenmiştir. Bununla beraber, Grep uygulaması için Hadoop ve Spark arasında göze değer bir fark görülmemiştir.

Anahtar Kelimeler: Büyük veri, MapReduce, Hadoop, Spark.

ACKNOWLEDGMENTS

I express sincere appreciation to my supervisor Prof. Dr. Ali Yazıcı for his guidance and insight throughout the research. Thanks also go to my co-supervisor Asst. Prof. Dr. Ziya Karakaya. Without their contributions, this thesis would never be as complete as it is for now. To my dear friend Mohammed Alayyoub, I offer sincere thanks for his continuous support during this period.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AM	-	Application Master
API		Application Programming Interface
BDGS		Big Data Generator Suite
CPU	-	Central Processing Unit
DAG		Directed Acyclic Graph
GB	-	Gigabyte
MB	-	Megabyte
OS		Operating System
OLAP		On-line Analytical Processing
OLTP		On-line Transaction Processing
RDD	-	Resilient Distributed Dataset
RM		Resource Manager
RQ	-	Research Question
SM	-	Systematic mapping
SQL		Structured Query Language
SSH	-	Secure Socket Shell
TB		Terabyte
YARN	-	Yet Another Resource Negotiator
3V		Volume, Velocity, Variety
5V		Volume, Velocity, Variety, Veracity, Value

CHAPTER 1

INTRODUCTION

1.1.Thesis Scope

The term "Big Data" refers to huge data sets that exceeds the ability of commonly used software tools to gather, manage, and process the data in within acceptable amount of time (Patel, 2012)

Traditional data processing models are struggling as the volume of data to be processed is increased in a tremendous speed. The data generated by applications such as web applications, social media, sensors data, and so on are no longer structured. The emergence of Big Data processing is the natural outcome of the rapid increase in volume, and velocity of data produced by those applications and technologies. For example Facebook and Yahoo handle more than 80 terabytes and 120 respectively every day (Ding, 2013).

To understand the "Big Data" paradox, it is necessary to describe the characteristics of it that is often described by 5V’s as shown in Figure 1.1

Figure 1.1: 5V's Big Data[1]

Volume refers to huge amount of datasets that are scaled with Terabytes (10^{12} Bytes), Zettabytes (10^{21} Bytes), or more. The capacities of Big Data technology to store and process this large amount of data is a crucial reason that led organization to use Big Data in their business.

Velocity refers to the speed at which data is created and processed.

Variety refers to disparate data types including: structured, semi-structured, and unstructured data that are generated from diverse data sources such as: social media, sensors, web logs, etc. Big Data processing includes gathering, storing, managing, and processing disparate data.

Veracity refers to reliability, accuracy, and trustworthiness of disparate big data sets.

Value is the most and crucial characteristics of Big Data which takes into account the costs and benefits of gathering, storing, and managing huge and disparate data sets. In other words there will be no sense if the data cannot be transferred into a value that is useful and profitable.

The characteristics described above states that the buzzword Big Data is not only dealing with huge amount of data sets, but also the data that is disparate and needed to be processed in the right-time manner or real-time.

[1] https://www.linkedin.com/pulse/20140306073407-64875646-big-data-the-5-vs-everyone-must-know

There are two most popular frameworks named Apache Hadoop[2] and Apache Spark[3] which are being used for Big Data analytics. Apache Hadoop is an open-source framework that allows large data sets to be processed in a distributed manner by using simple programming models across cluster of computers. Apache Hadoop consists of four primary modules; 1) Hadoop Distributed File System (HDFS): a distributed file system that provides a high throughput access to data application. 2) Hadoop MapReduce: A YARN-based system that allows large datasets to be processed in parallel. 3) Hadoop YARN: A framework for resource management and job scheduling. 4) Hadoop Common: Common Hadoop utilities to support other Hadoop modules[2].

Apache Spark, on the other hand, is an open-source, fast and in-memory data processing engine for large-scale data sets. Apache Spark uses an abstraction mechanism called RDD (Resilient Distributed Dataset). RDDs are fault tolerant and allow in memory computation. Spark grants the fault-tolerance with a handler to RDD that gets the metadata and build the RDD if the nodes crashes (Zaharia, 2010; Zaharia, 2012).

Studying and comparing the performance of these frameworks recently has attracted many researchers. Besides having their own way of data processing, both provides some advantages over the other.

1.2.Research Objectives

The aim of this study is to compare Apache Hadoop and Apache Spark frameworks, with three micro-benchmarks (WordCount, Grep, Sort) and one application benchmark (Naive-Bayes) based on BigDataBench[4] benchmark suite, to reveal the pros and cons of each and to understand which framework better performs under different scenarios and circumstances by taking their performance in terms of execution time, resource usage and speedup. By speedup, we mean the proportional execution time reduction under the same work load, when the computation resources are being increased. Within this study the following specific research questions are investigated in order to better compare the Hadoop and Spark frameworks;

[2] http://hadoop.apache.org/
[3] http://spark.apache.org/
[4] http://prof.ict.ac.cn/

1. Which configuration specifications lead to achieve the optimum performance of the frameworks for each workload?
2. Which of the frameworks has a shorter execution time in each workload?
3. Which of the frameworks has a better speedup ratio in each workload?
4. Do the resource usages of the frameworks differ for the same workloads?

1.3. Significance of the Study

Most of the studies in the literature compare Hadoop and Spark frameworks in terms of execution time and the majority of them do not even present any kind of measurements about their resource usage. Additionally, the versions of these frameworks used in those studies are older than that of used in this study. Having said so, in this study Hadoop and Spark frameworks are being compared with respect to their execution time, in addition to their CPU, memory, and network usage, by using up-to-date versions of these two frameworks. The comparisons are being made with using four different workloads; WordCount, Grep, Sort, and Naive-Bayes by the help of BigDataBench benchmark utility. Furthermore, the frameworks are being compared after finding their optimum parameter configurations that would result in their shortest execution time for each of the workloads listed above.

1.4. Thesis Outline

Chapter 1 includes the purpose and motivation of this thesis, followed by the clarification of the thesis scope through discussion of thesis objectives. Chapter 2 explores a literature review regarding Big Data and comparison of Big Data frameworks taking Apache Hadoop and Apache Spark into account, as of the consideration for this thesis work. In Chapter 3, a review of Apache Hadoop and Apache Spark frameworks are presented. Chapter 4 summarizes the laboratory environment settings that are being used to deploy and execute the workload tests on Apache Hadoop and Apache Spark. Chapter 5 presents overview information about BigDataBench benchmark utility, which is being used in this study to compare Hadoop and Spark Frameworks. In Chapter 6, the methodology, especially the test case scenarios for the comparison of the two frameworks are presented. Finally,

Chapter 7 presents the discussion of results, conclusions, limitations, and propositions for future studies.

CHAPTER 2

LITERATURE SURVEY

2.1.Related Work

This section includes three subsections: stating secondary studies regarding Big Data (Section 2.1.1), defining the systematic mapping (SM) and discussing the results from the SM study that is conducted for this thesis study (Section 2.1.2), and a literature review section (Section 2.1.3).

2.1.1. Related Works for Big Data

A description of Big Data and its characteristics are presented in (Katal, 2013). This paper also discusses the challenges and issues that need to be considered for the purpose of adopting Big Data technologies. In addition to stating best practices, the components of Hadoop framework are also explored within this study.

A comprehensive discussion related to Big Data is explored based on Hadoop framework in (Narasimhan, 2014). It explains 5V characteristics (Variety, Velocity, Volume, Value, and Veracity) of Big Data. It also gives extensive information about different Big Data components from the Hadoop point of view.

In (Ularu, 2012), the importance of Big Data and its challenges are presented from the perspective of data analytics. This study also demonstrates Big Data characteristics, and finally Hadoop frameworks and its components are explored.

A discussion on the 3V's of Big Data concerning Velocity, Volume, and Value is given in (Sagiroglu, 2013). This study also discusses the disciplines of gathering

information from complex data sets, and states some challenges of Big Data security and privacy.

Casado (Casado, 2015) not only states the 3V characteristic of Big Data but also the life cycle of Big Data is presented including; data acquisition, data storage, data analysis, and data exploitation of the results. Additionally, this paper explores technologies for Big Data processing including; Batch processing, real-time processing, and hybrid processing. This paper states that Hadoop is the winner for batch processing, and Spark brings impressive properties for some kind of applications.

In (Jonnalagadda, 2016), Apache Spark and its features are introduced. This paper also discusses the use cases that Spark has advantages over Hadoop, including; Iterative Algorithms in Machine Learning, Interactive Data Mining and Data Processing, Stream processing, Sensor data processing. In addition, this paper also illustrates the components of Spark framework.

Several issues of Big Data are discussed in (Samal, 2015) from the perspective of management, data storage, virtualization, distributed file systems and applications. In the same study, Hadoop framework and the several optimization techniques regarding MapReduce are presented.

In (Jagadish, 2014), the five phases of Big Data life cycle is demonstrated. And the issues; heterogeneity, scalability, and timeline that might occur in all or each of the phases are explored. Also, challenges regarding visualization of results and privacy of Big Data process are explained.

2.1.2. Systematic Mapping (SM) Study for Two Big Data Frameworks

SM is a study which aims to present an overview of a research field by a classification schema that maps the frequencies of the publications per categories to analyze the trends and find gaps in the research field (Petersen, 2015; Petersen, 2008).

A SM study is conducted in order to find the trends and gaps about current state-of-art about two of the Big Data frameworks including Apache Hadoop and Apache Spark, and for the purpose of finding the directions of this thesis based on the SM

results. The detailed SM study which is based on the guidelines in (Petersen, 2008) can be found in Appendix A.

From the SM results it was found that the community is more interested in contributing new method/approach/techniques for the purpose of performance enhancement of the these two frameworks. Based on the results of the SM study, it is found that 13 studies have compared their performance and among these studies only handful of them compared their performance and resource usage.

Based on these findings, the target of this thesis study specialized as comparing Hadoop and Spark frameworks from the perspective of their performance and resource usage, and speedup for four different workloads, including: WordCount, Grep, Sort, and Nave-Bayes by using BigDataBench benchmark utility.

2.1.3. Literature Review

The purpose of this thesis is to compare Hadoop and Spark frameworks to understand which of the two frameworks are better under different scenarios and circumstances taking the performance, computer resource usage, and speedup into consideration.

This section covers the literature review from the papers that are related to our study.

In (Shi, 2015), performance of Hadoop and Spark is compared by performing some extensive experiments on (WordCount, k-means, PageRank and Sort) workloads on Hadoop V2.4.0 and Spark V1.3.0. According to the results, it is reported that; Spark is 2.5x faster than Hadoop for WordCount workload, Spark is 5x faster than Hadoop for k-means workload, and regarding PageRank workload Spark is 5x faster than Hadoop. However, in the case of Sort workload Hadoop performs 2 times faster than Spark. In addition, Spark was deployed in standalone mode, and YARN was not utilized which allows using richer resource scheduling capabilities.

The performance of Hadoop and Spark is evaluated in terms of executions time by developing a recommendation system for each framework (Kupisz, 2015). The experiments run over Apache Hadoop Cloudera[5] V4.1.3 and Spark V0.7.3. It was found that the system based on the Spark platform was more efficient. The paper

[5] https://www.cloudera.com/products/apache-hadoop.html

itself states that the volume of input data is limited and experiments should be done on a larger volume to get conceivable outcomes. More importantly the paper only takes execution time as consideration and lacks comparisons in terms of CPU, memory, disk I/O, and network usage.

In (Gopalani, 2015), the performance of Hadoop (Mahout) and Spark (MLib) are compared in terms of execution time by a standard learning algorithm for clustering (K-Means). In its conclusion, it was stated that Spark performed noticeably faster than Hadoop. In the same work only execution time is considered and it lacks comparisons in terms of CPU, memory, disk I/O, and network usage.

The performance of Hadoop and Spark is compared in terms of execution time and scalability by carrying out experiments employing K-mean and PageRank algorithms on Hadoop (0.20.2) and Spark (0.6.0) by using Java API in a unified cloud platform that combines both frameworks (Lin, 2013). Overall, it was found that Spark performs remarkably faster than Hadoop, but the Spark performance degrades when the volume of data expands. However it is still faster than Hadoop. The paper takes execution time and scalability into consideration and lacks comparisons in terms of CPU, memory, disk I/O, and network usage.

In (Gu, 2013), the performance of Hadoop 0.20.205.0 and Spark 0.6.1 is compared over a classical PageRank algorithm, taking time and memory into the consideration. It was found that Spark outperforms Hadoop when there is sufficient memory, while its performance is degraded when there is insufficient amount of memory.

An algorithm C4.5 that is used to generate decision tree is performed on both Hadoop and Spark and the results are compared (Wang, 2014). It was found that Hadoop is not suited for a small volume of data processing, and Spark is better for algorithms with comprehensive I/O and limited CPU processing.

In (Deng, 2014), a popular PrefixSpan algorithm is performed on Hadoop MapReduce and Spark and their performance is compared in terms of execution time and scalability. In conclusion, Spark outperformed Hadoop. And same as (Lin, 2013), this paper only takes execution time as consideration and lacks comparisons in terms of CPU, memory, disk I/O, and network usage.

A comprehensive performance evaluation of Hadoop and Spark on three micro-benchmarks is presented in (Liang, 2014), the experiment results shows that Spark outperforms Hadoop.

In the above discussed studies, the framework versions used are old compared with the current versions (Apache Hadoop 2.7.2 and Apache Spark 1.6.0), that may cause different results if the experiments are repeated with the current version. Also above studies lack comparisons in terms of CPU, memory, and network usage, therefore this thesis will take above consideration into account in addition to the execution time and speedup.

CHAPTER 3

REVIEW OF HADOOP AND SPARK FRAMEWORKS

3.1. Review of Hadoop and Spark Frameworks

In this section review of Apache Hadoop and Apache Spark frameworks is presented.

3.1.1. Apache Hadoop

Apache Hadoop is an open-source framework that allows large data sets to be processed in a distributed manner by using simple programming models across cluster of computers.

It is designed in the form that can be scaled from single server to thousands of servers of machines, each server with its own local computation and storage capacity. To deliver high availability, it is designed to assign and handle failures at the application layer rather than depend on hardware.

The project includes the following modules:

- Hadoop Common: Common Hadoop utilities to support other Hadoop modules.
- Hadoop Distributed File System (HDFS): It is a distributed file system that provides a high throughput access to data application.
- Hadoop YARN: A framework for resource management and job scheduling.
- Hadoop MapReduce: A YARN-based system that allows large datasets to be processed in parallel.

3.1.1.1.Hadoop Distributed File System (HDFS)[6]

The Hadoop Distributed File System (HDFS) is a distributed file system that provides fault tolerance and it is designed to run on commodity hardware. It provides high throughput access to application data and suitable for applications that have large data sets.

HDFS has a master-slave architecture. Huge data sets are automatically divided into chunks which are managed by different nodes in the Hadoop cluster.

As show in Figure 3.1 an HDFS cluster consists of a single NameNode, a master node that monitors the namespace of the file system and access of clients to file system, and there are a number of DataNodes in the cluster, usually there is one DataNode per machine, that serve as the storage area. The input file is broken down into one or more blocks and they are stored in DataNodes.

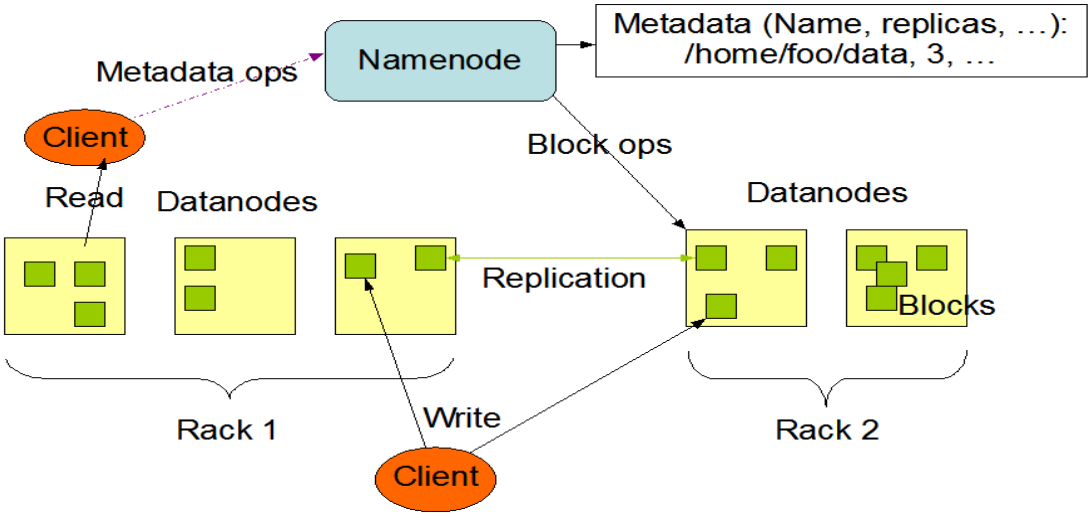

Figure 3.1: HDFS Architecture[6]

3.1.1.2.Hadoop MapReduce

Hadoop MapReduce is a software framework that allows large datasets to be processed in parallel in a reliable and fault-tolerant manner[7]. User assign a map function that process a key/value pair to create a set of intermediate key/value pairs

[6] http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html
[7] http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

and a reduce function that merges all intermediate results values related with the same intermediate key (Dean, 2008).

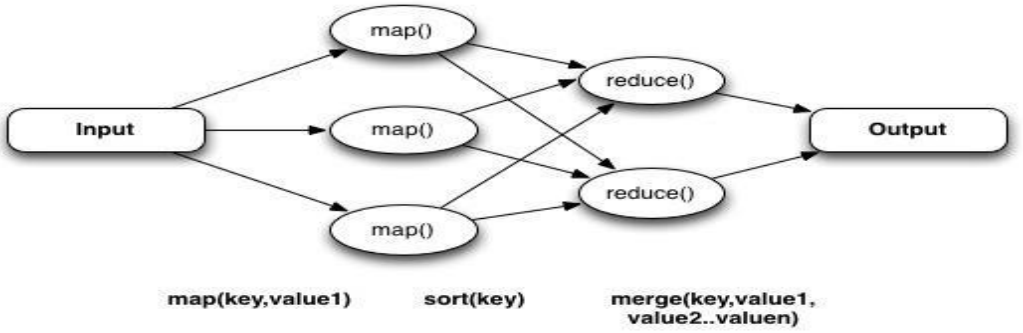

Figure 3.2: MapReduce Execution Flow (Shikhare)

3.1.1.3. Hadoop YARN[8]

YARN is also known as MapReduce 2.0 (MRv2). The concept in YARN is separating the responsibilities of resource management and job scheduling into two different daemons, unlike the MapReduce V.1 where JobTracker was responsible for the resource management and job scheduling duties.

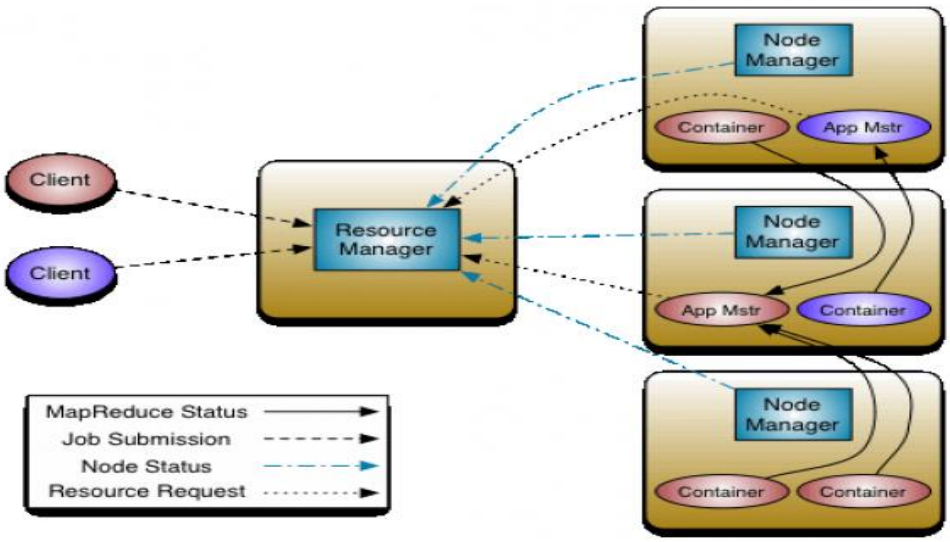

Figure 3.3: HADOOP YARN Architicure[8]

[8] http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

As shown in Figure 3.3, YARN has a global ResourceManager (RM), ApplicationMaster (AM) per application, NodeManager per node, and container(s) per node.

ResouceManager: located in master node, and monitors the resource scheduling for applications.

NodeManager: It runs in slave nodes and responsible for launching the application's containers, and reporting the resource usage of these containers to ResourceManager.

ApplicationMaster: responsible for the resource negotiating with ResourceManger for the applications, and in it is coordination with NodeManagers for scheduling of the tasks.

YARN allocates the cluster resources to the applications in the form collection of physical resources known as containers, and each container runs one task concurrently.

Hadoop YARN is not limited to MapReduce, but serves as a more general platform for other programming styles such as Directed Acyclic Graph (DAG), which extends Hadoop's capability beyond the simple MapReduce programming style[9].

3.1.2. Apache Spark[10]

Spark is an open-source distributed computing framework, and provides easy interfaces for Java, Scala, Python, and R programming languages. It is designed for fast and iterative computation on large data sets across clusters. It includes tools: Spark SQL for SQL and structured data, MLIB for machine learning, GraphX for graph processing, and Spark Streaming for streaming data processing[10] as shown in figure 3.4.

[9] https://opensource.com/life/14/8/intro-apache-hadoop-big-data

[10] https://spark.apache.org/docs/latest/

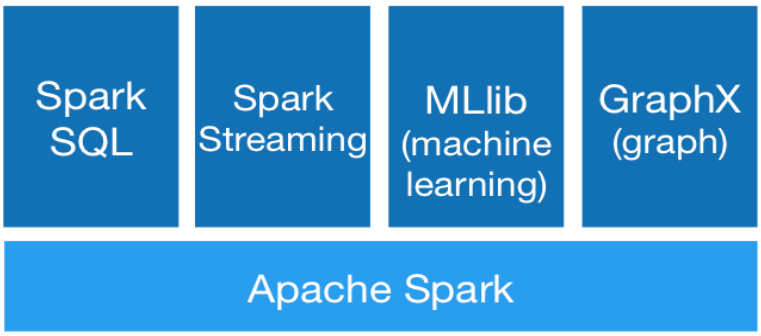

Figure 3.4: Apache Spark framework[11]

Apache Spark uses RDD (Resilient Distributed Dataset) abstraction. An RDD is a read-only collection of objects distributed across a set of machines. RDDs are fault tolerant and allow in memory computation. Spark grants the fault-tolerance with a handler to RDD that gets the metadata and build the RDD if the nodes crashes (Zaharia, 2010; Zaharia, 2012). RDDs can be created through four methods: 1) from Hadoop InputFormat (such as HDFS files) 2) By "parallelizing" a Scala collection (e.g., an array) in the driver program. 3) By transforming an existing RDD. 4) By changing the persistence of an existing RDD (Zaharia, 2010).

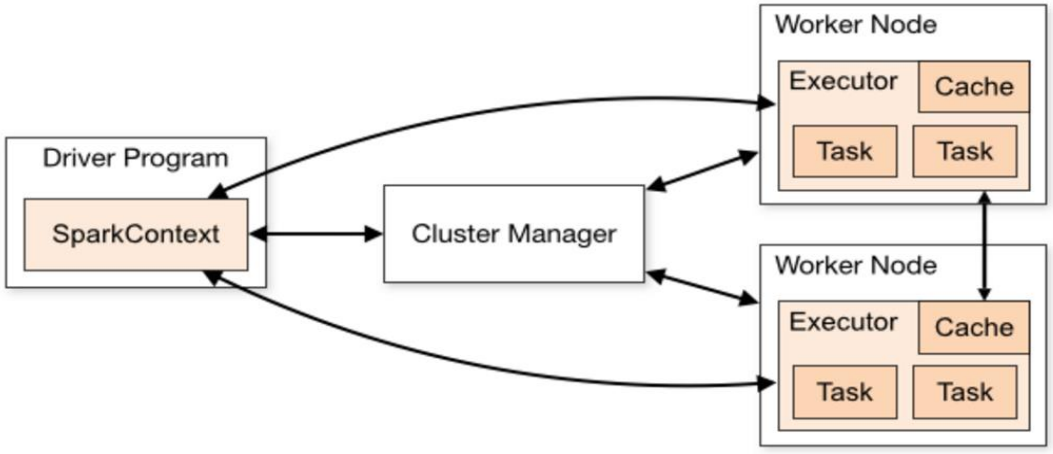

Figure 3.5: Spark Architecture[12]

Apache Spark has a master/worker architecture as shown in Figure 3.5. The driver program runs inside the master node and creates the SparkContext object. SparkContext can connect to various types of cluster managers (Mesos[13], Yarn or Spark's own standalone cluster manager) in order to allocate the resources for the

[11] http://spark.apache.org/
[12] https://spark.apache.org/docs/latest/cluster-overview.html
[13] http://mesos.apache.org/

applications. Once the connection is established, Spark gets the executors on nodes across the cluster. Finally, executors will receive the application code and the tasks to be processed on the executors[12].

Definition of some of the terminologies used for Apach Spark is given below:

- **Driver Program:** is the main function of the application and it is responsible for the creation of the SparkContext object.
- **SparkContext:** is the entry point to Spark for a Spark application.
- **Cluster Managers:** are the external service for bringing its resources to the cluster.
- **Worker Node:** is the node in the cluster that executes the application code.
- **Executor:** is the process that stores data for the application and executes the tasks.
- **Task:** is the unit of work[12].

The Spark applications can be executed in two deployment modes[12]:

- Client Mode: is the default deployment mode, where the driver program executes inside the machine that the spark application was launched.
- Cluster Mode: the driver program executes on any of nodes in the cluster.

CHAPTER 4

LABORATORY ENVIRONMENT

4.1. Laboratory Environment

This section presents the laboratory environment that is needed to deploy and execute tests on Apache Hadoop and Apache Spark.

4.2.Cluster Deployment

For this thesis study a laboratory environment is set up in the Software Engineering Department, Atilim University, Ankara, Turkey, consisting of 20 nodes. The hardware properties of the nodes (PC machine) are as follow:

- Processor: Intel® Core™ i7-3770 3.40 GHz Processor
- Memory: 8 GB
- Storage area: 100 GB (Linux reserved)
- Ethernet switch: 1 Gigabit
- Java version: Java Oracle 1.7
- Scala version: 2.10

Linux Ubuntu 14.04 LTS is used as the operating system.

4.3.Software Stack Installation

The software listed below are installed on the computers in the laboratory environment.

- Apache Hadoop
- Apache Spark

- Ganglia[14]

4.3.1. Apache Hadoop Installation

The latest stable version of Apache Hadoop which is currently Apache Hadoop 2.6.2 is installed on each node in the cluster. In order to obtain a multi-node Hadoop cluster we have setup the SSH connection between nodes.

4.3.2. Apache Spark Installation

The latest stable version of Apache Spark which is currently Apache Spark 1.5.2 is installed. The Hadoop YARN deployment model is chosen among the three deployment models for Apache Spark (Standalone, on Apache Mesos, on Hadoop YARN) for our cluster, and for this deployment model the Apache Spark is needed to be installed only on the master node, and it cares about the distribution of jobs among the other nodes.

4.3.3. Ganglia

Ganglia[14], is a resource monitoring tool that allows easy way to monitor resource usage of distributed clusters. Ganglia is installed on all nodes that are used for the test cases of this study.

[14] http://ganglia.info/

CHAPTER 5

BigDataBench

5.1. BigDataBench Benchmark

BigDataBench Benchmark is an open-source benchmarking tool for testing and comparing Big Data frameworks. The current version, BigDataBench 3.2, provides five regular and essential Big Data application domains:

- Search Engine
- Social Network
- E-commerce
- Multimedia Analytics
- Bioinformatics

Overall it covers 14 real-world data sets and 33 Big Data workloads considering offline analytics, streaming, Cloud OLTP, Graph, Data Warehouse workload types. The data types can be structured, semi-structured, and unstructured.

The BigDataBench has a tool called Big Data Generator Suite (BDGS) for generating expandable (scalable) big data for benchmarks by keeping their initial aspects and properties.

As shown below, it provides diverse implementations for the same benchmark specifications:

- Offline Analytics workloads for MapReduce, Spark, Flink[15], MPI.
- Interactive Analytics and OLAP workload for Shark[16], Impala[17], and Hive[18].

[15] https://flink.apache.org/

5.2.Why BigDataBench?

Table 5.1 indicates the claims, stated by the developers of tool, why BigDataBench is more mature and practical than other Big Data benchmark tools[19]. It allows the specification of five application domains with six workload types, while the others support fewer than that of BigDataBench, such as BigBench includes only one application domain with three workload types. BigDataBench contains larger amount of workload types than other benchmarks which in total includes thirty-three workloads. It contains eight scalable data sets, which is synthetic data generated from real data, while on the other hand BigBench, CloudSuite, and HiBench benchmarking tools have three scalable data sets. Furthermore, in contrary to other Big Data benchmark tools, BigDataBench provides multi-tenancy, subset, and stimulator features.

Multi-tenancy: It is used for flexible setting and recapping of mixed workloads based on workload traces in cases of modeling multi-application scenarios on Cloud.

Subset: In BigDataBench subset, a small number of workloads are selected among large amount Big Data workloads from specific perspectives for the purpose of reducing the cost benchmarking.

Stimulator: Provides collecting performance characteristics, and evaluating hardware design and detecting errors instead of real hardware.

[16] http://shark.cs.berkeley.edu/
[17] http://impala.apache.org/
[18] https://hive.apache.org/
[19] http://prof.ict.ac.cn/

Table 5.1: The Differences of BigDataBench from Other Benchmark Suites[19]

	Spec.	App. Domains	Workload Types	Work-loads	Scalable Data Sets	Diverse Implmen.	Multi-tenancy	Subset	Stimulator
BigDataBench	Y	Five	Six	Thirty-three	Eight	Y	Y	Y	Y
BigBench	Y	One	Three	Ten	Three	N	N	N	N
CloudSuite	N	N/A	Two	Eight	Three	N	N	N	N
HiBench	N	N/A	Two	Ten	Three	N	N	N	N
CALDA	Y	N/A	One	Five	N/A	Y	N	N	N
YCSB	Y	N/A	One	Six	N/A	Y	N	N	N
LinkBench	Y	N/A	One	Ten	N/A	Y	N	N	N
AMP Benchmarks	Y	N/A	One	Four	N/A	Y	N	N	N

The reason behind choosing BigDataBench for this thesis and preferring it over other existent Big Data benchmarks are:

- As it is shown in Table 5.1, it is argued to be more mature and more practical than other Big Data benchmark tools such as it includes typical Big Data application domains and micro-benchmarks that the author intended to use to compare the two Big Data frameworks: Apache Hadoop and Apache Spark.
- It provides the diverse implementations of the same workloads that can be used for both frameworks.
- In terms of usability, it is well organized and it provides an open-source data generation tool (BGDS) that allows generating expandable (scalable) synthetic data for benchmarks based on raw data sets and keeping their initial aspects and properties.

CHAPTER 6

A COMPARISION OF HADOOP AND SPARK FRAMEWORKS

6.1. Test Scenarios

This section presents the terminologies of both frameworks, and chosen workloads from the BigDataBench.

6.1.1. Terminologies in Both Frameworks

There are some aspects that need to be taken into consideration while working with Apache Hadoop and Apache Spark that may affect the performance and resource utilization of the jobs including:

- **HDFS block size** is the smallest unit of data that Hadoop uses to split the data and into chunks and store them in a distributed manner across the nodes in the cluster.
- **HDFS replication** is the factor that decides the number of the copies of HDFS blocks among the nodes in the cluster.
- **Number of concurrent tasks per node** determines how many map and reducer tasks can work in the same node.
- **Mapper** is responsible to process the input data and the number of map is controlled by the HDFS blocks size.
- **Reducer** is responsible to process output data of the map task, and its number can be controlled by the user in Hadoop, while in Spark it is controlled by spark.default.parallelism parameter[20].

[20] https://groups.google.com/forum/#!topic/spark-users/taYp76G9taQ

The aim of the tuning process is to find how the listed parameters above have impact on the performance of the two frameworks, so that the comparison process would be executed on optimized configurations of each framework and for each workloads.

6.1.2. Chosen Workloads

Four popular workloads, being selected to be used in this study, are WordCount, Grep, Sort, and Naïve-Bayes for benchmarking purposes.

WordCount, Grep and Sort are basic and widely used micro-benchmarks to evaluate the Big Data frameworks. WordCount calculates the occurrences of words in a text, therefore WordCount is a typical CPU intensive application with integer calculations (Han, 2015; Shahrivari, 2014). Grep is a workload that searches for given words or strings in a text and counting the number of occurrences of the matching string. If the search is made for a single word, then it becomes I/O intensive (Shahrivari, 2014). Sort is another typical I/O intensive workload that sorts the records of the given input depending on key values (Han, 2015; Yigitbasi, 2013). Since the number of data passed from Map to Reduce does not change, this process is known as Network intensive. Naive-Bayes is a machine learning algorithm for classification, based on the probabilistic algorithm, build on Bayes theorem with loose dependence expectations. Naive-Bayes is another CPU intensive workload with floating-point calculations of density and maximum like-hoods (Han, 2015; Liang, 2014).

BigDataBench Data Generator was used to generate input data for WordCount, Grep, Sort, and Naive-Bayes workloads.

6.2.Test Cases

This section includes two subsections: Parameter Configurations (Section 6.2.1) and Data Size (Section 6.2.2)

6.2.1. Parameter Configurations

Various combinations of the following parameters are tested together to find the optimum performance of the each framework for the chosen workloads. The values between { } brackets shows the default parameter values.

- Number of containers per node: 2, 4, and {6}

- HDFS Block Replication: 1, 2, and {3}
- HDFS block size: 64MB, {128MB}, 256MB, and 512MB
- Number of Reducers (for Hadoop): {1}, 2, 12, and 20

By default Hadoop assigns one reducer for the jobs.

In YARN, each task (Mapper/Reducer) is executed by one container therefore number of containers per node equals to the number of concurrent tasks per node. By default YARN creates the containers with 1 CPU and 1GB memory, so the number of containers in one node is equals to the number of CPU cores. The nodes used in these experiments have 4 physical cores (8 virtual cores) and 8GB memory. However, 2 virtual cores and 2 GB of memory are reserved for the operating system itself; therefore the maximum number of containers was allowed to be 6 per node.

Tests are conducted to get satisfactory answers to the research questions set by the author.

(RQ1) Which configuration specifications lead to achieve the optimum performance of the frameworks for each workload?

Tests are applied with the various parameter combinations. In each workload a combination of parameter yielded the shortest execution time within the clusters consisting of 2, 4, and 8 nodes. The optimum parameter configurations found for Hadoop and Spark are given in Table 6.1 and 6.2 respectively, and their deduction is explained in subsequent sections.

Table 6.1: Parameter combinations for the optimum performance of Hadoop for each workload.

Workload	Parameters			
	HDFS Block size	HDFS Block Replication	Number of Container/Node	Number of Reducers
WordCount	256MB,512MB	3	6	1
Grep	256MB,512MB	3	6	1
Sort	128MB	1	6	Maximum number
Naive Bayes	512MB	3	6	1

Table 6.2: Parameter combinations for the optimum performance of Spark for each workload.

Workload	Parameters		
	HDFS Block size	HDFS Block Replication	Number of Container/Node
WordCount	128MB	3	6
Grep	128MB	3	6
Sort	128MB	1	6
Naive Bayes	128MB	1	6

6.2.1.1. Hadoop Parameters Configurations

Regarding Hadoop, it is observed that the best performance of the workloads achieved with the combination of the parameters is shown in Table.6.1.

Tables 6.3, 6.4, 6.5, and 6.6 show how the best performance of Hadoop is achieved for each of the workloads.

HDFS block size has effects on the performance for WordCount, Grep, and Naïve-Bayes workloads. The reason behind is that the input data gets split and stored into the HDFS depending on the block size. Therefore, HDFS splits the input data into more blocks as the block size gets smaller, which in turn require more map tasks to be executed. As a result, the jobs with fewer numbers of blocks finish its job in a shorter time and become idle. Since the mapper tasks in Hadoop work concurrently, and synchronized at the end of all map tasks, there would be waste of resources and the total time spent for map phase would be equal to longest mapper.

On the other hand, the HDFS block size for Sort workload does not show remarkable difference in Hadoop's performance because most of the execution time is consumed during the sort progress. That is, the execution time is dominated by the time used in reducer stage, which is decreased as the number of reducer is increased.

Furthermore, the maximum number of containers per node, which is 6 in our case, performed better than 4 and 2 containers per node for all workloads. That is because

of having more containers allow Hadoop to perform more concurrent tasks on the HDFS blocks.

From the test results, it's also noticed that the number of HDFS replication does not have any remarkable effect on the performance of workloads except for sort, which showed better performance when the HDFS replication is set to one. This is because shuffle operation takes place in the basis of sorting, and shuffle is responsible for the movement of data across the cluster[21].

Within most Big Data application, the majority of the data pipelines initiates with a huge amount of input raw data and the data volume lessen through shuffle progress as result of refinement and removing irrelevant data or demonstrating data in a more solid manner[21]. But in Sort workload the reduction of data doesn't takes place along the pipeline. That means a shuffling of 100 TB of data occurs when sorting 100 TB of input data. Thus, the amount of output data produced by mappers would be same as the input data which internally requires more time to write on HDFS, as the number of replication increases.

Increasing the number of reducers shows a negative impact on the WordCount, Grep and Naïve-Bayes workload's performance because YARN distributes the mappers and reducers among the containers. Therefore, increasing number of reducer results in decreasing number of mapper, which means less number of mappers will be available in the cluster to work at the same time. On the other hand, increasing the number of reducer has a positive impact on the Sort workload. Because, the data doesn't get reduced in the pipeline and there is more computations occurs in the reduce phase of Sort workload. The best performance is obtained in Sort workload when giving the number of reducer same as the number of available container in the cluster.

[21] https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html

Table 6.3: Hadoop's WordCount Performance with respect to parameter configurations (4-node).

Test	Block Size (MB)	HDFS Replication	Number of Containers/Node	Number of Reducers	Execution Time(min)
1	128	3	6	1	11mins, 1sec
2	128	3	6	2	11mins, 20sec
3	128	3	6	12	14mins, 47sec
4	128	3	6	20	14mins, 38sec
5	128	3	4	1	13mins, 23sec
6	128	3	2	1	24mins, 58sec
7	128	2	6	1	11mins, 4sec
8	128	1	6	1	11mins, 43sec
9	64	3	6	1	13mins, 6sec
10	256	3	6	1	10mins, 5sec
11	512	3	6	1	10mins, 6sec

Table 6.4: Hadoop's Grep Performance with respect to parameter configurations (4-Node).

Test	Block Size (MB)	HDFS Replication	Number of Containers/Node	Number of Reducers	Execution Time (min)
1	128	3	6	1	4mins, 12sec
2	128	3	6	2	4mins, 22sec
3	128	3	6	12	5mins, 18sec
4	128	3	6	20	5mins, 12sec
5	128	3	4	1	5m14.001sec
6	128	3	2	1	9mins, 4sec
7	128	2	6	1	4mins, 17sec
8	128	1	6	1	5mins, 45sec
9	64	3	6	1	5mins, 59sec
10	256	3	6	1	3mins, 44sec
11	512	3	6	1	3mins, 48sec

Table 6.5: Hadoop's Sort Performance with respect to parameter configurations (8-Node).

Test	Block Size (MB)	HDFS Replication	Number of Containers/Node	Number of Reducers	Execution Time(min)
1	128	3	6	1	87min, 55sec
2	128	2	6	1	83min, 17sec
3	128	1	6	1	78mins, 52sec
4	128	1	6	2	39mins, 6sec
5	128	1	6	5	19mins, 36sec
6	128	1	6	10	15min, 27sec
7	128	1	6	14	14min, 54sec
8	128	1	6	15	13min, 41sec
9	128	1	6	25	13min, 39sec
10	128	1	6	30	13min, 7sec
11	128	1	6	35	13min, 12sec
12	128	1	6	40	13min, 2sec
13	128	1	6	46	12mins, 31sec
14	128	1	4	31	13min, 29sec
15	128	1	2	15	19min, 8sec
16	64	1	6	46	12min, 32sec
17	256	1	6	46	12min, 36sec
18	512	1	6	46	14mins, 50sec

Table 6.6: Hadoop's Naïve-Bayes Performance with respect to parameter configurations (4-Node).

Test	Block Size (MB)	HDFS Replication	Number of Containers/Node	Number of Reducers	Execution Time(min)
1	128	3	6	1	7mins, 50sec
2	128	3	6	2	7mins, 29sec
3	128	3	6	12	8mins, 24sec
4	128	3	6	20	8mins, 39sec
5	128	3	4	1	7mins, 54sec
6	128	3	2	1	13min, 20sec
7	128	2	6	1	7mins, 19sec
8	128	1	6	1	8mins, 20sec
9	64	3	6	1	9mins, 3sec
10	256	3	6	1	6mins, 38sec
11	512	3	6	1	5mins, 59sec

6.2.1.2.Spark Parameters Configurations

In Tables 6.7, 6.8, 6.9 and 6.10, it is observed that changing the default value of HDFS block size, and maximum number of containers per node which are 128 MB and 6 in our case, did not show remarkable impact on the workloads performance.

HDFS replication did not show noticeable effect on the performance of WordCount and Grep Workloads.

On the other hand Sort and Naive-Bayes workloads performed better with 1 HDFS replication. Regarding Sort, as discussed in Section 3.1.1, the reduction of data does not take place in the pipeline. It is obvious that the time of writing to HDFS file would increase as the replication number increases. In case of Naive-Bayes workload, the job whose HDFS replication is set to 1 has used more CPU than the jobs whose HDFS replication are sets to 2 and 3.

Table 6.7: Spark's WordCount Performance with respect to parameter configurations (4-Node).

Test	Block Size (MB)	HDFS Replication	Number of Containers/Node	Execution Time(min)
1	128	3	6	3mins, 26sec
2	128	3	4	3mins, 26sec
3	128	3	2	3mins, 23sec
4	128	2	6	3mins, 17sec
5	128	1	6	3mins, 33sec
6	64	3	6	4mins, 8sec
7	256	3	6	3mins, 32sec
8	512	3	6	3mins, 26sec

Table 6.8: Spark's Grep Performance with respect to parameter configurations (4-Node).

Test	Block Size (MB)	HDFS Replication	Number of Containers/Node	Execution Time (min)
1	128	3	6	2mins, 57sec
2	128	3	4	2mins, 58sec
3	128	3	2	2mins, 55sec
4	128	2	6	2mins, 58sec
5	128	1	6	3mins, 20sec
6	64	3	6	3mins, 20sec
7	256	3	6	3mins, 5sec
8	512	3	6	3mins, 21sec

Table 6.9: Spark's Sort Performance with respect to parameter configurations (8-Node).

Test	Block Size (MB)	HDFS Replication	Number of Containers/Node	Execution Time (min)
1	128	3	6	13mins, 29sec
2	128	3	4	12mins, 45sec
3	128	3	2	12mins, 21sec
4	128	2	6	11mins, 34sec
5	128	1	6	9mins, 37sec
6	64	3	6	14mins, 25sec
7	256	3	6	13mins, 9sec
8	512	3	6	13mins,9sec

Table 6.10: Spark's Naïve-Bayes Performance with respect to parameter configurations (4-Node).

Test	Block Size (MB)	HDFS Replication	Number of Containers/Node	Execution Time (min)
1	128	3	6	16mins, 41sec
2	128	3	4	16mins, 32sec
3	128	3	2	15min, 59sec
4	128	2	6	11mins, 13sec
5	128	1	6	7mins, 35sec
6	64	3	6	15mins, 28sec
7	256	3	6	16mins, 23sec
8	512	3	6	15mins, 56sec

6.2.2. Data Size

The workloads are tested in 2, 4, and 8 nodes for Hadoop and Spark with 50 GB of data size. Spark could not perform 50 GB of sorting, whose reason is explained in section 6.4.3, therefore, for a fair comparison 40 GB of data is used in Sort workload for both frameworks.

The reason behind limiting 50 GB of data size is due to the storage limits of the nodes. Each of the nodes has 100 GB of storage area and the BigDataBench benchmark suite first generates the data in the nodes then transfers it to the HDFS.

6.3.Speedup of the Frameworks

Speedup is the impact of increasing number of resources (number of nodes) to the same amount of work (data size) in order to gain a proportional reduction in execution time[22].

Speedup is defined as the ratio of T1 to T2, where T1 is the execution time of two-node cluster, and T2 is the execution time of cluster with more than two nodes.

6.4.Test Results

This section presents the performance, resource utilization and speedup of Hadoop and Spark frameworks. The tests cases are performed with WordCount, Grep, Sort, and Naive-Bayes workloads on 2, 4, and 8 nodes.

6.4.1. WordCount

(RQ2) Which of the frameworks has a shorter execution time in each workload?

From Figures 6.1 and 6.2, it can be observed that, Spark is 3.12x, 3.08x, and 2.58x faster than Hadoop under 2, 4, and 8 nodes respectively. The reason behind this is that; during the job execution the mapping phase of the Hadoop has used more CPU than Spark. Thus for mapping phase, the CPU time of Spark was shorter than that of Hadoop's. Most of the computations are being done during in the mapping phase; therefore reduction phase does not have a major impact on the total execution time. The data that comes out of each mapping phase is called as intermediate data. This data then gets sent to reduce phase for further computation. Spark uses the RDD concept that enables in-memory computation, which reduces the execution time by allowing the intermediate data to be kept and processed in memory. Contrarily, Hadoop reads and writes intermediate data onto disk, which increases the total execution time.

(RQ3) Which of the frameworks has a better speedup ratio in each workload?

When taking two nodes as a base, the test results showed that Hadoop has a better speed up than Spark with the ratios of 2.015 and 3.75 compared to the 1.98 and 3.10 ratios of Spark in 4 and 8 nodes respectively.

[22] https://exploredatabase.blogspot.com.tr/2013/11/scaleup-and-speedup.html

(RQ4) Does the resource usage of the frameworks differ for the same workloads?

Another reason that Spark outperforms Hadoop is the memory usage; the memory is used more efficiently in Spark compared to Hadoop. Both of the frameworks used CPU intensively, this is due to the CPU intensive nature of the WordCount workload as presented in (Section 6.1.2). As Figure 6.1 and 6.2 shows; Hadoop's CPU usage is higher than Spark, and network I/O of Spark is as twice as much of Hadoop's.

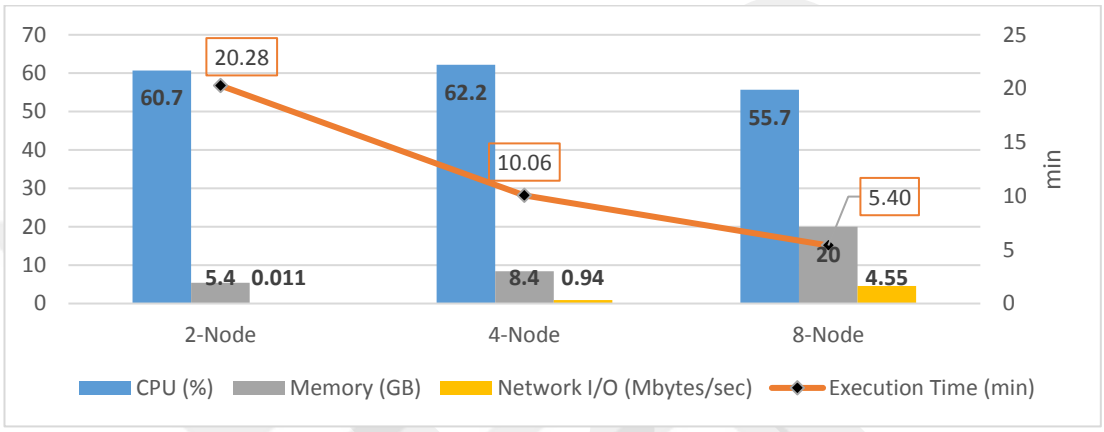

Figure 6.1: Performance and Resource Utilization of Hadoop WordCount

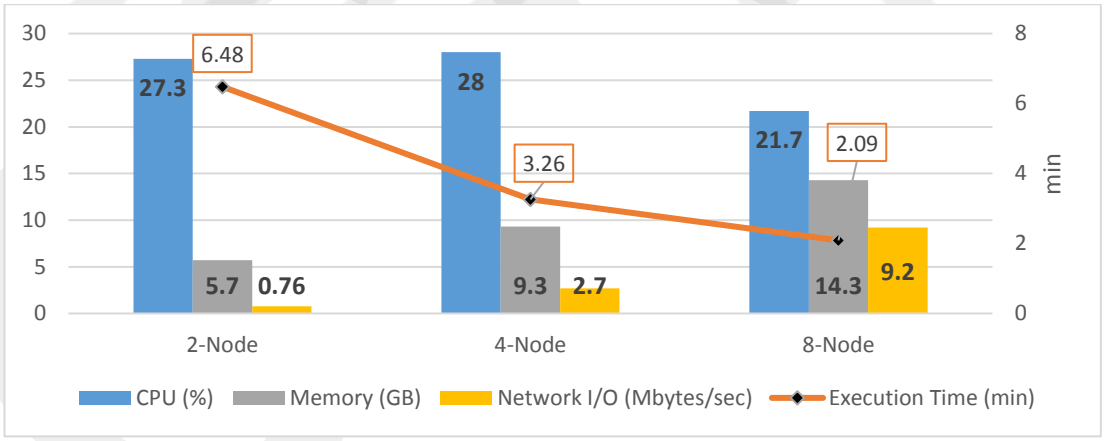

Figure 6.2: Performance and Resource Utilization of Spark WordCount

6.4.2. Grep

In this section, keeping in mind the research questions RQ2, RQ3, and RQ4, the findings are reported from the tests of Grep workload.

As discussed in Section 6.1.2, grep is searching for a given string pattern and counting the occurrence of that pattern matching in a given text. If the string has

matching one or several characters, which might occur frequently in many words, then it becomes slightly CPU intensive since the intermediate output data would contain more data to be flowed between mapper and reducer.

The Figures 6.3 and 6.4 shows that the grep is slightly CPU intensive for both frameworks since the searched expression was not a particular or a single word, which was containing a wildcard like “th*”. Therefore all the words starting with (th) were counted. The CPU usage of Hadoop was more than Spark, while Spark has used slightly more memory than Hadoop because of its in-memory computation nature.

There was no remarkable difference in the frameworks performance. Spark had better speedup ratio in 4 nodes with 2.4 compared to 2.1 of Hadoop, while Hadoop had better speedup ratio in 8 nodes with the ratio of 4.72 compared to 3.07 of Spark.

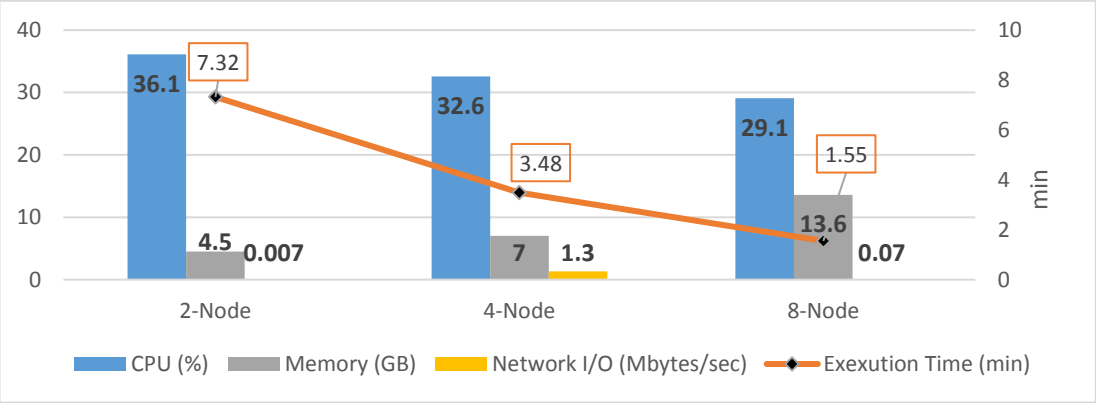

Figure 6.3: Performance and Resource Utilization of Hadoop Grep

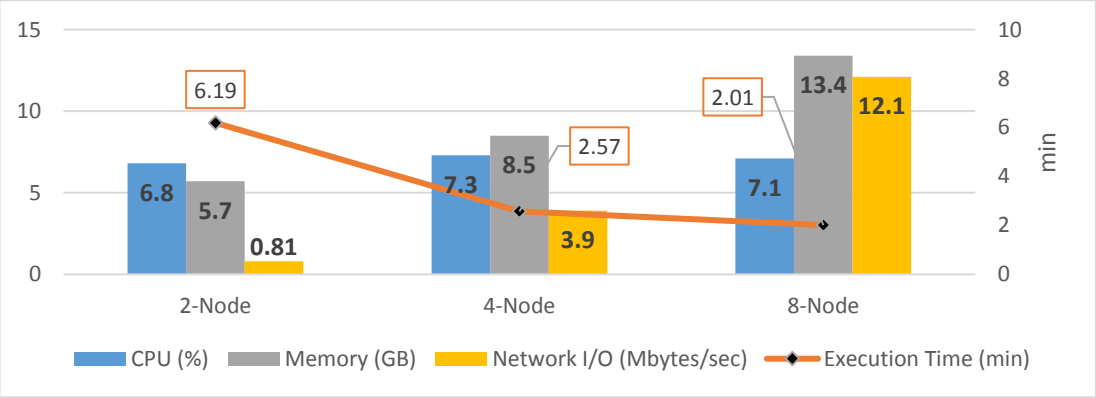

Figure 6.4 Performance and Resource Utilization of Spark Grep

6.4.3. Sort

The answers to the research question RQ2, RQ3, and RQ4 for Sort Workload are given in this section.

The first observation from the Figures 6.5, 6.6 and 6.7 shows high network I/O rate for both framework because sort transforms data from one representation to another which is the transformation of the input data into a sorted form. As it is mentioned before, the reduction of data doesn't takes place along the pipeline in Sort workload. As a result, the output data size is equal to the input data size. It is analyzed that CPU is used moderately and the value of CPUs´ IOWait was high for both frameworks. Thus, I/O should be considered as bottleneck when the input data size increases for the job. Spark has used more memory than Hadoop, and Network overhead of Spark is less than Hadoop. This is because Spark does the computation in-memory, where the intermediate data need not to be read from or written to disk as Hadoop does.

Hadoop had better speedup than Spark with the ratios of 2.96 and 6.55 compared to the 1.49 and 3.04 ratios of Spark in 4 and 8 nodes respectively for 40 GB data size.

Sort could not be tested with more than 40 GB because of the nature of Spark's Driver, which is responsible for task scheduling and establishing connection with the executors through the submitted jobs lifetime[23]. One RDD is created for each HDFS block, and these RDDs will be assigned as tasks for the executors[24]. Spark's Driver requires more heap memory when the number of tasks increases. In this thesis the maximum heap memory could be assigned to the Spark's Driver was 5.5 GB out of 6 GB that is reserved for each node.

For a fair comparison, Sort is tested with 40 GB in 2, 4, and 8 nodes in both frameworks. And from Figure 6.6 and 6.7, it can be seen that Spark outperforms Hadoop with 2.32x, 1.17x, and 1.08x in 2, 4, and 8 nodes respectively.

[23] http://spark.apache.org/docs/latest/cluster-overview.html
[24] https://spark.apache.org/docs/1.2.0/programming-guide.html

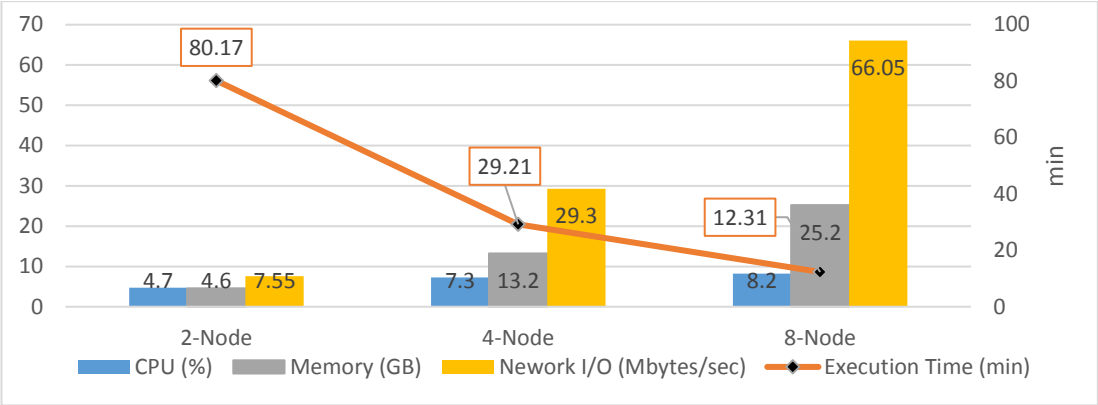

Figure 6.5: Performance and Resource Utilization of Hadoop Sort

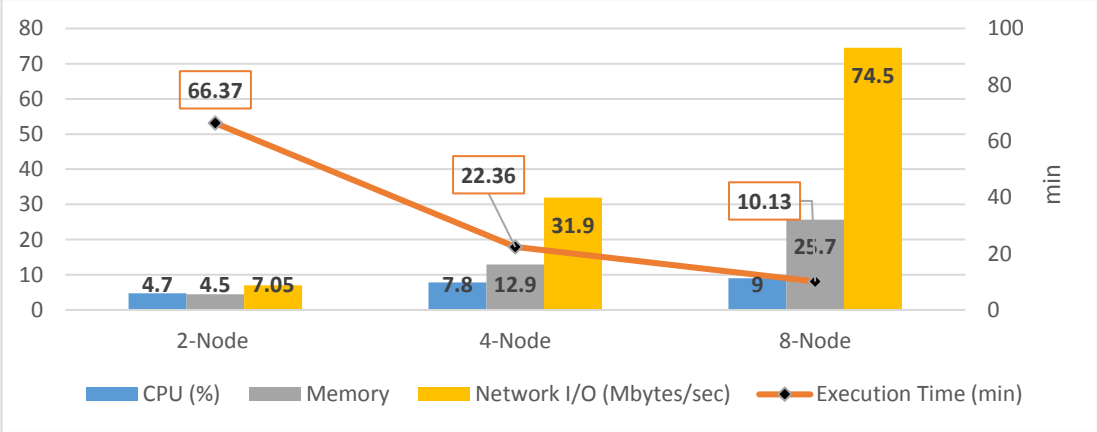

Figure 6.6: Performance and Resource Utilization of Hadoop Sort (40GB)

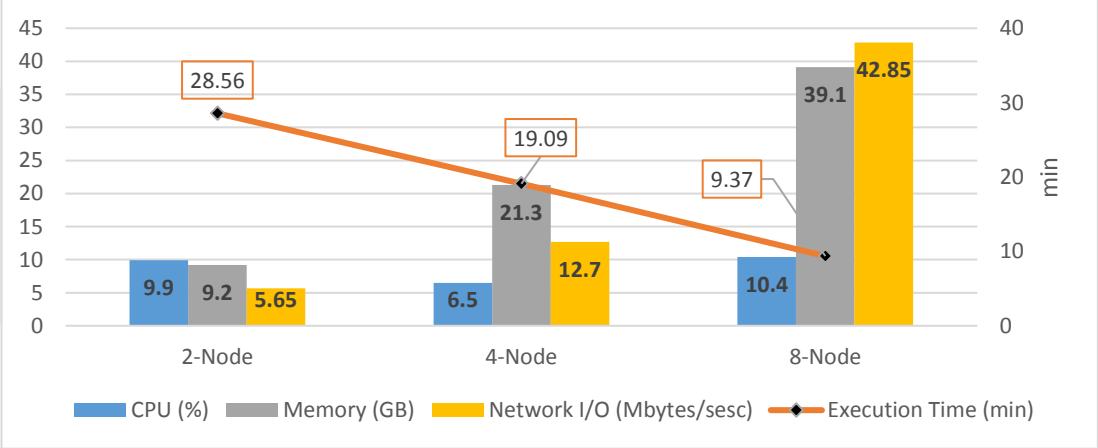

Figure 6.7: Performance and Resource Utilization of Spark Sort

6.4.4. Naïve-Bayes

This section presents the results from the test results of Naïve-Bayes workload for the research question RQ2, RQ3, and RQ4.

CPU was used intensively for both Hadoop and Spark as it is shown in figures 6.8 and 6.9 which were consistent with the explanation stated in Section 6.1.2.

It is observed that Hadoop outperforms Spark in the 2, 4, and 8 nodes. Further tests are performed on 4 node cluster of Hadoop and Spark to understand the reason of why Hadoop outperforms Spark. In cases of 5 and 10 GB Spark outperformed Hadoop, while in cases of 15 and 25 GB they had approximately same performance, and in case of 50 GB Hadoop outperformed Spark.

During the observation of Spark in Naive-Bayes, it is found that Spark fills up the reserved memory in cases of 15, 25 and 25 GB data size, which results in increasing CPU wait state (time spent by the CPU waiting for an IO operations to complete). Therefore, Spark loses its advantage of in-memory computation, where Hadoop does not get affected from this case because it is not processing data in-memory. As a result, the CPU usage of Hadoop was more than that of Spark, and Hadoop executed the job in a shorter total amount of time.

Hadoop had better speedup ratio in 4 nodes with 2.15 compared to 2.05 of Spark, while Spark had better speedup ratio in 8 nodes with the ratio of 4.24 compared to 3.87 of Hadoop.

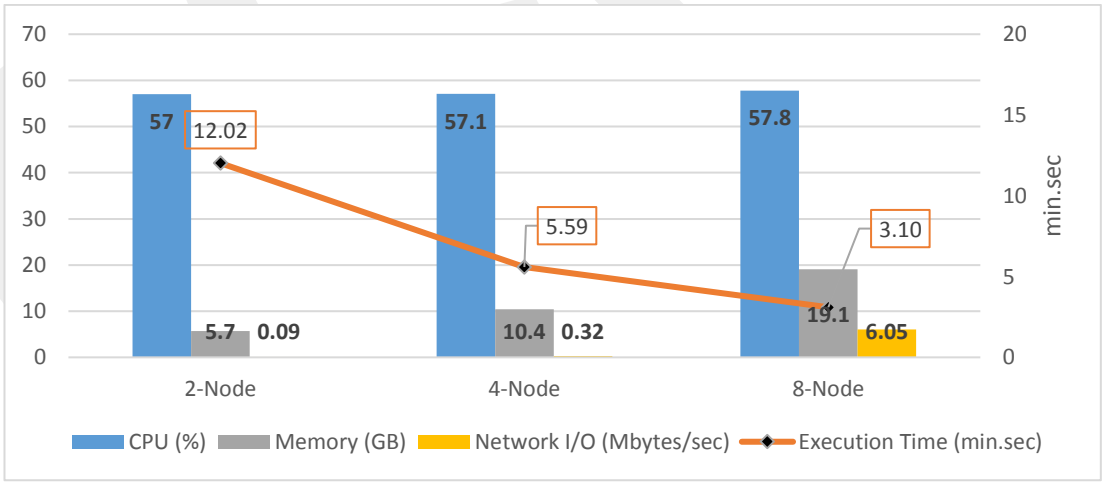

Figure 6.8: Performance and Resource Utilization of Hadoop Naive-Bayes

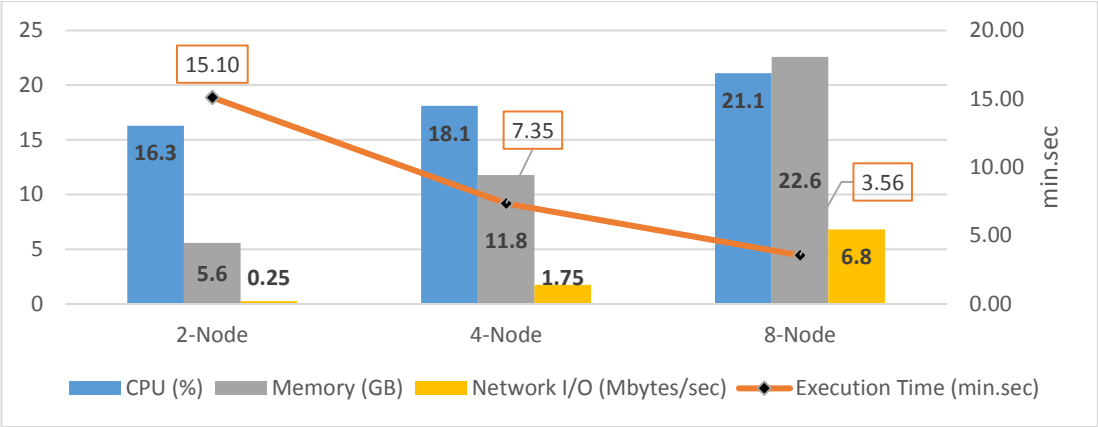

Figure 6.9: Performance and Resource Utilization of Spark Naive-Bayes

CHAPTER 7

DISCUSSION

7.1. Discussion of the Test Results

The comparison tests are performed to measure the performance, resource usage, and speedup of Hadoop and Spark frameworks on four different workloads (WordCount, Grep, Sort, and Naive-Bayes) by using BigDataBench.

In WordCount, both frameworks used CPU intensively where Spark outperforms Hadoop with 3.12x, 3.08x, and 2.58x in 2, 4, and 8 nodes respectively. This was due to CPU time of mapping phase of Spark was shorter and memory was used more efficiently for caching intermediate results. The results also showed that Hadoop had better speedup with the ratios of 2.015 and 3.75 compared to 1.98 and 3.10 ratios of Spark in 4 and 8 nodes respectively.

Both frameworks used CPU slightly more than expected for Grep workload, as Grep is known as I/O bound workload. The reason behind high CPU usage was related to the searched string pattern expression that was used in the test cases, where wildcards such as asterisk (*) was used. There was no remarkable difference in the frameworks performance, and Spark had used slightly more memory compared to Hadoop because of its in-memory computation nature. Spark had better speedup ratio in 4 nodes with 2.4 compared to 2.1 of Hadoop, while Hadoop had better speedup ratio in 8 nodes with the ratio of 4.72 compared to 3.07 of Spark.

For Sort workload, CPU was used moderately and the value of CPUs´ IOwait was high for both frameworks. This is because sort transforms data from one representation to another. Unlike the other workloads, reduction of data doesn't take place along the pipeline of Sort process. For a fair comparison, sort is tested with 40

GB in 2, 4, and 8 nodes in both frameworks since the Spark Driver's heap memory could not be given more than 5.5 GB of memory where only 6 GB of memory was available for the nodes. Spark's Driver is responsible for task scheduling and establishing connection with the executors through the submitted jobs lifetime. Spark outperformed Hadoop with 2.32x, 1.17x, and 1.08x in 2, 4, and 8 nodes respectively. Hadoop had better speedup than Spark with the ratios of 2.96 and 6.55 compared to the 1.49 and 3.04 ratios of Spark in 4 and 8 nodes respectively.

In Naive-Bayes workload, both frameworks have used CPU intensively due to its CPU bound behavior. Hadoop outperformed Spark in the 2, 4, and 8 nodes, and it was observed that Spark loosed its in-memory computation advantage when the data size increases. With the increase of data size it is noticed that reserved memory of Spark filled up and CPU starts to wait for I/O process, which in turn degrades Spark's performance. Hadoop had better speedup ratio in 4 nodes with 2.15 compared to 2.05 of Spark, while Spark had better speedup ratio in 8 nodes with the ratio of 4.24 compared to 3.87 of Hadoop.

All the workloads for Hadoop performed better with 6 containers per node because there were 6 spare cores on each node of cluster that could be assigned to containers. On the other hand, the number of containers per node for Spark did not show remarkable impact on the workloads performance. 2 CPU cores and 2 GB RAM was reserved for other parts of system itself. It is known that, by default YARN assign 1 CPU cores and 1 GB of memory to each container.

There are a few other studies in the literature comparing Hadoop and Spark. For example, Liang (Liang, 2014) compared them with WordCount, Grep, and Sort workloads but only WordCount and Sort are compared with by revealing their resource usage in their study. It is found that the resource usage of their results are consistent that of this study. This article also indicates that Spark outperforms Hadoop with approximately same ratios of this study for WordCount, Grep and Sort. As another example from literature in Shi's study (Shi, 2015) the performance of Hadoop and Spark were compared with WordCount, k-means, PageRank, and Sort workloads. Happily again, the results of their WordCount is also consistent with our results, and furthermore they stated that Spark outperforms Hadoop with similar ratios of this study. Regarding the Sort Workload, Shi indicates that Hadoop is 2

times faster than Spark and Shi reasons the execution model of shuffling process for this performance difference.

7.2. Conclusion, Limitations, and Future Work

Hadoop and Spark frameworks were compared with respect to their performance, resource usage, and speedup ratio. The comparison is done by using three micro-benchmarks (WordCount, Grep, Sort) and one application benchmark (Naive-Bayes) based on BigDataBench benchmark suite. One of the most important novalty of this study is tuning the resource parameters for the optimum performance of each frameworks, under different workloads. So that, up to us, more fair comparision would be achieved.

The experiments showed that Spark outperforms Hadoop with the ratios 2.92x and 1.52x for Wordcount and Sort workloads respectively. However, Hadoop outperforms Spark in Naive-Bayes workload. And there was no remarkable difference in the frameworks performance for Grep workload.

Disk throughput of the frameworks could not be evaluated because the preferred resource monitoring tools did not provide a reliable way of measuring the disk throughput. In this thesis, Ganglia tool was used for resource monitoring. Ganglia does not provide feature for measuring the disk throughput. A patch for Ganglia source, found in the Internet, is tried in order to measure the disk throughput results but the measured results were not consistent in different runs, which prevent it from being used in scientific study. Having said so, disk throughput measurement became one of the limitation of this thesis. Other limitation is the limited hardware resources.

For future work, we propose Hadoop and Spark be tested with other Big Data workloads in different application domain. Additionally, the tests may also be repeated with more than 8 nodes for further speedup analysis. Futhermore, measuring the disk throughput can also be added to the frameworks evaluation.

REFERENCES

Anuradha, J. (2015). A brief introduction on Big Data 5Vs characteristics and Hadoop technology.Procedia computer science, 48, 319-324.

Casado, R., & Younas, M. (2015). Emerging trends and technologies in big data processing.Concurrency and Computation: Practice and Experience, 27(8), 2078-2091.

da Silva Morais, T. Survey on Frameworks for Distributed Computing: Hadoop, Spark and Storm.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters.Communications of the ACM, 51(1), 107-113.

Deng, J., Qu, Z., Zhu, Y., Muntean, G. M., & Wang, X. (2014, January). Towards efficient and scalable data mining using spark. In Information and Communications Technologies (ICT 2014), 2014 International Conference on (pp. 1-6). IET.

Dittrich, J., Quiané-Ruiz, J. A., Jindal, A., Kargin, Y., Setty, V., & Schad, J. (2010). Hadoop++: making a yellow elephant run like a cheetah (without it even noticing). Proceedings of the VLDB Endowment, 3(1-2), 515-529.

Ding, L., Wang, G., Xin, J., Wang, X., Huang, S., & Zhang, R. (2013). ComMapReduce: An improvement of MapReduce with lightweight communication mechanisms. Data & Knowledge Engineering, 88, 224-247.

Gopalani, S., & Arora, R. (2015). Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means. International Journal of Computer Applications, 113(1).

Gu, L., & Li, H. (2013, November). Memory or time: Performance evaluation for iterative operation on hadoop and spark. In High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on (pp. 721-727). IEEE.

Han, R., Zhan, S., Shao, C., Wang, J., John, L. K., Xu, J., ... & Wang, L. (2015, August). Bigdatabench-mt: a benchmark tool for generating realistic mixed data center workloads. InWorkshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware (pp. 10-21). Springer International Publishing.

Hu, H., Wen, Y., Chua, T. S., & Li, X. (2014). Toward scalable systems for big data analytics: A technology tutorial. IEEE Access, 2, 652-687.

Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., & Shahabi, C. (2014). Big data and its technical challenges. Communications of the ACM, 57(7), 86-94.

Jonnalagadda, V. S., Srikanth, P., Thumati, K., & Nallamala, S. H. A Review Study of Apache Spark in Big Data Processing.

Kamburugamuve, S., Fox, G., Leake, D., & Qiu, J. (2013). Survey of Apache Big Data Stack(Doctoral dissertation, Ph. D. Qualifying Exam, Dept. Inf. Comput., Indiana Univ., Bloomington, IN).

Katal, A., Wazid, M., & Goudar, R. H. (2013, August). Big data: issues, challenges, tools and good practices. In Contemporary Computing (IC3), 2013 Sixth International Conference on (pp. 404-409). IEEE.

Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering. In Technical report, Ver. 2.3 EBSE Technical Report. EBSE.

Kupisz, B., & Unold, O. (2015, March). Collaborative filtering recommendation algorithm based on Hadoop and Spark. In Industrial Technology (ICIT), 2015 IEEE International Conference on (pp. 1510-1514). IEEE.

Lee, D., Kim, J. S., & Maeng, S. (2014). Large-scale incremental processing with MapReduce.Future Generation Computer Systems, 36, 66-79.

Lee, K. H., Lee, Y. J., Choi, H., Chung, Y. D., & Moon, B. (2012). Parallel data processing with MapReduce: a survey. AcM sIGMoD Record,40(4), 11-20.

Liang, F., Feng, C., Lu, X., & Xu, Z. (2014, March). Performance benefits of DataMPI: a case study with BigDataBench. In Workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware (pp. 111-123). Springer International Publishing.

Lin, X., Wang, P., & Wu, B. (2013, November). Log analysis in cloud computing environment with Hadoop and Spark. In Broadband Network & Multimedia Technology (IC-BNMT), 2013 5th IEEE International Conference on (pp. 273-276). IEEE.

Narasimhan, R., & Bhuvaneshwari, T. (2014). Big data—a brief study. *Int. J. Sci. Eng. Res*, *5*, 1-4.

Patel, A. B., Birla, M., & Nair, U. (2012, December). Addressing big data problem using Hadoop and Map Reduce. In Engineering (NUiCONE), 2012 Nirma University International Conference on (pp. 1-5). IEEE.

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008, June). Systematic mapping studies in software engineering. In 12th international conference on evaluation and assessment in software engineering (Vol. 17, No. 1, pp. 1-10). sn.

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update.Information and Software Technology, 64, 1-18.

Polato, I., Ré, R., Goldman, A., & Kon, F. (2014). A comprehensive view of Hadoop research—A systematic literature review. Journal of Network and Computer Applications, 46, 1-25.

Sagiroglu, S., & Sinanc, D. (2013, May). Big data: A review. In Collaboration Technologies and Systems (CTS), 2013 International Conference on (pp. 42-47). IEEE.

Sakr, S., Liu, A., & Fayoumi, A. G. (2013). The family of MapReduce and large-scale data processing systems. ACM Computing Surveys (CSUR), 46(1), 11.

Shahrivari, S. (2014). Beyond batch processing: towards real-time and streaming big

Samal, N., & Mishra, N. (2015). Big Data Processing: Big Challenges and Opportunities.Journal of Computer Sciences and Applications,3(6), 177-180. data.Computers, 3(4), 117-129.

Shi, J., Qiu, Y., Minhas, U. F., Jiao, L., Wang, C., Reinwald, B., & Özcan, F. (2015). Clash of the titans: MapReduce vs. Spark for large scale data analytics. Proceedings of the VLDB Endowment,8(13), 2110-2121.

Shikhare, A., & Kulkarni, S. A Case Study Comparing Different Big-Data Handling Approaches Using Hadoop-Hive VS Spark-Shark.

Ularu, E. G., Puican, F. C., Apostu, A., & Velicanu, M. (2012). Perspectives on big data and big data analytics. Database Systems Journal,3(4), 3-14.

Wang, H., Wu, B., Yang, S., Wang, B., & Liu, Y. (2014). Research of Decision Tree on YARN Using MapReduce and Spark. In Proceedings of the 2014 World Congress in Computer Science, Computer Engineering, and Applied Computing.

Yigitbasi, N., Willke, T. L., Liao, G., & Epema, D. (2013, August). Towards machine learning-based auto-tuning of mapreduce. In 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (pp. 11-20). IEEE.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster Computing with Working Sets. HotCloud, 10, 10-10.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., & Stoica, I. (2012, April). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. InProceedings of the 9th USENIX conference on Networked Systems Design and Implementation(pp. 2-2). USENIX Association.

Zhang, Y., Gao, Q., Gao, L., & Wang, C. (2012). imapreduce: A distributed computing framework for iterative computation. Journal of Grid Computing, 10(1), 47-68.

APPENDIX A

Systematic Mapping for Comparing Two Big Data frameworks: Hadoop and Spark

Abstract. There have been lots of debates about which Big Data frameworks are better. They have different advantages and disadvantages as they possess different data processing techniques. The goal of this SM study is to present a literature review and difference in two most popular frameworks named Apache Hadoop and Apache Spark. This study based on the guidelines presented in (Petersen, 2008). To achieve the objectives, 10 research questions are raised, in which 97 studies were evaluated. The results can help researches to find the trends and gaps about current state-of-art about the two Big Data frameworks

A.1. Introduction

The rapid development of the Internet has resulted to the explosion in the size of the datasets generated by Web, social media, sensors and mobile devices that led to emerging Big Data term. Big Data stands for datasets that consists of large volumes of structured and unstructured data and needs to be processed in high speed. These requirements poses big challenges to traditional data processing models.

To deal with such challenges, a variety of cluster computing frameworks have been proposed to support large-scale data-intensive applications on commodity machines (Gu, 2013), MapReduce, introduced by Google, is one such successful framework for processing large data sets in a scalable, reliable and fault-tolerant manner (Dean, 2008).

Apache Hadoop (Apache Hadoop, 2016), is open source implementation of MapReduce. MapReduce is a cyclic data flow that reads the same data iteratively and materializes intermediate results in local disks in each iteration, requiring lots of disk accesses, I/Os and unnecessary computations (Gu, 2013). While this data flow programming model is useful for a large class of applications, there are applications

that cannot be expressed efficiently as acyclic data flows, such as iterative jobs and interactive analytics that reuse a working set of data across multiple parallel operations. Apache Spark (Spark, 2016) that is in memory and MapReduce like computing framework is designed to overcome this shortage of Hadoop (Zaharia, 2010). Spark offers an abstraction called resilient distributed datasets (RDDs), through which reused data and intermediate results can be cached in memory across cluster of machines during the whole iterative process. With RDDs Spark outperforms Hadoop by up to 20 in iterative applications and can be used interactively to query hundreds of gigabytes of data (Zaharia, 2012).

This SM study is structured as follow; related work is presented in Section A.2. Section A.3 explains the research methodology and research questions. Classification scheme is presented in Section A.4. The results of this SM study is given in section A.5. Discussion of the results is provided in section A.6.

A.2. Related Work

This section explores a short review of secondary studies regarding Big Data frameworks.

In a survey (da Silva Morais, 2015), the work flows and architecture of Hadoop, Spark and Storm are explored. It states that Spark handles data analytics in a faster way through in-memory computation. This paper also indicates that Storm is not as replacement but complement to Hadoop for real-time. Additionally, in the same paper a short comparison between Spark streaming and Storm is given.

In (Kamburugamuve, 2013), the layered architecture that shows the relationship between the layers and their projects of Apache Open Source Big Data projects is presented. Apache Hadoop and Apache Spark are included in Runtime Systems layer in the Stack.

A literature review is conducted in (Polato, 2014), to help the Apache Hadoop research contributors in finding gaps and encourage them for new studies. The results indicate that few impressive ideas developed while never integrated into the framework. In general, it indicates Hadoop as an evolved and solid Big Data processing framework.

The cons and pros of MapReduce framework and optimization strategies for the framework in the recent years are discussed in (Lee, 2013). Additionally, in the same paper open issues and challenges regarding parallel data processing with MapReduce is given.

In a survey (Sakr, 2013), the mechanisms and approaches of processing huge datasets that are established on the initial concept of the MapReduce framework are conducted. It also includes few popular systems that support supplementary interfaces above the MapReduce framework. Finally, it indicates Hadoop which is an open-source implementation of MapReduce examined to be adequately sophisticated and extensively used by industry and academia in various solutions.

An extensive discussion of big data from Hadoop framework perspective is given in (Anuradha, 2015). And in the same paper Big Data is identified. In Addition the detailed information about different components of Hadoop is presented.

A literature survey considering few mechanisms and methods in various Big Data phases is explored in (Hu, 2014). This paper also introduces Hadoop and the lifecycle of Big Data.

A.3. Research Methodology

SM approach is used in this study. SM allows an effective way to describe the state-of-art of a specific research area. It also provides the ideas about existing studies by classifying them into categories. This study based on the guidelines presented in (Petersen, 2008), as shown in Figure A.1.

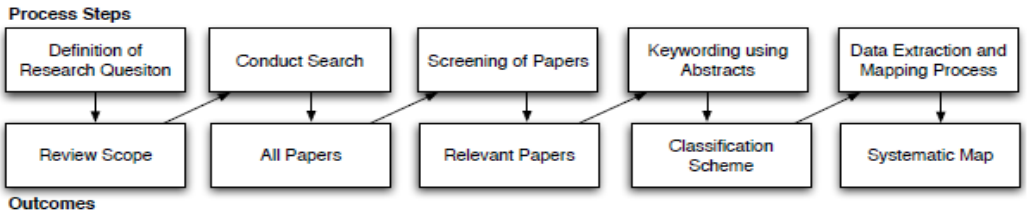

Figure A.1: The Systematic Mapping Process

The output of each step in Figure A.1 is the input for the next step of the process and terminates with systemic mapping report.

The process includes (article selection, classification scheme, and the conduction of systematic and its results) sections.The rest of this section describes goal and research questions and paper selection strategy.

A.3.1. Goal and Research Questions

The primary aim of this SM is to present an overview about current state-of-art and analyzing the difference between two Big Data framework: Hadoop and Spark. The goal of this SM study is demonstrate the trends in the area of Big Data by calculating the studies per categories that is designed based on research questions. Therefor the research questions are declared to shed light on the trends and the differences of the two Big Data framework. The questions that are considering the properties of the two framework were kept out of scope, since they will not advice for finding trend and describing their difference.

RQ1: Number of papers by contribution facet: What types of contributions are made by the papers? How many studies propose approaches/methods/techniques, frameworks, tool, model, architecture, analyze, comparison, empirical (case) study, overview? In order to advise the researches about directions in the research field that needs more consideration.

RQ2: Number of papers by research facet: What type of research methods are used in the papers? The study might introduce a solution or it might include experimental study approaches (Petersen, 2008). The aim of this research question is classify the studies based on research facets and understand the maturity of the research area.

RQ3: Big Data framework(s): Which Big Data framework is used/examined in the papers? The goal of this research questions it to find out which of the frameworks are adopted by the papers. Furthermore, it can be used to find the correlation with other categories.

RQ4: Purpose of the Contribution: What is the purpose of the contribution for each paper? The aim of this research question is to find out whether the paper considering performance enhancement or use-case enhancement or both of them.

RQ5: Programming Language: Which programming language is most commonly used? In order to find out which programming language is mostly used by which Big Data framework.

RQ6: Efficiency Cost: What is the efficiency cost of each framework? The aim is to find the challenges in terms of computing efficiency of each framework.

RQ7: Data sources: What is/are the data source(s) used in the papers? The purpose is to understand which data types are more adopted by which framework.

RQ8: Use-Case: What are the use-cases that Hadoop and Spark are best suit for? In order to understand which of the frameworks mostly used for which type of use-cases, such as: Batch Processing, Real-time Processing.

RQ9: Big Data analytic type: Which of the two the frameworks is preferred for a specific Big Data analytic type? The aim is to find which of the frameworks best suits for which Big Data analytics.

RQ10: Demographic and bibliometric: The research questions listed below are to figure out the bibliometric and demographic of publications.

- **RQ 10.1: Publication count by year:** What is the sum of publications per each year
- **RQ 10.2: Top-cited papers:** Which papers have been cited more frequently by other studies?
- **RQ 10.3: Top venues types:** Which venues types are more preferred by the papers in Big Data research area?
- **RQ 10.4: Preferred Venues:** Which venues have a higher ratio of publications?
- **RQ 10.5: Author Affiliation:** What are the author's affiliations in the studies?

A.3.2. Paper Selection Strategy

This SM study begins with collecting of related papers. The steps listed below are followed:

- Source selection and developing search keywords (Section A.3.2.1).

- Exclusion criteria (Section A.3.2.2).
- Inclusion criteria (Section A.3.2.3).
- Final pool of papers and the online repository (Section 3.3.2.4).

A.3.2.1 Source Selection and Developing Search Keywords

The search engines listed below are used to find the related papers.

- ACM Digital Library[25]
- CiteSeerX[26]
- IEEE Digital Library[27]
- Google Scholar[28]
- Science Direct[29]

Different key words are tried to minimizing the possibility of missing any papers that is related to the selected frameworks. The final list of key words is listed in the Table A.1.

The papers are collected in the second half of 2015. Pool of studies include studies between the years 2007 and 2015 also including the papers from second half of 2015.

Table A.1: Search Keywords used in our SM study

• Hadoop • Spark • MapReduce	Both frameworks' names are used in the search, as Hadoop is implementation of MapReduce, therefore MapReduce also included as the keyword search.
• Big Data frameworks • Hadoop or Spark • Hadoop vs. Spark • Why Hadoop? • Why MapReduce? • Why Spark?	The combo search keywords included to minimize the risk of missing related papers for Hadoop and Spark frameworks. This strategy enables us to find many papers but the papers had to be checked if they will be included or excluded that we will explain them next section.

[25] http://dl.acm.org/
[26] http://citeseerx.ist.psu.edu/index
[27] http://ieeexplore.ieee.org/Xplore/home.jsp
[28] https://scholar.google.com
[29] http://www.sciencedirect.com/

A.3.2.2. Exclusion Criteria

The following principles are used for the exclusion of the papers.

- The studies that don't takes Hadoop and Spark as the main consideration.
- The papers that are not written in English.
- Short papers.
- The manual papers that are only introducing the frameworks.
- Duplicate papers.

A.3.2.3. Inclusion Criteria

The next step is paper inclusion phase. Below methods are followed to lessen the risk of missing papers the:

- Studies that are referenced by the papers already exist in the pool.
- Studies from the popular Big Data research venues.

A.3.2.4. Final Pool of Papers and the Online Repository

At the beginning the pool contained 160 papers. Number of the papers dropped to 97 papers after the exclusion.

A.4. Classification Scheme

The classification scheme is developed after an extensive analysis. The aim was to develop a classification scheme to hold the answer for each research questions. To achieve this goal, firstly common grounds are specified. During the data extraction, the initial classification scheme is changed iteratively as a result of discovering new categories. The corrections are terminated when it is believed that the classification scheme can include all papers into its categories.

The classification scheme is presented in Table A.2. The table contains four columns for research questions, attributes, value sets, and single or multiple choice attributes. The multi-choice attributes indicate that the paper might contain multiple attributes for the same category.

Table A.2: Classification scheme developed and used in our study

RQ	Attribute	Value set	(M)ultiple/(S)ingle attribute
1	Type of paper – Contribution facet	{Method/Technique/Approach, Framework, Model, Tool, Architecture, Platform, Empirical (Case) study, Analyze, Comparison, Overview, Other}	M
2	Type of paper – Research facet	{Solution Proposal, Validation Research, Evaluation Research, Experience Papers, Philosophical Papers, Opinion Papers, Other}	M
3	Big Data frameworks used/examined.	{Hadoop, Apache Spark}	M
4	Purpose of the Contribution	{Use-Case enhancement, Performance enhancement, None}	M
5	Programming Language(s) used.	{Java, Python, Scala, Ruby, Others, Not mentioned}	M
6	Efficiency Cost	{Memory Efficiency, Performance Efficiency, Staff Availability, Other, Not mentioned}	M
7	Data source(s)	{Sensor data, Social Media data, Graphical data, Geospatial data, Log data, Raw data, Generic, Other, None}	M
8	Use-Case(s)	{Batch Processing, Real-time Processing, Not mentioned }	S
9	Big Data analytic type	{Iterative/Algorithm Jobs, Machine Learning, Interactive Analytics, Other, Not mentioned }	M
10	Demographic and bibliometric		
10.1	Publication count by year	Year: Integer	
10.2	Top-cited papers	Number of Citations: Integer	
10.3	Top Venues	{Conference, Journal, Workshop, Symposium, Thesis, Book, Magazine}	
10.4	Preferred venues	Venues: String[]	
10.5	Author affiliation	{Academic, Industry, Collaborative}	

A.4.1. Classification Scheme Explanation

This section includes the clarification of the attributes in the categories of the classification scheme in Table A.2.

Type of paper: Contribution facet (RQ 1)

The term "contribution facets" is described in (Petersen, 2008). Contribution facets stands for the type of the contributions, like: approach/method/technique/, model, tool. For this SM study additional contribution facets are added such as: architecture, analyze, framework, overview, empirical (case) study, comparison, framework, other.

Type of paper: Research facet (RQ 2)

Research facet stands for the type of the research approach used in the studies. This category follows the guidelines in (Petersen, 2008). The following are the research facets used for this study:

Solution Proposal: A solution for a problem is proposed, the solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution can be shown by a small example or a good line of argumentation.

Validation Research: Techniques investigated that have not been implemented yet in practice and are novel. Techniques used in experiments, i.e., work done in labs.

Evaluation Research: Techniques are implemented in practice and an evaluation of the technique is conducted.

Experience Papers: Experience papers are the personal experiences of the authors and they explain on what and how something has been done in practice.

Big Data framework(s) used or examined (RQ 3)

This study contains two popular Big Data frameworks: Apache Hadoop and Apache Spark. This category clarifies which of the two frameworks is used or examined in the studies

Purpose of the Contribution (RQ 4)

The purpose of the contribution facet might be either:

- **Use-Case Enhancement:** This means the study is considering the enhancement of one of the use-cases, regardless of its performance, by integrating new models, methods, techniques, etc.
- **Performance Enhancement:** This mean the article is considering with enhancing the performance of one of the frameworks by altering or introducing new mechanism, methods, algorithms, architecture, etc.

Programming Language(s) used (RQ 5)

Hadoop and Spark frameworks provide API for different languages, therefore the aim is to examine which programming language mostly used by which of the frameworks in the studies.

Efficiency Cost (RQ 6)

Computing resources are vital to get the desired performance, therefore it is important to understand the resource efficiency cost of the two frameworks. Resource efficiency is categorized as memory efficiency and performance efficiency. In addition to resources, the staff availability is also considered as the cost.

Data Source(s) (RQ 7)

In a Big Data there are few data sources, and this category classifies these data sources, as the Big Data frameworks are capable to process disparate data sources.

Use-Case(s) (RQ 8)

The Big Data frameworks are capable of processing data in batch mode and real-time mode. We interested to know which of our selected frameworks is more suitable to process data as batch processing or real-time processing.

Big Data Analytics Type (RQ 9)

It’s a great output to understand which framework is capable of handling which type of Big Data analytics efficiently

Demographic and Bibliometric (RQ 10)

This category considers the bibliometric and demographic of the studies, the clarification in Table A.2 should be self-explanatory

A.5. Results of Systematic Mapping

The results of the SM study are explored in this section that are derived from the data extraction of ninety seven papers. The charts are used to illustrate the results of each research question

A.5.1. Results from Research Questions

This section presents the results for 10 RQs defined in section A.3.1

RQ 1: What types of contributions are made by the papers?

Figure A.2 shows the dispersion of the 97 studies included in this SM study into contribution facet types. Some of studies were classified under more than one facet. For example (Dittrich, 2010) proposes two contributions: (1) a technique and (2) a framework

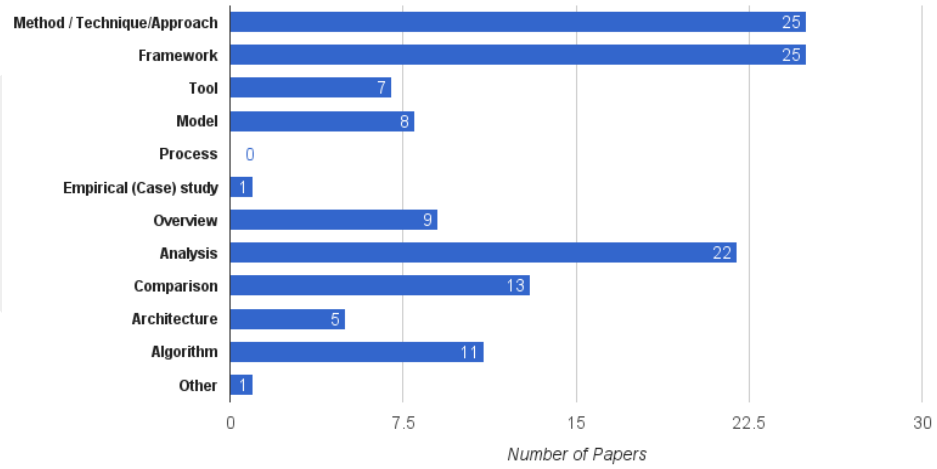

Figure A.2: Search Keywords used in our SM study

Figure A.2 shows that, presenting new methods/techniques/approaches, proposing new frameworks, and papers that present analysis have attracted the researchers with 25, 25, and 22 papers respectively. Also, comparing the existing frameworks,

proposing new algorithms and overview of the frameworks has taken high proportion with 13, 11, and 9 papers respectively. There were 1 paper which could not be categorized into our contribution facet categories, thus this it is categorized under 'Other'.

RQ 2: What type of research methods are used in the papers?

Figure A.3 depicts the distribution of the type of papers by research facet. The Solution proposal and Evaluation researches papers have dominated the research facet with 40 papers for each of them. Validation researches follow them with the proportion of 14 papers. There are only 2 experience studies.

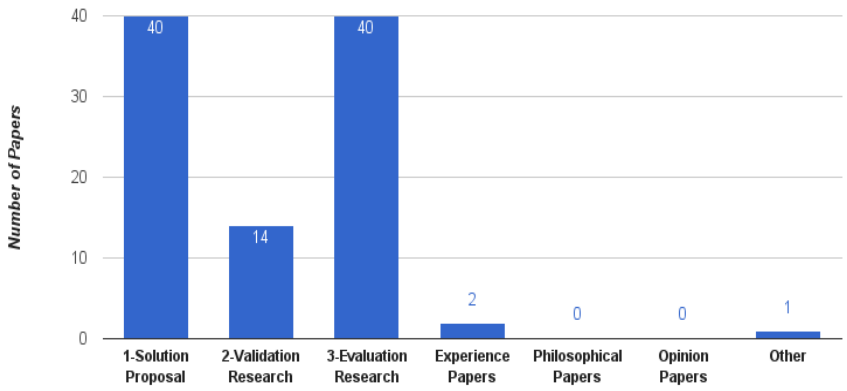

Figure A.3: Research Facets

RQ 3: Which Big Data frameworks are used/examined in each of paper?

Figure A.4 illustrates the counts of Big Data frameworks that are either used or examined in the papers. Hadoop has the major proportion of the counts in the papers with approximately two thirds of papers. And the rest of the papers are about Spark and comparison papers of the Hadoop and Spark frameworks with 15, 13 papers respectively.

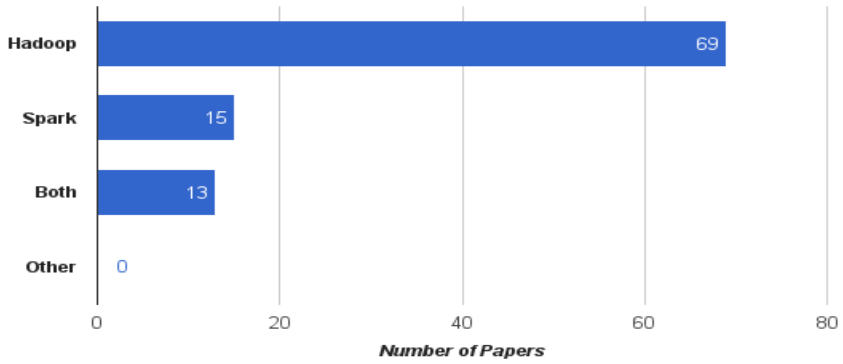

Figure A.4: Number of publication counts for each Big Data frameworks

RQ 4: What is the purpose of the contribution for each paper?

Figure A.5 shows the distribution of the purpose of the contribution which can be either performance or use-case enhancement. For example; a proposed method can be intended for performance enhancement by modifying the frameworks internals and/or its design. Also another paper proposing a new method can be intended for usage enhancement, for example; in (Lee, 2014), a modified Hadoop architecture tailored to large-scale incremental processing with conventional MapReduce algorithms.

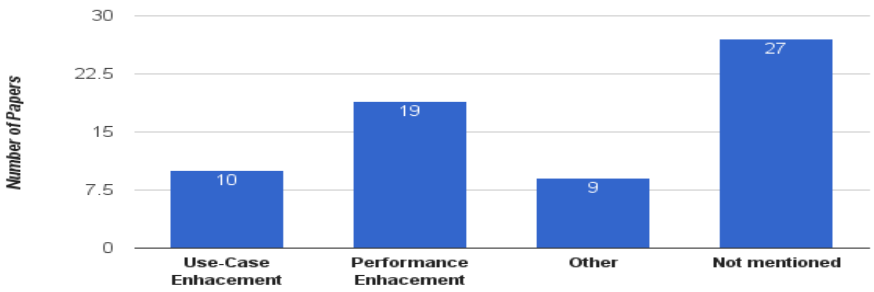

Figure A.5: Purpose of the contributions

As Figure A.5 shows the community interest is in performance enhancement more than that of use-case enhancement, and from the analyzations it is found that more of the use-case enhancements were about iterative jobs and algorithms in MapReduce of Hadoop.

RQ 5: Which programming language is most commonly used?

Figure A.6 shows the counts for each programming language used in implementation phase in the papers. Overall Java has been used by quarter of the papers and 11 papers have used Scala for their experiments. Python has been used in only two papers.

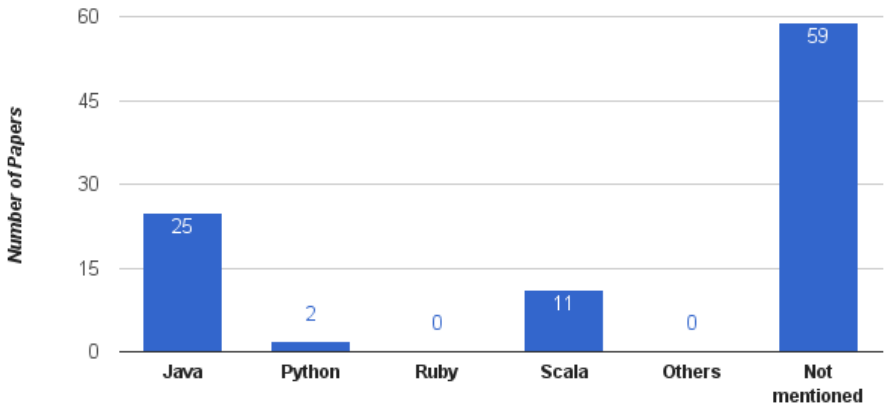

Figure A.6: Programming languages used in experimentation

RQ 6: What is the Efficiency Cost of each framework?

Figure A.7 illustrates the challenges for efficiency cost of each frameworks. Majority of the papers didn't mention them in their evaluation. One fourth of the papers mentioned performance and memory efficiency challenges in their experiments with 18 and 9 papers respectively and only 1 paper mentioned staff availability as challenge.

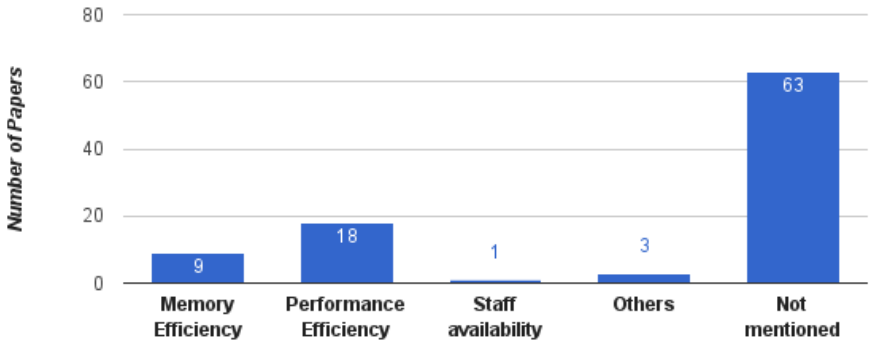

Figure A.7: Efficiency Cost

Figure A.8 shows that all the memory efficiency cost papers are related to Spark and all the performance efficiency cost papers are related to Hadoop. And the only one paper that is regarding staff availability is related to Spark.

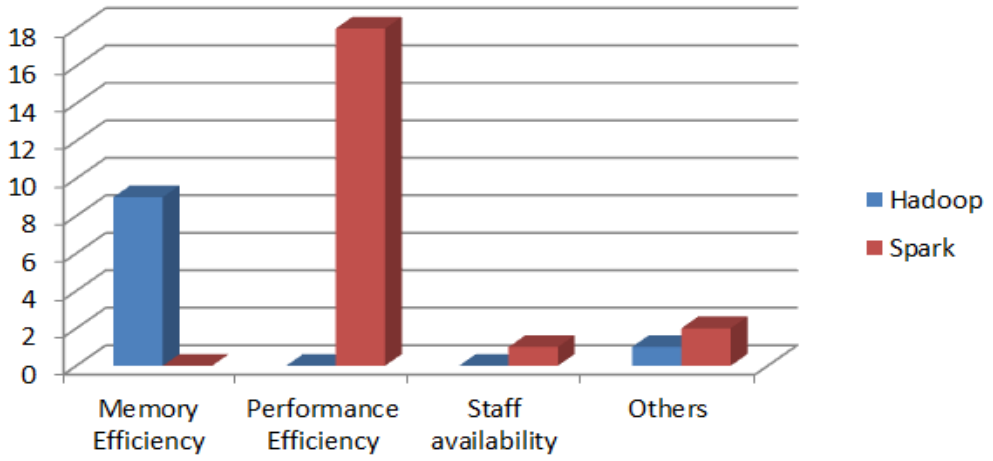

Figure A.8: Efficiency Cost for Hadoop and Spark

RQ 7: What is/are the data source(s) used in the papers for experimentation?

Figure A.9 depicts the data source(s) used for the experiments in the papers. The most used data type is raw data (CSV and text files) with the count of 35 times. 11 papers used more than one data type for their evaluation. Only a small number of the papers used graphical, social media, and sensor data with 3, 3, and 1 papers respectively.

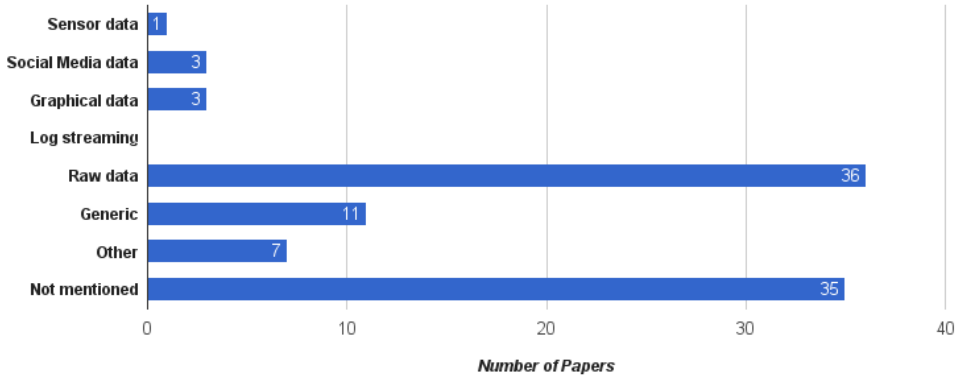

Figure A.9: Data sources used in the papers

RQ 8: What are the use-cases that they (Hadoop and Spark) best suit for?

As its clear from Figure A.10, only 9 papers mentioned about use-cases suitability for the frameworks, that is 3 papers for batch processing and 6 papers for real-time processing.

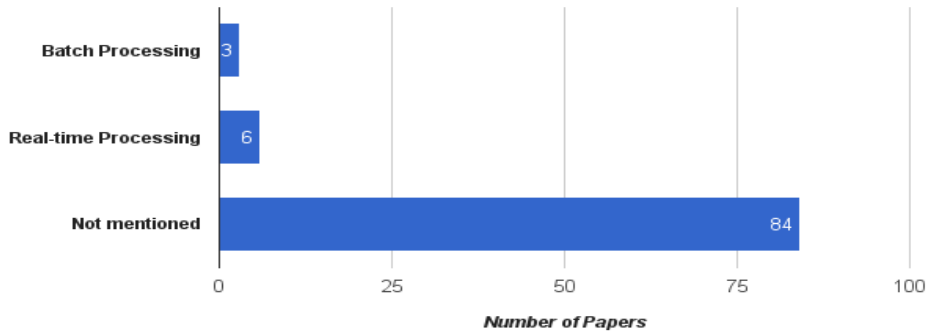

Figure A.10: Use-Cases

Figure A.11 shows that all the three batch processing papers indicates that Hadoop is suitable for batch processing. Regarding real-time processing, five of the papers indicates that Spark is suitable for real-time processing and only one paper states that Hadoop is suitable for real-time processing

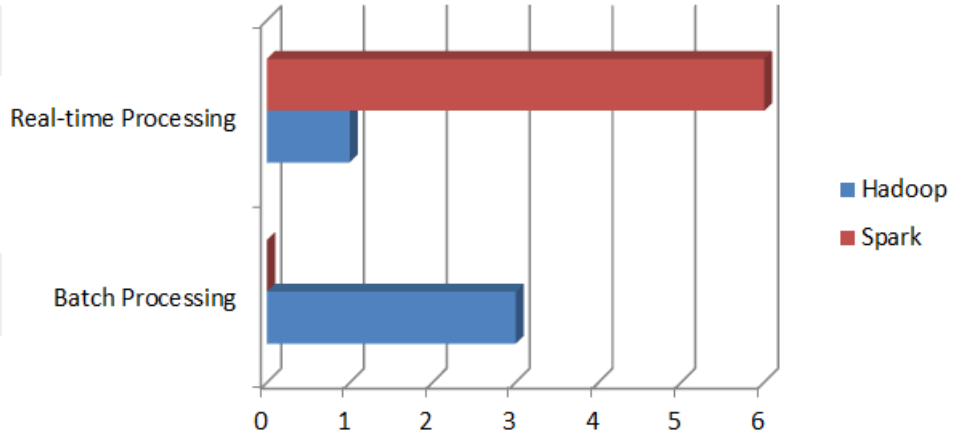

Figure A.11: Use-Cases for Hadoop and Spark

RQ 9: Which of the two frameworks is preferred for a specific Big Data analytic type?

Figure A.12 shows which of the two frameworks is suitable for Big Data analytic types. From the figure A.12 it can be seen that Spark is suitable for all type of Big Data analytics. Because in-memory computation characteristic of Spark makes it able to address the execution time problem involved in iterative or machine learning algorithms (Shikhare). There were seven papers out of fifteen papers, that shows Hadoop can be applied to Iterative/Algorithm Jobs by adding some techniques to the Hadoop MapReduce, for example (Zhang, 2012).

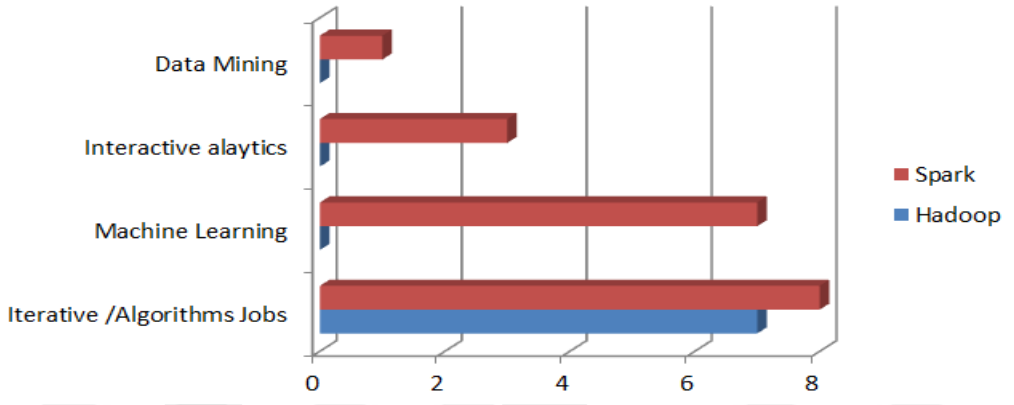

Figure A.12: Big Data analytics

RQ 10 – Demographic and Bibliometric

The results about demographic and bibliometric of the publications are presented in this section.

RQ 10.1: What is the number of publications per each year?

Figure A.13 shows the number of publications per year. There is a slight degradation in number of papers between the years 2009 and 2011. And there is a sharp increase in number of papers between years 2014 and 2015 from 13 papers to 28 papers. Figure A.13 shows that, during that years 2014 and 2015 the researchers are more interested in Big Data field comparing to the 2009 and 2011 years.

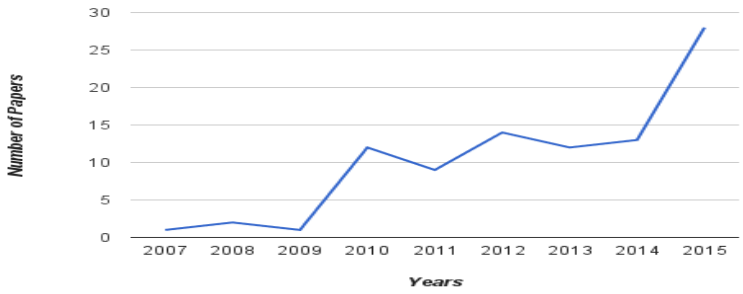

Figure A.13: Publication count by year

RQ 10.3: Which venues types are more preferred by the studies in Big Data research area?

Figure A.14 shows the number of papers published per venue types. It is identified that 88 papers gave information about venues, and 9 of the papers did not include any information about venues. Figure A.14 shows that most of the papers are published in Conferences, Journals, and Symposium with 33, 3, and 20 publications respectively.

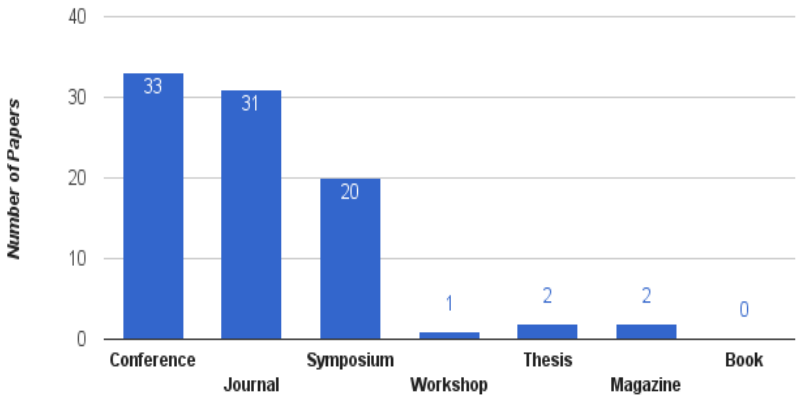

Figure A.14: Distribution of the venues for the contributions

RQ 10.4: Which venues include the higher ratio of publications?

Figure A.15 shows eight venue abbreviations that the papers in the study pool have been published in these venues for more than three times and more. International

Journal on Very Large Data Bases (VLDB) has the highest ration of publications with 7 papers.

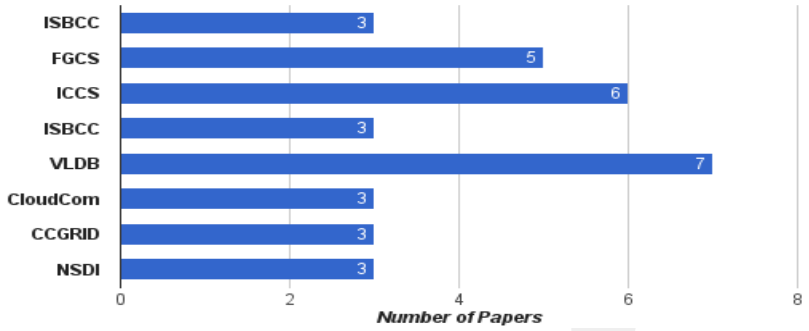

Figure A.15: Preferred venues for the contributions

Table A.3: Name and abbreviations of the venues in Figure A.15

Venue acronyms	Venue names
ISBCC	International Symposium on Big Data and Cloud Computing
FGCS	Future Generation Computer Systems
ICCS	International Conference on Computational Science
VLDB	The International Journal on Very Large Data Bases
ISBCC	International Symposium on Big Data and Cloud Computing
CloudCom	IEEE International Conference on Cloud Computing Technology and Science
CCGRID	IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing
NSDI	USENIX Symposium on Network System Design and Implementation

RQ 10.5: What are the author's affiliations in the studies?

Figure A.16 shows the ratio of publications per affiliation type. It can be seen from the figure that the researcher's affiliations are mostly from academy with %69.1 ratio. Collaboration means the research conducted with the authors from both academy and industry field.

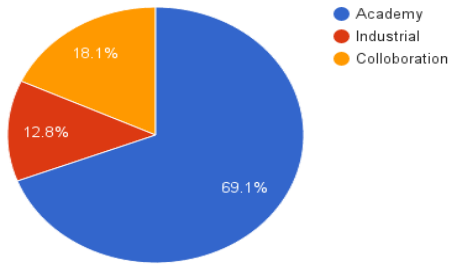

Figure A.16: Author Affiliations

A.6. Discussion

From the research question results, it is found that Hadoop and Spark are widely used frameworks in the field of Big Data. From Figure A.4 it is realized that the majority of the papers examined Hadoop framework. Also, the research community is more interested in contributing for new method/approach/techniques or presenting new frameworks by adding some features on the top of existing frameworks as shown in Figure A.2. Figure A.5 shows that the intention of the contributions is mostly related to performance enhancement and use-case enhancements. Majority works of use-case enhancements are about iterative jobs and algorithms in MapReduce of Hadoop such as in (Lee, 2014).

From the Big Data analytics perspective, the results in Figure A.12 shows that Apache Spark is suitable for all types of Big Data analytics. This is because of the in-memory computation characteristics of Spark that makes it able to address the performance problems involved in iterative or machine learning algorithms.

Figure A.13 states that in the recent years the researchers are more interested to Big Data field comparing to the previous years. Majority of the publication's affiliations are from academy with %69.1 ratio as shown in Figure A.16