# OPTIMISATION MODEL FOR SCHEDULING MAPREDUCE JOBS IN BIG DATA PROCESSING

IBRAHIM ABAKER TARGIO HASHEM

FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2017

# OPTIMISATION MODEL FOR SCHEDULING MAPREDUCE JOBS IN BIG DATA PROCESSING

**IBRAHIM ABAKER TARGIO HASHEM**

**THESIS SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY**

**FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY**
**UNIVERSITY OF MALAYA**
**KUALA LUMPUR**

**2017**

**UNIVERSITY OF MALAYA**

**ORIGINAL LITERARY WORK DECLARATION**

Name of Candidate: Ibrahim Abaker Targio Hashem Registration/Matric No: WHA120037

Name of Degree: Doctor of Philosophy

Title: Optimization Model for Scheduling MapReduce Jobs in Big Data Processing

Field of Study: Big Data Scheduling

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

**UNIVERSITI MALAYA**
**PERAKUAN KEASLIAN PENULISAN**

Nama: Ibrahim Abaker Targio Hashem
No. Pendaftaran/Matrik: WHA120037
Nama Ijazah: Doctor of Philosophy
Tajuk Kertas Projek/Laporan Penyelidikan/Disertasi/Tesis: Optimisation Model for Scheduling MapReduce Jobs in Big Data Processing

Bidang Penyelidikan: Big Data Scheduling

Saya dengan sesungguhnya dan sebenarnya mengaku bahawa:

(1) Saya adalah satu-satunya pengarang/penulis Hasil Kerja ini;
(2) Hasil Kerja ini adalah asli;
(3) Apa-apa penggunaan mana-mana hasil kerja yang mengandungi hakcipta telah dilakukan secara urusan yang wajar dan bagi maksud yang dibenarkan dan apa-apa petikan, ekstrak, rujukan atau pengeluaran semula daripada atau kepada mana-mana hasil kerja yang mengandungi hakcipta telah dinyatakan dengan sejelasnya dan secukupnya dan satu pengiktirafan tajuk hasil kerja tersebut dan pengarang/penulisnya telah dilakukan di dalam Hasil Kerja ini;
(4) Saya tidak mempunyai apa-apa pengetahuan sebenar atau patut semunasabahnya tahu bahawa penghasilan Hasil Kerja ini melanggar suatu hakcipta hasil kerja yang lain;
(5) Saya dengan ini menyerahkan kesemua dan tiap-tiap hak yang terkandung di dalam hakcipta Hasil Kerja ini kepada Universiti Malaya ("UM") yang seterusnya mula dari sekarang adalah tuan punya kepada hakcipta di dalam Hasil Kerja ini dan apa-apa pengeluaran semula atau penggunaan dalam apa jua bentuk atau dengan apa juga cara sekalipun adalah dilarang tanpa terlebih dahulu mendapat kebenaran bertulis dari UM;
(6) Saya sedar sepenuhnya sekiranya dalam masa penghasilan Hasil Kerja ini saya telah melanggar suatu hakcipta hasil kerja yang lain sama ada dengan niat atau sebaliknya, saya boleh dikenakan tindakan undang-undang atau apa-apa tindakan lain sebagaimana yang diputuskan oleh UM.

Tandatangan Calon Tarikh:

Diperbuat dan sesungguhnya diakui di hadapan,

Tandatangan Saksi Tarikh:

Nama:
Jawatan:

## ABSTRACT

With the fast development of Internet-based technologies, data generation has increased drastically over the past few years, coined as big data era. Big data offer a new paradigm shift in data exploration and utilization. The major enabler for underlying many big data platforms is certainly the MapReduce computational paradigm. Scheduling plays an important role in MapReduce, mainly in reducing the execution time of data-intensive jobs. However, despite recent efforts toward improving MapReduce performance, scheduling MapReduce jobs across multiple nodes have shown to be multi-objective optimization problem. The problem is even more complex using virtualized clusters in a cloud computing to execute a large number of tasks. The complexity lies in achieving multiple objectives that may be of conflicting nature. These conflicting requirements and goals are challenging to optimize due to the difficulty of predicting a new incoming job's behavior and its completion time. In this study, we aim to optimize task scheduling and resource utilization using an evolutionary algorithm based on the proposed completion time and monetary cost of cloud service models. The multi-objective approaches which are, Sorting Genetic Algorithm II (NSGA-II) and Strength Pareto Evolutionary Algorithm II (SPEA2) are applied to find the Pareto front of the Makespan and total cost. The result of our experiment analysis reveals that the advantage of NSGA-II over the SPEA2 on the tested problems based on the adopted measuring criteria. In addition, NSGA-II algorithm was able to find the optimal solutions. We then proposed a multi-objective scheduling algorithm framework that considers resource allocation and task scheduling in a heterogonous cloud environment. The proposed algorithm is evaluated using tasks scheduling in the scheduling load simulator and

validated using statistical modeling. The simulation results acquired from the experiments showed the effectiveness of the proposed framework and algorithm.

## ABSTRAK

Dengan perkembangan pantas teknologi berasaskan Internet, penjanaan data telah meningkat secara drastik sejak beberapa tahun kebelakangan ini, dikenali sebagai era data yang besar. Data besar menawarkan anjakan paradigma baru dalam penerokaan data dan penggunaannya. Pemboleh utama bagi untuk platform data besar adalah pengkomputeran MapReduce paradigma. Penjadualan memainkan peranan yang penting dalam MapReduce, terutama sekali dalam mengurangkan masa pelaksanaan pekerjaan intensif-data. Walau bagaimanapun, walaupun usaha terbaru ke arah meningkatkan prestasi MapReduce sedang dilakukan, kerja penjadualan MapReduce merentasi pelbagai nod telah menunjukkan ia menjadi masalah objektif-pelbagai pengoptimuman. Masalah menjadi bertambah kompleks dengan menggunakan kelompok maya dalam pengkomputeran awan untuk melaksanakan jenis tugas yang besar. Kerumitan ini terletak dalam mencapai pelbagai objektif yang mampu menjadi percanggahan semula jadi. Keperluan yang bercanggah dan matlamat adalah mencabar untuk pengoptimuman berikutan kesukaran meramalkan kelakuan kerja masuk yang baru dan waktu ia siap. Dalam kajian ini, kami berhasrat untuk optimumkan penjadualan tugas dan penggunaan sumber dengan menggunakan algoritma evolusi berdasarkan cadangan waktu dan kos kewangan dalam model perkhidmatan awan. Pendekatan pelbagai-objektif adalah, Menyusun Algoritma Genetik II (NSGA-II) dan Kekuatan Pareto Evolusi Algoritma II (SPEA2) digunakan untuk mencari Pareto awal daripada waktu dan jumlah kos. Keputusan analisis eksperimen mendedahkan kelebihan NSGA-II lebih pada SPEA2 pada masalah yang diuji berdasarkan kriteria pengukur yang digunakan. Di samping itu, algoritma NSGA-II mampu mencari penyelesaian yang optimum. Kemudian kami cadangkan rangkakerja algoritma penjadualan pelbagai-objektif mengambilkira peruntukan sumber dan penjadualan tugas dalam persekitaran awan yang pelbagai. Algoritma yang dicadangkan menggunakan tugasan penjadualan dalam simulator

penjadualan dan disahkan menggunakan pemodelan statistik. Keputusan simulasi dari eksperimen menunjukkan keberkesanan rangka kerja dan algoritma yang cadangkan.

## ACKNOWLEDGEMENTS

I wish to give my gratitude to the almighty Allah for giving me the opportunity to complete the thesis. My sincere appreciation goes to my supervisor, Dr. Nor Badrul Anuar Jumaat, Faculty of computer Science and Information Technology for taking his time to guide and thoroughly go through each and every line of the thesis despite his tight schedules. I believed the constructive comments of my supervisor have significantly improved the quality of the thesis which could have not being so without his inputs. I would like also to express my sincere gratitude to Prof. Abdullah Gani for the continuous support of my Ph.D. study and research.

Thanks are not enough to be given to my parents, for their ongoing supports. There are no words that are valuable to equalize their love for me. This thesis would not have been possible without the support of my brother, Jamal, who always support me and encouraged me to pursue my study.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| ACID | Atomicity, Consistency, Isolation, Durability |
| ASF | Apache Software Foundation |
| B2B | Business-to-Business |
| Doc | Document |
| DSMS | Data Stream Management System |
| EC2 | Amazon Elastic Compute Cloud |
| GFS | Google File System |
| HDDs | Hard Disk Drives |
| HDFS | Hadoop Distributed File System |
| IaaS | Infrastructure as a Service |
| ICT | Information Communication Technology |
| BAR | BAlance-Reduce |
| IT | Information Technology |
| JSON | JavaScript Object Notation |
| KV | Key Value |
| NAS | Network Attached Storage |
| NoSQL | Not Only SQL |
| OLM | Online Lazy Migration |
| PaaS | Platform as a Service |
| PDF | Portable Document Format |
| RDBMS | Relational Database Management System |
| SAN | Storage Area Network |
| SQL | Structured Query Language |
| SDLM | Scientific Data Lifecycle Management |

| | |
|---|---|
| S3 | Simple Storage Service |
| SaaS | Software as a Service |
| URL | Uniform Resource Locator |
| XML | Extensible Markup Language |
| NSGA-II | Non-dominated Sorting Genetic Algorithm II |
| SPEA2 | Strength Pareto Evolutionary Algorithm 2 |
| ZB | Zettabyte |
| SLA | Service-level agreement |
| FIFO | First in, first out |
| QoS | Quality of Service |
| LAN | Local Area Network |
| CPU | Central processing unit |
| I/O | Input/Output |
| MOEA | Multi-objective Evolutionary Algorithms |

# CHAPTER 1: INTRODUCTION AND OVERVIEW

With the fast development of Internet-based technologies; data generation has increased drastically over the past few years. These data can be stored in low-cost, commodity computers in a distributed network to be used for analytics to extract knowledge as well as other purposes. Data creation is occurring at a record rate (Villars et al., 2011), referred to herein as big data, and has emerged as a widely recognized trend. The term "big data" refers to a set of processing techniques and technologies that require new forms of integration to uncover largely hidden values from large datasets that are diverse, complex, and in a massive scale. Big data processing assists data scientists in uncovering hidden patterns and other useful information from huge volumes of data that conventional processing and business intelligence cannot solve. Moreover, big data is transforming healthcare, science, engineering, finance, business, and eventually, the society. The major enabler for underlying many big data applications is certainly the MapReduce computational paradigm (Dean & Ghemawat, 2008).

MapReduce is the most popular framework for processing the existing large-scale data (Dean & Ghemawat, 2008) (Dean & Ghemawat, 2010) primarily because of its important features that include scalability, fault tolerance, ease of programming, and flexibility. Nowadays, MapReduce is primarily used for expressing distributed computations on the massive amounts of data and an execution framework for large-scale data processing on clusters of commodity servers.

Clusters in the cloud computing environment can include more than one MapReduce jobs running simultaneously. Each job often consists of multiple tasks, many of them periodically scheduled. Hence, for the optimizer to decide on the best execution plan is a critical factor in a scheduling process (Kllapi et al., 2011) (H. Chang et al., 2011). This level of optimization is called job scheduling in MapReduce programming model

(Hammoud et al., 2012; Kc & Anyanwu, 2010; D. Wang et al., 2013). Scheduling has been widely studied in distributed computing literature (Blazewicz et al., 1983; Liu & Layland, 1973; Pinedo, 2012); however, the cost and complexity of adopting traditional scheduling models to big data platforms have increased.

## 1.1 Big Data

Since the invention of computers, large amounts of data have been generated at a rapid rate. This condition is the key motivation for current and future research frontiers. Advances in mobile devices, digital sensors, communications, computing, and storage have provided means to collect data (Bryant et al., 2008). According to the renowned IT company Industrial Development Corporation (IDC; 2011), the total amount of data in the world has increased nine times within five years (Gantz & Reinsel, 2011). This figure is expected to double at least every two years (M. Chen et al., 2014b). Big data is a novel term that originated from the need of large companies, such as, Yahoo, Google, and Facebook, to analyze large amounts of data (Garlasu et al., 2013). The three major motives for big data technology are to minimize hardware costs, check the value of big data before committing significant company resources, and reduce processing costs (Leavitt, 2013). Well-managed big data should exhibit availability, reliability, security, and maintainability (Khan et al., 2014).

## 1.2 MapReduce programming model

MapReduce (Dean & Ghemawat, 2008) is a simplified programming model for processing large amounts of datasets pioneered by Google for data-intensive applications. The model is stunningly simple, and it effectively supports parallelism (Lämmel, 2008). Such a model is adopted through Hadoop implementation, quickly spreading and transforming into a dominant force in the field of big data (Polato et al., 2014). MapReduce enables an inexperienced programmer to develop parallel programs

and create a program that can run in the cloud. In most cases, programmers are required to execute only two functions, namely, the map (mapper) and reduce functions (reducer), which are commonly utilized in functional programming. The mapper regards the key/value pair as input and generates intermediate key/value pairs, and the reducer merges all pairs associated with the same (intermediate) key and then generates an output. The map function is applied to each input (key1, value1), in which the input domain is different from the generated output pairs list (key2, value2). The elements of the list (key2, value2) are then grouped by a key. After grouping, the list (key2, value2) is divided into several lists [key2, list (value2)], and the reduce function is applied to each list [key2, list (value2)] for generating a final result list (key3, value3).

### 1.3 Scheduling

The widespread popularity of big data processing platforms is the growing demand to further optimize their performance for various purposes. In particular, enhancing resources and jobs scheduling are becoming critical since they fundamentally determine whether the applications can achieve the performance goals in different use cases. There are many works published that focus on big data optimization (Facchinei et al., 2014; Richtárik & Takáč, 2015), which aim to improve the processing and completion time of as many tasks as possible (Pop & Cristea, 2015). There are still areas to explore, particularly with respect to scheduling and resource management related to big data processing. According to (Kc & Anyanwu, 2010) the research studies on scheduling optimization problem can be categories into two types: single objective optimization and multi-objective optimization.

*(a) Single objective optimization*. A single objective optimization problem is presented in (J. D. Knowles et al., 2001; Seada & Deb, 2015) by finding an optimal solution that is corresponding to the minimization or maximization based on a single function.

Considering the nature of the study, a single objective can be used to provide decision makers with insights of the problem. However, such solution is unable to handle an alternative result that has conflicting objectives (Savic, 2002).

*(b) Multi-objective optimization*. Most real-world scheduling optimization problems are multi-objective in nature since they normally have possibly conflicting objectives that must be satisfied at the same time. Scheduling often requires considering several objectives in an optimization process. For example, a trade-off between resource consumption and performance while scheduling multiple tasks in the cloud. This optimization procedure manages two or more objectives and is called multi-objective optimization. Several relevant studies include (Rasooli & Down, 2014) (Nita et al., 2015) (S.-Q. Long et al., 2014) are focused primarily on the objective functions, constraints, minimum schedule length and global optimality on solving scheduling big data platforms.

## 1.4 Statement of Problem

Scheduling tasks in MapReduce across multiple nodes have shown to be multi-objective optimization problem. The problem is even more complex using virtualized clusters in a cloud computing to execute a large number of tasks (Smanchat & Viriyapant, 2015). The complexity lies in achieving multiple objectives that may be of conflicting nature (Ruzika & Wiecek, 2005). These conflicting requirements and goals are challenging to optimize due to the difficulty of predicting a new incoming job's behavior and its completion time.

Scheduling plays an important role in big data, mainly in reducing the execution time and cost for processing. MapReduce scheduling model assumes that the processing time of a task on a particular node is fixed and can perform work at roughly the same rate (Zaharia et al., 2008). However, in a most realistic situation apart from the nodes, it

requires additional resources to process jobs and the processing time of a job is determined internally by the amount of the resources allocated. Moreover, Hadoop MapReduce assumed that the resources are similar and the data locality is often the only scheduling constraint (Krish et al., 2014). However, with the use of virtual machines, it leads to Hadoop cluster becoming increasingly heterogeneous, in that a cloud may have several clusters each boasting different characteristics (Krish et al., 2014) (F. Dong & Akl, 2007). Resource heterogeneity in the cluster may compress of either heterogeneous or homogeneous (Polo et al., 2010). Homogeneous clusters means that the nodes in the cluster have similar resources such as CPU, memory, storage and networking capabilities, whereas, heterogeneous clusters, consist of different resources with the nodes in terms of CPU, memory, storage or communication speeds (Lopes & Menascé, 2015) (Zaharia et al., 2008).

Although, MapReduce can achieve better performance with the allocation of more compute nodes from the cloud to speed up computation; however, this approach of "renting more nodes" is less effective in particular the cost in a pay-as-you-go environment (Jiang et al., 2010). Furthermore, MapReduce adopts a runtime scheduling scheme. The scheduler assigns data blocks to the available nodes for processing one at a time. This scheduling strategy introduces runtime cost and may slow down the execution of the MapReduce job (Jiang et al., 2010) as shown in Figure 1-1. In these situations, both the time to complete the jobs and the cost of the resources allocated should be taken into account.

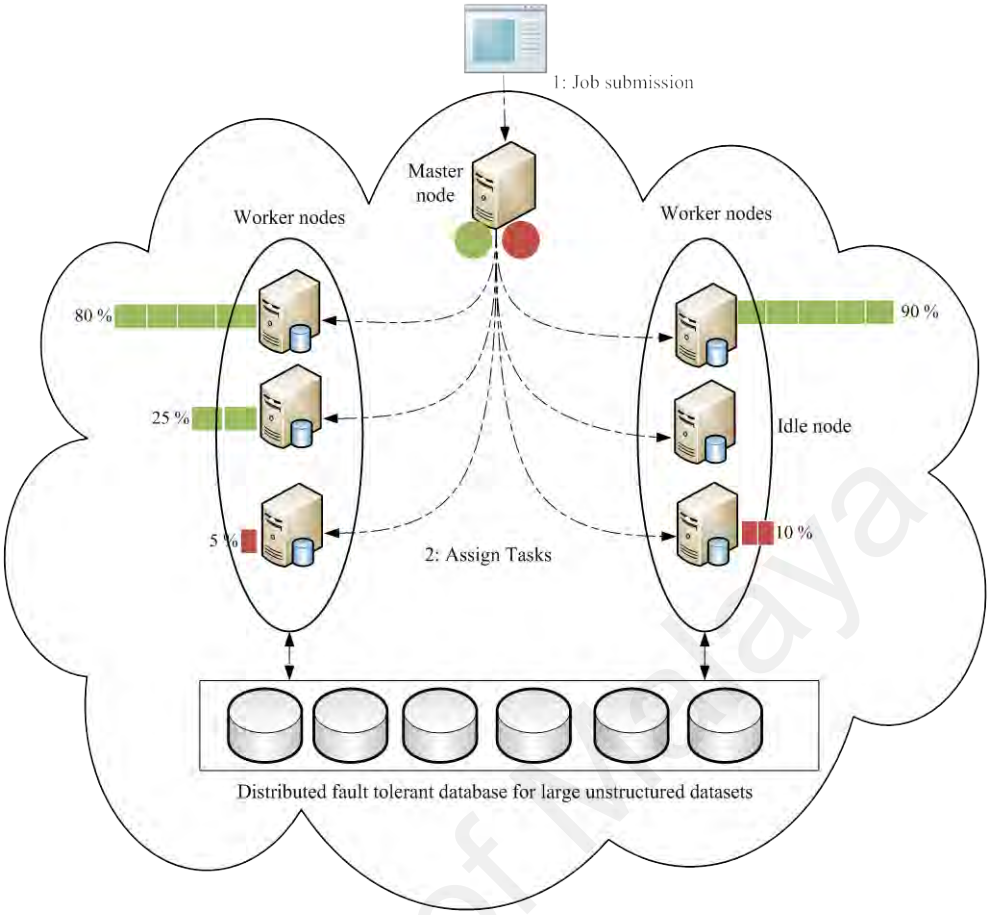



**Figure 1-1: Exemplifies the big data scheduling process using the Hadoop system**

The scenario illustrated in Figure 1-1, exemplifies the big data scheduling process using the MapReduce in a cloud computing environment. The tasks are divided across many virtual nodes in the cloud to be executed in parallel. However, it is possible that a few nodes slow down the overall execution of the tasks. The execution of the tasks may be slow due to various reasons such as software misconfiguration and hardware degradation. When the client submits the jobs to the master node, the jobs will be broken down into tasks. These tasks will be executed by the worker nodes in which the total execution is dominated by the slowest worker nodes in the cluster.

Given the limitations stated above, we believe that a multi-objective approach is more appropriate for the scheduling tasks in MapReduce.

## 1.5 Aims and objectives

The aim of the research is to propose an optimization model for scheduling in big data platform. To achieve the aim of this research, the following objectives need to be accomplished:

a) To study the research advancements of the domain of big data and identify the key issues with respect to scheduling in big data platforms.

b) To investigate the time and cost trade-off to explore the space alternative schedules using genetic algorithms.

c) To design a new optimization model based upon a multi-objective scheduling algorithm to minimize time and cost in a heterogeneous cloud environment.

d) To evaluate the performance of a proposed algorithm by validating it with current scheduling algorithms in different scenarios.

The primary objective of this thesis is to develop a multi-objective task scheduling model. The aim is to minimize two different objectives: completion time and a total budget of each node in the cloud. Two stages are identified to achieve the goal. The first, being the construction of the model. The second stage, being the application of multi-objective genetic algorithms, adopts genetic algorithms. Moreover, the trade-off solutions of Makespan and cost computed by sorting genetic algorithm II (NSGA-II) and Strength Pareto Evolutionary Algorithm 2 (SPEA2) are analyzed based on different workflow in order to find an optimal solution between the two conflicting objectives. The classic one point crossover operator is used, which is important for the creation of the children. Also, two columns are randomly selected during the mutation operator and in a ranking operation, different solutions assigned belong to many dominated fronts. The analysis of the models is carried out using genetic algorithms. It is well known that multi-objective genetic algorithms are among the most useful approaches for multi-

objective. For computational experiments, randomly non-dominated solutions for each job are generated using testbed. These non-dominated solutions generated during the experiment are copied to the Pareto archive. The above-mentioned objectives are based on the research activities carried out in this study; in next section, the structure of the study is introduced.

## 1.6 Thesis Structure

Chapter 2 aims to survey the research undertaken in the field of scheduling in big data platforms. The chapter provides knowledge of scheduling in big data to identify the key issues with respect to scheduling and requirements for scheduling in big data platforms such as data locality, SLA-based, load balancing, time, cost, and scheduling. Moreover, this chapter highlights the scheduling algorithms for big data and investigates the related scheduling algorithms. This section discusses several approaches to the scheduling problem. These approaches consider different scenarios, which take into account the application types, the execution platform, the type of algorithms used and the various constraints that might be imposed. Moreover, the chapter discusses the multi-objective optimization by focusing on genetic algorithms like NSGA-II and SPEA2.

Chapter 3 presents a review of the research conducted within the framework of MapReduce in solving problems of scheduling. Experimental results of the comparison between Hadoop physical cluster and cloud cluster is provided. The chapter also provides problem definition and introduces time with a budget constraint model and cost with deadline constraint model.

Chapter 4 presents a new proposed framework for multi-objective algorithms, which try to identify the importance of resource allocation and task scheduling in the cloud, by considering both completion time and the cost minimization models. The proposed

framework algorithm is designed based on the combination of two main models which are adaptive control and cost decision module in order to meet performance goals and maximize the efficiency of a Hadoop cluster in the cloud. Also, to establish the relationship between resource allocation and task scheduling, new scheduling algorithms are proposed. This combination of the resource allocation and task scheduling helps in achieving one of the objectives of this study. Moreover, beside the main objective to propose new scheduling algorithm, the algorithms also address the limitation of the resource allocation and task scheduling, which was identified in previous chapters.

Chapter 5 presents a proposed multi-objective scheduling algorithm; it is important to design a systematic evaluation procedure in order to provide a verification of its use. This chapter offers performance evaluation and statistical modeling based on the proposed algorithm which aims to compare it regarding the performance with other algorithms from throughput and execution time perspective. First, the chapter provides a description of big data benchmarks that used for the evaluation of the proposed algorithm. Second, the simulation environment is described in details. Then, statistical analysis that is derived for validation of the findings is described. A series of experiments to show platform-independence of our proposed solution is described and the comparative study that is designed to demonstrate the proposed algorithm regarding throughput and execution time is described. Finally, the chapter investigates the performance of the proposed algorithm by comparing with the most used algorithms: FIFO and Fair.

Chapter 6 concludes the major contributions of the thesis. It also outlines the limitations and opportunities to further improve or extend the work presented in the thesis. To this

end, this thesis stands as a substantial effort to optimize big data scheduling in the cloud from two dimensions simultaneously, including resource allocation and task scheduling.

# CHAPTER 2: OVERVIEW OF BIG DATA AND SCHEDULING

Recent trends in big data have shown that the amount of data continues to increase at an exponential rate. This trend has inspired many researchers over the past few years to explore new research direction of studies related to multiple areas in big data. However, only a few research works are available to address the issues of scheduling in big data platforms. The big data stored in a distributed fashion require processing in parallel (Philip Chen & Zhang, 2014). As a result, the new knowledge and innovation can be mined within a reasonable amount of time. Data processing has been successfully adopted in some applications (Hsu, 2014), such as data mining, data analytics, scientific computation, and search engine. Nevertheless, processing big data has been challenged by these applications because of the complexity of the data that should be processed and the scalability of the underlying scheduling methods and algorithms that support such processes (Labrinidis & Jagadish, 2012).

Scheduling plays an important role in big data, mainly in reducing the execution time and cost for processing. To understand scheduling in big data studies, this chapter presents an introduction to the rise of big data in cloud computing and scheduling in big data platforms, requirements for big data processing, scheduling algorithms, and multi-objective optimization, which are close, linked to big data processing studies. This chapter begins by giving an introduction to big data in order to establish a solid starting point to pursue the proposed study. This chapter also discusses different scheduling algorithms used in big data platforms. Moreover, in order to highlight the results of the study of the previous survey conducted, the findings are strengthened. Having presented the concept of scheduling in big data platforms, various optimization options are identified.

## 2.1 Cloud computing

Cloud computing is a fast-growing technology that has established itself in the next generation of IT industry and business. Cloud computing promises reliable software, hardware, and infrastructure as a service delivered over the Internet and remote data centers (Armbrust et al., 2010). Cloud services have become a powerful architecture to perform complex large-scale computing tasks and span a range of IT functions from storage and computation to database and application services. The need to store, process, and analyze large amounts of datasets has driven many organizations and individuals to adopt cloud computing (Huan, 2013). A large number of scientific applications for extensive experiments are currently deployed in the cloud and may continue to increase because of the lack of available computing facilities in local servers, reduced capital costs, and increasing volume of data produced and consumed by the experiments (Pandey & Nepal, 2013). In addition, cloud service providers have begun to integrate frameworks for parallel data processing in their services to help users access to cloud resources and deploy their programs (Warneke & Kao, 2009).

Cloud computing "is a model for allowing ubiquitous, convenient, and on-demand network access to many configured computing resources (e.g., networks, server, storage, application, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Peter Mell & Timothy Grance, 2011). Cloud computing has several favorable aspects to address the rapid growth of economies and technological barriers. Cloud computing provides a total cost of ownership and allows organizations to focus on the core business without worrying about issues, such as infrastructure, flexibility, and availability of resources (Giuseppe et al., 2013). Moreover, combining the cloud computing utility model and a rich set of computations, infrastructures, and storage cloud services offers a highly attractive environment where scientists can perform their experiments (Gunarathne et al., 2013).

## 2.2 The relationship between cloud and big data

Cloud computing and big data are conjoined. Big data provide users the ability to use commodity computing to process distributed queries across multiple datasets and return resultant sets in a timely manner. Cloud computing provides the underlying engine through the use of Hadoop, a class of distributed data processing platforms. The use of cloud computing in big data is shown in Figure 2-1. A large data source from the cloud and Web are stored in a distributed fault-tolerant database and processed through a programming model for large data sets with a parallel distributed algorithm in a cluster.

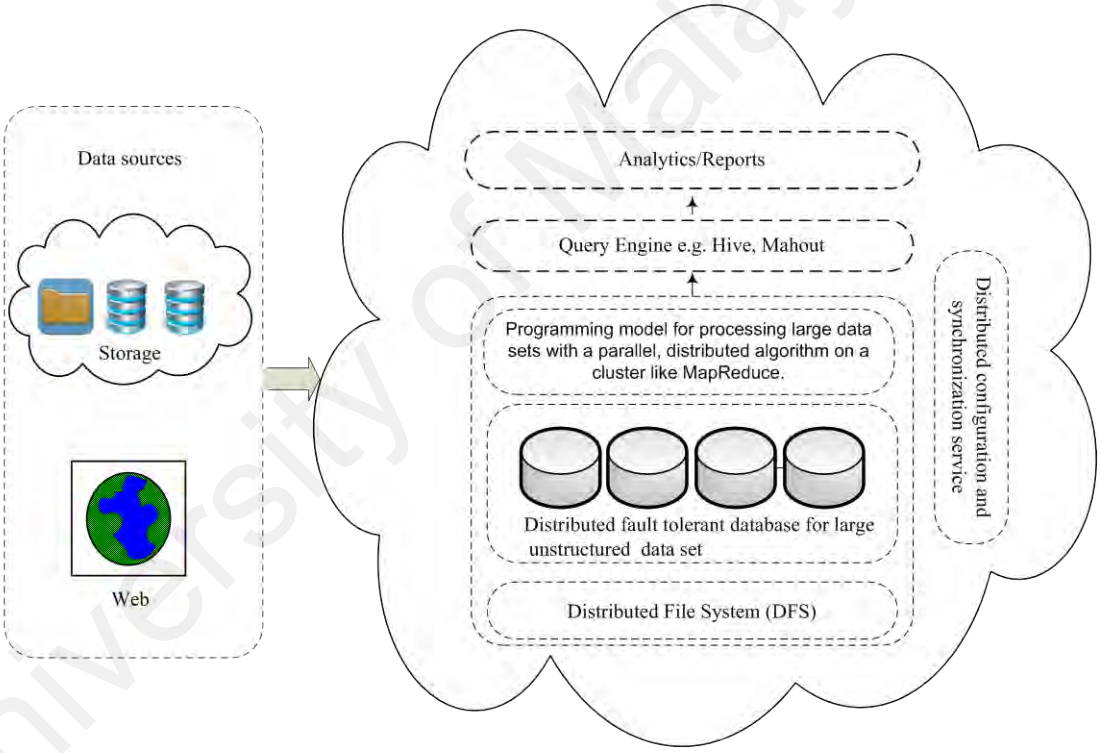



**Figure 2-1: Cloud computing usage in big data**

Furthermore, cloud computing is correlated with a new pattern for the provision of computing infrastructure and big data processing method for all types of resources available in the cloud through data analysis. Several cloud-based technologies have coped with this new environment, however, dealing with big data for concurrent processing has become increasingly complicated (Ji et al., 2012). MapReduce (Dean &

Ghemawat, 2008) is a good example of big data processing in a cloud environment; it allows for the processing of large amounts of data sets stored in parallel in the cluster. Cluster computing exhibits good performance in distributed system environments, such as computer power, storage, and network communications. Likewise, (Bollier & Firestone, 2010) emphasized the ability of cluster computing to provide a hospitable context for data growth. Tables 2-1 show the comparison of several big data cloud providers.

**Table 2-1: Comparison of several big data cloud platforms**

| | Google | Microsoft | Amazon | Cloudera |
|---|---|---|---|---|
| Big data storage | Google cloud services | Azure | S3 | |
| MapReduce | AppEngine | Hadoop on Azure | Elastic MapReduce (Hadoop) | MapReduce YARN |
| Big data analytics | BigQuery | Hadoop on Azure | Elastic MapReduce (Hadoop) | Elastic MapReduce (Hadoop |
| Relational database | Cloud SQL | SQL Azure | MySQL or Oracle | MySQL, Oracle, PostgreSQL |
| NoSQL database | AppEngine Datastore | Table storage | DynamoDB | Apache Accumulo |
| Streaming processing | Search API | Streaminsight | Nothing prepackaged | Apache Spark |
| Machine Learning | Prediction API | Hadoop + Mahout | Hadoop + Mahout | Hadoop + Oryx |
| Data import | Network | Network | Network | Network |
| Data Sources | A few sample datasets | Windows Azure marketplace | Public Datasets | Public Datasets |
| Availability | Some services in private beta | Some services in private beta | Public production | Industries |

## 2.3 Big data scheduling challenge

Although cloud computing has been broadly accepted by many organizations, research on big data in the cloud remains in its early stages. A few attempts have been made to address the issues of big data. Moreover, new challenges continue to emerge from applications by the organisations. In the subsequent sections, some of the key research challenges related to scheduling are highlighted.

The continuous increase in computational capacity over the past years has produced an overwhelming flow of data or big data, which exceeds the capabilities of conventional processing tools. Big data signify a new era in data exploration and utilization. The MapReduce computational paradigm is a major enabler for underlying numerous big data platforms. MapReduce is a popular tool for the distributed and scalable processing of big data. It is increasingly being used in different applications primarily because of its important features, including scalability, fault tolerance, ease of programming, and flexibility. The impact on data processing and analytics is the need to re-think the approaches and solutions to better performance. In this context, scheduling models and algorithms have an important role, which based on a large variety of solutions for specific applications. Scheduling in big data offers an important contribution to the big data optimization, particularly in reducing execution time during the processing by schedule resources and tasks to minimize job completion times (Pop & Cristea, 2015). Scheduling has been widely explored from various aspects by many researchers in recent years. However, most former research work mainly considers optimized designed of algorithms and frameworks under a relatively a homogeneous environment. Moreover, numerous mechanisms are used for resource allocation in the cloud, which is heterogeneous and widely distributed (Pop & Cristea, 2015). Although many scheduling methods are used in big data processing frameworks; however, find the best method for a particular processing request remains a significant challenge.

Based on our discussion above on the big data challenge, we summarize that there are several key research challenges related to big data such as scalability, availability, quality, heterogeneity, privacy, governance, and scheduling.

Before discussing scheduling big data platforms and its algorithms, first, the brief MapReduce and its implementation systems are provided to understand the state-of-the-art framework. The following section provides an overview of Hadoop MapReduce.

## 2.4 Apache Hadoop

Hadoop (Hadoop, 2011) (Shvachko et al., 2010) is an open-source Apache Software Foundation (ASF) project written in Java that provides cost-effective, scalable infrastructure for the distributed processing of large datasets across clusters of the commodity.

Hadoop was "inspired" by Google File System GFS (Ghemawat et al., 2003) and Google's MapReduce distributed computing environment. Initially, started as a distributed Nutch search engine project and named by developer Doug Cutting after his son's toy elephant (White, 2009). It has been successfully used for processing highly distributable problems across a large amount datasets using commonly available servers in a very large cluster, where each server has a set of inexpensive internal disk drives. The current Hadoop project consists of two main modules, that is, the distributed file system called Hadoop Distributed File System (HDFS) and MapReduce engine.

### 2.4.1 HDFS

HDFS is an open source version of the Google's GFS designed to run on commodity hardware. Like other Hadoop-related technologies, HDFS have become a key tool for managing pools of big data and supporting big data analytics applications and also is a highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS is a Master/Slaver architecture illustrated in Figure 2-2, consisting of NameNode called master, Secondary NameNode called checkpoint, and several DataNode called slaves. NameNode is the controller that handles all files system operations; hence, any request that comes to the file system such as create, delete, and read a file goes through it. The

meta-data are present in the NameNode, which registers attributes such as access times, modification, permission, and disk space quotas. NameNode also handles block mappings. In particular, the file is divided into blocks (Default 64MB), in which each block is independently replicated across DataNode for redundancy and periodically sends a report of all existing blocks to the NameNode. DataNode manages the block creation, deletion, and replication upon the instruction from the NameNode. The cluster may incorporate thousands of DataNode and tens of thousands of HDFS clients per cluster given that each DataNode may concurrently execute multiple application tasks.

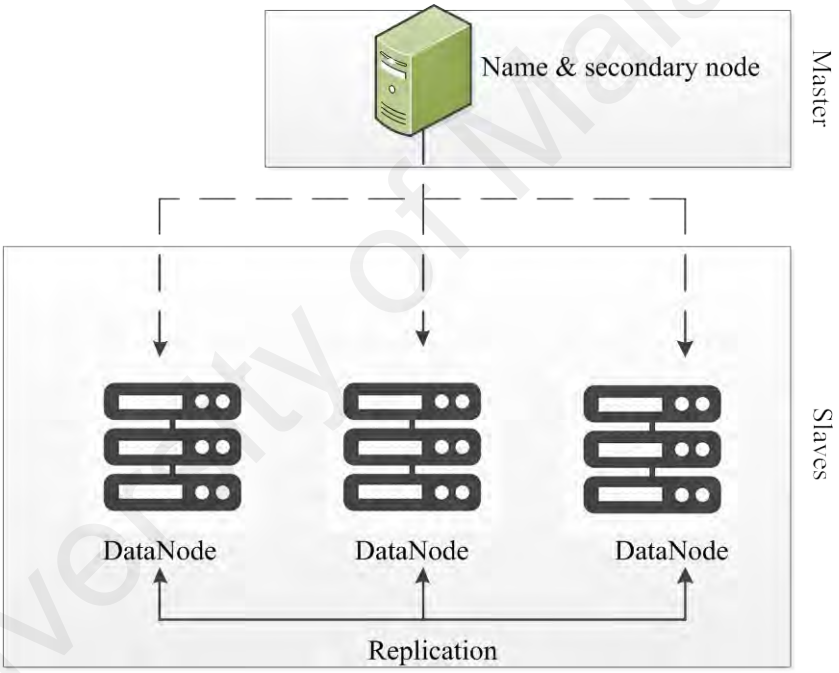



**Figure 2-2: HDFS Architecture**

Moreover, DataNode sends a Heartbeat message to the NameNode periodically, the default heartbeats interval is three seconds. If the NameNode is unable to detect heartbeat message due to loss of connectivity between the NameNode and DataNode. The NameNode consider being out of service and the block replicas hosted by that DataNode to be unavailable or dead and does not forward any new requests to that particular DataNode. The NameNode then schedules the creation of new replicas of

those blocks on other DataNode. Meanwhile, the job of the Secondary NameNode is not to be secondary to the NameNode, but only to periodically read the file system changes log and provides backup for the former, thereby updating it. In the larger cluster environment, Secondary NameNode usually runs on a different machine than the primary NameNode since its memory requirements are on the same order as the primary NameNode. This task enables NameNode to start up faster next time.

### 2.4.2 Hadoop MapReduce

MapReduce engine comprises several components as shown in Figure 2-3. In particular, the main component is job client, which submits the job to the clusters. Job tracker oversees the task tracker and provides execution plans, coordinates the jobs, and schedules them across the task tracker. Meanwhile, task tracker breaks down the jobs into Map and Reduce tasks. Each task record has slots for execution map, gradually reduces, and reports the progress.

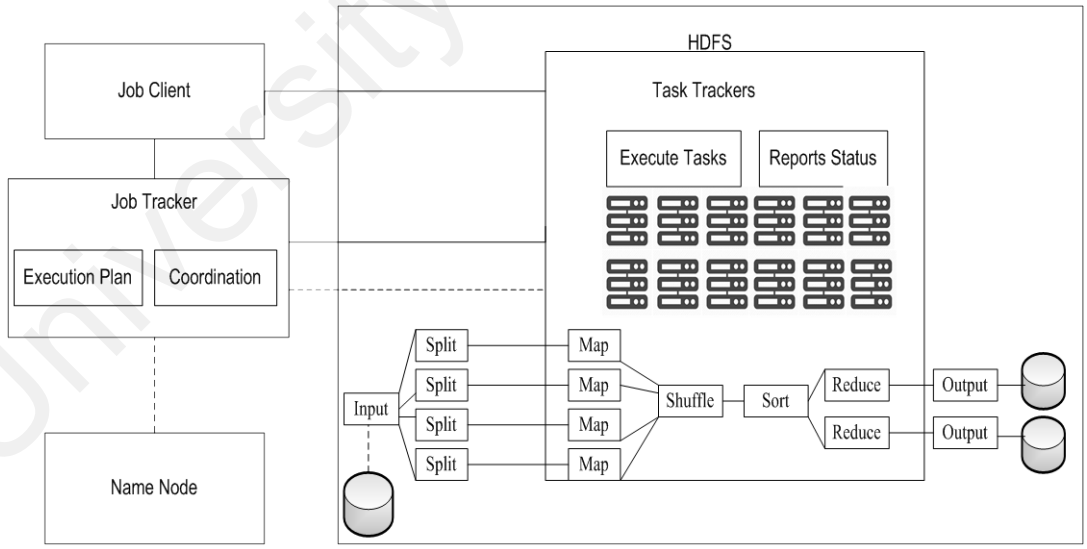



**Figure 2-3: MapReduce architecture**

Then the input data are divided into input splits based on the input format. Input splits equate to a map task, which runs in parallel. Input format determines how the files are

parsed into the MapReduce pipeline. Map transforms the input splits into intermediate key/value pairs based on the user-defined code. Shuffle and sort: moves intermediate key/value pairs outputs to the reducers and sorts them by the key. The reducer merges all pairs associated with the same (intermediate) key and then generates an output based on the user-defined code.

Job scheduling in Hadoop is performed by a master node, which manages many worker nodes in the cluster. Each worker has a fixed number of map slots and reduces slots, which can run the map and reduce tasks respectively. The number of maps and reduce slots is statically configured. Based on the availability of free slots and the scheduling policy, the master assigns map and reduce tasks to slots in the cluster (Z. Zhang et al., 2013).

## 2.5 Scheduling in big data platforms

A few attempts have been made in the recent past to study scheduling in big data platforms. The report by (Tiwari et al., 2015) provides a comprehensive and structured survey of scheduling algorithms used for big data platforms. The study proposed a multidimensional classification framework based on quality requirements, scheduling entities and the adaptation of dynamic environments. Moreover, there has been a lot of work published related to scheduling big data in MapReduce, which all aim to improve the performance of big data platforms (Dean & Ghemawat, 2008, 2010; Doulkeridis & Nørvåg, 2014; J.-J. Li et al., 2011; Sakr et al., 2013). However, there are still areas to explore, particularly with respect to scheduling and resources management in cloud heterogeneous environments. Existing studies have so far focused on reducing the execution time, overhead, resource consumption and data locality (Z. Guo, G. Fox, M. Zhou, et al., 2012; Zaharia et al., 2008; X. Zhang et al., 2011). The studies on scheduling in big data have been classified into three: data storage, processing engine, and resource manager.

*(i) Data storage:* The rapid growth of data has restricted the capability of existing storage technologies to store and manage data. Over the past few years, traditional storage systems have been utilized to store data through structured RDBMS (M. Chen et al., 2014a). However, most storage systems have limitations and are inapplicable to the storage and management of big data. A storage architecture that can be accessed in a highly efficient manner while achieving availability and reliability is required to store and manage large datasets.

Several storage technologies have been developed to meet the demands of massive data. Existing technologies can be classified as direct attached storage (DAS), network attached storage (NAS), and storage area network (SAN). In DAS, various hard disk drives (HDDs) are directly connected to the servers. Each HDD receives a certain amount of input/output (I/O) resource, which is managed by individual applications. Therefore, DAS is suitable only for servers that are interconnected on a small scale. Given the aforesaid low scalability, storage capacity is increased but expandability and upgradeability are limited significantly. NAS is a storage device that supports a network. NAS is connected directly to a network through a switch or hub via TCP/IP protocols. In NAS, data are transferred as files. Given that the NAS server can indirectly access a storage device through networks, the I/O burden on a NAS server is significantly lighter than that on a DAS server. NAS can orient networks, particularly scalable and bandwidth-intensive networks. Such networks include high-speed networks of optical-fiber connections. The SAN system of data storage is independent with respect to storage on the local area network (LAN). Multipath data switching is conducted among internal nodes to maximize data management and sharing. The organizational systems of data storages (DAS, NAS, and SAN) can be divided into three parts: (i) disc array, where the foundation of a storage system provides the fundamental guarantee, (ii) connection and network subsystems, which connect one or more disc

arrays and servers, and (iii) storage management software, which oversees data sharing, storage management, and disaster recovery tasks for multiple servers.

*(ii) Processing engine:* MapReduce accelerates the processing of large amounts of data in a cloud; thus, MapReduce, is the preferred computation model of cloud providers (Zhifeng & Yang, 2013). MapReduce is a popular cloud computing framework that robotically performs scalable distributed applications (Srirama et al., 2012) and provides an interface that allows for parallelization and distributed computing in a cluster of servers (O'Leary, 2013). (Srirama et al., 2012) presented an approach to apply scientific computing problems to the MapReduce framework where scientists can efficiently utilize existing resources in the cloud to solve computationally large-scale scientific data. Currently, many alternative solutions are available to deploy MapReduce in cloud environments; these solutions include using cloud MapReduce runtimes that maximize cloud infrastructure services, using MapReduce as a service, or setting up one's own MapReduce cluster in cloud instances (Gunarathne et al., 2010). Several strategies have been proposed to improve the performance of big data processing. Moreover, the effort has been exerted to develop SQL interfaces in the MapReduce framework to assist programmers who prefer to use SQL as a high-level language to express their task while leaving all of the execution optimization details to the backend engine (Sakr et al., 2013). Table 10 shows a summary of several SQL interfaces in the MapReduce framework available in existing literature.

*(iii) Resource manager:* Resource manager allows data and computational resources to be shared and accessed by the nodes in the cluster (Vavilapalli et al., 2013). There exist some important resource management studies related to big data scheduling as below:

*(a) Adaptive resource management*: The resource manager considers data, physical resources, and workload while taking scheduling decisions (Tiwari et al., 2015), this

include fair share (Zaharia, 2009), capacity (Zaharia, 2009) , delay (Tan et al., 2012; Zaharia et al., 2010), and resource-aware (Polo et al., 2011; Sousa et al.), which all aim to reduce the completion time and resource consumption in distributed environment.

*(b) Non-adaptive resource management*: The resource manager assigns jobs/tasks a fixed number of resources at runtime. For example, FIFO (Hadoop), is the default Hadoop scheduler and the most popular algorithm in the non-adaptive scheduling of Hadoop. Possibly the most straightforward approach to schedule task is to maintain a FIFO run queue based on policies or the solution of some optimization problems.

The requirements of scheduling in big data platforms require a cluster resource management, scheduling, and execution engines, which based on new information processing frameworks that include applications and the storage resources. This requirement encompasses the need to share resources and decide when tasks run. The general requirements for scheduling in big data can be described are as follows:

*(a) Data locality.* Data locality is defined as how close the compute and input data are, and it has different levels – node-level, rack-level, etc. Data locality is one of the most important factors considered by schedulers in data parallel systems (Herodotou et al., 2011). Hadoop MapReduce determines whether cluster/rack is being scheduled based on the availability of the data locally. It assumes that nodes within the same rack have higher bandwidth than those that are not resident in the same cluster. Knowing this, the scheduler can simply increase data locality for tasks. Good data locality reduces cross-switch network traffic - one of the bottlenecks in data-intensive computing (Z. Guo, G. Fox, & M. Zhou, 2012). Some scheduling policies in Hadoop consider the effect of data locality, which can be classified as cluster and rack (Park et al., 2012) (J.-J. Li et al., 2011). For instance, to assign map tasks to a node, the Hadoop default FIFO chooses the job from the queue and schedule its local map tasks. When the job does not have any

map task locally, it will be assigned to the non-local map in the cluster (He et al., 2011). Furthermore, (Abad et al., 2011) proposed a distributed adaptive data replication algorithm that helps the scheduler to achieve better data locality. The advantages of this approach are to allow many replicas to be allocated for each file and make use of probabilistic sampling and a competitive aging algorithm independently at each node. (X. Zhang et al., 2011) emphasizes on data locality problem of MapReduce. The author introduces a next-k-node scheduling method, which has implemented in Hadoop-0.20.2. Also, (J. Jin et al., 2011) suggest that initial task allocation is produced first before the job completion time can be reduced gradually. The author introduces a heuristic task scheduling algorithm called BAlance-Reduce (BAR), which adjust data locality dynamically according to network state and cluster workload by tuning the initial task allocation using a global view. The experimental result of the algorithm shows that BAR is able to outperform previous related algorithms in terms of the job completion time and deal with large problem instances in a few seconds.

*(b) SLA-based.* Scheduling Hadoop tasks in virtual machines in the cloud demand resources of the cloud, typically, users are aware of the deadline of when the job is completed (Wu et al., 2011). However, in a cloud computing environment, all machines compete for resources to execute the jobs (Buyya et al., 2009). These resources are controlled by batch queue systems, which may not offer guarantee deadline during the task execution, only if the priority used for resource reservation which is a restricted level of service.

*(c) Load balancing.* Load balancing is one of the scheduling methods, which provides the cluster with the share of the load of the resources between all the machines participate in the cluster and obtain the best performance for distributed task scheduling systems (Fang et al., 2010; K. Wang et al., 2014). However, some resources may not

match with tasks properties which are a challenging requirement need to be considered. Many approaches has been developed to tackle this issue including round-robin scheduling (Shreedhar & Varghese, 1996; K. Wang et al., 2014), however, new approaches that cope with large scale and heterogeneous systems is proposed such as slow start time (Khiyaita et al., 2012), agent-based adaptive balancing (Q. Long et al., 2011) and least connections (McGuire, 2006).

*(d) Time.* In a heterogeneous environment, cluster usually contains nodes with different computing capacity in which some of the nodes may poorly perform a task execution; this can create some challenges to the overall execution time. Hadoop assumes that all the machines are homogeneous, but as a matter of fact, in most cases, machines are not homogeneous, especially in a cloud computing environment where the hardware could be in different generations and virtualized data center as the uncontrollable several of virtualized resources. The scheduler should improve the performance of scheduled jobs as much as possible using different heuristics and state estimation suitable for specific task models. Multitasking systems can process multiple datasets for multiple users at the same time by mapping the tasks to resources in a way that optimizes their use.

*(e) Cost.* The cost implemented in the cloud is pay-as-you-go model, where the services are charged as per the QoS requirements of the users. The resources in the cloud, such as network bandwidth and storage are charged at a specific rate (Hussain et al., 2013). Thus, the cost has become an important objective in big data cloud scheduling research. The total cost incurred by processing big data can comprise many cost components such as compute cost and data transfer cost. Cloud computing providers lease computing resources that are typically charged based on a per time quantum pricing scheme. Hence, the scheduler should lower the total cost of execution by minimizing the total number of resources used and respect the total money budget. This aspect requires

efficient resource usage and can be done by optimizing the execution of mixed tasks using the high-performance queuing system and by reducing the computation and communication overhead. Table 2-2 shows the comparison of scheduling algorithms that are based on the single objective optimization approach.

**Table 2-2 A summary of scheduling algorithms based on a single objective optimization approach**

| Algorithm | Requirements | Constraints |
|---|---|---|
| FIFO (Hadoop, 2011) | Execution time | Deadline<br>Data dependency |
| Fair (Hadoop, 2009) | Response time, Data locality, and Cost | Data dependency |
| Capacity (Hadoop) | Response time, Data locality, and cost | Data dependency |
| Delay (Zaharia et al., 2010) | Locality and<br>Fairness | Fairness, Resource usage |
| LATE (Zaharia et al., 2008) | Response time, SLA-based | The cost to run speculative task scheduling is expensive |
| Quincy (Isard et al., 2009) | Min-cost flow | Resource usage |
| MCP (Q. Chen et al., 2013) | Response time, SLA-based | The cost of running speculative task scheduling is expensive |
| SHadoop (Gu et al., 2014) | Response time, data locality | Throughput Performance |
| MRA++ (Anjos et al., 2015) | Perform data intensive computing in heterogeneous environments | Delay |
| Maestro (Ibrahim et al., 2012) | Improving the locality execution of map task efficiency, SLA-based and Cost | - |
| ARIA (Verma et al., 2011) | Completion time | SLO |

## 2.6 Scheduling algorithms for big data

Several approaches to the scheduling problem have been considered over time. These approaches consider different scenarios, which take into account the application types, the execution platform, the type of algorithms used and the various constraints that might be imposed. The existing schedules suitable for large environments and also big data platforms are as follows:

*(a) Fair scheduler*. A fair scheduler is a method of assigning equal resources for jobs in a Hadoop cluster by helping small tasks to run in parallel with other tasks that requires more CPU time. Fair scheduler developed by Facebook and subsequently released to the Apache Hadoop community. Unlike the default Hadoop scheduler, which forms a queue of jobs, fair scheduler allows an equal share of the cluster capacity between pools over time. With each pool allocated a guaranteed minimum number of Map and Reduce slots. When a new job is submitted, idle resources in the pool are assigned to that new job, so that each job ultimately gets approximately the same amount of resources. It uses priorities applied as weights to manage the fraction of total resources for each job assigned over time. Moreover, fair scheduler support preemption where jobs are dismissed from pools for a certain period of time. For example, all the tasks that are belonging to a particular job will be allocated to slots for computation. Subsequently, the scheduler checks the time deficit against the ideal fair allocation for that job (Rao & Reddy, 2012). As soon as the slots freed for the next scheduling of the task with the highest time deficit, it will be assigned to the available slot. Gradually, such process will have an effect of guaranteeing that the jobs get roughly equal amounts of resources. The jobs with a small number of tasks allocate sufficient resources in order to finish the tasks very fast. However, the problem with such schedule is the resource congestion when dealing with a lot of numbers of tasks.

*(b) Capacity scheduling*. Capacity scheduling is originally developed at yahoo to run Hadoop applications as a shared, where the fair allocation of computational resource is critical to the user. The main idea is that Hadoop Map-Reduce cluster is partitioned using the available resources between users who collectively use the cluster based on computing needs. Besides, the capacity scheduler is similar to the fair scheduler; it uses a queue to allocate jobs to users, but with the major difference of using prioritized queue jobs. Each queue is given a configured capacity, which contents a scheduling that

operates on a modified priority queue basis with specific user limits. The queue with the least number of jobs is selected if the slot becomes available in order to schedule the task for that job. Generally, this may have an impact on the cluster capacity sharing among users than among jobs, as was the case in the Fair Scheduler.

*(c) Delay scheduling.* Delay scheduling (Zaharia et al., 2010) is used to improve data locality in MapReduce, in a situation where there is a conflict between fairness in scheduling and data locality. The idea is that if the next jobs can be scheduled according to fairness, then that job cannot launch local tasks. Such solution is suitable if the jobs to be scheduled are less and the task is not been scheduled locally. Thus, delaying its scheduling time can significantly improve data locality. The result shows that delay scheduling can achieve nearly optimal data locality and its outperformance fair sharing by its simplicity under a wide variety of scheduling policies. Also, (Tan et al., 2012) suggest an analytically tractable model under different schedulers such as default FIFO Scheduler and the popular Fair Scheduler for job processing delay distribution in MapReduce.

*(d) Maestro.* (Ibrahim et al., 2012) proposed scheduling algorithm named Maestro, which, is designed to improve the performance of the MapReduce computation. The current Hadoop schedulers perform inefficient scheduling of map tasks by degrading the replicas distribution. Maestro scheduler has two objectives; first, each data node is equipped with the empty slots based on the replication scheme for their input data and the number of hosted map tasks. Secondly, it considers the runtime of each scheduling tasks and the replicas of the task's input data determine the scheduling of the map task on a particular node. With these objectives, the scheduler can achieve a higher locality of the map tasks and also balanced intermediate data distribution for the shuffling

phase. The results of presenting Maestro algorithm are very promising compared to the current Hadoop scheduler.

*(e) SHadoop* (Gu et al., 2014) is a scheduling method used for improving the performance of Hadoop MapReduce by optimizing the job and task execution mechanism. SHadoop incorporates the following main optimization approaches: (1) Optimizing the job initialization and termination stages, thereby shortening the startup and cleanup time of all jobs; (2) providing an instant messaging communication mechanism for efficient critical event notification that can benefit the majority of the short jobs with large deployment or many tasks. However, such optimization may induce a little more burden to the JobTracker because it needs to create and delete an empty temporary directory for each job. Compared with the standard Hadoop, SHadoop can averagely achieve 25 percent of performance improvement for various tested Hadoop benchmark jobs and Hive applications.

*(f) Quincy* (Isard et al., 2009) addresses the problem of scheduling jobs on distributed computing clusters that are close to application data stored on the computing node. Each job in the node is managed by a root task that is assigned by the scheduler in the cluster. Such node is responsible for submitting a list of the worker to the schedule in which these works have no dependency relationship. For each worker, the root is calculated based on the preference list of computers and racks that have a high rate of data in the computer in the rack of computers. Quincy originally designed for DryadLing, but it can be applied to other systems such as MapReduce. The system implemented the concept of the queue based on the hierarchical nature of the cloud networks to allow data to be executed locally and close to computation. Queues exist for machines, racks, and the system. This system works well when data locality is even and job lengths are approximately equal.

*(g) ARIA* (Verma et al., 2011) is a framework for addressing the problem of job scheduler in MapReduce environment. This framework consists of three components, which are as follows: (i) Job profile of continuing executing jobs; (ii) MapReduce for a particular job; and (iii) time to execute the tasks based on the amount of the resource estimated for a specific job with a time frame. Moreover, a novel SLO-based schedule has been proposed by the authors in order to select job ordering and the amount of resources to be allocated for meeting the job deadline. Resource allocations can be increased by expediting job completion and can be realized during the Map and Reduce stages.

*(h) MRA++* (Anjos et al., 2015) has been presented as a scheduling and data placement framework design on MapReduce that is suitable for the heterogeneous environment. MRA++ adapts the amount of data processed during the Map phase to the distributed computing capability of the computers. Based on the sum granularity factors, the reduced data is partitioned into smaller sizes. This signifies that the fastest computers process more data than the slower ones (i.e. the efficiency of the computer depends on its configurations). The idea behind this framework is to efficiently perform intensive data computation in a heterogeneous environment. The system consists of two main features, including task scheduling job control and the heterogeneity of nodes during data distribution. Moreover, the implementation proposed in MRA++ is concerned about whether the workload can be adapted to the computing capability of each machine.

*(i) Throughput Scheduler:* (Gupta et al., 2013) proposed throughput scheduler which reduces the overall job completion time on clusters of heterogeneous nodes. The idea is, the scheduler uses optimal matching job requirements to actively schedule the tasks based on the node capabilities. Node capabilities are learned by running probe jobs on

the cluster. The scheduler utilizes two methods: a Bayesian and active learning scheme to learn the resource requirements of job on-the-fly. The result showed that throughput scheduler can reduce average mapping time by 33% compared to other Hadoop schedulers.

According to the scheduling strategy discussion on the current research improvement, we summarize the implementation approach and limitations for MapReduce scheduling algorithms in Table 2-3. For each work, we present the objective of the algorithm, the implementation approach and the limitation of each algorithm. We observe that most of the algorithms are a focus on improving the performance of jobs or tasks scheduling. On the basis of our analysis, we anticipate, gradually the challenges can be reduced to achieve accurate scheduling algorithm by exploring the underlying research progress on scheduling.

**Table 2-3: Summary of MapReduce scheduling algorithms**

| **Algorithm** | **Objectives** | **Implementation approach** | **Limitation** |
|---|---|---|---|
| FCFS | To minimize the completion time | • Run a queue<br>• Compare each task's progress to the average progress. | • Designed only for the single type of a job.<br>• Low performance when running multiple jobs.<br>• Poor response times for short jobs compared to large jobs. |
| LATE | To improve response time for short jobs. | Progress rate is calculated by dividing the progress score | • Only takes action on appropriate slow tasks.<br>• 2. However, it does |

| | | | |
|---|---|---|---|
| | | | not compute the remaining time for tasks correctly, and cannot find real slow tasks in the end.<br>• Poor performance due to the static manner in computing the progress of the tasks |
| Quincy | Reduce the scheduling problem to the min-cost flow problem | Fairness and data locality | • Resource sharing<br>• Predictability and consequently fairness |
| MCP | Improves the effectiveness of speculative execution | Both the progress rate and the bandwidth are used within a phase to select slow tasks. Exponentially weighted moving average is used to predict process speed and calculate a task's remaining time, and Determine which the task to backup based on the load on a cluster using a cost-benefit model | Poor performance due to the static manner in computing the progress of the tasks |
| SHadoop | Improving the performance of job and task | • Optimize the setup and cleanup tasks<br>• Messaging | • Hadoop cluster can only be statically configured. |

| | | | |
|---|---|---|---|
| | optimizing the job and task execution. | communication mechanism | • It does not work well in heterogeneous |
| MRA++ | Develop algorithms allow the use of data-intensive applications in large-scale environments with the use of the internet | • Grouping,<br>• Data distribution<br>• Task scheduling | • Low performance when running multiple types of jobs. |
| Maestro | Improve the overall performance of MapReduce computation | Using a replica-aware execution of map tasks | Maestro algorithm can only be statically configured. |
| ARIA | To provides scheduler mechanism for job completion deadline. | Uses job profiles and a soft deadline that determines job ordering and many resources to allocate for meeting the job deadlines. | It does not consider node failures. |
| Flex | To optimize any of the variety of standard scheduling theory metrics | Flexible scheduling allocation scheme. | Schemes are very metric dependent |

## 2.7 Resource scheduling frameworks

During the last decade, a lot of resource allocation and job scheduling frameworks have emerged and also become popular, including Yarn, Mesos, and Corona. Table 2-4

shows a comparison of MapReduce default, Yarn, Mesos and Corona scheduling frameworks.

**Table 2-4: A comparison of resources scheduling frameworks**

| **Features** | **MapReduce** | **YARN** | **Mesos** | **Corona** |
|---|---|---|---|---|
| Resources | Request-based | Request-based | Offer-based | Push-based |
| Scheduling | Memory | Memory | Memory/CPU | Memory/CPU/Disk |
| Cluster utilization | Low | High | High | High |
| Fairness | No | Yes | Yes | Yes |
| Job latency | High | Low | Low | Low |
| Scalability | Medium | High | High | High |
| Computation model | Job/task-based | Cluster-based | Cluster-based | Slot-based |
| Language | Java | Java | C++ | - |
| Platform | Apache Hadoop | Apache Hadoop | Cross-platform | Cross-platform |
| Open Source | Yes | Yes | Yes | Yes |
| Developer | ASF | ASF | ASF | Facebook |

*(a) YARN* (Vavilapalli et al., 2013) is a resource manager that represents a generational shift in the architecture of Apache Hadoop. It utilizes the MapReduce programming framework by default to perform efficient data processing by separating the processing engine and resource management capabilities of MapReduce. Hence, it makes the Hadoop environment highly suitable for operational applications that cannot wait for batch jobs to be completed. This feature simplifies the support of maintaining a multi-tenant environment, managing and monitoring workloads, implementing security controls, cluster utilization and providing high- scalability for Hadoop framework. Moreover, YARN enables programmers to design and implement distributed frameworks while sharing a common set of resources and data (Murthy et al., 2013). A resource manager, the central entity of YARN, employs a node manager to launch containers that could either map or reduce tasks and monitor the operations of individual cluster nodes. When a job request comes into the YARN resource manager, YARN evaluates all the resources available, and it places the job.

*(c) Apache Mesos* (Hindman et al., 2011) is an open source cluster manager, originally developed at the University of California at Berkeley to offer effective heterogeneous resources isolation and allocation across distributed applications. Apache Mesos is being used by many companies such as Twitter, MediaCrossing, Xogito, Airbnb and Apple. It offers an abstraction of computing resources (CPU, storage, network, memory, and file system) from nodes aiming to deploy and manage applications in large-scale clustered environments more efficiently. Mesos adapting features of the modern kernel—"cgroups" in Linux, "zones" in Solaris and reside between the operating system layer and the applications layer to provide isolation for CPU, memory, I/O, file system, rack locality, which allow applications like Hadoop, Spark, Kafka to run dynamically across shared pool of nodes. Mesos determines which resources are available, and it makes offers back to an application scheduler (Scott, 2015). Those offers can be accepted or rejected by the framework. The Hadoop framework uses Mesos-aware schedule in order to register against the Mesos master. The Mesos master sends offers with available resources to the registered Hadoop. When MapReduce requires launching tasks, it replies to the master that it is taking part of the offers to launch the tasks. Hadoop often uses a pre-defined set of resources to task for one of its specific tasks

*(d) Corona:* is an extension of the MapReduce framework; it provides high scalability and cluster utilization for small tasks. This extension was designed to overcome some of the important Facebook challenges, such as scalability, low latency for small jobs, and processing needs (Facebook, 2012). Facebook has rewritten its scheduling framework in Corona, which is based on a task resource requirement rather than a count of the map and reduces tasks. Cluster manager is also introduced in Corona to monitor nodes in the cluster and report their available resources. For each job, a dedicated job tracker is initialized in either small or large job with a separate process.

A list of existing resource scheduling framework and MapReduce scheduling algorithms that have been modified into on Hadoop framework is provided in Table 2-5 below:

**Table 2-5: Modifications induced by existing scheduling approach to MapReduce**

| System | Modification in Hadoop systems |
|---|---|
| Yarn (Vavilapalli et al., 2013) | Yes, build on top of Hadoop to provide resource management |
| Mesos (Hindman et al., 2011) | No, runs on every machine and provides applications Hadoop with API's for resource management and scheduling |
| Corona (Facebook, 2012) | No, separates cluster resource management from job coordination |
| FIFO (Hadoop, 2011) | Yes, integrated into Hadoop |
| Fair scheduler (Hadoop, 2009) | Yes, integrated into Hadoop |
| Capacity scheduler (Hadoop) | Yes, integrated into Hadoop |
| Late (Zaharia et al., 2008) | No, add-on algorithm uses estimated finish times to speculatively execute the tasks that hurt the response time the most |
| Quincy (Isard et al., 2009) | No, introduce a powerful and flexible new framework for scheduling concurrent distributed jobs with fine-grain resource sharing |
| MCP (Qi et al., 2014) | No, add-on algorithm modified to choose proper worker nodes for backup tasks |
| SHadoop (Gu et al., 2014) | Yes, integrated into the Intel Distributed Hadoop (IDH) |
| MRA++ (Anjos et al., 2015) | No, a new MapReduce framework design that considers the heterogeneity of nodes during data distribution, task scheduling, and job control |
| Maestro (Ibrahim et al., 2012) | No, extension based on replica-aware Map Scheduling |
| ARIA (Verma et al., 2011) | No, implemented on top of Hadoop to determine job ordering and several resources to allocate for meeting the job deadlines |
| Flex (Wolf et al., 2010) | No, add-on module |
| MTSD (Tang et al., 2012) | No, extensional MapReduce Task Scheduling algorithm for Deadline constraints in Hadoop platform |
| COSHH (Rasooli & Down, 2014) | No, add-on algorithm considers heterogeneity |
| MOMTH (Nita et al., 2015) | No, a tool integrated into Hadoop |
| MORM (S.-Q. Long et al., 2014) | Different system |

## 2.8 Related Scheduling algorithms

Essentially the scheduling optimization problems in MapReduce are similar to query optimization in RDBMS in two aspects, first, its research space is vast, because of different optimization opportunities that cloud computing offers, and second the dimension of the optimization is content more than one criterion with the monetary cost of using the public cloud being at least as important as completion time (L. Guo et al., 2012).

The cloud computing paradigm is different from clusters, in term of the cost model (Armbrust et al., 2010; Peter Mell & Tim Grance, 2011). The cluster represents a fixed capital investment made up-front and relatively small operational cost paid over time (Eisen et al., 1998). On the other hand, clouds are characterized by elasticity and offer their users the ability to lease resources only for as long as needed, based on per quantum pricing scheme.

The work on scheduling data processing in distributed computing systems has been studied in the current literature. Many taxonomies exist that can present the grounds for this study (Casavant & Kuhl, 1988; Rimal et al., 2009; Su et al., 2013; Venugopal et al., 2006; Yu & Buyya, 2005), especially, the studies on which multi-objective functions are involved (Tsai et al., 2013; X. Wang et al., 2014; F. Zhang et al., 2014). The most relevant studies related to this study are concerned the time and cost (Bittencourt & Madeira, 2011; S.-Q. Long et al., 2014; Nita et al., 2015; Rasooli & Down, 2014).

Rasooli and Down (2014) propose a job scheduler to improve the performance of Hadoop job completion time in a heterogeneous cluster environment which consider both the application and cluster level of the system. The schedule is competitive with respect to other performance measures such as fairness, locality, and minimum share satisfaction. The authors also designed a scheduling algorithm which classifies the job

based on their requirements and finds an appropriate matching of resources and jobs in the system. The classification part detects changes and adapts the classes based on the new system parameters. A typical Hadoop scheduler receives two main messages from the Hadoop system: a message signaling a new job arrival from a user to store the incoming job in an appropriate queue, and a heartbeat message from a free resource by triggering the routing process to assign a job.

Nita et al. (2015) proposed multi-objective scheduling algorithm of many tasks in Hadoop for big data processing, named MOMTH. For that reason, two objective functions related to users and resources are considered with constraints such as deadline and budget. In order to evaluate the algorithm in the scheduling load simulator, a collaboration platform known as MobiWay that exposes interoperability between a large number of sensing mobile devices and a wide-range of mobility applications is used for performance analysis of the MOMTH. The algorithm is compared with FIFO and fair schedules and it obtained similar performance for the same approach.

S.-Q. Long et al. (2014) proposes a multi-objective offline optimization approach for replica management, in which we view the various factors influencing replication decisions such as mean file unavailability, mean service time, load variance, energy consumption and mean access latency as five objectives. It makes decisions about replication factor and replication layout with an improved artificial immune algorithm that evolves a set of solution candidates through clone, mutation and selection processes. The proposed algorithm named Multi-objective Optimized Replication Management (MORM) seeks the near optimal solutions by balancing the trade-offs among the five optimisation objectives. Table 2.6 shows the comparison of scheduling algorithms based on a multi-objective. It shows the objectives of the proposed

scheduling algorithms with different constraints. These scheduling algorithms are dealing with more than one objective.

**Table 2-6: Comparison of related scheduling algorithms**

| Algorithm | Objectives | Constraints | Minimum schedule length |
|---|---|---|---|
| COSHH (Rasooli & Down, 2014) | Fairness, locality, and minimum share satisfaction | Heterogeneity | Yes |
| MOMTH (Nita et al., 2015) | Avoiding resource contention and having an optimal workload of the cluster | Deadline and budget | Yes |
| MORM (S.-Q. Long et al., 2014) | Optimized replication management | Load variance, energy consumption, and Latency. | Yes |

## 2.9 Multi-objective optimization

Most real-world scheduling parallel machine optimization problems are multi-objective in nature since they possibly have conflicting objectives that must be satisfied at the same time using Meta-heuristic algorithms (Deb, 2014; Marler & Arora, 2004). The multi-objective optimization problem can be defined as "a vector of decision variables, which satisfies constraints and optimizes vector functions from a mathematical description of performance criteria which are usually in conflicts with each other". A general multi-objective optimization problem can be formally defined as follows:

$$\begin{gathered} \min f(x) = \left( f_1(x), \ldots \ldots f_{n(x)} \right) \\ subject\ to\ g_{i(x)} \leq 0, i = \{1, \ldots . m\} \\ h_{i(x)} \leq 0, j = \{1, \ldots . p\} \end{gathered} \quad \textbf{2-2-1}$$

where minimize $f(x)$ is the convex objective function, $x$ is a n-dimensional decision variable vector $x = (x_1, \ldots . x_n)$, $n$ represents the number of objectives to be optimized (two in this study, completion time and cost), $g_{i(x)}$ are the convex inequality constraint

functions, and $h_{i(x)}$ are the equality constraint functions which can be expressed in linear form $m + p$.

Scheduling often requires considering several objectives in an optimization process (Marler & Arora, 2004). For example, a trade-off between resource consumption and performance while scheduling MapReduce jobs on the cloud. This optimization procedure manages two or more objectives functions and is called multi-objective optimization (Y. Jin & Sendhoff, 2001). The primary objective is to develop multi-objective tasks scheduling research flow. Two stages are identified to achieve the goal. The first, being the construction of the model. A mathematical model and schedule evaluation criteria are established in the next chapter. The second stage, being the application of multi-objective genetic algorithms, adopts genetic algorithms. The research flow of the chapter is illustrated in Figure 2.4, details of which are explained in coming sections.

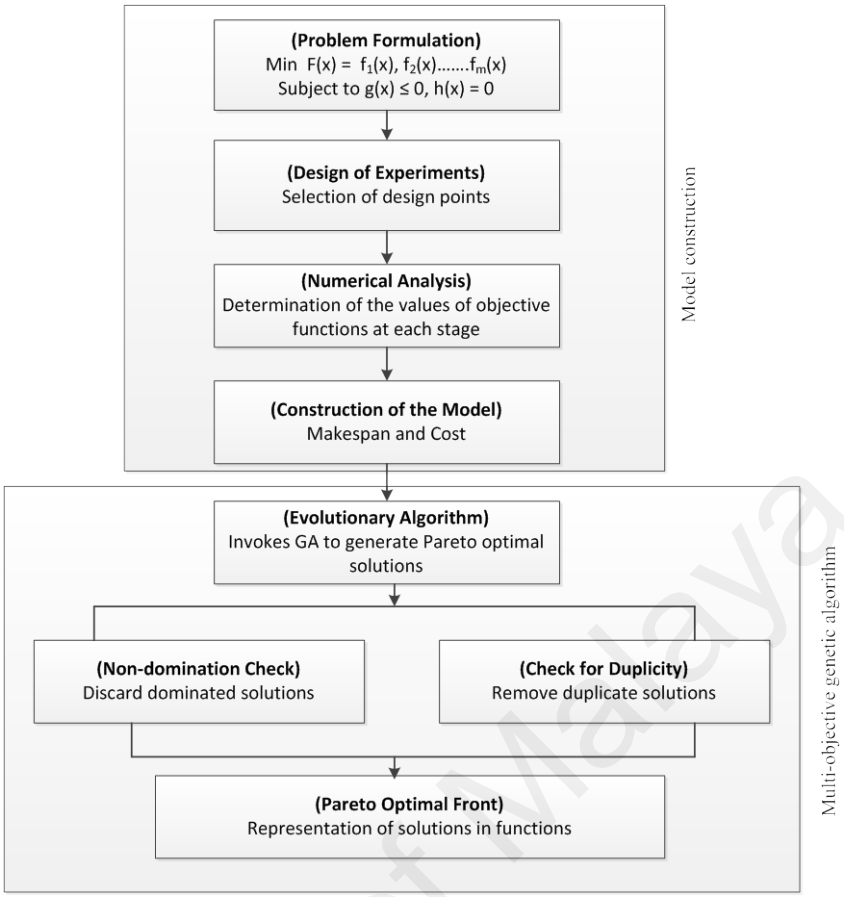



**Figure 2-4: Research Flow**

### 2.9.1 Multi-objective genetic algorithms

It is well known that multi-objective genetic algorithms are among the most useful approaches for multi-objective nonlinear discrete optimization problems. Recent research on the applications of evolutionary algorithms to solve the MapReduce job scheduling has been reviewed in (Tiwari et al., 2015).

A genetic algorithm has been successfully used to solve various optimization problems using a set of procedures by enabling solutions to be found for the specific problems (Horn et al., 1994; Ponnambalam et al., 2000). Generally, a genetic algorithm is parallel in nature and so, it suits better to the MapReduce job scheduling problem. A genetic algorithm is an optimization technique based on the guided random search mechanism. It uses the principle of evolution “Survival of the fittest and inheritance” (Bagchi,

1999). Moreover, a genetic algorithm is based on the payoff function to guide the search and utilizes the probabilistic transition rules (Goldberg & Holland, 1988). It can produce continuous populations of different solutions. These solutions can be reproduced until acceptable results are obtained. Meanwhile, the quality of the individual solutions for each population is improved.

Various implementations of genetic algorithms have been used over years such as NSGA-II (non-dominated sorting genetic algorithm) (Deb et al., 2002), SPEA2 (strength Pareto evolutionary algorithm) (Zitzler et al., 2001). Two popular genetic algorithms like NSGA-II and SPEA2 are used to solve the problem of the scheduling optimization.

#### 2.9.1.1 NSGA-II algorithm

NSGA-II is a non-dominated genetic sorting algorithm; which is the updated version of the NSGA algorithm and it is widely used to solve the multi-objective optimization problem. In 2000 the algorithm was proposed by (Deb et al., 2000) as the first fast, reliable and efficient multi-objective genetic algorithm for non-dominated solutions using Pareto dominance relationship. NSGA-II uses the elitist principle and explicit diversity preserving mechanism. NSGA-II begins with an initial population by determining the number of the chromosome, generation as well as mutation and crossover rate value. Moreover, the possible solutions of the population are generated randomly and evaluation of fitness value of chromosomes is carried out by calculating objective functions. Different solutions are realized at each generation in the population, selection, crossover, mutation operations and ranking. The parent, child and ranked populations are combined in order to select the individuals. If the selecting of the individuals stopping criteria met, then the report of the final population is created. However, if the criteria do not meet, then the process of selection will start. During the

parent selection operation, binary tournament technique can be more suitable. For this study, the classic of one point crossover operator is used, which is important for the creation of the children. Also, two columns are randomly selected during the mutation operator and all the different non-dominated fronts solutions are assigned in a ranking operation. Once the non-dominated fronts ranking is complete, the crowding distance is assigned to sort the solutions in the same fronts. The Pseudocode of the NSGA-II algorithm is described in Algorithm 1(Deb et al., 2000).

**Algorithm 1** : Pseudocode of the NSGA-II algorithm

**Input:** A maximum number of generations $max_{gen}$

**Output:** A population evolved

**begin**

T= 0;

Initialize $P_t$ with the size N Random, G,A, Min Min

Evaluate population according to the objective functions;

**While** $t < max_{gen}$ **do**

Apply selection probability, crossover and mutation (explained in section 4.2.2) to $P_t$ to generate $Q_t$

$R_t = P_t \cup Q_t$;

Assign a hierarchy based on Pareto dominance to $R_t$;

Assign crowding distance to $R_t$;

Select N individuals of $R_t$, according to the crowding comparison operator to generate $P_t + 1$;

end

end

#### 2.9.1.2 SPEA2 algorithm

SPEA2 is used to find or approximate the Pareto-optimal set of multi-objective optimization problems (P.-C. Chang et al., 2007). A regular population and archive mechanism (Pareto archive) are used in SPEA2. All the non-dominated solutions are

copied to the Pareto archive during the generation of the population. Unlike SPEA, the size of the Pareto archive is fixed and is fed to the non-dominated solutions till are satisfied. If the dominated solutions are not completed, then the dominated solutions with the finest goal values are likewise selected. However, if more than one solution found for non-dominated over archive size, then only the best among them with a decreasing order of distance is considered. The pseudo-code of SPEA2 is presented in algorithm 2. The algorithm requires the same input parameters as the NSGA-II.

**Algorithm 2** : Pseudocode of the SPEA-II algorithm

**Input:**

$P_t$ - Population size

$Q_t$ - Archive size

t - Maximum number of generations

**Output:** A (non-dominated set)

**begin**

Set t= 0;

$P_t$ = 0;

Initialize $P_t$ and create the empty archive

**While** (stop condition) is false **do**

Compute fitness of each individual in $P_t$ and t

Copy all individual evaluating to non-dominated vector

Use the truncation operator to move elements from t when the capacity in $P_t$ to t

Preform binary tournament selection with replacement to fill the mating pool

Apply crossover and mutation to the mating pool

**end**

**end**

### 2.9.2 Strengths of Genetic Algorithm

A genetic algorithm has the ability to accommodate different types of problems either continuous or discrete, which makes the genetic algorithm more flexible in dealing with

a range of optimization problems such as scheduling. Moreover, genetic algorithm offers a good performance by exploring the solution space in multiple directions at once. As a result, many researchers are attracted by such approach to tackling different problems in different domains (Nagata & Chu, 2003). A genetic algorithm does not require the derivation of information before solving the problem. As typically holds a single solution to a problem at a particular time, and search to find out the next direction of movement based on the gradient function of the latest solution. When the genetic algorithm decided the distance to move, then new solution is selected.

Moreover, the strength of genetic algorithms is performed on the problem with more complexes such as changes over time, noisy and the fitness function is discontinuous. Solving problems in multi-objective usually require huge alternative solutions impossible to search exhaustively, which difficult to select the best solution (Marczyk, 2004). Local optima can effectively be avoided by the genetic algorithm and to provide several alternative solutions to a problem. Local optima are a poor solution that pretends to be the best through which algorithms can be deceived from reaching the optimal solution, but the genetic algorithm has the capability to avoid the local optima. This is one of the major attractive characteristics of the genetic algorithm (Chiroma et al., 2015).

## 2.10 Summary

Scheduling plays an important role in big data, mainly in reducing the execution time and cost during processing of data. To understand scheduling in big data studies, this chapter presented an introduction to the rise of big data, which include current big data trends and big data challenges. The chapter also provided an overview of Apache Hadoop focusing on HDFS, and Hadoop MapReduce. Cloud computing and the relationship between cloud and big data are discussed. Moreover, scheduling in big data

platforms and its algorithms, which are closely, linked to big data processing studies is provided. It also underlined the requirements for scheduling in big data platforms based on data locality, SLA-based, load balancing, time and cost. Moreover, it highlighted the current state-of-the-art scheduling algorithms and investigated the related scheduling algorithms by comparing their objectives and unique characteristics.

This chapter also discusses different scheduling algorithms used in big data platforms. Furthermore, in order to highlight the results of the study of the previous survey conducted, the findings are strengthened. By presented the concept of scheduling in big data platforms, various optimization options are identified. Having said that, a methodical approach, which can identify the importance of scheduling big data on the cloud is crucial.

Moreover, in this chapter scheduling of MapReduce jobs of the large volume of data processing are studied. The study considers the minimization of completion time given a fixed budget, minimization of the monetary cost given a deadline and the trade-offs between the completion time and monetary cost of using cloud computing.

Strengthened by the result of the study, this chapter has also revealed the issues that led to performance degradation in resource allocation and task scheduling problems, especially, when large amounts of data are being processed by a framework like MapReduce in a distributed environment. Therefore, this thesis is devoted to addressing those issues, by investigating the scheduling mechanism in Hadoop MapReduce. More specific, this thesis seeks optimization solutions to tackle the issue of resource allocation and task scheduling in Hadoop MapReduce. Moreover, this chapter highlights the scheduling algorithms for big data and investigates the related scheduling algorithms. This section discusses several approaches to the scheduling problem. These approaches consider different scenarios, which take into account the application types,

the execution platform, the type of algorithms used and the various constraints that might be imposed. Moreover, the chapter discusses the multi-objective optimization by focusing on genetic algorithms like NSGA-II and SPEA2.

In this next chapter, we investigate and model the problem of scheduling a MapReduce job on the cloud as an optimization problem between time and cost within budget and deadline constraints.

# CHAPTER 3: SCHEDULING MAPREDUCE JOBS AND THE PERFORMANCE ISSUES

This chapter aims to provide analysis on the performance of scheduling MapReduce jobs using physical and cloud cluster. This analysis has an impact on the processing in terms of time and cost. Normally in the scheduling process, the processing time of the job on a machine is assumed to be fixed in advance. However, in reality, it demands resources to complete the job and the execution time is determined internally by many resources allocated. Using analytical analysis mathematical equations is derived to identify a time-cost model for the processing of big data, which demonstrate the significance of the execution time and budget when utilizing cloud resources. We formulate the completion time with a budget constraint model and cost with deadline constraint model.

The rest of the chapter organized as follows. Section 3.1 presents our analysis of the performance of executing MapReduce jobs when using physical and cloud cluster. We describe problem identification in Section 3.2. Section 3.3 offers time-cost models analysis using the multi-objective evolutionary algorithm. Section 3.4 provides computational results.

## 3.1 Hadoop physical vs. cloud cluster analysis

This section provides analysis of the impact of using physical versus cloud cluster when process large amount of data. The reason to conduct such analysis is to identify the important of cluster usage in terms of the cost of execution time and the utilization of the resources to complete the tasks. A physical cluster is a group of computers connected by a local area network (LAN), where Hadoop distribution is installed directly on the physical machines that are bounded by disk I/O. in contrast, cloud computing is a type of computing that relies on sharing heterogeneous computing

resources to deliver computing services over the Internet in a convenient and scalable manner (Armbrust et al., 2010).

In the cloud, Hadoop distribution is installed on the virtual machines, where multiple "machines" does not require full physical resources at all times because the underlying infrastructure is shared (White, 2012). Moreover, Hadoop has specifically designed for storing and processing unstructured data in a homogeneously distributed computing environment, which run on commodity hardware. More recently, there has been some effort to proposed Hybrid environment for MapReduce framework. For instance, Sharma et al. (2013) introduce hierarchical scheduler for hybrid data centers with two-phase named HybridMR. The algorithm comprises of multiple virtual machines and physical to make use of both paradigms. Firstly, the information acquired from HybridMR profiles to estimate virtualization overheads based on incoming MapReduce jobs to gauge can automatically guide placement between physical machines and virtual machines. Secondly, HybridMR builds run-time resource prediction models and performs dynamic resource orchestration to minimize the interference within and across collocated interactive and MapReduce applications.

The experiments were carried out in a MYREN cloud and physical cluster machines. We use five PCs as well as five Virtual Machine (VM) with the following configurations: 2.80 GHz processor, 2 GB main memory, and 1000 GB disk space. Hadoop cluster is used on Linux Ubuntu 14.04 where one machine runs a NameNode and ResourceManager, and the remaining are running DataNode and DataManager. Moreover, we use PingER data sets of different sizes varying from 500MB to 2 GB.

Table 3-1 provides a result of the comparison between the physical cluster and cloud cluster in terms of execution time.

**Table 3-1: Comparison between physical and cloud cluster in terms of execution time**

| Data size (GB) | Physical Cluster (s) | Cloud (s) |
|---|---|---|
| 1 | 46.28 | 42.04 |
| 2 | 53.44 | 51.67 |
| 3 | 69.30 | 50.57 |
| 4 | 97.57 | 60.39 |
| 5 | 124.92 | 90.23 |

As shown in Figure 3-1 the experimental result illustrates that increasing number of data size in the cluster, significantly increase the time necessary to run the application on both physical cluster and cloud. Figure 3-1 shows for both physical cluster and cloud require less time to finish the jobs. Hence, running jobs on a cluster with a large number of data is the main motivation for using Hadoop to process big data. Given the relatively low commodity hardware of physical cluster, the results were fairly significant. Using five nodes of machines to run the jobs practically increase the runtime, compared with one node. Thus, increasing the number of machines could lead to increase in runtime as the cluster size increased.

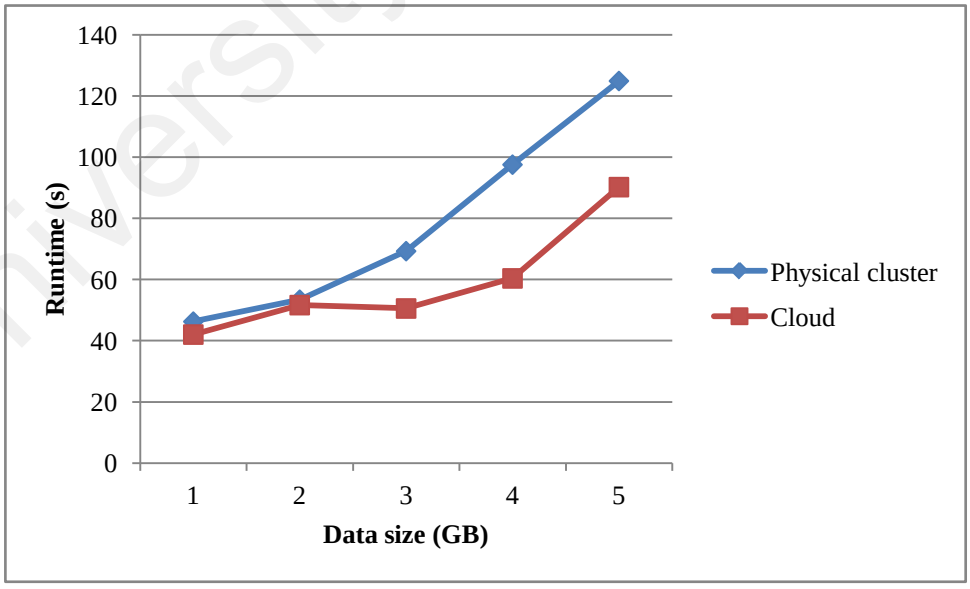



**Figure 3-1: Comparison between physical cluster and cloud in terms of runtime**

Nevertheless, the results indicate the unavailability of free resources. Running five cloud nodes put a considerable load on the host computer running the virtualization software and pushed the CPU utilization to 100%. This indicates that the use of cloud virtual machines helped better utilize the resources of the host computer. However, these machines require an optimal scheduling algorithm in order to reduce the overall execution time. The monetary cost to complete the entire workflow based on cloud platforms (e.g., Amazon EC2) is also an important metric as the resources are claimed on demand and will be charged as long as it is used. The monetary cost is closely related to the completion time. However, they do not always correlate due to the pricing scheme in the cloud.

### 3.2 Analytical Time-Cost analysis

Several common assumptions are made in this study given the relatively high complexity of MapReduce job scheduling. Some of these assumptions have been used in Nita et al. (2015) and Wang and Shi (2014). These assumptions are as follows. (i) One or more free slot(s) are available at a given time in each node $N = \{n_1, n_2, \ldots, n_m\}$ in the cluster, where the minimum number of tasks for the map is reduced to less than or equal to the available slots. (ii) Big data processing for each query is translated into one or more MapReduce job(s) $J = \{j_1, j_2, \ldots, j_h\}$, where each job has multiple tasks $T = \{t_1, t_2, \ldots, t_n\}$, which consist of a known number of map tasks $N_m$ and reduce tasks $N_r$. (iii) The reduce tasks can only be launched when all the map tasks have been completed. (iv) For each map task, the exact amount of data processed $S_m$ is known from the beginning and is equally distributed among map nodes. (v) Each job has arrival time $A$, deadline $D$, and allocated budget $B$ for using the node. (vii) Sufficient resources are allocated for each task in the cloud, which implies that a node is never completed by more than one tasks, and its allocation is charged based on the actual time that it is used and the fixed service rate. Thus, before discussing the

model for completion time and monetary cost, the definition of the problem is described as follows: A MapReduce job $J$ is modeled as a workflow that consists of multiple tasks $T$. This workflow is a collection of independent map and reduce tasks executed in parallel and denoted as $t = \{t_{m_1}, t_{m_2}, \ldots, t_{m_u}, t_{r_1}, t_{r_2}, \ldots\ldots t_{r_u}\}$. Each map/reduce task is run in a cloud VM known as a "node" with a possibly distinct performance configuration, and a different charge rate for each machine is deployed in the cluster. Each job has a particular number of slots assigned; these slots can be used by map and reduce tasks at any given time, where no reduce task can be started until all the map tasks for the job are completed. However, the same slots can be used by the mapper and the reducer. For each task, $t_i$, $0 \leq i \leq j$, where $t_i^u \leq u \leq N$ represents the time to run tasks $t_i^u$ on node $N$. Table 3-2 shows notations associated with the problem description and modeling.

### 3.2.1 Completion time with budget constraint model

Modeling completion time is an essential part of this study because it is the basis for the rest of the work, other calculations, and the proposed algorithms. This procedure is one of the most widely accepted methods for modeling the optimization problem (Heintz et al., 2012). Many variants of this model are available, but one variant is particularly related to the map and reduce task assignment problem with budget constraints (Wang & Shi, 2014). In this problem, the goal is to minimize the Makespan given a particular budget constraint. To achieve this goal, the execution time $T_{(t_i,b)}$ of a task $t_i$ with a specific budget $b = \frac{the\ total\ budget\ (B)}{the\ total\ numer\ of\ tasks\ (T)}$ should be defined as the time required to complete the task within the specific budget. The shortest time to complete the task is denoted as

$$T_{(t_i,b)} = t_i^u \ , c_i^{u+1} < B < c_i^{u-1} \qquad \textbf{3-1}$$

where the estimation of the budget per map/reduce task can be described as:

$$b \leq 1, \forall t_i \in J \quad \textbf{3-2}$$

The time to complete task $t_i$ with budget $b$, denoted as $t_i(b)$, is defined as the time consumed when all the tasks are completed within the given budget as follows:

$$t_i(b) = \max_{t_i \in J} \{T_{(t_i,b)}\} \quad \textbf{3-3}$$

For the query, the reduce task is started immediately after map tasks complete. Therefore, the total Makespan with budget $B$ to complete all tasks for particular job is defined the sum of all tasks times. The goal is to minimize this time within the given budget $B$.

$$t(B) = \min_{\sum_{t_i \in J} b \leq B} \sum_{t_i \in J} T(B) \quad \textbf{3-4}$$

### 3.2.2 Cost with deadline constraint model

Pay-as-you-go is a well-known pricing model implemented by cloud service providers to charge users based on quality of service (QoS) requirements. The charges for some resources in cloud-like network bandwidth and storage are at a particular rate.

The pricing model implemented in the cloud is a pay-as-you-go model, where services are charged as per the QoS requirements of the users. The resources in the cloud, such as network bandwidth and storage, are charged at a specific rate (Hussain et al., 2013). Thus, cost has become an important objective in scheduling. Total cost incurred by processing big data can comprise many cost components, such as computation and data transfer costs. Cloud computing offers a variety of resources and services per manner of use. These computational resources are basically used per time quantum pricing

scheme. This quantum is typically 1 h, although recently, an alternative seems to be receiving increasing interest.

Given the deadline for the job, the minimum cost to complete all the tasks is:

$$C_{(N_m+N_r)}\,(D) \;=\; \sum_{t_i \in J} C_i\;(D_i) \qquad \textbf{3-5}$$

where $C_i\,(D_i)$ is the minimum cost to complete the task within the $D_i$. Thus, $t_m \leq D_i$ and $t_r \leq D_i$

$$C_{(N_m+N_r)}\,(D) = \min_{t_i \in J} \sum_{t_i \in J} C_i\;(D_i) \qquad \textbf{3-6}$$

The computation cost is defined based on resource $R_j$, such that, for each task $t_i$ executed on resource $R_j$, two timestamps will be recorded, that is, $A$ when the task starts and $E$ when the task finishes its execution. The value $E$ can be defined as $A + t_{(i,b)} + \max_{i \in J} \frac{Size\ of\ the\ data}{Bandwith}$. These timestamps indicate the period during which the resources should to be utilized because of the execution of task $i$.

**Table 3-2: Notations associated with the problem description and modeling**

| Symbol | Definition |
|---|---|
| $J$ | The number of jobs $j = 1, \ldots, n$ |
| $N$ | The number of nodes $N = \{n_1, n_2, \ldots, n_m\}$ |
| $T$ | The number of tasks $T = \{t_1, t_2, \ldots, t_n\}$ |
| $c_j^u$ | The cost for each job |
| $C_m$ | The cost of executing a single map task |
| $C_i$ | Completion time of each task |
| $D_i$ | Deadline of each task |
| $t(B)$ | The total budget of all tasks during the execution |

| | |
|---|---|
| $k_i$ | Performance degradation perimeter |
| $t_i(b)$ | The time consumed when all task is completed within the given budget |
| $R_j$ | Resources |

### 3.3 Time-Cost models analysis using multi-objective evolutionary algorithm

The analysis of the models is carried out using the evolutionary algorithm. It is well known that multi-objective evolutionary algorithm is among the most useful approaches for multi-objective. This fact makes multi-objective genetic algorithms suitable for solving the MapReduce job scheduling as the objective functions and the decision variable is an integer in the MapReduce job scheduling. Recent research on the applications of evolutionary algorithms to solve the MapReduce job scheduling has been reviewed in (Tiwari et al., 2015).

The following describes the multi-objective genetic algorithms, focusing on the problem specific features, designed to solve the time-cost problem based on the NSGA-II and SPEA2 algorithms discussed in Chapter 2.

*(a) Representation of solutions:* For the problem formulated each solution is represented by the one chromosome with N genes; one or more genes are considered for each node. The first gene contains the task to which the corresponding node is assigned in the cluster. The second gene contains the selected resources for the corresponding node. It is seen that by this representation at least one resource and one task is selected for each node and therefore constraints $b \leq 1, \forall t_i \in J$ and $t_m \leq D_i$ , $t_r \leq D_i$ are satisfied automatically.

*(b) Initial Population:* The first step in the functioning of a genetic algorithm is the generation of an initial population, which is a set of randomly generated of populations to find an optimal solution (Goldberg & Holland, 1988). Each solution in the population

is named individually, and this individual is referred as a chromosome, which involved genes and its value can be bit string, real number, permutations of elements, program elements or data structure. From the initial population, the individuals are being selected and its operations are transformed to the next generation. The mating chromosomes are selected based on some specific criteria.

*(c) Fitness Function:* Fitness is used to measure the quality of the represented solutions in the population according to the given optimization objective. The fitness values are then used in the process of natural selection to choose which potential solutions will continue to the next generation. Maximization can be straightforwardly achieved using fitness functions $F(x)$ in genetic algorithms, where it first derived from the problem's objective function $F(x) = f(x)$, however, minimization is required multiple transformations for example $F(x) = \frac{1}{1+f(x)}$, This transformation does not alter the location of the minimum but it converts the original minimization problem into a maximization problem.

Since the objective space is explored in two directions, in this case, time and cost, then the cost fitness function of an individual is defined as expressed by Eq (3-10) adapted from (Yu & Buyya, 2006).

$$F_{C_{(N_m + N_r)}}(I) = \frac{\sum_{t_i \in J} C_i}{B^{\alpha} \times maxCost^{(1-\alpha)}} \quad \textbf{3-10}$$

where the $\sum_{t_i \in J} C_i$ is the sum of the execution cost of the task, $maxCost$ is the most expensive solution of the current population and $\alpha$ is the binary variables used in this function. The cost fitness module for the budget constrained scheduling encourages the formation of the solutions that satisfy the budget constraint. For the deadline

constrained scheduling, it encourage the genetic algorithm to choose individuals with less cost.

The time fitness component for the budget constrained is constructed to encourage the genetic algorithm to choose individuals with earliest completion time from the current population. The time fitness function of an individual is defined as expressed by Eq (3-11) adapted from (Yu & Buyya, 2006).

$$F_{t_{(N_m+N_r)}}(I) = \frac{\max\limits_{t_i \in J} \{T_{(t_i,b)}\}}{D^{\beta} \times maxTime^{(1-\beta)}} \quad \textbf{3-11}$$

where $\max_{t_i \in J} \{T_{(t_i,b)}\}$ the completion time of the individual is, $maxTime$ is the largest completion time of the current population.

In equation (3.11) and (3.12), the fitness of time and cost with respect to the budget and deadline constraints. The combination of the both fitness functions is defined as follows:

$$F(I) = \begin{cases} \alpha \times F_{C_{(N_m+N_r)}}(I) + \beta \times F_{t_{(N_m+N_r)}}(I), if\ F_{C_{(N_m+N_r)}}(I) > 1 \\ F_{C_{(N_m+N_r)}}(I)^{\beta} \times F_{t_{(N_m+N_r)}}(I)^{\alpha}\ otherwise \end{cases} \quad \textbf{3-12}$$

*(d) Encoding:* Encoding problem in the genetic algorithm has been studied by many researchers in the past, where considerably different problems need completely different genetic encoding for a good solution to be established. Some different encoding methods are proposed in the literature (Jia et al., 2003; Orero & Irving, 1998). The encoding strategy can be divided into many types which are the binary, permutation, value, tree, direct chromosome representation and indirect chromosome representation. This study adopted direct representation to give viability and legibility to a chromosome. The encoding strategy of the mentioned scheduling problem is obtained to assist in the development of the methods as shown in Figure 3-2. A chromosome with

an (N * 3) matrix is proposed to enable the scheduling problem. Each task assigned to the node represented as a gene of the chromosome. There are two rows represent the encoding solution. The first row in the chromosome displays the node in which the task is executed. The second row displays the budget on each node.

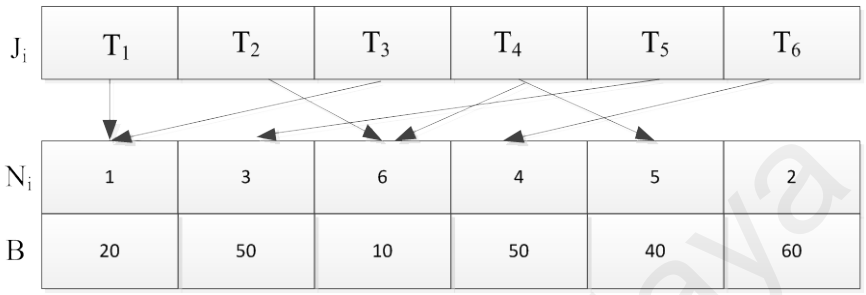



**Figure 3-2: Encoding strategy**

where J is a job which consists of multiple tasks $T = \{t_1, t_2, \ldots, t_n\}$ and known number of map tasks $N_m$ and reduce tasks $N_r$, respectively. N is the number of nodes in the cluster and B is the budget that allocated to each node.

| Order | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Tasks | 1 | 2 | 1 | 6 | 3 | 4 |

**Figure 3-3: Illustration of problem encoding**

As shown in Figure 3-3 the chromosome has divided into two strings to represent the order and tasks. The string order is a vector containing a permutation of all tasks indexes.

*(e) Selection:* In order to determine the probability of various individuals genetic to the next generation, the proportion selection operator is used to breed a new generation. The proportional selection operator means the probability, which is selected and genetic to next generation sets is related to the size of the individual's fitness. According to the

fitness value generated by the fitness function, the selection probability can be determined by using equation 3-13.

$$P_i = \frac{F_i}{\sum_{i=1}^{N} F_i} \qquad \textbf{3-13}$$

where $P_i$ is the selection probability of the string $i$, $F_i$ the fitness value of string $i$ and N is the population size.

*(f) Crossover:* Crossover in genetic algorithms is used to evolve the programming of one or more chromosomes from one generation to another. The purpose of a crossover is to improve the new chromosome from their parents if it inherits the better characteristics from each of the parents. It happens during evolution according to a user definable crossover probability. A popular implementation of crossover uses the single point crossover process (see Figure 3-4) in which a crossing site is randomly chosen along the string length, and all bits to the right side of the crossing site are exchanged between the two parent strings.

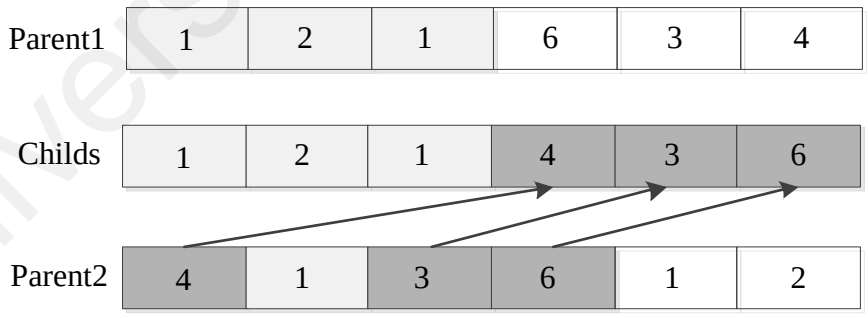



**Figure 3-4: Crossover (Single point)**

*(g) Mutation:* While a crossover operator attempts to produce new strings of superior fitness by affecting large changes in a string's makeup, the need for local jumps in search around a current solution also exists. Mutation is used to keep genetic diversity from one generation of a population of genetic algorithm chromosomes to the other,

where the chromosome coding series were replaced by the other gene values in order to generate a new individual. Figure 3-5 shows the mutation of a single point.

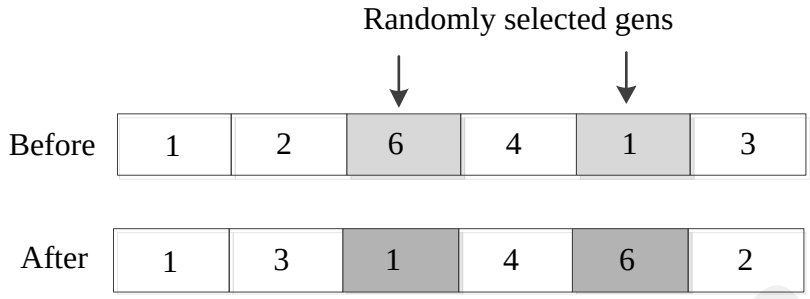



**Figure 3-5: Mutation (Single point)**

In the following subsection, the detail of the simulation environment and experimental setting are discussed. Then simulation, computational results are presented.

### 3.3.1 An experimental setting

The problem is implemented according to its descriptions in the above sections and compared it using two popular genetic algorithms like NSGA-II and SPEA2 on multi-objective evolutionary algorithms (MOEAs) framework. Section 2.9.1 provides a brief introduction to NSGA-II and SPEA2 algorithms. For both algorithms, the parameters setting used are shown in Table 3.3. The population size $P_t$ is 100 with max evolution of 1000; archive size $Q_t$ for SPEA2 is 50. However, NSGA-II has no archive population. The crossover probability is 0.7 and, the mutation probability is 0.1 to decide the convergence and the diversity of the result.

**Table 3-3: Parameter Setting Summary**

| Parameters | Setting | |
|---|---|---|
| | NSGA-II | SPEA2 |
| Population size | 500 | 500 |
| Archive size | N/A | 50 |
| Max evaluations | 1000 | 1000 |
| Crossover probability | 0.7 | 0.7 |
| Mutation probability | 0.1 | 0.1 |

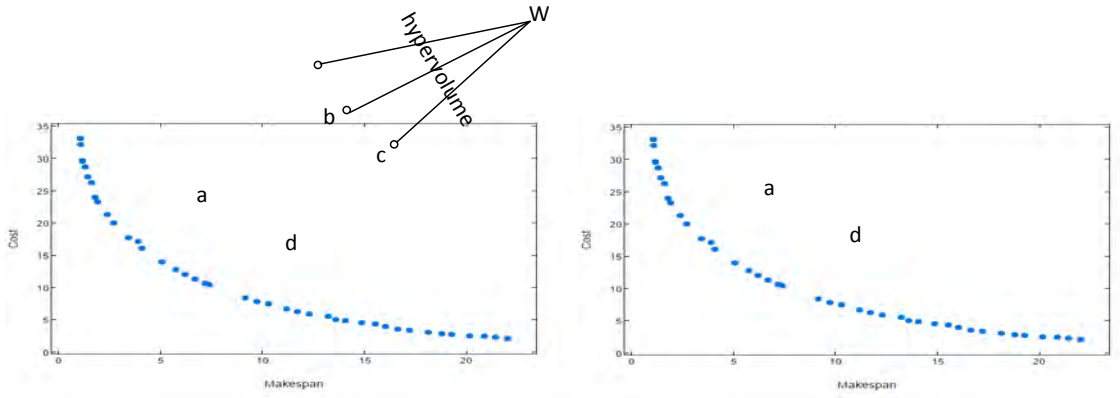
This study validated the performance of the proposed multi-objective scheduling model through experiments. The results obtained by MOEA Framework are compared to validate the quality of the solutions by genetic algorithm NSGA-II and SPEA2.

Specific parameter settings for all the considered algorithms are described in Table 3.3. The parameters used in the experiment are based on the previous research findings. Each experiment was repeated 10 times with different random seeds. That is being said. First, the shortest Makespan of the schedules is computed using NSGA-II and SPEA2 algorithms. Second, the aspect of the cost of the schedules regarding money is considered, analyzing the cheapest solution using NSGA-II and SPEA2 algorithms. Lastly, the hypervolume is defined in order to evaluate the quality of computed trade-off solutions described in 3.3.2.

The trade-off solutions of Makespan and cost computed by NSGA-II and SPEA2 are analyzed based on different workflow in order to find an optimal solution between the two conflicting objectives. For this analysis, the graphical representation is used for the solution computed by the two algorithms.

### 3.3.2 Multi-objective tradeoff solutions

The dominance is introduced in this study as it is not possible to find an optimal solution that minimizes both the completion time and cost at the same time. A solution $y$ dominates a solution $z$, if the completion time and cost of $y$ are less than those in $z$. Conversely, two solutions are said to be non-dominated whenever none of them dominates the other, for example, one is better at compilation time and, the other is better in cost.




**3-6: Multi-objective tradeoff solutions**

Figure 3-6 shows the multi-objective tradeoff solutions, in which the solutions marked (a) and (d) dominates the marked (b) and (c) because it has better Makespan and cost. Meantime, (a) and (d) are represented by the non-dominated solutions, in which (a) is good in Makespan, while (d) is good in cost. This set of non-dominated solutions represented a set of trade-off solutions among the different mapping of the workflow tasks with different Makespan and cost and known as the Pareto front (the trend line containing the (a), (d), and (e) solutions).

For multi-objective, the minimized or maximized optimal result is represented by a multi-dimensional vector of values. This vector of the solution is called the Pareto optimal set (ps). The image of this Pareto set of the objective functions is called a Pareto front. The PS comprises many solutions, as there exist different trade-off solutions, each describing a compromise between the different objective functions.

That is being said; hypervolume is used to measure the quality of a set tradeoff solutions. Given a set of tradeoff solutions, the hypervolume measures the area enclosed between the points in (b), (c) and (W) as it can be seen in Figure 3-6, usually selected as the maximum objective value. Thus, the better and the more diverse the points contained in X are, the higher hypervolume.

### 3.4 Computational Results

Recently many researchers have studied measurement criteria for comparing non-dominated solutions. For example, a spread metric has been introduced by (Ranjithan et al., 2001) to defines the high number of choice represented by the non-dominated solutions and a coverage metric that characterizes the distribution of solutions. J. Knowles and Corne (2002) have proposed an *R* metrics, which tend to dominate alternatives regarding their profile of a particular property of a Pareto set approximation which does not require knowledge of the true Pareto front and fairly scalable to many objectives. They require, however, using reference set and a set of points *S* output from an optimization run, and *R* metrics provides a single scalar value that estimates the 'utility of *S*.

For computational experiments, randomly non-dominated solutions for each job are generated using testbed. These non-dominated solutions generated during the experiment are copied to the Pareto archive. Compared with SPEA2, the Pareto archive size is fixed in this algorithm. The task data include the completion time *T*, cost for each job to be completed within specific budget *B* and the deadline for each task *D*. For each task, the completion time and cost are generated using a uniform distribution. The numerical value of all obtained solutions of 50 tasks and ten nodes problem is presented in Table 3-4.

**Table 3-4: Comparison of NSGA-II and SPEA2 with the 50 tasks and ten nodes problem**

| | **NSGA-II (s)** | | **SPEA2 (s)** | |
|---|---|---|---|---|
| | **x-value** | **y-value** | **x-value** | **y-value** |
| Seed 1 | 54.509 | 26.451 | 30.480 | 43.641 |
| Seed 2 | 10.134 | 40.767 | 45.104 | 71.035 |
| Seed 3 | 44.629 | 31.643 | 51.580 | 42.350 |
| Seed 4 | 23.322 | 19.432 | 43.621 | 42.862 |
| Seed 5 | 56.757 | 24.876 | 43.389 | 56.760 |
| Seed 6 | 57.583 | 49.501 | 54.741 | 19.424 |
| Seed 7 | 34.516 | 54.637 | 64.431 | 62.519 |

| | | | | |
|---|---|---|---|---|
| Seed 8 | 29.698 | 25.583 | 17.901 | 59.871 |
| Seed 9 | 52.750 | 29.653 | 42.564 | 43.574 |
| Seed 10 | 38.633 | 46.432 | 37.299 | 67.891 |

The comparison of NSGA-II and SPEA2 are presented in Table 3-4 with the values of each one of them, and we repeat the experiment for ten times. Figure 3-7 and 3-8 below show the approximation set of each algorithm on the effect of the different x-values and y-values of the distribution. Both figures show the domination of all solutions based with respect to the cost and makespan. Some reference sets are chosen for this problem and are shown in the black square. For a particular population size and a chosen number of reference sets.

Moreover, the figures show that there exist many solutions that dominate other solutions and reference sets is represented non-optimal solutions. While, in the reference sets the shortest distances is most preferred, however, consider non-dominated solutions over dominated solutions allows Pareto-optimal solutions to be found (Fonseca et al., 2009). Thus, if the user is interested in knowing optimal trade-off solutions in (minimum execution time and cost) the proposed procedure is able to find solutions near the reference sets, instead of finding Pareto-optimal front, thereby allowing the user to consider only a few solutions and that too solutions, which lie in the regions of user's interest.

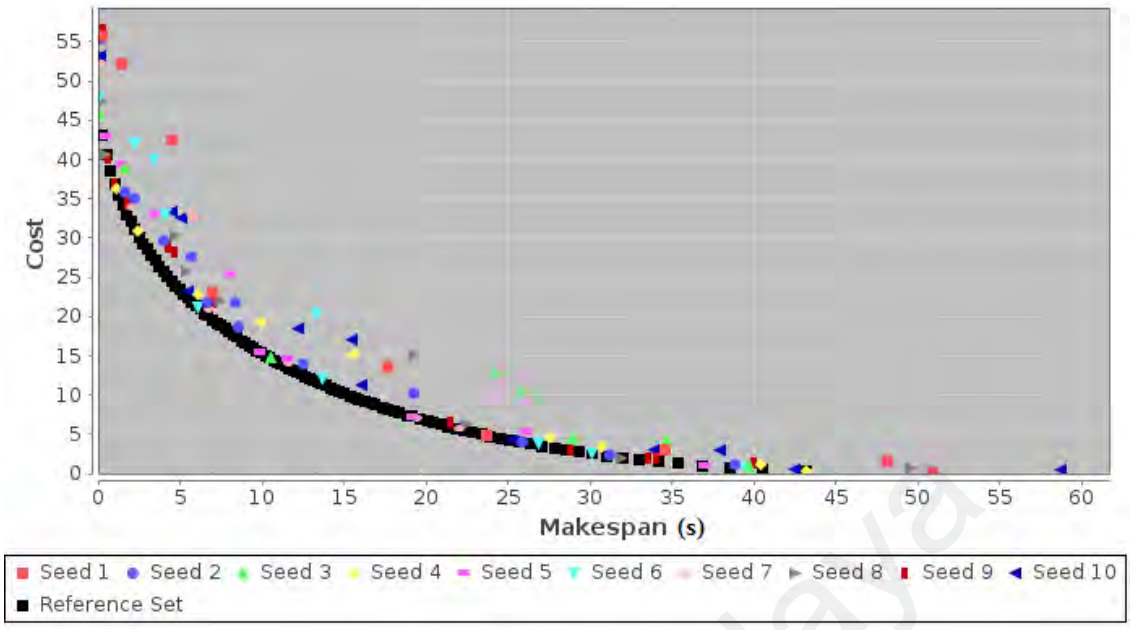



**Figure 3-7: Comparisons of the 50 tasks and 10 nodes using NSGA-II**

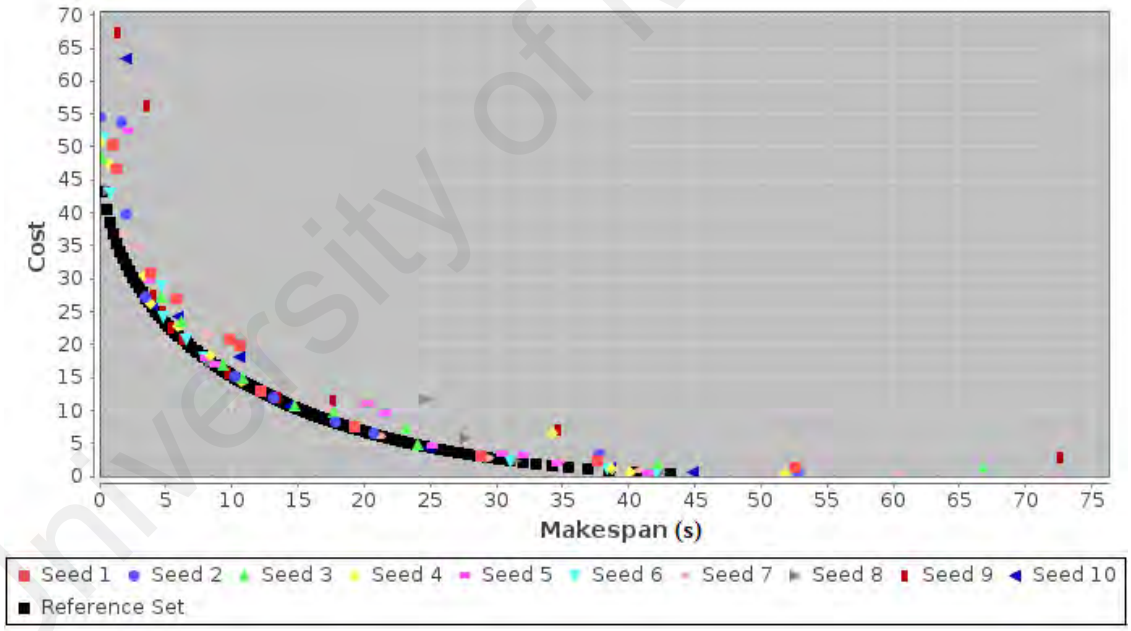



**Figure 3-8: Comparisons of the 50 tasks and ten nodes using SPEA2**

This result developed to find a solution for the multi-objective scheduling problem. There are two phases applied in the algorithm: the first phase, is the sub-populations, which focus on the exact search space and careful avoids all individuals from changed to a local optimal and regrouped as a single big population based on the subpopulation,

which explores solution space ignored or missed in the first phase. In the second phase, each's chromosome in this big population is randomly assigned a weight value to explore for more solution spaces.

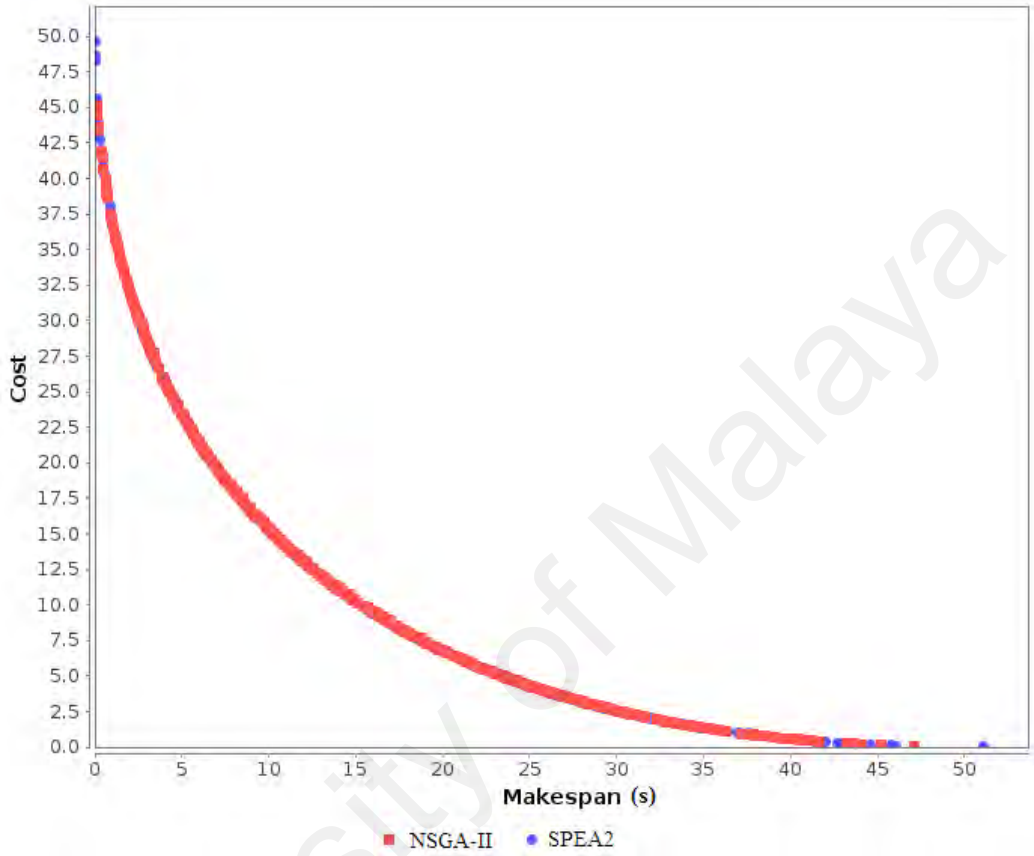



**Figure 3-9: Example of NSGA-II/SPEA2 comparison**

Figure 3-9 shows an example where 50 tasks should be scheduled on ten identical nodes. The objective values of all the obtained solutions are shown in Table 3.4. As shows in Figure 3-9, a set of alternative solutions, called non-dominated solutions or Pareto optimal solutions, is obtained by the two algorithms, from which the users may select one according to their preference. If the users preferred less time to complete the job with a high budget, the user can choose the two left solutions in Figure 3-9, whose no y-value more than 30, while when there is a less budget to be allocated for cloud resources, users could choose the bottom right solution, whose x-value is 10.134 and

17.901 for both algorithms respectively. We can observe that the front obtained by NSGA-II dominates the other obtained by SPEA2. The comparison confirms the optimality of genetic algorithm.

## 3.5 Summary

This chapter provides analysis on the performance of scheduling MapReduce jobs using physical and cloud cluster, which has an impact on the processing time and cost. It provides analysis of the impact of using physical versus cloud cluster when processes large amounts of data. The reason to conduct such analysis is to identify the importance of cluster usage in terms of the cost of execution time and the utilization of the resources to complete the tasks. The experiment result shows increasing in execution time, which indicate the unavailability of free resources. Running five cloud nodes put a considerable load on the host computer running the virtualization software and pushed the CPU utilization to 100%. This indicates that the use of cloud virtual machines helped better utilize the resources of the host computer. However, these machines require an optimal scheduling algorithm to reduce the overall execution time. The monetary cost to complete the entire workflow based on cloud platforms (e.g., Amazon EC2) is also an important metric as the resources are claimed on demand and will be charged as long as it is used. The monetary cost is closely related to the completion time. However, they are not always correlated due to the pricing scheme in the cloud.

The analysis of the models is carried out using genetic algorithms. It is well known that multi-objective genetic algorithms are among the most useful approaches for multi-objective. For computational experiments, randomly non-dominated solutions for each job are generated using testbed. These non-dominated solutions generated during the experiment are copied to the Pareto archive. We can observe that the front obtained by

NSGA-II dominates the other obtained by SPEA2. The comparison confirms the optimality of genetic algorithm.

# CHAPTER 4: FRAMEWORK FOR MULTI-OBJECTIVE SCHEDULING ALGORITHMS

This chapter offers comprehensive details about the proposed scheduling algorithms by considering resource allocation and task scheduling in a heterogeneous cloud environment. These algorithms are an extension to the multi-objective list-based scheduling algorithm for optimizing the workflow (Durillo & Prodan, 2014) and greedy Cost-Time distribution (Yu & Buyya, 2006). The aim of the proposed algorithms is to improve the performance of MapReduce scheduling for big data processing to meet the deadline and budget constraint. The deadline and budget constraints in big data processing scheduling are important as it provides a cost-effective allocation of the cloud resources among Hadoop nodes. Given that, this chapter details the resource allocation and task scheduling strategies to optimize workflow big data processing in the cloud in terms of completion time and the monetary cost of using the cloud.

Efficient large-scale data processing is one of the major aspects of MapReduce framework characterization. Scheduling tasks in virtual machines in the cloud demand resources of the cloud, typically, users are aware of the deadline of when the job is completed. However, in cloud computing environment, all machines compete for resources to execute the jobs. These resources are controlled by batch queue systems, which may not offer guarantee deadline during the task execution, only if the priority used for resource reservation, which is a restricted level of service.

Whereas, in MapReduce with multiple jobs workloads running simultaneously, resource-ware is important for enhancing resource utilization across nodes (Polo et al., 2011). It provides fast reconfigurable architectures capable of adapting at runtime according to changing requirements and constraints (Sousa et al.). For example, to minimize contention for CPU and I/O Yong et al. (2009) proposed resource-aware

schemes on the slave's nodes in order to improve the performance of the cluster. Such solution can offer a learning mechanism in which tasks can be classified based on their CPU bound and IO bound categories and assign jobs as appropriate. However, preventing batch processing jobs from interfering with foreground workloads is a challenging task (W. Zhang et al., 2014). Furthermore, (Z. Guo, G. Fox, M. Zhou, et al., 2012) offer a new way of resource aware, named, resource stealing to allow tasks are running in the cluster to steal resources kept for idle slots and release them whenever a new task is assigned to that slots. Resource stealing can utilize wasted resources being used by others without interfering with normal job scheduling. The results show that resource stealing may improve the performance improvement for compute-intensive and network intensive applications. However, a heterogeneity property in the cloud is more difficult when the cluster is shared among multiple jobs.

The proposed framework for multi-objective algorithms tries to identify the importance of resource allocation and task scheduling in the cloud, by considering both completion time and the cost minimization models. These models are based on similar work by (Kc & Anyanwu, 2010; Nita et al., 2015; W. Zhang et al., 2014), who proposed models that assist MapReduce jobs to meet the performance deadline with the monetary cost of using the cloud. Selecting suitable schemes to adopt in the algorithms is essential, particularly, considering the technical aspects of the chosen approach. The proposed framework algorithms are designed based on the combination of two main models which are adaptive control and cost decision module in order to meet performance goals and maximize the efficiency of a Hadoop cluster in the cloud.

Lastly, in order to establish the relationship between resource allocation and task scheduling, new scheduling algorithms are proposed. This combination of the resource allocation and task scheduling helps in achieving one of the objectives of this study.

Moreover, beside the main objective to propose new scheduling algorithms, the algorithms are also addressing the limitation of the resource allocation and task scheduling, which was identified in Chapter 3. The following discussion offers an in-depth description of the algorithms.

## 4.1 Multi-objective Scheduling algorithm

In this section, the proposed algorithm is described based on the objectives discussed in Chapter 1, which related to user preferences with respect to completion time and cost. Thus, users are allowed to specify the values for deadline and budget constraints. However, to satisfy these constraints, the information of map and reduce tasks are required during the implementation. Moreover, the information about memory, IO, CPU must be known at the time of execution. However, since the proposed algorithm is designed to work on Hadoop framework deployed on cloud, only the cost of the map and reduce are considered and not the measure of information exchanged from outside. Furthermore, Hadoop comes with storage module, where the datasets are stored in HDFS and accessible at running time for any job.

The multi-objective earliest finish time algorithm has been used to optimize the workflow in the cloud and to iteratively map the workflow tasks onto the resources. Aside from mapping every task onto the resource, the algorithm also maps resources onto tasks to establish a trade-off among the considered objectives. This algorithm is described in the study of Durillo and Prodan (2014), in which a positive value should be returned by the service function if the mappers and reducers are sufficient to complete the tasks for a specific job within the given budget and deadline.

**Algorithm 3: Earliest finish time scheduling**

**Input**

Q: the task queue, where all tasks $T \in J$ and $T \leq t_m + t_r$

N: many nodes in the cluster, where $N \geq Slots$

1. **For** each task t ∈ $T_{m+r}$ **do**
2. Assign $T$ to an available compute $resources$
3. **End for**
4. Repeat
5. **For all** $t_m$ & $t_r$ **do**
6. Assign ready task $t_i$ to any available slots
7. **End for**
8. Dispatch all the mapped tasks
9. Wait for tasks_queue
10. Update the running tasks_queue
11. tasks in the ready list are zero

The pseudocode described in Algorithm 3 presents the multi-objective earliest finish time algorithm, which begins with the required inputs of all the tasks that belong to a particular job in the cluster. The tasks are then split into map and reduce tasks represented by the job. The map tasks will be scheduled first, followed by the reduce tasks. The total of both tasks are scheduled in some nodes, depending on the availability of the slots. Subsequently, the mapping and reducing phases of the algorithm begin by iterating over the list of tasks of the map and reduce tasks sorted according to their order in the queue. The tasks are assigned to available resources in the cluster. Therefore, only trade-off solutions are saved to avoid assigning performance degradation. We only consider the solution belong to non-dominated by other solutions as discussed in chapter 3. All the map tasks, outputs are dispatched, and new inputs for reduce begins

the iteration. Each map task is input to reduce, which is in the queue ready to be executed. The process continues for all tasks on the queue till the tasks listed in the ready queue are completed.

For the basic scheduling policy, the framework has a scheduler that allows the resource allocation decision to be made at the time of the task submission. A scheduling algorithm for the multi-objective heterogeneous earliest finish time algorithm is listed in algorithm 3. First, the tasks are assigned to available resources. The map tasks without parents, which will be on top of the list, are assigned to the first available resources. These available resources should be able to accept new tasks for execution and should not exceed the limit for accepting tasks to new slots. All of the tasks are stored in a queue and updated throughout the scheduling process. Afterward, the tasks that are ready will be assigned by the scheduler to the available cloud resources and slots. The optimum choice for the earliest finish time depends on the number of tasks in the application, the scheduling policy, and the decision model, which are configurable by the user before executing any workflow application in the cloud.

---

**Algorithm 4:** Workload information gathering

---

1. **If** a $t_m$ of job $j$, is finished **then**
2. Update the execution time of a map tasks $t_m$ in the log
3. **If** reduce tasks of job $j$ is finished **then**
4. Update the execution time of reduce $t_r$ in the log

---

Once the map and reduce tasks have completed the execution, the current workload information should be updated, as shown in Algorithm 2. After the map and reduce tasks are completed, the execution time is collected and reported to the “Job Tracker” in the current Hadoop system. The following subsection describes the process of the scheduling algorithm framework.

Gathering more system information can have a significant impact on making better scheduling decisions. Gathering information from the Hadoop system is critical, especially during the scheduling decision. The algorithm works with the first check if all the map tasks have completed in every node in Hadoop cluster. Then, the output of the results of the execution of the map tasks and the time is saved in the logs. Similarly, for the reduce tasks, the completed tasks of the execution and the time are updated in the logs. The importance of gathering the information about the execution is to help analysis the result in terms of the completion time, latency and the utilization of the CPU.

Since the birth of the Hadoop paradigm, the MapReduce programming model has been one of its main components as discussed in the previous section. The traditional implementation of MapReduce has revealed high latencies during execution of Hadoop MapReduce jobs. The submitted jobs are performed based on the steps structure in which the data will be split, Map, shuffle, sort and then reduce. This problem is exacerbated for more complex processing involving statistical MapReduce jobs which require time on the order of minutes, hours, or longer – even with fairly small data volumes. Table 4-1 shows the information collector in Hadoop.

**Table 4-1: The information collector in Hadoop**

| |
|---|
| ./bin/Hadoop fs -put /tmp/logs /var/logs |
| ./bin/hadoop dfs -ls /var/logs |
| public static class LogEntryMapper extends Mapper<Object, Text, Text, IntWritable> { |
| ./bin/hadoop jar loganalyzer.jar loganalyzer /var/logs /var/logs-output |

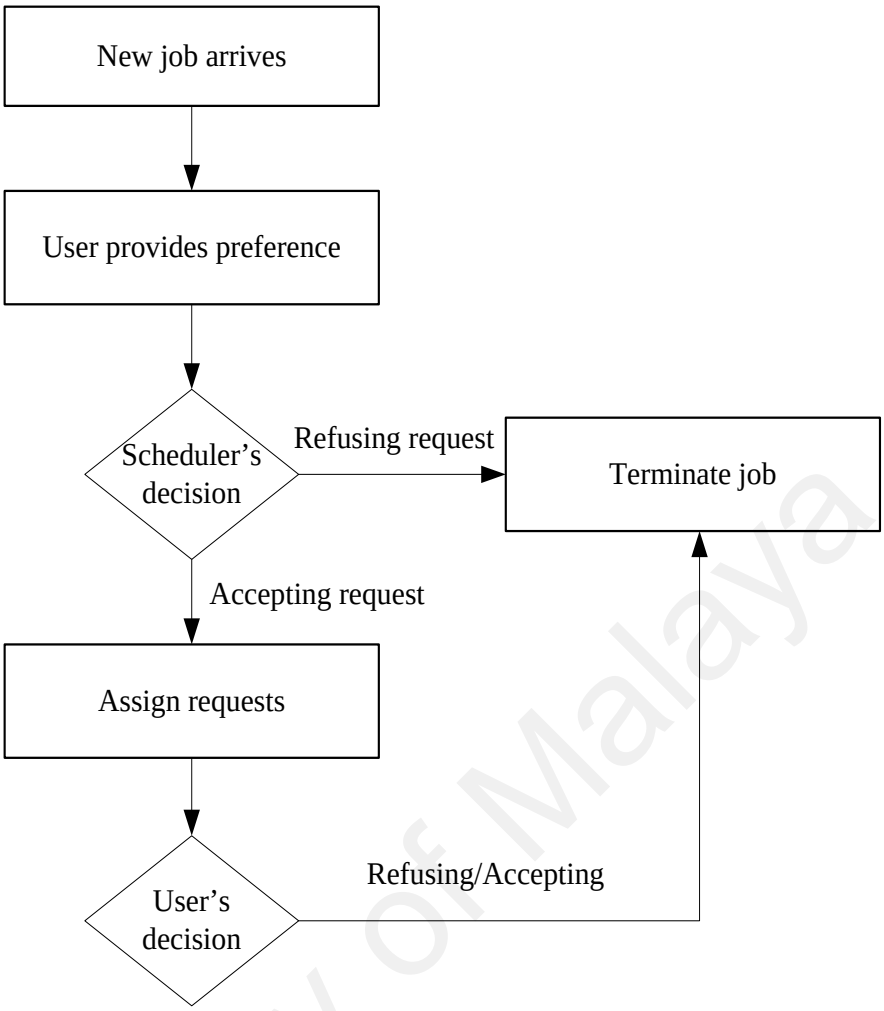



**Figure 4-1: Scheduling process**

For the sake of comparing and describe the performance characteristics of the Hadoop MapReduce in the cloud environment, a series of performance indicators is required. This section mainly focuses on measuring the working capability of the MapReduce jobs, including the measurement of throughput and respond time of each MapReduce job, and the processing duration, CPU utilization of the node. The scheduling process is shown in Figure 4.1. Thus, an experiment is conducted to illustrate the MapReduce scheduling performance under difference scenarios.

## 4.2 Scheduling framework

The proposed framework is considered based on user preferences and the current cloud state, the scheduler confirms a scheduling decision by assigning or refuses the offer, this allows the model to decide what the best fit for the job to run. Since no priority jobs considered in the proposed model, a scheduling policy has been used to share resources between the running jobs. If the request is refused, the job has to be resubmitted. Upon receiving a request, a user can choose to accept or reject it.

Finally, the scheduling process is terminated, if all slots in a cluster have been fully scheduled, the scheduling system for the given slot is closed. The parameters are given by users' requests and assign tasks on resource nodes according to status information and scheduling policy. Scheduling policy is critical for saving energy and satisfying QoS requirements.

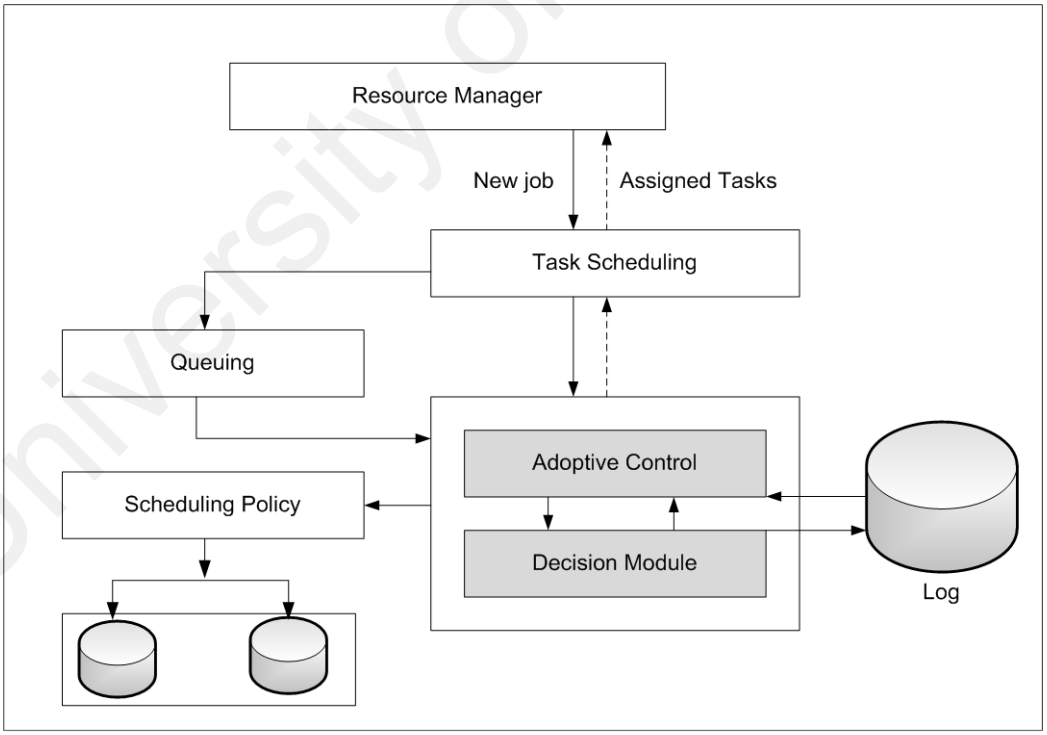



**Figure 4-2: System architecture**

As shows in Figure 4-2 system architecture, various running applications are allocated resources by the scheduler subject to some constraints. Such schedule provides no guarantees on resuming failed tasks because of either application failure or hardware failures. The scheduling components as described below.

*(a) Resource manager* represents a generational shift in the architecture of Apache Hadoop. It utilizes the MapReduce programming framework by default to perform efficient data processing by separating the processing engine and resource management capabilities of MapReduce. Hence, it makes the Hadoop environment highly suitable for operational applications that cannot wait for batch jobs to be completed. This feature simplifies the support of maintaining a multi-tenant environment, managing and monitoring workloads, implementing security controls, cluster utilization and providing high- scalability for Hadoop framework.

The resource manager consists of two interfaces, namely, clients submitting applications and application masters who dynamically negotiate with an access to resources and others toward node managers. Application masters codify their need for resources in terms of one or more resource requests, each of which tracks the number of containers (Vavilapalli et al., 2013).

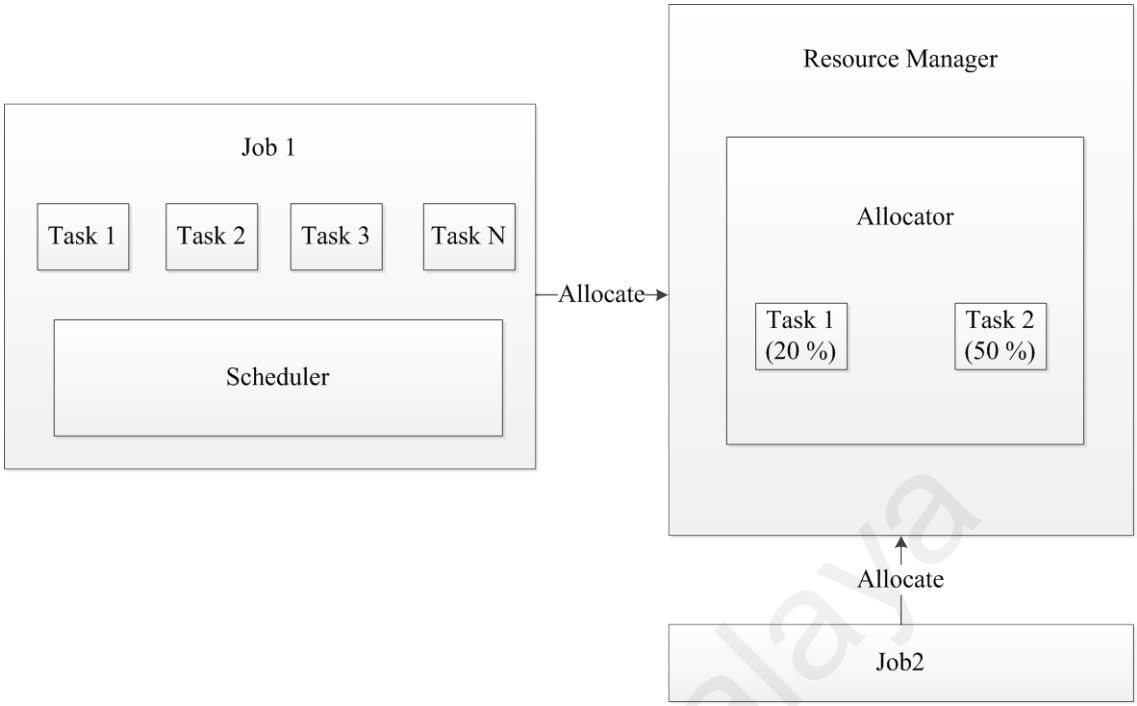



**Figure 4-3: Resource allocation process**

In addition, Yarn uses two steps for resource allocation based on a pluggable solution for dynamic policy loading (Pop & Cristea, 2015) as shows in Figure 4-3. The yarn is responsible for resource allocation, whereas application responsible for dividing the jobs into multiple tasks then schedule them. This makes Yarn more generic while allowing flexibility of scheduling strategies. The resource manager consists of two interfaces, namely, clients submitting applications and application masters who dynamically negotiate with an access to resources and others toward node managers. Application masters codify their need for resources in terms of one or more resource requests, each of which tracks the number of containers.

*(b) Task scheduling:* the resource manager assigns resources to the jobs within the cluster. Each job will have many execution slots allocated by the resource manager. Assigning more slots to a job lead to more resources allocated to the job. The jobs are divided into multiple tasks, one or more tasks are assigned to slots in form of queue. This queue is updated during the task scheduling. The scheduler then assigns these

ready tasks from the queue to resources based on the availability of each resource and slots. The optimum choice for earliest finish time depends on the number of tasks in the application, scheduling policy and decision model, which is configurable by the user before executing any workflow application in the cloud.

*(c) Scheduling policy:* The scheduling policy is used to control the ordering request for different resources. The policy is considered a priority, locality, deadlines, budget, and the system behavior. Resource allocation is done by YARN, and task scheduling is done by the application, which permits the YARN platform to be a generic one while still allowing flexibility of scheduling strategies. The specific policies in YARN are oriented on resource splitting according to schedules provided by the applications. In this way, the YARN can decide on which cluster to be allocated the resource and how much based on their availability and on the configured sharing policy. Altering the configuration of a queue using time, the queue-level time-based policies can be used including allowing jobs to be submitted.

*(d) Adaptive control:* The main objective of the adaptive control is to identify tasks that unable to complete in a given time with the available resources and provides a feedback for the users. The users may adjust their preferences for the tasks and take appropriate steps to meet the availability of resources in the Hadoop cluster. The preferences with regard to nodes and time slots are assumed to be independent of each other. Moreover, adaptive control contains, in addition to a feedback control with adjustable parameters.

## 4.3 Summary

This chapter has provided a framework for multi-objective scheduling algorithm, in order to improve the execution time of tasks in the big data platforms. A framework is designed based on MapReduce framework, where resource manager is responsible for allocating the resources among the Hadoop clusters. The framework has introduced two

main components, which are adaptive control to identify tasks that unable to complete in a given time with the available resources and provides a feedback for the users and the decision module to decide what the best fit for the job to run.

The algorithm works with the first check if all the map tasks have completed in every node in Hadoop cluster. Then, the output of the results of the execution of the map tasks and the time is saved in the logs. Similarly, for the reduce tasks, the completed tasks of the execution and the time are updated in the logs. The importance of gathering the information about the execution is to help analysis the result in terms of the completion time, latency and the utilization of the CPU.

Having established the proposed framework using multi-objective scheduling algorithm strategies and models, the next chapter presents the evaluation of the framework and is followed by the resulting discussion. It is important to understand that the results provide an evaluation and verification of the usefulness and suitability of the framework in improving the execution time.

In conclusion, this chapter highlighted the main point of the research objective in this study by design a new optimization proposal based upon a multi-objective algorithm to minimize time and cost in a heterogeneous cloud environment and offered a detail of the framework.

# CHAPTER 5: EVALUATION OF MULTI-OBJECTIVE SCHEDULING ALGORITHM

The objective of this chapter is to provide performance evaluation method used to evaluate and validate the proposed multi-objective scheduling algorithm. The scheduling algorithm adopts more accurate methods to determine the execution time. The purpose is to improve resource allocation and tasks scheduling in big data processing. Thus, in order to outline the importance of the proposed algorithm as discussed in Chapter 4, this evaluation study is significant. The performance of MapReduce scheduling for big data processing has been investigated by many researchers ranging from job scheduling to an adaptive and on-demand fault, replication and placement to new resource allocation models. However, evaluating of these models have been overly by simplified setting in most MapReduce performance enhancement solutions, which presents significant challenges to the analysis and compare the effectiveness of these solutions (Sangroya et al., 2016).

Having proposed multi-objective scheduling algorithm, it is important in this chapter to design an evaluation procedure in order to provide a verification of its use. This chapter also offers validation using statistical analysis, which aims to validate the results of the proposed algorithm in terms of the performance and compare with the FIFO and Fair schedulers. The evaluation results are validated using linear regression models. First, the chapter provides a description of the benchmarks that used for the evaluation of the proposed algorithm. Second, the simulation environment and the datasets used for the experiment are described in details. Finally, the chapter investigates the performance of the proposed algorithm by comparing it with the most used scheduling algorithms: FIFO and Fair schedulers.

The chapter is organized as follows. Section 5.1 provides benchmark description that is used for the evaluation of big data scheduling algorithms. Section 5.2 highlights the datasets used in the experimental stage. Section 5.3 presents the list of experiment and their procedure, as well as a description of this study. Section 5.4 presents a statistical model include the coefficient of determination. Section 5.5 presents performance evaluation results for the throughput and the execution time. Section 5.6 offers validation of results and the discussion is in Section 5.7. Section 5.8 provides a summary of the chapter.

## 5.1 Benchmark Description

In order to evaluate Hadoop schedulers in a proper way, we assume that various big data applications vary dynamically when running in a real application in cloud computing. This evaluation assumes that users can scale-up or scale-down the workload traces in terms of both data and workload scales according to their requirements. The purpose of using the benchmarks is to evaluate the performance of proposed scheduling algorithm regarding execution time and throughput.

The performance of MapReduce job is measured based upon the execution time (Shankar et al., 2014). Also, many other factors can influence the performance of the MapReduce job significantly, such as many map tasks and reduce tasks, the underlying network, intermediate shuffle data pattern, and the shuffle data size.

The scheduling algorithm performance of virtualized Hadoop cluster deployed on the cloud computing is evaluated. Figure 5-1 shows the process of big data processing using Hadoop cluster. The cluster is configured in a virtualized environment where each cluster node installed in a separate virtual machine. The cluster consists of the master virtual machine running JobTracker, NameNode, and multiple workers virtual machine running TaskTracker and DataNode. When the user places a query, the NameNode

located the DataNode within the cluster. Each DataNode holds a portion of the datasets and constantly communicates with the NameNode to complete a certain task. Once the datasets located within the cluster the TaskTracker begins processing the data using the MapReduce framework mechanism, which start from Map, combine, shuffle and then reduce. The result of the data processing can be stored back in HDFS.

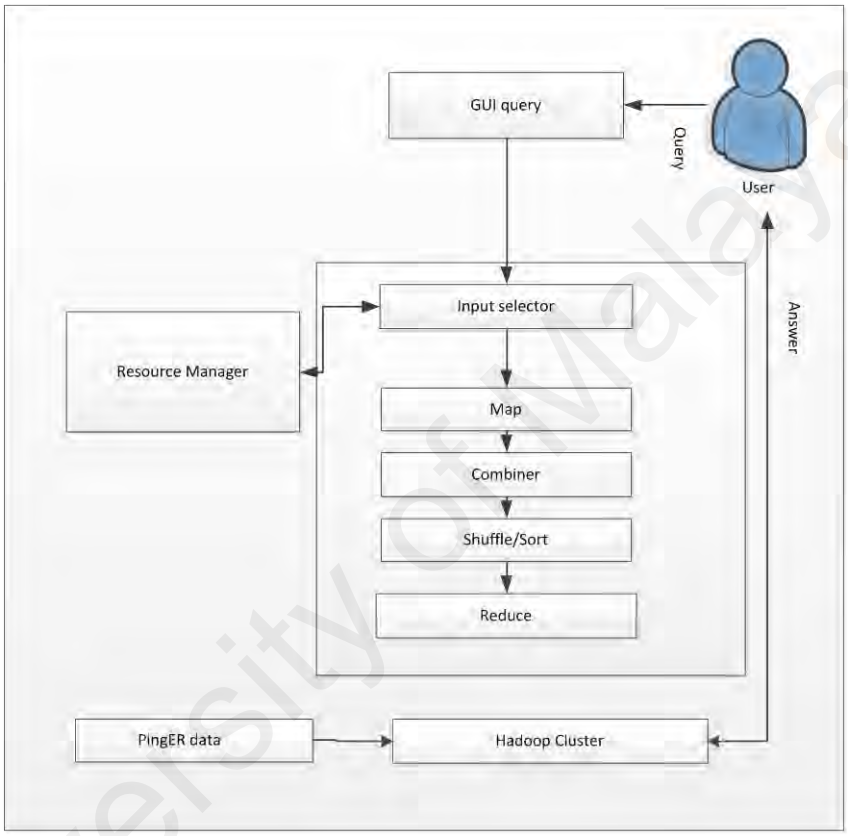



**Figure 5-1: Process of big data processing**

Hadoop, in general, is known to be low CPU usage system. Some internal restrictions imposed by the implementation do not allow boosting the CPU usage to its maximum if a single job running on each node. To address this Hadoop scheduler load simulator is used to obtain high average CPU usage per node, which increases the cluster performance for the non I/O bound workloads.

Table 5.1 shows some of the big state-of-the-art data benchmarks efforts used for evaluating different methods and algorithms of applications. Some of these benchmarks

can be used to evaluate scheduling algorithms in some big data platforms like MapReduce. To cover diverse and representative workloads, the important workloads from four applications are highlighted in order to help in the selection of suitable benchmarks.

**Table 5-1: Comparison of Bi data benchmarking efforts**

| Benchmark | Datasets | Software stacks | Examples | Status |
|---|---|---|---|---|
| Micro (Islam et al., 2014) | Unstructured text data | Apache Hadoop | WordCount and Sort | Open source |
| HiBench (Huang et al., 2011) | Unstructured/Semi-structured data | Apache Hadoop + Hive | Offline analytics | Open source |
| BigDataBench (L. Wang et al., 2014) | Unstructured text, graph data | NoSQL | Online analytics | Open source |

That is being said; Hadoop is used as the basic software stack. For the same big data application, the scale of the simulator running big data applications is mainly decided by the size of data input. Since the evaluation of the algorithms is based on text datasets, the benchmarks used in these experiments are Micro Benchmark which can be tuned depends on the cluster and workload characteristics. For the analysis of the scheduling algorithm, the WordCount and Sort applications are used on a Hadoop cluster deployed in the cloud virtual machines. These applications are commonly employed in a computation of workloads to measure the factors which include CPU usage, throughput and execution time.

WordCount is a program which as its name suggests, is used to count the number of times each word is found in a text. The Hadoop distribution comes with several benchmarks, which are bundled in Hadoop-*examples*.jar. Sort is a synthetic, which also included in Hadoop distribution. It allows generating variable-sized data sets and sorting the generated data. This benchmark has been used by Google and Yahoo to evaluate their MapReduce frameworks and it is considered to be a good measure of the

performance characteristic of the underlying platform. Next section introduces the PingER datasets that are used in the research.

## 5.2 PingER datasets

This evaluation uses datasets that represent different ends of the spectrum in terms of difficulty. The datasets used referred to herein as PingER, is the datasets generated using the point-to-point network.

The PingER data sets generate data about the Internet in order to anticipate the performance of internet links between laboratories, universities and research institutions (Cottrell, 2012). The project started in laboratories for modern High Energy Nuclear and Particle (HENP) Physics. The goal is to help the projects group to monitor performance and assess the feasibility of the computing goals of the future experiments (Cottrell, 2012). The laboratories faced a significant challenge in their wide-area networks. Thus, the HENP networking community used End-to-End performance monitoring infrastructure to closely examine malfunctions across a wide range of networks and connections and chart long-term trends. According to ICFA-SCIC Monitoring Working Group in 2014, the pingER project has over 86 active monitoring nodes in 22 different countries around the world monitoring over 810 remote nodes at over 775 sites in 166 countries that are around 8000 monitor nodes.

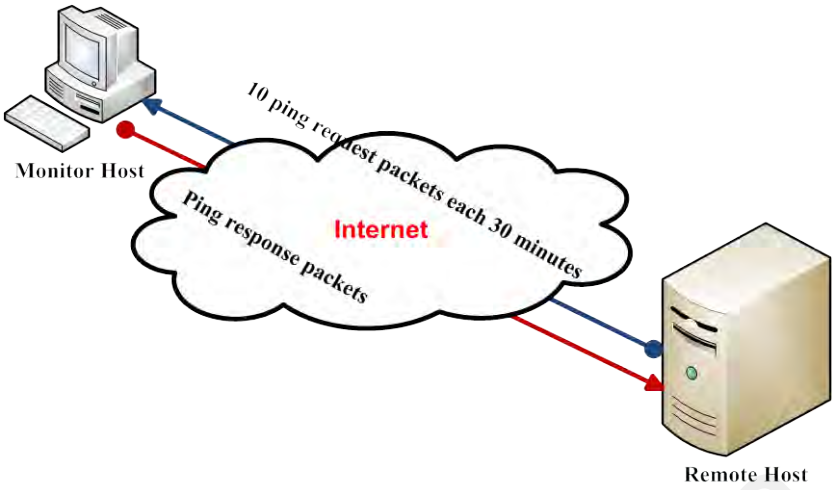



**Figure 5-2: Data collection method: PingER**

As demonstrated in Figure 5-2 each monitoring node sends ping messages at regular intervals to single or group of selected worldwide remote nodes and data is collected at archive site via HyperText Transport Protocol (HTTP) for analysis purpose (Cottrell et al., 2013). This process resulted in a massive amount of data being produced by PingER. Figure 5.3 represent the PingER hourly data compressed for each 100 bytes of pings, which stored in PingER archive and made available via anonymous ftp via ftp://ftp.slac.stanford.edu/users/cottrell (Cottrell, 2012).

Over the past 15 years, PingER project has generated a tremendous amount of data stored in flat CSV files in a form of Linked open data, which have been used to anticipate the performance of internet links between laboratories, universities and research institutions (Cottrell, 2012). Hence, to access such data, Pingtable application (Cottrell et al., 2013) is used to retrieve row data stored in the archive and load it into a normal HTML page.

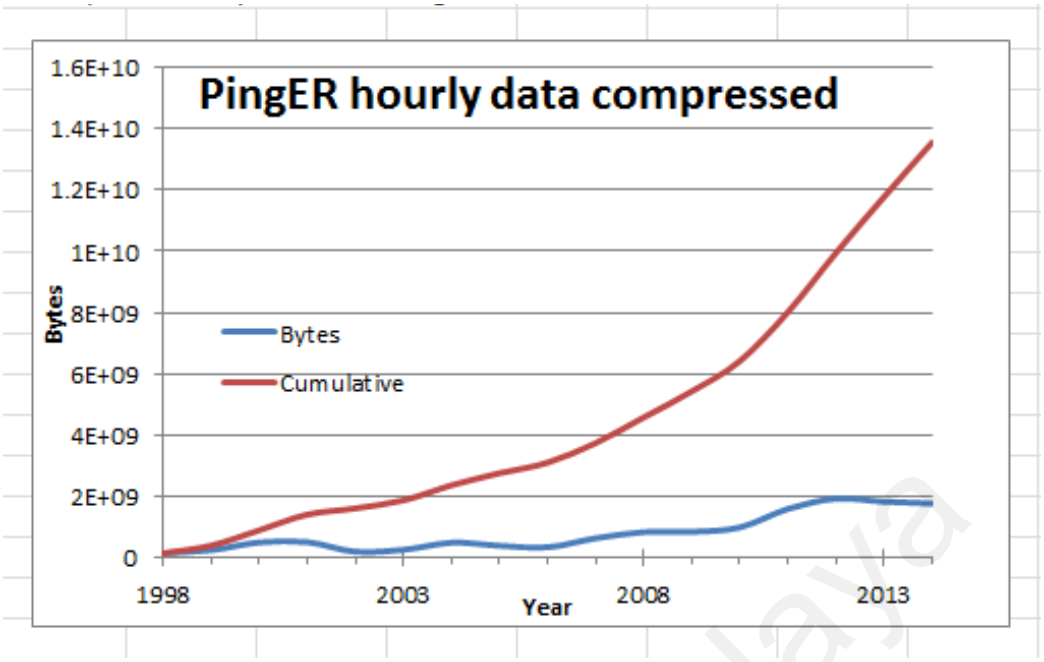



**Figure 5-3: Pinger volume of compressed hourly data for 100Byte pings**

The data sets are stored in the Hadoop distributed file system, which accommodates a large number of PingER files. Moreover, this dataset stored is in many pieces across many nodes in the cloud to facilitate parallel computation.

In this study, we used up to 10 GB of PingER datasets. The reason of using this PingER data is that the data available to the public for research purposes and it is suitable for experiment with WordCount and Sort benchmark applications due to the complexity of the data.

## 5.3 Experimental and procedure description

This section presents a systematic performance evaluation of the proposed algorithm using divers' sets of workloads, including Map, Shuffle, and Reduce workload. In addition, the Hadoop scheduling simulator is used for the evaluation. The scheduling algorithms FIFO and Fair schedulers are compared with the proposed algorithm using Hadoop version 2.6.0. The newest version content YARN as a successor of Hadoop, which offers a new framework for resource management. Thus, based on the performance evaluation, the algorithms implemented within YARN such as Fair

scheduler is working well. Although, YARN still in its early stage of development, therefore it is not fully stable yet for very large-scale execution (Kulkarni & Khandewal, 2014).

The experiments were conducted using the Hadoop cluster with 10 VMs installed on Linux Ubuntu 14.04. One of the VMs runs NameNode and ResourceManager, whereas the other VMs run DataNode and DataManager. Each VM has the following configuration: 2.80 GHz processor, 8 GB main memory, and 1000 GB disk space. Hadoop version 2.6.0 was used for the high-level query. The maximum replication factor "dfs.replication.max" was applied to set the replication limit of data blocks. A benchmark representative set of CPU and IO intensive applications included in the Hadoop distribution, such as WordCount and Sort, for performance analysis was used to efficiently evaluate the MapReduce task scheduling algorithms (Huang et al., 2011). The reason for choosing these two benchmarks is because both are used most often as a baseline benchmarks for MapReduce. Table 5.2 provides an example of MapReduce of the basic settings for the experiment.

**Table 5-2: Example MapReduce Settings**

| Property | Value |
|---|---|
| mapreduce.map.memory.mb | 1536 |
| mapreduce.reduce.memory.mb | 2560 |
| mapreduce.map.java.opts | -Xmx1024m |
| mapreduce.reduce.java.opts | -Xmx2048m |
| yarn.scheduler.minimum-allocation-mb | 512 |
| yarn.scheduler.maximum-allocation-mb | 4096 |

## 5.4 Statistical Models

Scheduling jobs on MapReduce have conflicting requirements and goals to optimize due to the difficulty of predicting a new incoming job's behavior and its completion time. Given the possible presence of the cloud heterogeneity, deterministic modeling of scheduling jobs on MapReduce is difficult. As such, proven statistical scheduling

models are required to represent the real life scenario of the data analysis regarding validation. In order to perform data analysis, the statistical model is used to identify various forms of inferences from the benchmarking experiments. Hence, the purpose of the statistical model is to construct a model that approximates the true structure as accurately as possible through the use of available data. This statistical model is based on two factors which are throughput and execution time.

In realistic settings, a lot of factors can impact the performance of the Hadoop scheduling mechanism, in the following, critical factors which are closely correlated with the system performance are elaborated. The purpose of analyzing these factors is summarized in Table 5-3.

**Table 5-3: Summary of the purpose of analysis throughput and execution time**

| **Factors** | **Purpose of analysis** |
|---|---|
| Throughput | Compare the average task throughput of FIFO, Fair scheduler, and the proposed scheduling algorithm under different data sizes in order to verify that the proposed algorithm can foster average task throughput. |
| Execution time | Compare the performance of execution time of FIFO, Fair scheduler; confirms that to the proposed scheduling algorithm can enhance the execution time performance of MapReduce in the cloud. |

For the statistical analysis model, we used alteration method to produce numerical values, which content the execution time and throughput for each workload. These workloads are independent in nature. Moreover, in order to identify the correlation between the workloads and the execution time and the throughput of the experimental output, we use a regression model to determine the value of the execution time and throughput from the values of one or more variables, so that the dependent variable can be predicted. That is being said; the dependent variable is execution time and throughput that mainly depends on the independent variables.

The execution time and throughput predictions can be made from the set of the data collected from the experiments. A method to predict the execution time and throughput of each task on a Hadoop cluster running on the cloud is required for an efficient mapping of the task to the CPU and the cost to complete each task within a deadline. The workload of the cluster offers knowledge of the amount of work at the CPU at any given time. The load is obtained as a numerical value taking into consideration the amount of time to complete the tasks and the CPU time and an average number of jobs in the run queue.

Linear regression is considered due to independent variables such as size and the number of tasks running on the virtual cluster. Thus, for the execution time and throughput considering the size and the numbers of tasks, linear regressions are employed in order to find the significance of execution time and throughput. The regression line is depicted by the relationship between two variables in a straight line in order to predict the dependent variable from independent variables. Equation 5-1 shows an example of the mathematical relationship between two variables $x - axis$ and $y - axis$ as the linear regression relationship. The parameters $b$ and $m$ represented the coefficients of the regression model which are fixed values.

$$\boldsymbol{y = m * x + b} \qquad \textbf{5-1}$$

For the several independent variable relationships, linear regression is used to predict variables and dependent variables. The main goal of linear regression is to offer a probabilities model that relates a dependent variable to one independent variable. The dependent variable is represented by the time to execute the tasks, and the independent variables are the amount of the data that is being executed by Hadoop.

For validation the result of the regression model, we used a numerical methods strategy. Consequently, the generated data from the experiments are used to validate the result of

the model. As such, the data split randomly into two: first, to develop the model and the second, is to measure predictive accuracy to identify the correlations between the variables.

### 5.4.1 Coefficient of determination

Basically, the measurement of predicted output corresponds to the actual output accuracy is given by the coefficient of determination. It is represented by $R^2$, which magnitude varies between zero and one. The value with high number indicates the high accuracy of the prediction. As mentioned earlier in the execution time and throughput are defined by the linear regression models, the values of $m$ and $b$ illustrated in equation 5-1 are to be determined. These parameters cannot be directly measured and have to be estimated. The estimation is performed by collecting a set of data for example, CPU, total respond time and runtime. Such estimation of the parameters is done through drawing an estimated regression line through the actual data. For any data point there exists a residual or error of fit represented as $r$ given by:

$$\boldsymbol{r_i = \alpha_i + \beta_i} \qquad \textbf{5-2}$$

Where $\alpha_i$the actual execution time and $\beta_i$ is the predicted execution time. The $\alpha_i$ can be less than $\beta_i$ if the observation shows below the estimated regression line. Using the method of ordinary least squares, the criterion for an accurate prediction is that the sum of the squared residuals be as small as possible, which unexplained variation is represented as:

$$min \sum_{i=1}^{n} r_i^2 \qquad \textbf{5-3}$$

For any observation $I$, the execution time $\alpha_i$ deviates from the mean of the original execution times. This deviation is equal to the deviation of the predicted value from the

mean plus the residual. The regression explains the predicted value, while the residual remains unexplained.

The proportion of the total variation in the execution time and throughput is measured by the coefficient of determination $R^2$ that is explained by the independent variables, which are the number of tasks and the data size, as such the $R^2$ can be defined as:

$$1 - \sum_{i=1}^{n} r_i^2 = \frac{unexplained\ variation}{total\ variation} \quad \textbf{5-4}$$

The assessment of the magnitude of $R^2$ depends on the nature of the process being analyzed. A $R^2$ value of 0.05 or higher is said to be a relatively good predication.

### 5.4.2 Execution time

In order to perform the mapping of tasks to resources for execution, a matrix is required. There are two ways to obtain the result of execution time, which are the measurement of the tasks on particular resources on a Hadoop cluster. Meanwhile, statistical analysis defines many primitive types. The analysis, determine the performances of the nodes in the cluster based on the execution time estimation. As described in section 5.4, the regression model can be used to predict the completion time of all the tasks executed by MapReduce. The algorithm generates a scheduling matrix by making use of the predicted execution times to complete the tasks at near optimal solutions, and the tasks executed according to the scheduling time given by the algorithm. For the tasks to be successively executed, the user's preference regarding the budget, the deadline must be captured. A decision module is necessary for the allocation of resources. Several experiments were performed on Hadoop deployed on cloud environment to evaluate the performance of the improved scheduling model. Frist, we evaluated the performance of the MapReduce scheduler in the cloud based on, throughput, the execution time, CPU response time, total response time by Map, and total response time by Reduce metrics.

As such, the completion time is predicted during the execution time. There are two types of time we considered in this study also which are the response time of each task and the total response time of CPU.

*Total CPU respond time:* CPU is an important resource element during data processing, especially in the Map and Reduce phases. We observed that intensive computation in the map and reduced code can increase the CPU utilization as shown in Figure 5.4. Usually, CPU resources consume more by computationally intensive jobs as compare to other resources like bandwidth and I/O throughput. Consider the CPU utilization as given in Equation 5.3.

$$utilization = \frac{requirements}{capacity} \quad \textbf{5-3}$$

Increase CPU utilization can create a bottleneck problem. However, most of the problems come from insufficient CPU utilization that is because, with current hardware, the CPU can to process a large amount of data faster than any other resources such as storage I/O and network. Thus, in many cases, when data is flowing through the MapReduce pipeline, the CPU is waiting for other resources to feed in data before it can proceed to the actual computation.

That is being said. First, the proposed algorithm is measured regarding CPU utilization in order to evaluate the performance of CPU scheduling. This is because running Hadoop in a shared environment is maybe affected by the limits of available resources. The job completion and resource utilization are part of the user's requirements, in a sense that the user specifies the values in the scheduler configuration file for the for the deadline and the budget

A set of tasks run in a Hadoop cluster with ten nodes is measured based on the CPU utilization ranging from 0 to 100%. The x-axis represents the CPU utilization of the

cluster nodes before each task is started; we measured task execution time by the average completion time when running the same type of task on a node in a Hadoop cluster.

The result from Table 5-4 shows the toal response time of each map task is higher compared to the time spent on the reducer and CPU that is because, during the scheduling process, the map tasks require more computational to finish the tasks. Moreover, the number of tasks given by the scheduler is more compared the Reduce phase.

Figure 5-4 illustrates that a number of resources consumed by different applications run on the virtual Hadoop cluster. The execution time of the application is increased significantly for some tasks due to CPU intensive job. This shows that WordCount has high CPU utilization regarding IO compare with the other applications. As such, for both applications, better performance can be achieved if carefully allocate the resources in Hadoop cluster that offer an efficient performance.

**Table 5-4: The job profile of total response time in the cloud**

| **No** | **Total respond time (s)** | | |
|---|---|---|---|
| | CPU | Map | Reduce |
| 1 | 485030 | 132919 | 18122 |
| 2 | 47090 | 102730 | 5878 |
| 3 | 53440 | 117224 | 11323 |
| 4 | 29030 | 124923 | 21761 |
| 5 | 46280 | 154398 | 25451 |

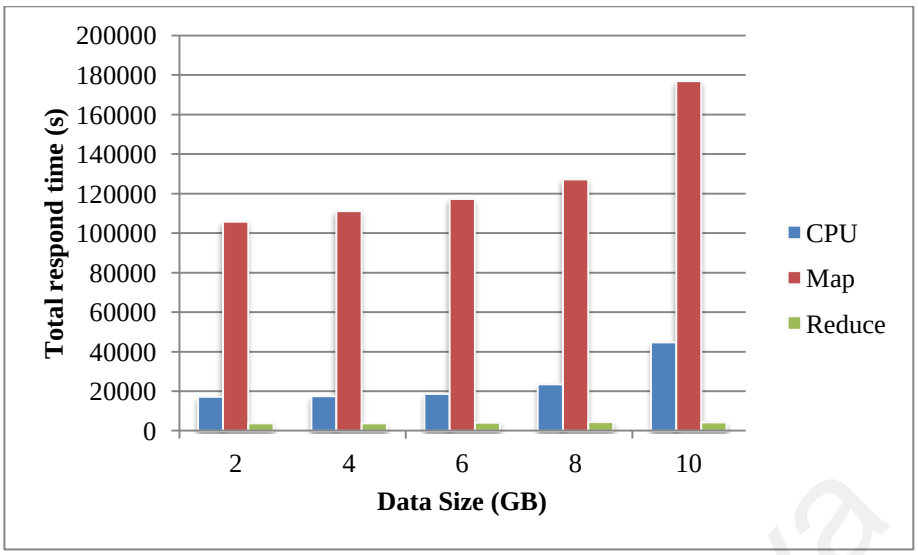



**Figure 5-4: Total respond time**

We can also observe that the completion of the task of each phase is changed when the input size varies. As we have shown in the Figure 5-5 the completion time in the reduce phase stays fairly constant, whereas the others increase as the input data becomes larger, especially in Map phases where it takes most of the time to complete the tasks. A heterogeneous cluster with the numbers of VMs in cloud computing is created. Performance evaluation results

Results of performance evaluation generated using two selected benchmarks: WordCount and Sort, for each one the result is collected based on the configured algorithm. The same data set size is specified for all algorithms in order to avoid bias during the evaluation. Moreover, the number of tasks executed in each job is determined by the input data set during the processing, which is handled by the MapReduce framework. These inputs are split into independent chunks which processed by the map tasks in a completely parallel manner. The performance of the proposed work compared with the default Hadoop scheduler and the Fair scheduler algorithms. Both Sort and WordCount benchmarks were run on the Hadoop Scheduler Load Simulator on the MYRAN cloud (see Appendix B) with 10 VMs to process input PingER data sets of different sizes varying from 2GB to 10 GB. The benchmark applications are executed

multiple times each using an average of the execution and throughput. In following, we present an explanation of statistical models of throughput and execution time. We have collected the data not only based on the size but also the number of tasks that is given by ResourceManager to the DataManager to be executed. The next section we discuss the performance result of both the throughput and execution time. For each one, we conducted a regression analysis to predict the performance of the results.

### 5.4.3 Throughput

This section presents the result of throughput when executing PingER datasets on MapReduce using a WordCount and Sort applications. Statistical analysis of section 5.4 is applied to predict and evaluate the performance of the result. In most cases, higher throughput shows that the system can complete more tasks in a given deadline, and the resources can be utilized sufficiently. Many charts and tables are used to demonstrate the finding.

Table 5-5 provides the comparison between the three scheduling algorithms: FIFO, Fair, and the proposed algorithm. As shown from the table the proposed algorithm has high throughput percentage to the overall average of (78.31) compare to the Fair scheduler (69.31) and FIFO (62.18). The default Hadoop scheduler Throughput of FIFO can be shown to be low compared with other scheduling algorithms like Fair and the proposed algorithm.

**Table 5-5: Comparison of the proposed algorithm, FIFO, and Fair schedule regarding the throughput**

| Data Size (GB) | Proposed algorithm | Fair Scheduler | FIFO |
|---|---|---|---|
| 2 | 46.280 | 47.090 | 42.045 |
| 4 | 53.440 | 44.519 | 51.670 |
| 6 | 69.309 | 54.030 | 50.570 |
| 8 | 97.579 | 88.613 | 67.398 |
| 10 | 124.923 | 111.398 | 99.230 |
| Total | 391.531 | 345.650 | 310.913 |

The throughput of the proposed algorithm is predicted from executing MapReduce jobs using linear regression over different data sizes. This predicted throughput is analyzed for all algorithms using same data sizes in order to find the correction between them.

Table 5.6 shows predefined fit function using linear regression for throughput. It presents the coefficient of determination; $R^2$ "R squared" value of the results gathered from tasks processed in MapReduce that is deployed on the cloud. An $R^2$ 0.95 and higher are generally considered a good prediction. Considering that even at higher load the value of $R^2$ is consistently closer to 1 and in most cases above 0.95 this process that the regression model is accurate in its prediction (see equation 5-1).

**Table 5-6: predefined fit function using multiple regression regarding throughput**

| **Attribute** | **Value** |
|---|---|
| R2 | 0.95 |
| Mean squared error | 8.4 |
| Fit parameters | b=17.9 +/- 8.8<br>m=10.1 +/- 1.3 |
| Unformatted fit parameters | b=17.878700000000023<br>m=10.071249999999997 |

The data presented in Table 5-5 were analyzed using ANOVA, and the results are reported in Table 5-7, 5-8, 5-9, 5-10, 5-11,5-12.

Table 5-7 shows that the significant is (0.008). The result suggests that the regression procedure, which estimated by the model is significant at the level of 0.05. Thus, at least one of the regression coefficients is different from zero. The output of the analysis indicates a statistically significant difference F (2, 2) = 124.69, p < 0.008 = 124.692,

therefore, the throughput accuracy of the proposed algorithm is significantly better than the FIFO algorithm.

Moreover, Table 5-7 shows the temporal data related to throughput using regression model. The table presents mean coefficients, standard errors, and throughput of the two algorithms (Proposed algorithm and FIFO) with 95% confidence interval. The coefficients here means the response variable when the predictor variable changed, whereas holding other predictors in the model constant. The higher values of one variable tend to be related to lower values for the other variable. A positive correction indicates that higher values of one variable tend to be related to higher values of the other variable. As shown in the table, there is a significant high throughput when performing tasks execution using proposed algorithm on the cloud. The performance has a direct correlation with latency, meaning that the system can produce low latency using the proposed algorithm.

Thus, the performance accuracy of the proposed algorithm regarding throughput significantly better than that of the FIFO. The accuracy of the results is most likely because the datasets used are of approximately equal size.

**Table 5-7: Analysis of multiple regression for throughput time execution of proposed scheduling algorithm and FIFO scheduler**

| | df | SS | MS | F | Significance F |
|---|---|---|---|---|---|
| Regression | 2 | 39.682 | 19.841 | 124.692 | 0.008 |
| Residual | 2 | 0.318 | 0.159 | 0.000 | 0.000 |
| Total | 4 | 40.000 | 20.000 | 124.692 | 0.008 |

While, as shown in Table 5-8 the p-value for the estimated coefficients of x-axis and y-axis, are respectively 0.028 and 0.048, indicating that they are significantly related at the level of 0.05. Therefore, our proposed algorithm framework demonstrates high throughput when executing big data compare to the FIFO scheduling algorithm. The t-

value and p-value give an indication of the impact and the significance of each independent variable.

**Table 5-8: t-test significance of difference between the proposed algorithm and FIFO with respect to the throughput**

| | Coefficients | Standard Error | T-value | P-value |
|---|---|---|---|---|
| Intercept | -6.019 | 1.007 | -5.977 | 0.027 |
| Proposed algorithm | 0.752 | 0.128 | 5.883 | 0.028 |
| FIFO | -0.540 | 0.122 | -4.420 | 0.048 |

Table 5-8 reveals that p-value is less than 0.05 for proposed algorithm and FIFO. Since the P-value is less than 0.05, the result is accepted at the 5 percent level of significance. Therefore, it is concluded that there no significant difference between proposed algorithm and the FIFO when small execution size of data. Furthermore, Table 5-8 shows the coefficients that give intercept and the regression coefficients for each explanatory variable. The intercept value -6.019 represents the contestant which predicted throughput time. The finding of the study also reveals that the perception of the proposed algorithm and FIFO with respect to throughput is different. A coefficient score of proposed algorithm is 75.2.

**Table 5-9: Analysis of multiple regression for throughput time execution of proposed scheduling algorithm and Fair scheduler**

| | df | SS | MS | F | Significance F |
|---|---|---|---|---|---|
| Regression | 2 | 39.430 | 19.715 | 69.190 | 0.014 |
| Residual | 2 | 0.570 | 0.285 | - | - |
| Total | 4 | 40.000 | 20.000 | - 69.190 | 0.014 |

The throughput of the proposed algorithm and fair analysis of multiple regressions given in Table 5-9 yielded an F value of 69.190. When compared to the table value, the result is highly significant.

**Table 5-10: t-test significance of difference between the proposed algorithm and Fair with respect to the throughput**

| | Coefficients | Standard Error | T-value | P-value |
|---|---|---|---|---|
| Intercept | -6.784 | 1.592 | -4.260 | 0.051 |
| Proposed algorithm | 0.742 | 0.175 | 4.235 | 0.051 |
| Fair | -0.503 | 0.159 | -3.166 | 0.087 |

Table 5-10 shows the coefficients that give intercept and the regression coefficients for each explanatory variable. The coefficient value of the proposed algorithm 0.742 represents the contestant which predicted throughput time in comparison with the Fair. Therefore, our proposed algorithm framework demonstrates high throughput when executing big data compare to the Fair scheduling algorithm.

**Table 5-11: Analysis of multiple regression for throughput time execution of proposed scheduling algorithm and Fair scheduler**

| | df | SS | MS | F | Significance F |
|---|---|---|---|---|---|
| Regression | 3 | 39.728 | 19.243 | 48.627 | 0.105 |
| Residual | 1 | 0.272 | 00.757 | - | - |
| Total | 4 | 40.000 | 20.000 | 48.627 | 0.105 |

The summary analysis of linear regressions for the three algorithms sections is listed in Table 5-11. A closer inspection of Table 5-11 and 5-12 revealed that the summary of linear regression for predicting scheduling algorithm in MapReduce jobs using the cloud. This table contained information obtains from the analysis of linear regression for all possible combinations of the three selected predictor variables. Moreover, included in the table were computed F values, standard errors of estimated multiple correlation coefficients, F and the significance of F. The data in this table were then used to select the best possible combination of the independent variable for predicting the performance of the scheduling algorithm in MapReduce. The regression equation is used to predict the dependent variable shown in (5-2). The result revealed that multiple correlations of 0.737 and standard error of 0.171 were recorded from three variable analysis. The multiple correlations indicated that the three-variable models accounted

for over 50 percent of the variance involved. The F ratio that was 48.627 was highly significant beyond the one percent significant level. From the data, the results indicated that the prediction of proposed scheduling algorithm improves the performance of executing tasks in MapReduce using cloud computing is possible.

**Table 5-12: t-test significance of difference between the proposed algorithm and FIFO and Fair on the throughput**

| | **Coefficients** | **Standard Error** | **T-value** | **P-value** |
|---|---|---|---|---|
| Intercept | -5.366 | 2.065 | -2.598 | 0.234 |
| Proposed algorithm | **0.737** | 0.171 | 4.301 | 0.145 |
| FIFO | -0.879 | 0.841 | -1.045 | 0.486 |
| Fair | 0.335 | 0.816 | 0.411 | 0.752 |

### 5.4.4 Analysis of the results regarding throughput

The correlation analysis of the proposed algorithm, FIFO, and the Fair identical dependent variables reveals 0.145 significant correction counts for proposed algorithm, 0.486 significant corrections for FIFO and 0.752 counts for the Fair scheduler. From an independent variable set perspective, we observed the proposed algorithm set dominates the significant correction count 0.737.

Figure 5-5 and 5-6 present the throughput of different data sizes to be processed by the MapReduce framework in a cloud computing environment using the WordCount and Sort benchmarks, respectively. This processing of datasets is scheduled by different algorithms, that is, FIFO scheduler, Fair scheduler, and the proposed scheduling algorithm. Data size affects the type of scheduler required to execute the tasks at a targeted performance level. This metric significantly influences task scheduling, where the execution time of each task has to be minimized considering the heterogeneity of the cluster.

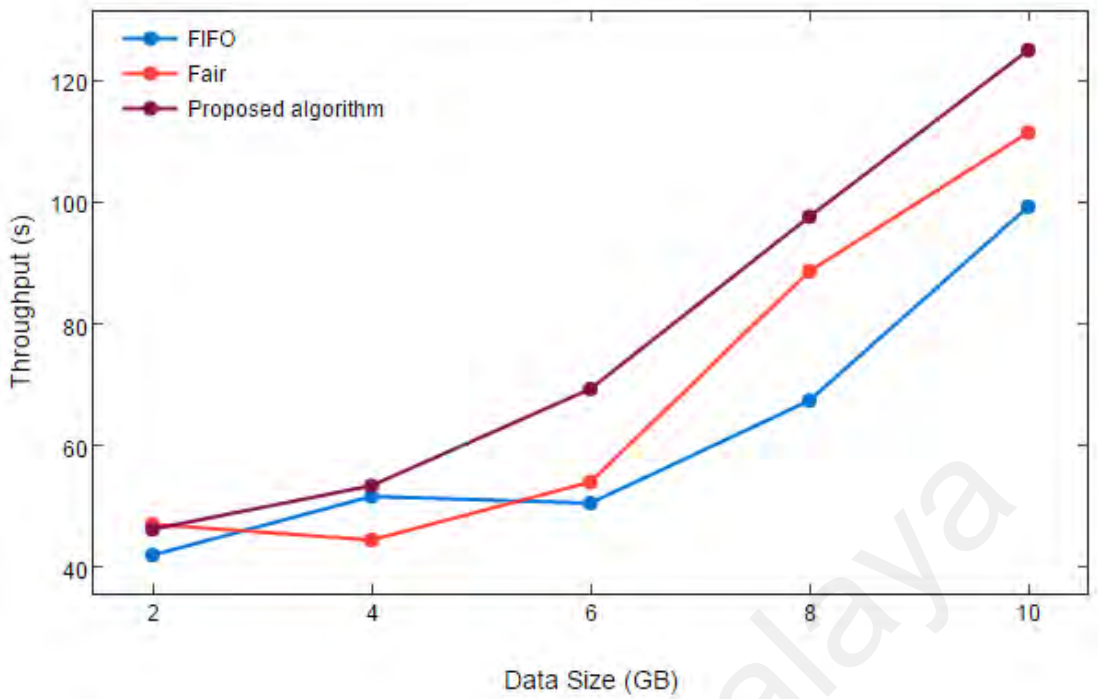



**Figure 5-5: Throughput using WordCount benchmark**

As can be seen from Figure 5-5 shows that the proposed algorithm can provide higher throughput compared with the other scheduling algorithms, namely, FIFO and Fair. The main purpose of achieving high throughput is to reduce the processing time of the workload, particularly when a large amount of data is involved. The resource utilization rate is the reflection of system throughput, which is the useful computation cost over the total cost, including the overhead for starting up the cluster. Many cloud service providers offer hour-based or minute-based charges to users who are availing of computing service on the cloud to reduce the unnecessary CPU cycles spent on overhead, which may consume a large amount of resources to be allocated elsewhere to meet the demands of users (Armbrust et al., 2010).

Figure 5-6 presents the number of allocated data inputs in the cluster to test the proposed algorithm. The experiment conducted using the WordCount benchmark is similar for FIFO, Fair, and the proposed scheduling algorithm.

.

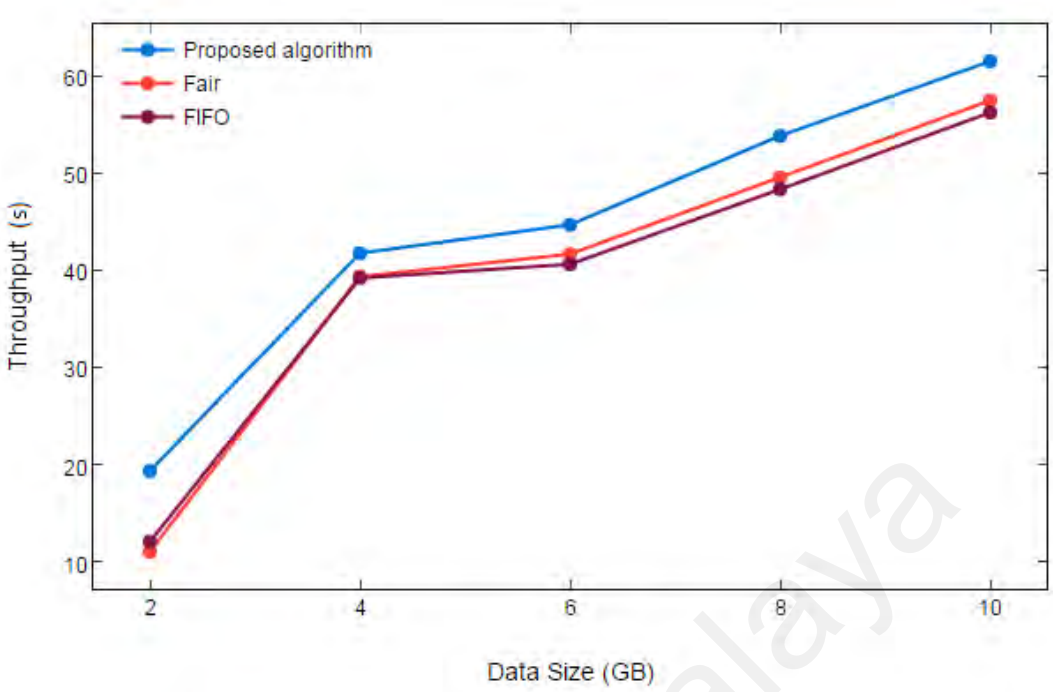



**Figure 5-6: Throughput using Sort benchmark**

Figure 5-6 presents the throughput obtained by executing datasets with the same amount of data using the Sort benchmark. The tested result of the three scheduling algorithms also used a dataset with the same size as that used for the WordCount benchmark. Figure 4 shows that the proposed algorithm has high throughput compared with FIFO and Fair schedulers. However, a simple technique to achieve good performance in the FIFO and Fair algorithms is to assign an available slot to the pool with the least amount of running tasks (Hadoop, 2009). The overall throughputs are insignificantly different under FIFO and Fair.

As shown in Figure 5-5 and 5-6, show that the amount of resources consumed by each node increases as throughput time becomes longer during the execution. Thus, the task in Hadoop scheduling should be matched carefully to the VM in the cloud environment to achieve good performance. In this manner, the system can effectively use the resources to improve the progress of executing the tasks in the Hadoop cluster.

### 5.4.5 Execution time

This section presents the results of the execution time of the MapReduce job in the cloud using the WordCount and Sort benchmarks. The design of the experiment is based on the MapReduce framework running on the cloud. The execution of the MapReduce job depends on the scheduling algorithms deployed for each experiment, which includes the proposed algorithm, FIFO, and Fair. Data related to execution time are collected using benchmarking in this section. Several tables and charts used to demonstrate the findings. Table 5.13 shows predefined fit function using linear regression for throughput. It presents the $R^2$ value of the tasks being processed in MapReduce. An R2, 0.90 and higher are generally considered a good prediction. Considering that even at higher load the value of $R^2$ is consistently closer to 1 and in most cases above 0.90 this process that the regression model is accurate in its prediction (see eq 5-1)

**Table 5-13: predefined fit function using multiple regressions regarding execution time**

| **Attribute** | **Value** |
|---|---|
| $R^2$ | 0.90 |
| Mean squared error | 4.6e+3 |
| Fit parameters | b=144.3766 +/- 1.7e+3<br>m=19.3 +/- 2.1 |
| Unformatted fit parameters | b=144.37656306548365<br>m=19.303353662617386 |

Table 5-13 shows the historical data related to throughput using regression model. The table presents mean coefficients, standard errors, and execution time of the two algorithms (Proposed algorithm and FIFO) with 90% confidence interval. The coefficients here represent the mean change in the response variable for one algorithm

of change in the predictor variable, whereas holding other predictors in the model constant. The higher values of one variable tend to be related to lower values for the other variable. A positive correction indicates that higher values of one variable tend to be related to higher values of the other variable. As shown in the table, there is significant in execution time when performing tasks execution using proposed algorithms on the cloud. The performance has a direct correlation with completion time and cost of using the cloud, meaning that the system can reduce the execution time using the proposed algorithm.

**Table 5-14: Analysis of multiple regression for execution time execution of proposed scheduling algorithm and FIFO scheduler**

| | **df** | **SS** | **MS** | **F** | **Significance F** |
|---|---|---|---|---|---|
| Regression | 2 | 4130753.341 | 2065376.670 | 35.923 | 0.000 |
| Residual | 8 | 459951.205 | 57493.901 | - | - |
| Total | 10 | 4590704.545 | 2122870.571 | - | - |

Table 5-14 shows that the significant F is (0.000), which suggests that the regression procedure estimated by the model is significant at the level of 0.05. The minimum of the coefficients is different from zero. $F(2, 8) = 35.923$, $p < 0.000$. As shown in Table 5-14 the estimated coefficients of the p-value of x-axis and y-axis, are respectively 0.889 and 0.998, indicating that they are significantly related at the level of 0.05. The t-value and p-value give an indication of the impact and the significance of each independent variable. The T-value and P-value represent a significant difference between mean proposed algorithm and FIFO execution time values. The positive T-value of the proposed algorithm (0.145) and FIFO (-0.002) demonstrate that the execution time of the proposed algorithm takes less time than the FIFO. Thus, the completion time and cost of our proposed algorithm is more significant compared to FIFO.

**Table 5-15: t-test significance of difference between the proposed algorithm and FIFO on the execution time**

| | **Coefficients** | **Standard Error** | **T-value** | **P-value** |
|---|---|---|---|---|
| Intercept | 133.475 | 92.478 | 1.443 | 0.187 |

| | | | | |
|---|---|---|---|---|
| Proposed Algorithm | 0.053 | 0.364 | 0.145 | 0.889 |
| FIFO | -0.001 | 0.276 | -0.002 | 0.998 |

Table 5-15 shows the coefficients that give intercept and the regression coefficients for each explanatory variable. The coefficient value of the proposed algorithm 0.742 represents the contestant, which predicted execution time in comparison with the Fair. Therefore, our proposed algorithm framework demonstrates execution time when executing big data compare to the FIFO scheduling algorithm.

**Table 5-16: Analysis of multiple regression for throughput time execution of proposed scheduling algorithm and Fair scheduler**

| | **df** | **SS** | **MS** | **F** | **Significance F** |
|---|---|---|---|---|---|
| Regression | 2 | 4133289.341 | 2066644.671 | 36.145 | 0.000 |
| Residual | 8 | 457415.204 | 57176.901 | - | - |
| Total | 10 | 4590704.545 | 2123821.572 | 36.145 | 0.000 |

Table 5-16 shows that the significant is (0.000), which suggests that the regression procedure estimated by the model is significant at the level of 0.05. Therefore, at minimum one of the coefficients is different from zero. $F(2, 8) = 36.145$, $p < 0.000$. As shown in Table 5-16 the p-value for the estimated coefficients of x-axis and y-axis, are respectively 0.891 and 0.838, indicating that they are significantly related at the level of 0.05. The t-value and p-value give an indication of the impact and the significance of each independent variable. The T-value and p-value give an indication of the impact and the significance of each independent variable. Therefore, the T-value and P-value represent a significant difference between mean proposed algorithm and Fair execution time values. Negative T-value of the proposed algorithm (-0.141) demonstrate that the execution time of the proposed algorithm takes more time than the Fair at some point. Thus, the completion time and cost of our proposed algorithm is less significant compared to Fair at some point. However, the value can change as shown the P-value of the proposed algorithm (0.891) is positive.

**Table 5-17: t-test significant of difference between the proposed algorithm and Fair with respect to the execution time**

| | Coefficients | Standard Error | T-value | P-value |
|---|---|---|---|---|
| Intercept | 139.369 | 90.218 | 1.545 | 0.161 |
| Proposed Algorithm | -0.105 | 0.745 | -0.141 | 0.891 |
| Fair | 0.133 | 0.631 | 0.211 | 0.838 |

Table 5-17 shows the coefficients that give intercept and the regression coefficients for each explanatory variable. The coefficient value of the proposed algorithm -0.105 represents the contestant which predicted execution time in comparison with the Fair scheduler. Therefore, our proposed algorithm framework demonstrates execution time when executing big data compare to the Fair scheduling algorithm.

**Table 5-18: Analysis of multiple regression of execution time execution of proposed scheduling algorithm, FIFO, and Fair schedulers**

| | df | SS | MS | F | Significance F |
|---|---|---|---|---|---|
| Regression | 3 | 4137223.409 | 1379074.470 | 21.288 | 0.001 |
| Residual | 7 | 453481.136 | 64783.019 | - | - |
| Total | 10 | 4590704.545 | 1443857.489 | - | - |

Table 5-18 shows that the significant is (0.001). This implies that the model estimated by the regression procedure is significant at the level of 0.05. Thus, at least one of the regression coefficients is different from zero. $F(7, 10) = 21.288$, $p < 0.000$. As shown in Table 5-18 the P-value for the estimated coefficients of x and y, are respectively 0.788, 0.761 and 0.812, indicating that they are significantly related at the level of 0.05. The T-value and p-value give an indication of the impact and the significance of each independent variable.

**Table 5-19: t-test significance of difference between the proposed algorithm, FIFO, and Fair with respect to the execution time**

| | Coefficients | Standard Error | T-value | P-value |
|---|---|---|---|---|
| Intercept | 134.310 | 98.201 | 1.368 | 0.214 |
| Proposed Algorithm | 0.505 | 1.807 | 0.280 | 0.788 |
| Fair | 0.342 | 1.081 | 0.316 | 0.761 |
| FIFO | 0.116 | 0.473 | 0.246 | 0.812 |

As shown in Table 5-19 the T-value and P-value give an indication of the impact and the significance of each independent variable. The t Stat and P-value represent a significant difference between mean proposed algorithm, FIFO, and Fair execution time values. The positive T-value of the proposed algorithm (0.280), FIFO (0.246) and Fair (0.316) demonstrate that the execution time of the proposed algorithm takes less time than the Fair. Thus, the completion time and cost of our proposed algorithm is more significant compared other algorithms.

### 5.4.6 Analysis of results regarding execution time

Figure 5-7 shows the differences in execution time. The straight line denotes the difference in achievement and the correlation between workload and execution time. Initially, execution is unstable in terms of time due to the small size of the data. However, execution becomes relatively stable with the increase in data size and the number of tasks to be executed by the framework on the cloud.

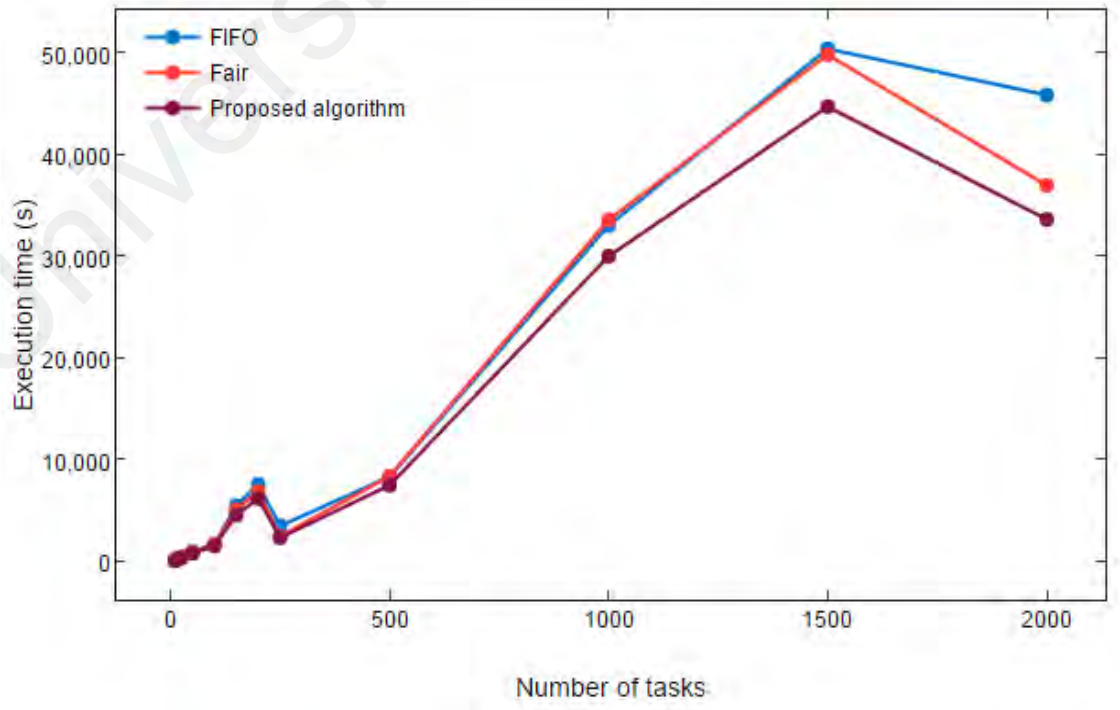



**Figure 5-7: Execution time using WordCount benchmark**

Figure 5-7 presents the execution time of several tasks using the WordCount benchmark with FIFO, Fair, and the proposed scheduling algorithms in virtual cluster nodes with 12 Hadoop jobs of different sizes. The figure shows that the completion time of the overall processing is also increased. In the first scenario, the default algorithm FIFO is used on the Hadoop nodes without tuning the Hadoop parameters. Figure 5 illustrates that the FIFO algorithm slightly degrades the performance of Hadoop in terms of execution time and resource utilization, where data are shared among multiple users. The Fair scheduler and the proposed algorithm appear to exhibit better performance compared with the FIFO algorithm, but the data locality feature is hindered. The proposed algorithm can finish the tasks faster than the other two schedulers using WordCount to process the data. The completion times change based on the type of workload given that different workloads have various resource demands. The sharing of resources during workflow execution regardless of the size are typically relayed on the structure, a number of modules of the workflow, and the complexities. However, only a limited amount of resources that are shared among the nodes can be utilized by the small number of modules in each layer.

In Addition, as shown in Figure 5-7 the sign of dropping after 1500 number of tasks probably indicate that, many nodes in the clusters participate during the process of data since Hadoop replicate its datasets across the cluster. In addition, Hadoop relies on data locality during the processing and as the number of tasks increases the possibility of finding data in a designated node is high compared to less number of tasks which only fewer nodes are participating.

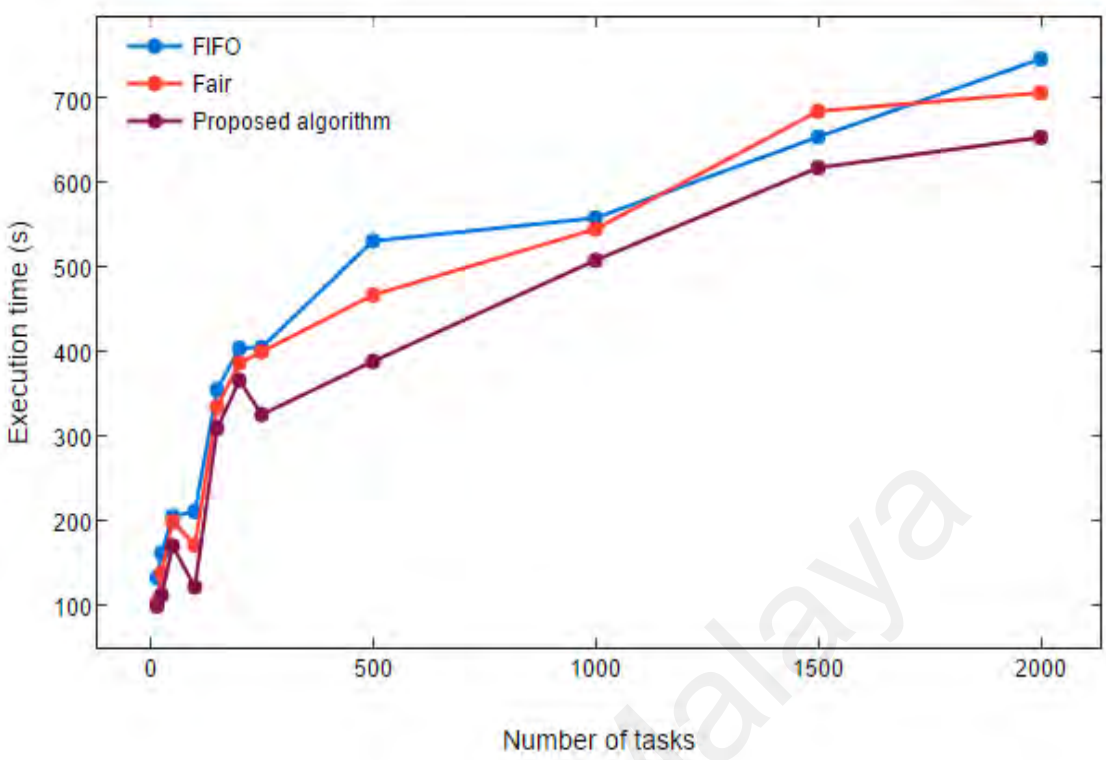



**Figure 5-8: Execution time using Sort benchmark**

In Figure 5-8, shows the use of the Sort benchmark to simulate the FIFO, Fair, and proposed scheduling algorithms. The result slightly differs from that of the WordCount benchmark, where the proposed algorithm achieves a noticeable reduction in task execution time. The Sort benchmark consumes more resources than the WordCount benchmark because of the intensive data flow and the computation of aggregate functions that must perform the Sort benchmark.

The comparison of the aforementioned algorithms indicates that performance has significantly improved using the Sort benchmark, which relies completely on the sharing of resources. Thus, the number of maps and reduce tasks is scaled. The proposed algorithm occasionally exhibits better performance compared with the other algorithms, such as FIFO, given the limited resources to be shared among active nodes.

Table 5.20 provides the comparison between the three algorithms. As shown from the table the proposed algorithm has low execution time regarding the number of tasks for each job in most of the cases as compare to FIFO and Fair scheduler. The default Hadoop scheduler FIFO execution time can be shown to be high regarding execution compare with other scheduling algorithms like fair and the proposed algorithm.

**Table 5-20: Comparison between the proposed algorithm, FIFO, Fair schedules regarding execution time**

| Number of tasks | Proposed algorithm | Fair Scheduler | FIFO Default |
|---|---|---|---|
| 15 | 30 | 39 | 37 |
| 25 | 148.19 | 199.08 | 204.06 |
| 50 | 255.2 | 358.58 | 354.12 |
| 100 | 674.68 | 787.44 | 930.02 |
| 150 | 1250.2 | 1494.53 | 1631.95 |
| 200 | 4048.04 | 4525.55 | 5454.49 |
| 250 | 5342.59 | 6164.7 | 7537.13 |
| 500 | 2115.08 | 2307.18 | 3457.25 |
| 1000 | 6329.22 | 7464.96 | 8303.04 |
| 1500 | 25267.13 | 29980.91 | 33001.93 |
| 2000 | 37964.17 | 44690.48 | 50369.89 |

From the Figure 5-7 and 5-8, from the curves, we can observe that the execution time of the WordCount and Sort tasks scheduled are different in some cases. The reason is that WordCount process requires heavy disk I/O and network throughput. Table 5-20 shows the comparison of different performance measurement based on the result of the experiments conducted using three algorithms FIFO, Fair and the proposed algorithm. Various criteria are presented in the table for the comparison. The summary of the discussion has shown that the proposed algorithm is high in terms of the throughput, resource utilization, and CPU.

The results of execution time show the significant improvement in processing big data with the MapReduce framework in the cloud when our proposed algorithm is used. This significant achievement is because of many factors, including, the flexibility of utilizing the cloud resources when executing a large amount of data, high throughput, low

latency deployed Hadoop cluster on the cloud. The result of this section is comparable with and supporting the finding of the statistical analysis. Finally, the completion time of MapReduce job can achieve with a high probability of the proposed algorithm prediction on the cloud. Moreover, it is able to make good trade-off decisions using multi-objective mechanism.

## 5.5 Validation of Results

The performance accuracy of the multi-objective optimization model was recorded while considering throughput and execution time.

In order to measure the results of our proposed algorithm, validation feature is needed to confirm the quality of the results. This section presents the validation of the proposed algorithm results collected from the experiments for the performance evaluation in section 5.4. Based on the statistical analysis conducted, the results are related to throughput and execution time, which show overall effectiveness of our algorithm. Statistical analysis of our finding via multiple regression models is presented, and their individual values are plotted using figures. We use t-test results to confirm significant of the proposed algorithm. The result of the t-test shows significant differences between mean values of throughput and execution time. The prediction scheme is proving to be accurate considering the high values of the coefficient of determination with respect to the proposed algorithm.

### 5.5.1 Throughput

MapReduce in terms of throughput is limited to hardware and software. As such, the throughput could increase linearly with several jobs submitted to the cluster, which leads to high intensive workload and consequently increases in a total throughput at a constant rate. This shows that linear regression is a clear choice for predicting throughput of the MapReduce. Figure 5-9 illustrates the result of statistical analysis

using linear regression where x-axis gives data size, and y-axis gives the throughput regarding seconds. Moreover, it illustrates the actual and predicted throughput obtained from the experimental test of the proposed algorithm used by linear regression. In Figure 5-10, we observe that the predicted throughput is very close to the actual throughput.

The P-values obtained from section 5.5.1 test the statistical association between the proposed algorithm and other algorithms like FIFO and Fair. Figure 5-6 shows the scatter plot of actual proposed algorithm values and estimated or predicted values obtained by using the $R^2$ method.

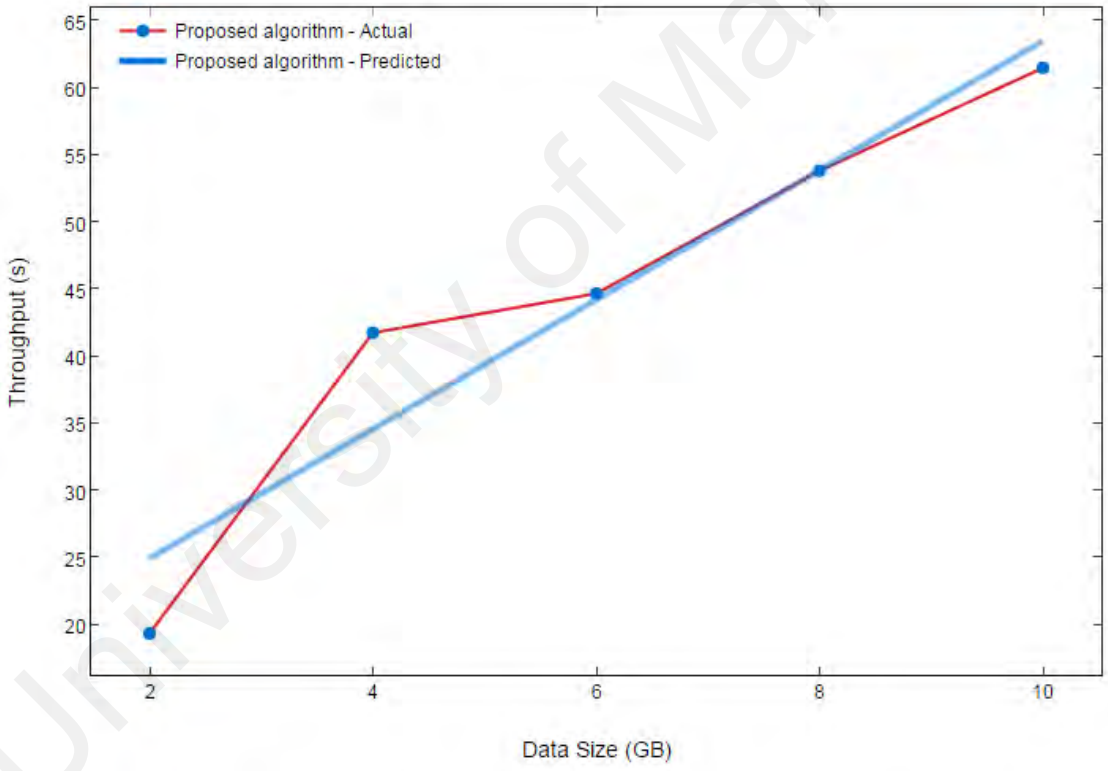



**Figure 5-9: Actual vs. predicted throughput times**

Figure 5.9 shows the throughput of both predicted and actual performance using Hadoop cluster in the cloud for a given size distribution of the job. The runtime is shown as the average value of the runtime on the virtual resource that was used to execute the tasks. Obviously, the predicted throughput of the tasks is very close to the

actual throughput even under varying and high load. This observation is supported by the high value of the unformatted fit parameters. The magnitude of $R^2$ lies between 0 and 1 with a higher magnitude indicating a better prediction. The regression analysis showed a significant relationship between the actual and predicted proposed algorithm, $R^{2 =} 0.95$.

### 5.5.2 Execution time

Figure 5-9 illustrates the result of statistical analysis using linear regression where x-axis gives the number of tasks and y-axis gives the runtime in terms of seconds. Moreover, it shows the actual and predicted execution time obtained from the experimental test of the proposed algorithm used by linear regression. In Figure 5-11, we observe that the predicted execution time is very close to the actual execution time in the early stage.

It can be observed in Figure 5-10 that there is an improvement regarding execution time with respect to predicted ones. This is because increasing number of tasks in Hadoop cluster would speed the process of executing jobs of the datasets. Thus, it is indicated from the performance evaluation that the algorithm was able to reduce the time to execute Hadoop jobs using cloud computing. The comparison results indicated that the proposed methodology presented in the research was found to be better than the FIFO and Fair scheduler regarding the performance metrics (throughput and execution time). The proposed algorithm outperforms the FIFO and Fair in both throughput and execution time.

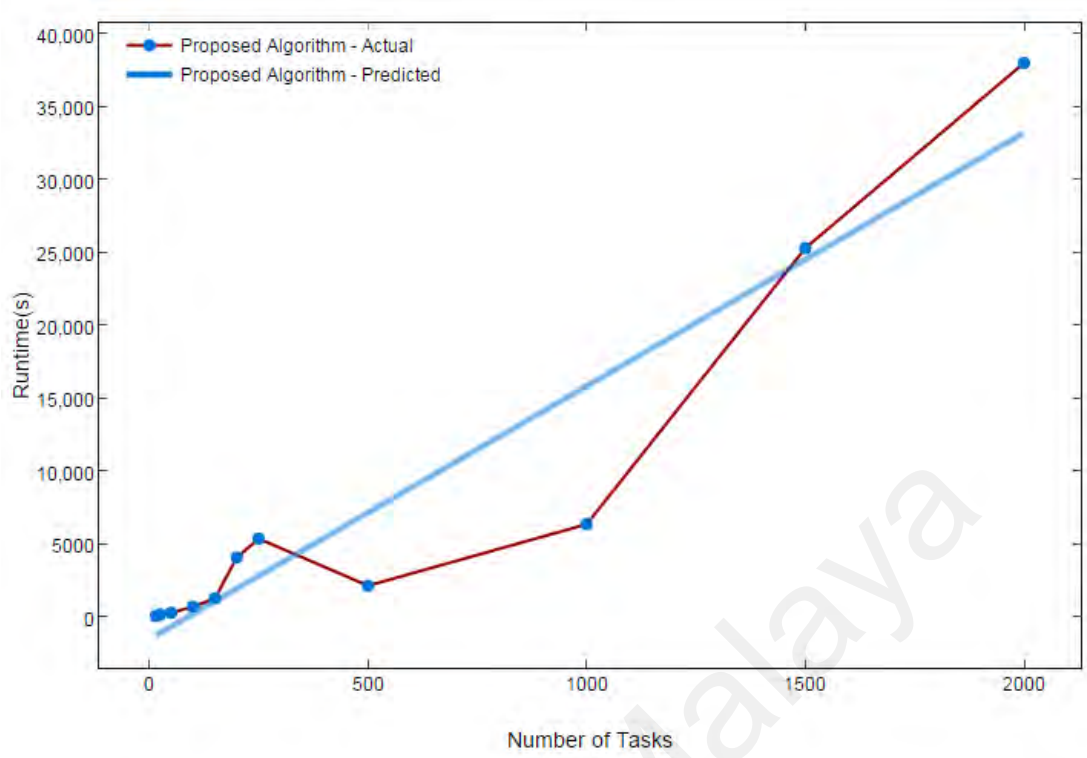



**Figure 5-10: Actual vs Predicted tasks execution time**

Figure 5.10 presents the actual and predicted execution time of the tasks. The predicted and actual execution times are compared for a given number of tasks in Hadoop cluster. The coefficient of determination and the average load on the cluster at the time the task was executed is shown. The regression analysis showed a significant relationship between the actual and predicted proposed algorithm, $R^2 = 0.90$. Moreover, Figure 5.10 demonstrates remarkable improvement in execution time of the MapReduce job using proposed scheduling algorithm in the cloud; further analysis is undertaken in tables 5-17, 5-18 and 5-19.

For the execution of MapReduce tasks, the resources of cloud computing like CPU, memory, and I/O used to complete the Map and Reduce tasks. The initial stage of the execution, all the data is loaded in HDFS, and then when the user submits the application, the data is loaded into memory. After the processing of the data is completed, the output is saved in a separate file in the storage to present to the user.

Such process does not entertain unnecessary virtual resources. However, when the computation requires a high memory and CPU and RAM intensive are not available. Such limitations cause completion time take longer than expected. In addition, because of the constraint set by the cloud providers in terms of cloud resources, it is not possible to load the entire data into memory for the high intensive workload. Therefore, the time consuming by I/O process continues to load the data into memory. Nonetheless, significant differences in execution MapReduce tasks in the cloud allow users to adjust the resources depends on the extremely large workload.

## 5.6 Discussions

In the previous sections, we provide a performance evaluation and validation of our proposed algorithm framework thought experiments and statistical analysis. In this section, we provide a discussion of the resulting findings and statistical analysis in order to validate the finding. The statistical analysis is implemented in order to ensure the accuracy achieved by the proposed algorithm. The research used a t-test to measure the significant of the proposed algorithm. Two independent samples are employed for testing difference of the two means which are throughput and execution time. The t-test is performed under the assumption that the completion time and the actual cost have no significant difference between them as shown by the throughput and the execution time.

### 5.6.1 Throughput

We compared the results of the experiments and the statistical modeling. Table 5-21 provides a comparison of the statistical analysis of throughput results. As illustrated in the table, the results of the algorithms and statistical analysis show significant high throughput in using our proposed algorithm. The average of the high throughput of our proposed algorithm is significant 78.31 % when WordCount benchmark application is used. In addition, the coefficient is measured by 0.737 when using the proposed

algorithm. Such, approximately and strong support of findings advocate reliability and validity of the throughput.

**Table 5-21: evaluation, comparison: Proposed algorithm, Fair, and FIFO in terms of Throughput times**

| Evaluation | Throughput Time | Coefficients | Standard Error | P-value | $R^2$ |
|---|---|---|---|---|---|
| Proposed algorithm | High | 0.737 | 0.171 | 0.145 | 0.91 |
| Fair | Medium | 0.335 | 0.816 | 0.752 | 0.86 |
| FIFO | Low | -0.879 | 0.841 | 0.486 | 0.85 |

Moreover, the table shows the coefficient of determination $R^2$ "R squared" value of the results gathered from tasks processed in MapReduce that is deployed on the cloud. $R^2$ values are given for the executed tasks under cloud computing environment. An $R^2$ for the proposed algorithm (0.91), Fair (0.86), and FIFO (0.85) is captured which is higher than 0.05 that is considered a good prediction and prove that the regression model is accurate in its prediction. Thus, the actual and predicted throughput times are compared to a given size. The standard error on the workload on the cluster at the time the tasks were executed is also displayed. As shown in Table 5-21 the P-value of the estimated coefficients of x-axis and y-axis, are 0.145, 0.752and 0.752, for the proposed algorithm, Fair, and FIFO respectively, indicating that they are significantly related at the level of 0.05.

### 5.6.2 Execution time

The execution time also compared using statistical analysis models to predict the execution times of all the tasks on all nodes from the data gathered by the algorithms. Thesis times represent the current cluster workload conditions in the cloud. Note that the schedule generated by the proposed algorithm clearly outperforms the Fair and FIFO schedulers.

**Table 5-22: Evaluation comparison: Proposed algorithm, Fair, and FIFO regarding execution times**

| Evaluation | Execution Time | Coefficients | Standard Error | P-value | $R^2$ |
|---|---|---|---|---|---|
| Proposed algorithm | Low | 0.505 | 1.807 | 0.214 | 0.90 |
| Fair | Medium | 0.342 | 1.081 | 0.788 | 0.90 |
| FIFO | High | 0.116 | 0.473 | 0.761 | 0.90 |

Table 5-22 shows the coefficient of determination $R^2$ "R squared" value of (0.90) from the results gathered from tasks processed in MapReduce that is deployed on the cloud. This $R^2$ is higher than 0.05, which considered a good prediction and proved that the regression model is accurate in its prediction. Therefore, the actual and predicted execution times are compared regarding given many tasks. The standard error on the workload on the cluster at the time the tasks were executed is also displayed. As shown in Table 5-22 the P-value for the estimated coefficients of x-axis and y-axis, are 0.214, 0.788 and 0.761, for the proposed algorithm, Fair, and FIFO respectively, indicating that they are significantly related at the level of 0.05.

## 5.7 Summary

As a new emerging technology, scheduling in MapReduce has been widely explored from various aspects by many researchers in recent years. However, most former research work mainly considers optimized and designed of algorithms and frameworks under a relatively a homogeneous environment. As a matter of fact, many performance issues, like hardware failure, software error and the heterogeneity of the machines and data have brought a great challenge in the performing data analysis. Therefore, it is essential to consider the performance tuning in MapReduce scheduling, such that the performance of algorithms and frameworks can be guaranteed for a different environment.

The goal of the proposed multi-objective algorithm in this chapter is to decrease the execution time of the tasks in the MapReduce framework in the cloud in order to achieve the minimization of the time and cost objective. This algorithm can decide the task start time through the running situation of the tasks. This evaluation set out to test the efficiency of the proposed algorithm against very common scheduling algorithms used for big data processing on Hadoop cluster. Using Hadoop benchmarks, a good performance was achieved in a different scenario. The evaluation metric revealed that running time of multiple tasks in a parallel environment is reduced under proposed algorithm and the throughput revealed that the scheduling could offer low latency with high throughput.

To measure and calculate the performance of each node on the cluster, different benchmarks are used. Also, for the verification of the effectiveness of the result, we use formulas in the experiment to confirm the correctness of the results obtained. For the evaluation, we use Hadoop MapReduce program on the heterogeneous parallel virtual computer.

In this chapter, the result of performance evaluation of the proposed algorithm using benchmarking and statistical modeling is presented and discussed using tables and figures. Linear regression analysis was performed on throughput and execution time using single predictor variable. Moreover, F significance, R-square, coefficient, and standard error were produced using standard Summary Report. For checking the linearity of the regression Residual Plots and Line Fit Plots were used. The result is based on p-values which typically 0.05 at 95% confidence interval. The t-test results are shown in Table 5-21, and 5-22 showed that the t-test is not significant ($p>0.05$). Thus, it is accepted that the completion time and the actual cost has no significant difference between them as shown by the throughput and the execution time. This means that the

throughput by the proposed algorithm is statistically equivalent to the cost. Thus, the algorithm has the potential for representing the real world system because it was able to produce results that are statistically the same as the actual real world system. We show to demonstrate the results in section 5.2, which highlight the significance of the minimization of the execution time and latency via increase the throughput from the benchmarking and statistical analysis. The performance evaluation of the proposed algorithm is conducted using workloads of the two applications in order to show the significance of the execution and throughput.

# CHAPTER 6: CONCLUSION AND FUTURE DIRECTION

This chapter concludes the major contributions of the thesis. It also outlines the potential opportunities to improve further or extend the work presented in the thesis. To this end, this thesis stands as a substantial effort to optimize scheduling of MapReduce in cloud computing from two dimensions simultaneously, including resource allocation and task scheduling.

## 6.1 Aim and objectives of the study

In this thesis, we aimed to achieved multi-objective scheduling algorithm that minimizes both the completion time and cost of using cloud computing by using evolutionary algorithms in order to improve resource allocation and tasks scheduling during the processing of big data. In the following, we highlight the contribution achievement of the thesis.

### 6.1.1 Study the domain of big data and identify the key issues with respect to scheduling in big data platforms

We accomplish this objective by reviewing the rise of big data in cloud computing. The characteristics and classification of big data along with some discussions on cloud computing are introduced. The relationship between big data and cloud computing and Hadoop technology are also discussed. We discussed the background of Hadoop technology and its core components, namely, MapReduce and HDFS. We also present scheduling in big data platforms, requirements for big data processing, scheduling algorithms, and multi-objective optimization, which is close, linked to big data processing studies.

Moreover, in this objective also we discuss different scheduling algorithms used in big data platforms. In order to highlight the results of the study of the previous survey

conducted, the findings are strengthened. Having presented the concept of scheduling in big data platforms, various optimization options are identified. We studied scheduling of MapReduce jobs of the large volume of data processing. The study considers the minimization of completion time given a fixed budget, minimization of the monetary cost given a deadline and the trade-offs between the completion time and monetary cost of using cloud computing.

Strengthened by the result of the study, this objective has revealed the issues that led to performance degradation in resource allocation and task scheduling problems, especially, when large amounts of data are being processed by a framework like MapReduce in a distributed environment.

### 6.1.2 Investigate and identify the research problem

This objective aims to provide analysis on the performance of scheduling MapReduce jobs in terms of the processing using the physical and cloud cluster, which has an impact on the processing time and cost. Using analytical analysis mathematical equations is derived to identify the time-cost model in the processing of big data which demonstrate the significant of the execution time and budget when utilizing cloud resources. We formulate the completion time with budget constraint model and cost with deadline constraint model. The initial findings are verified through experiments using two genetic algorithms: NSGA-II and SPEA2. The analysis of the models is carried out using genetic algorithms. It is well known that multi-objective genetic algorithms are among the most useful approaches for multi-objective. For computational experiments, the testbed is developed in which the tasks data for each job are randomly generated. Moreover, the trade-off solutions of Makespan and cost computed by sorting genetic algorithm II (NSGA-II) and Strength Pareto Evolutionary Algorithm 2 (SPEA2) are analyzed based on different workflow in order to find an

optimal solution between the two conflicting objectives. The classic one point crossover operator is used, which is important for the creation of the children. Also, two columns are randomly selected during the mutation operator, and all the different solutions are assigned to many non-dominated fronts in a ranking operation. During the generation, all the non-dominated solutions are copied to the Pareto archive. Compared with SPEA2, the Pareto archive size is fixed in this algorithm. The Pareto archive is filled with the non-dominated solutions until the considered archive size.

### 6.1.3 Design and propose a new multi-objective algorithm

This objective has achieved by providing a framework for multi-objective scheduling algorithm, in order to improve the execution time of tasks in the big data platforms. The framework designed is based on MapReduce framework, where resource manager is responsible for allocating the resources among the Hadoop clusters. The framework has introduced two main components which are adaptive control to identify tasks that unable to complete in a given time with the available resources and provides a feedback for the users and decision module to decide what the best fit for the job to run.

Hadoop checks if all the map tasks have completed in in the cluster.,then, the output of the results of the execution of the map tasks and the time is saved in the logs. Similarly, for the reduce tasks, the completed tasks of the execution and the time are updated in the logs. The important of gathering the information about the execution is to help analysis the result in terms of the completion time, latency and the utilization of the CPU.

### 6.1.4 Evaluate the performance of a proposed algorithm

In this objective, the result of performance evaluation of the proposed algorithm using benchmarking and statistical modeling is presented and discussed using tables and figures. We demonstrate the results in section 5.2, which highlight the significant of the

minimization of the execution time and latency via increase the throughput from the benchmarking and statistical analysis. Linear regression analysis was performed on throughput and execution time. In each case presented single predictor variable was used. R square, Adjusted R square, intercept, coefficient, standard error, and F significance were generated by the standard Summary Report. Residual Plots and Line Fit Plots were used to check for the linearity of the regression. The result is based on p-values which typically 0.05 at 95% confidence interval. The t-test results are shown in Table 5-21, and 5-22 showed that the t-test is not significant ($p>0.05$). Thus, it is accepted that the completion time and the actual cost has no significant difference between them as shown by the throughput and the execution time. This means that the throughput by the proposed algorithm is statistically equivalent to the cost. Thus, the algorithm has the potential for representing the real world system because it was able to produce results that are statistically the same as the actual real world system.

The performance evaluation of the proposed algorithm is conducted using workloads of the two applications in order to show the significant of the execution time and throughput.

## 6.2 Limitations and Future Research Directions of the study

In this thesis, we developed a multi-objective algorithm framework to improve the resource allocation and job scheduling in the cloud. The framework is based on a multi-objective algorithm, which considers execution time is an important factor. There are several limitations that extend from the work presented here. The research studies only two objective functions and constraints in developing the models. However, many other objective functions need to be considered, which may have a significant impact on the MapReduce performance while processing big data. Moreover, this thesis focused only one job with multiple tasks to be execution. However, several jobs with multiple with different resources can be considered. Moreover, the study does not focus on services

level agreements as the whole for resource allocation, in which the providers are not aware of the quality metrics set by the users regarding resource allocation decision.

While existing research on Hadoop scheduling has shown great improvement, numerous challenging yet to be solved. Hadoop composed of one master node and several data nodes dependents on the size of the cluster. Users can make their choices according to the availability of the resources and the nodes. This research offers insights into the future research opportunities as follows:

*(a) QoS-Based Scheduling:* In the last few years, cloud computing has become a fully service-oriented paradigm, which effectively allows users to consume based on their QoS requirement. The decision on scheduling is based on assumptions, in which the dependencies of workflow tasks are properly defined (Yu, 2007). Thus, monitoring task execution, resource capability, and service time based on the SLA are significant. Moreover, there is a lack of algorithms that consider the tradeoff of multiple quality requirements (Tiwari et al., 2015). The scheduler should be able to find an alternative service and request an SLA for the task execution with respect to its currently accepted set of SLAs and expected the return of unscheduled tasks. Moreover, most of the previous scheduling is based on focuses on deterministic Directed Acyclic Graphs (DAGs). Moreover, using Hadoop on cloud require a good QoS based workflow execution. It is important that new benchmarks put in place to ensure QoS-based workflow scheduling algorithms by comparing and evaluating different workflow applications that are suitable for Hadoop framework.

*(b) Multi-dimensional resource scheduling:* Traditional scheduling systems based on a single-resource optimization, like processors, fail to provide near optimal solutions (Sheikhalishahi et al., 2016). Multi-dimensional resources may consider using a series of resources such as network bandwidth, CPU, and memory. For example, slot

management scheme (Y. Yao et al., 2015) is proposed in order to enable dynamic slot configuration in Hadoop. The idea behind slot management scheme is to improve resource utilization and reduce the Makespan of multiple jobs. Thus, there is a lack of related metrics for scheduling algorithms considering multi-dimensional resource. For future research in, the main idea of the scheduling algorithm with multi-dimensional resources such as CPU and memory is to achieve a less completion time through efficient management of existing cloud resources (Khoo et al., 2007) (Z. Yao et al., 2015). Moreover, the scheduling algorithms should be able to consider diverse resource requirements of different tasks and shown to obtain a minimal execution schedule through efficient management of available cloud computing resources.

*(c) Event-based scheduling:* Previously, the offline problem has been the main focus in scheduling research in order to minimize execution time for a single workflow with known task runtimes (X. Dong et al., 2011). Heterogeneity brings new challenging issues to the Hadoop scheduling the low support for complex requirements in current queue-based scheduling algorithms and arising problems of the schedule-based solution when applied in a dynamic environment with uncertainties. Optimization of these scheduling algorithms for Hadoop demanded to schedule many jobs in the queue. Thus, it is essential to overcome the traditional method and develop automatic synchronization activity execution for enabling workflows to exchange data with other workflows or other applications (Casati & Shan, 2007) (Ilyushkin et al., 2015).

*(d) Energy consumption and efficiency models for Map-reduce jobs:* Although a new power-aware MapReduce application model has introduced in (Y. Li et al., 2011) to be used for power-aware computing with consideration of users' requirements. There is a need of detailed energy efficiency model for MapReduce environments to predict the energy consumed for mix workload scenarios (Goiri et al., 2012). It should also

consider the background HDFS activities carried out for availability checks. It should be able to incorporate the idle nodes energy as well. The performance of MapReduce framework can possibly use for prediction of the map and reduce task timings depending on the data volume, their distribution, underlying hardware, etc. Energy and performance models can be combined to evaluate various scheduling algorithms for predicting the energy consumptions and thus decide which one maximizes the performance and energy efficiency.

Moreover, the continuous growth in the size of the data-centers containing Hadoop MapReduce clusters of hundreds and thousands of machines to support many users has let in a tremendous increase in the energy consumed to operate these large-scale data centers. Consequently, energy efficiency becoming a key open issue in the development of different techniques and approaches to optimize power management in Hadoop clusters (Ibrahim et al.). Furthermore, in interactive data analysis, MapReduce workload runs in large clusters, whose size and cost make energy efficiency a critical concern on MapReduce, particularly in a cloud environment in which large equipped infrastructure is involved. For example, (Maheshwari et al., 2012) and (Wirtz & Ge, 2011) addressed the problem of energy conservation for large data centers that run MapReduce jobs. Accordingly, the tremendously increased amount of energy is consumed to operate these data centers (i.e., electricity used for operating and cooling them) and ends up with a high money bill in the order of millions of dollars. To this end, the system may offload some computation tasks to the sources in the distributed data-centric environment to avoid the expensive data movement costs (Kambatla et al., 2014) and to achieve fast response time with minimal energy consumption.

*(e) Mapping scheme:* It assigns multiple inputs to a set of reducers in such a manner that for each output, a reducer receives all inputs while performing the computation.

Owing to the limited capacity of reducers, only specific inputs can be assigned. However, the size of each individual input may vary, which results in a high communication cost. Consideration and restriction of input size are important in the MapReduce framework and can help optimize the communication cost between map and reduce phases (Afrati et al., 2014). Several solutions have been proposed to minimize the number of copies of inputs being sent to reducers. Research is required to establish an efficient mapping scheme that can minimize the communication cost without affecting the performance of a specified task.

*(f) Performance optimizations:* Hadoop/MapReduce is useful because of its multiple characteristics, such as scalability, fault tolerance, and large amounts of data processing. MapReduce comprises several factors, namely, task initialization time, scheduling, and monitoring performance degradation in different types of applications (Kalavri & Vlassov, 2013). In addition, Hadoop/MapReduce does not support data pipelining or overlapping of the map and reduce phases. To improve the MapReduce performance, further optimization is required in the different aspects of MapReduce, such as index creation, reuse of previously computed results, and fast query execution. MapReduce will achieve better performance if the said issues are solved.

*(g) Optimized data shuffling:* In MapReduce, intensive disk input/output during the shuffling phase increases the overall execution time, which in turn degrades the performance of overall systems (Lin et al., 2013). Reducing the execution time has become challenging. Many solutions have been proposed to address this problem, but no solution has solved this problem completely in an efficient manner. Only new optimized techniques can solve this problem completely and efficiently. Research in this area can increase the performance of MapReduce by reducing the shuffle phase time.

*(h) Automation and configuration:* Automatic tuning and configuration help while deploying the Hadoop/MapReduce cluster by setting several parameters. To perform proper tuning, both hardware and workload characteristics must be known. While performing configuration, a small mistake can cause inefficient execution of jobs, which leads to performance degradation (Lama & Zhou, 2012). To overcome this issue, several new techniques and algorithms are required; these techniques and algorithms perform the calculation in which basic setting can be performed in an efficient manner. Creating such algorithms that receive input from the user, understanding the characteristics of underlying hardware by using machine learning, and suggesting a proper setting for better performance are challenging.

## REFERENCES


Abad, C. L., Lu, Y., & Campbell, R. H. (2011). *DARE: Adaptive data replication for efficient cluster scheduling.* Paper presented at the Cluster Computing (CLUSTER), 2011 IEEE International Conference on, pp. 159-168.

Afrati, F., Dolev, S., Korach, E., Sharma, S., & Ullman, J. D. (2014). Assignment Problems Of Different-Sized Inputs In Mapreduce, 11(2), 18.

Anjos, J. C., Carrera, I., Kolberg, W., Tibola, A. L., Arantes, L. B., & Geyer, C. R. (2015). MRA++: Scheduling and data placement on MapReduce for heterogeneous environments. *Future Generation Computer Systems, 42*, 22-35.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., . . . Stoica, I. (2010). A view of cloud computing. *Communications of the ACM, 53*(4), 50-58.

Bagchi, T. P. (1999). *Multiobjective scheduling by genetic algorithms*: Springer Science & Business Media, Kluwer Academic Publishers.

Bittencourt, L. F., & Madeira, E. R. M. (2011). HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications, 2*(3), 207-227.

Blazewicz, J., Lenstra, J. K., & Kan, A. R. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics, 5*(1), 11-24.

Bollier, D., & Firestone, C., M. (2010). *The promise and peril of big data*: Aspen Institute, Communications and Society Program Washington, DC, USA.

Bryant, R., Katz, R. H., & Lazowska, E. D. (2008). Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science and Society, pp 1-15.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems, 25*(6), 599-616.

Casati, F., & Shan, M.-C. (2007). Event-based scheduling method and system for workflow activities: Google Patents, U.S. Patent No. 7,240,324. 3.

Casavant, T. L., & Kuhl, J. G. (1988). A taxonomy of scheduling in general-purpose distributed computing systems. *Software Engineering, IEEE Transactions on, 14*(2), 141-154.

Chang, H., Kodialam, M., Kompella, R. R., Lakshman, T., Lee, M., & Mukherjee, S. (2011). *Scheduling in mapreduce-like systems for fast completion time.* Paper presented at the INFOCOM, 2011 Proceedings IEEE, (pp. 3074-3082).

Chang, P.-C., Chen, S.-H., & Liu, C.-H. (2007). Sub-population genetic algorithm with mining gene structures for multiobjective flowshop scheduling problems. *Expert Systems with Applications, 33*(3), 762-771.

Chen, M., Mao, S., & Liu, Y. (2014a). Big Data: A Survey. *Mobile Networks and Applications*, 1-39.

Chen, M., Mao, S., & Liu, Y. (2014b). Big Data: A Survey. *Mobile Networks and Applications, 19*(2), 171-209.

Chen, Q., Liu, C., & Xiao, Z. (2013). Improving mapreduce performance using smart speculative execution strategy, 63(4), 954-967.

Chiroma, H., Abdulkareem, S., & Herawan, T. (2015). Evolutionary Neural Network model for West Texas Intermediate crude oil price prediction. *Applied Energy, 142*, 266-273.

Cottrell, L. (2012). *PingER: Actively measuring the worldwide Internet's end-to-end performance.* Paper presented at the Workshop at the University of Malaysia in Sarawak.

Cottrell, L., Matthews, W., & Logg, C. (2013). Tutorial on Internet Monitoring & PingER at SLAC. Retrieved from http://www.slac.stanford.edu/comp/net/wan-mon/tutorial.html

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

Dean, J., & Ghemawat, S. (2010). MapReduce: a flexible data processing tool. *Communications of the ACM, 53*(1), 72-77.

Deb, K. (2014). Multi-objective optimization *Search methodologies* (pp. 403-449): Springer.

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II.* Paper presented at the Parallel problem solving from nature PPSN VI.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on, 6*(2), 182-197.

Dong, F., & Akl, S. G. (2007). *PFAS: a resource-performance-fluctuation-aware workflow scheduling algorithm for grid computing.* Paper presented at the Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International.

Dong, X., Wang, Y., & Liao, H. (2011). *Scheduling mixed real-time and non-real-time applications in mapreduce environment.* Paper presented at the Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on.

Doulkeridis, C., & Nørvåg, K. (2014). A survey of large-scale analytical query processing in MapReduce. *The VLDB Journal, 23*(3), 355-380.

Durillo, J. J., & Prodan, R. (2014). Multi-objective workflow scheduling in Amazon EC2. *Cluster Computing, 17*(2), 169-189.

Eisen, M. B., Spellman, P. T., Brown, P. O., & Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences, 95*(25), 14863-14868.

Facchinei, F., Sagratella, S., & Scutari, G. (2014). *Flexible parallel algorithms for big data optimization.* Paper presented at the Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.

Facebook. (2012). Facebook Engineering. Under the Hood: Scheduling MapReduce jobs more efficiently with Corona. Retrieved from https://www.facebook.com/notes/facebook-engineering/under-the-hood-scheduling-mapreduce-jobs-more-efficiently-with-corona/10151142560538920

Fang, Y., Wang, F., & Ge, J. (2010). A task scheduling algorithm based on load balancing in cloud computing *Web Information Systems and Mining* (pp. 271-277): Springer.

Fonseca, C. M., Gandibleux, X., Hao, J.-K., & Sevaux, M. (2009). *Evolutionary Multi-Criterion Optimization: 5th International Conference, EMO 2009, Nantes, France, April 7-10, 2009, Proceedings* (Vol. 5467): Springer.

Gantz, J., & Reinsel, D. (2011). Extracting value from chaos. *IDC iview*, 1-12.

Garlasu, D., Sandulescu, V., Halcu, I., Neculoiu, G., Grigoriu, O., Marinescu, M., & Marinescu, V. (2013). *A big data implementation based on grid computing.* Paper presented at the Roedunet International Conference (RoEduNet), 2013 11th.

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). *The Google file system.* Paper presented at the ACM SIGOPS Operating Systems Review.

Giuseppe, A., Alessio, B., Walter, D., & Antonio, P. (2013). Survey Cloud monitoring: A survey. *Comput. Netw., 57*(9), 2093-2115. doi:10.1016/j.comnet.2013.04.001

Goiri, Í., Le, K., Nguyen, T. D., Guitart, J., Torres, J., & Bianchini, R. (2012). *GreenHadoop: leveraging green energy in data-processing frameworks.* Paper presented at the Proceedings of the 7th ACM european conference on Computer Systems, (pp. 57-70). ACM.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning, 3*(2), 95-99.

Gu, R., Yang, X., Yan, J., Sun, Y., Wang, B., Yuan, C., & Huang, Y. (2014). SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters. *Journal of Parallel and Distributed Computing, 74*(3), 2166-2179.

Gunarathne, T., Wu, T.-L., Qiu, J., & Fox, G. (2010). *MapReduce in the Clouds for Science.* Paper presented at the Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, (pp. 565-572).

Gunarathne, T., Zhang, B., Wu, T.-L., & Qiu, J. (2013). Scalable parallel computing on clouds using Twister4Azure iterative MapReduce. *Future Generation Computer Systems, 29*(4), 1035-1048.

Guo, L., Zhao, S., Shen, S., & Jiang, C. (2012). Task scheduling optimization in cloud computing based on heuristic algorithm. *Journal of Networks, 7*(3), 547-553.

Guo, Z., Fox, G., & Zhou, M. (2012). *Investigation of data locality in mapreduce.* Paper presented at the Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012).

Guo, Z., Fox, G., Zhou, M., & Ruan, Y. (2012). *Improving resource utilization in mapreduce.* Paper presented at the Cluster Computing (CLUSTER), 2012 IEEE International Conference on.

Gupta, S., Fritz, C., Price, B., Hoover, R., De Kleer, J., & Witteveen, C. (2013). *ThroughputScheduler: Learning to Schedule on Heterogeneous Hadoop Clusters.* Paper presented at the ICAC.

Hadoop, A. Apache Hadoop. (2008, july 12) Retrieved from https://hadoop.apache.org/

Hadoop, A. Capacity Scheduler Guide. (2013, Aug 12) Retrieved from https://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html

Hadoop,A.(2009 May 22). Fair Scheduler.Retrieved from https://hadoop.apache.org/docs/stable1/fair_scheduler.html

Hadoop, A. (2011). Apache Hadoop. Retrieved from https://hadoop.apache.org/

Hammoud, M., Rehman, M. S., & Sakr, M. F. (2012). *Center-of-gravity reduce task scheduling to lower mapreduce network traffic.* Paper presented at the Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. (pp. 49-58). IEEE.

He, C., Lu, Y., & Swanson, D. (2011). *Matchmaking: A new mapreduce scheduling technique.* Paper presented at the Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. (pp. 40-47). IEEE.

Heintz, B., Chandra, A., & Sitaraman, R. K. (2012). Optimizing mapreduce for highly distributed environments. *arXiv preprint arXiv:1207.7055*.

Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F. B., & Babu, S. (2011). *Starfish: A Self-tuning System for Big Data Analytics.* Paper presented at the CIDR, (11), pp. 261-272.

Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A. D., Katz, R. H., . . . Stoica, I. (2011). *Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center.* Paper presented at the NSDI, (11), pp. 22-22.

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). *A niched Pareto genetic algorithm for multiobjective optimization.* Paper presented at the Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on. (pp. 82-87). Ieee.

Hsu, C.-H. (2014). Intelligent big data processing. *Future Generation Computer Systems, 36*(0), 16-18. doi:http://dx.doi.org/10.1016/j.future.2014.02.003

Huan, L. (2013). Big Data Drives Cloud Adoption in Enterprise. *Internet Computing, IEEE, 17*(4), 68-71.

Huang, S., Huang, J., Dai, J., Xie, T., & Huang, B. (2011). The HiBench benchmark suite: Characterization of the MapReduce-based data analysis *New Frontiers in Information and Software as Services* (pp. 209-228): Springer.

Hussain, H., Malik, S. U. R., Hameed, A., Khan, S. U., Bickler, G., Min-Allah, N., . . . Ghani, N. (2013). A survey on resource allocation in high performance distributed computing systems. *Parallel Computing, 39*(11), 709-736.

Ibrahim, S., Jin, H., Lu, L., He, B., Antoniu, G., & Wu, S. (2012). *Maestro: Replica-aware map scheduling for mapreduce.* Paper presented at the Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on. (pp. 435-442). IEEE.

Ibrahim, S., Phan, T.-D., Carpen-Amarie, A., Chihoub, H.-E., Moise, D., & Antoniu, G. Governing energy consumption in Hadoop through CPU frequency scaling: An analysis. *Future Generation Computer Systems*(0). 54, 219-232.

Ilyushkin, A., Ghit, B., & Epema, D. (2015). *Scheduling workloads of workflows with unknown task runtimes.* Paper presented at the Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on. (pp. 606-616). IEEE.

Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K., & Goldberg, A. (2009). *Quincy: fair scheduling for distributed computing clusters.* Paper presented at the Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. (pp. 261-276). ACM.

Islam, N. S., Lu, X., Wasi-ur-Rahman, M., Jose, J., & Panda, D. K. D. (2014). A micro-benchmark suite for evaluating HDFS operations on modern clusters *Specifying Big Data Benchmarks* (pp. 129-147): Springer.

Ji, C., Li, Y., Qiu, W., Awada, U., & Li, K. (2012). *Big data processing in cloud computing environments.* Paper presented at the Pervasive Systems, Algorithms and Networks (ISPAN), 2012 12th International Symposium on. (pp. 17-23). IEEE.

Jia, H., Nee, A. Y., Fuh, J. Y., & Zhang, Y. (2003). A modified genetic algorithm for distributed scheduling problems. *Journal of Intelligent Manufacturing, 14*(3-4), 351-362.

Jiang, D., Ooi, B. C., Shi, L., & Wu, S. (2010). The performance of mapreduce: An in-depth study. *Proceedings of the VLDB Endowment, 3*(1-2), 472-483.

Jin, J., Luo, J., Song, A., Dong, F., & Xiong, R. (2011). *Bar: An efficient data locality driven task scheduling algorithm for cloud computing.* Paper presented at the Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. (pp. 295-304). IEEE.

Jin, Y., & Sendhoff, B. (2001). Multi-objective optimization: Google Patents. U.S. Patent Application No. 10/007,906.

Kalavri, V., & Vlassov, V. (2013). *Mapreduce: Limitations, optimizations and open issues.* Paper presented at the Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on. (pp. 1031-1038). IEEE.

Kambatla, K., Kollias, G., Kumar, V., & Grama, A. (2014). Trends in big data analytics. *Journal of Parallel and Distributed Computing, 74*(7), 2561-2573.

Kc, K., & Anyanwu, K. (2010). *Scheduling hadoop jobs to meet deadlines.* Paper presented at the Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. (pp. 388-392). IEEE.

Khan, S., Shiraz, M., Abdul Wahab, A. W., Gani, A., Han, Q., & Bin Abdul Rahman, Z. (2014). A Comprehensive Review on Adaptability of Network Forensics Frameworks for Mobile Cloud Computing. *The Scientific World Journal, 2014*, 27. doi:10.1155/2014/547062

Khiyaita, A., Zbakh, M., El Bakkali, H., & El Kettani, D. (2012). *Load balancing cloud computing: state of art.* Paper presented at the Network Security and Systems (JNS2), 2012 National Days of of (pp. 106-109). IEEE..

Khoo, B. B., Veeravalli, B., Hung, T., & See, C. S. (2007). A multi-dimensional scheduling scheme in a Grid computing environment. *Journal of Parallel and Distributed Computing, 67*(6), 659-673.

Kllapi, H., Sitaridi, E., Tsangaris, M. M., & Ioannidis, Y. (2011). *Schedule optimization for data processing flows on the cloud.* Paper presented at the Proceedings of the 2011 ACM SIGMOD International Conference on Management of data.

Knowles, J., & Corne, D. (2002). *On metrics for comparing nondominated sets.* Paper presented at the Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on. (Vol. 1, pp. 711-716). IEEE.

Knowles, J. D., Watson, R. A., & Corne, D. W. (2001). *Reducing local optima in single-objective problems by multi-objectivization.* Paper presented at the Evolutionary multi-criterion optimization. (pp. 269-283). Springer Berlin Heidelberg.

Krish, K., Anwar, A., & Butt, A. R. (2014). *[phi] Sched: A Heterogeneity-Aware Hadoop Workflow Scheduler.* Paper presented at the Modelling, Analysis &

Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on. (pp. 255-264). IEEE.

Kulkarni, A. P., & Khandewal, M. (2014). Survey on Hadoop and Introduction to YARN. *International journal of emerging technology and advanced engineering, 4*(5), 82-87.

Labrinidis, A., & Jagadish, H. (2012). Challenges and opportunities with big data. *Proceedings of the VLDB Endowment, 5*(12), 2032-2033.

Lama, P., & Zhou, X. (2012). *Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud.* Paper presented at the Proceedings of the 9th international conference on Autonomic computing. (pp. 63-72). ACM

Lämmel, R. (2008). Google's MapReduce programming model — Revisited. *Science of Computer Programming, 70*(1), 1-30.

Leavitt, N. (2013). Bringing big analytics to the masses. *Computer, 46*(1), 20-23.

Li, J.-J., Cui, J., Wang, D., Yan, L., & Huang, Y.-S. (2011). Survey of MapReduce parallel programming model. *Dianzi Xuebao(Acta Electronica Sinica), 39*(11), 2635-2642.

Li, Y., Zhang, H., & Kim, K. H. (2011). *A power-aware scheduling of mapreduce applications in the cloud.* Paper presented at the Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on. (pp. 613-620). IEEE.

Lin, M., Zhang, L., Wierman, A., & Tan, J. (2013). Joint optimization of overlapping phases in MapReduce. *Performance Evaluation, 70*(10), 720-735.

Liu, C. L., & Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM), 20*(1), 46-61.

Long, Q., Lin, J., & Sun, Z. (2011). Agent scheduling model for adaptive dynamic load balancing in agent-based distributed simulations. *Simulation Modelling Practice and Theory, 19*(4), 1021-1034.

Long, S.-Q., Zhao, Y.-L., & Chen, W. (2014). MORM: A Multi-objective Optimized Replication Management strategy for cloud storage cluster. *Journal of Systems Architecture, 60*(2), 234-244.

Lopes, R., & Menascé, D. (2015). A Taxonomy of Job Scheduling on Distributed Computing Systems. Retrieved from http://cs.gmu.edu

Maheshwari, N., Nanduri, R., & Varma, V. (2012). Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework. *Future Generation Computer Systems, 28*(1), 119-127.

Marczyk, A. (2004). Genetic algorithms and evolutionary computation. *The Talk Origins Archive: http://www. talkorigins/faqs/genalg/genalg.html*.

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization, 26*(6), 369-395.

McGuire, J. M. (2006). Highly scalable least connections load balancing: Google Patents. Patent No. 6,996,615. Washington, DC: U.S. Patent and Trademark Office

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., . . . Daly, M. (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome research, 20*(9), 1297-1303.

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing (draft). *NIST special publication, 800*(145), 7.

Murthy, A. C., Vavilapalli, V. K., Eadline, D., Niemiec, J., & Markham, J. (2013). *Apache Hadoop YARN: Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2*: Pearson Education.

Nagata, Y., & Chu, K. H. (2003). Optimization of a fermentation medium using neural networks and genetic algorithms. *Biotechnology letters, 25*(21), 1837-1842.

Nita, M.-C., Pop, F., Voicu, C., Dobre, C., & Xhafa, F. (2015). MOMTH: multi-objective scheduling algorithm of many tasks in Hadoop. *Cluster Computing*, 1-14.

O'Leary, D. E. (2013). Artificial Intelligence and Big Data. *Intelligent Systems, IEEE, 28*(2), 96-99. doi:10.1109/MIS.2013.39

Orero, S., & Irving, M. (1998). A genetic algorithm modelling framework and solution technique for short term optimal hydrothermal scheduling. *Power Systems, IEEE Transactions on, 13*(2), 501-518.

Pandey, S., & Nepal, S. (2013). Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond. *Future Generation Computer Systems, 29*(7), 1774-1776.

Park, J., Lee, D., Kim, B., Huh, J., & Maeng, S. (2012). *Locality-aware dynamic VM reconfiguration on MapReduce clouds.* Paper presented at the Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing.

Philip Chen, C. L., & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences, 275*(0), 314-347. doi:http://dx.doi.org/10.1016/j.ins.2014.01.015

Pinedo, M. L. (2012). *Scheduling: theory, algorithms, and systems*: Springer Science & Business Media.

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. doi:http://dx.doi.org/10.1016/j.jnca.2014.07.022

Polo, J., Carrera, D., Becerra, Y., Beltran, V., Torres, J., & Ayguadé, E. (2010). *Performance management of accelerated mapreduce workloads in heterogeneous clusters.* Paper presented at the Parallel Processing (ICPP), 2010 39th International Conference on. (pp. 653-662). IEEE.

Polo, J., Castillo, C., Carrera, D., Becerra, Y., Whalley, I., Steinder, M., . . . Ayguadé, E. (2011). Resource-aware adaptive scheduling for mapreduce clusters *Middleware 2011* (pp. 187-207): Springer.

Ponnambalam, S., Aravindan, P., & Naidu, G. M. (2000). A multi-objective genetic algorithm for solving assembly line balancing problem. *The International Journal of Advanced Manufacturing Technology, 16*(5), 341-352.

Pop, F., & Cristea, V. (2015). The Art of Scheduling for Big Data Science. Taylor & Francis Group LLC.

Qi, C., Cheng, L., & Zhen, X. (2014). Improving MapReduce Performance Using Smart Speculative Execution Strategy. *Computers, IEEE Transactions on, 63*(4), 954-967. doi:10.1109/TC.2013.15

Ranjithan, S. R., Chetan, S. K., & Dakshina, H. K. (2001). *Constraint method-based evolutionary algorithm (CMEA) for multiobjective optimization.* Paper presented at the International Conference on Evolutionary Multi-Criterion Optimization.

Rao, B. T., & Reddy, L. (2012). Survey on improved scheduling in Hadoop MapReduce in cloud environments. *arXiv preprint arXiv:1207.0780*.

Rasooli, A., & Down, D. G. (2014). COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems. *Future Generation Computer Systems, 36*, pp (1-15).

Richtárik, P., & Takáč, M. (2015). Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, pp (1-52).

Rimal, B. P., Choi, E., & Lumb, I. (2009). *A taxonomy and survey of cloud computing systems.* Paper presented at the 2009 Fifth International Joint Conference on INC, IMS and IDC. pp (44-51).

Ruzika, S., & Wiecek, M. M. (2005). Approximation methods in multiobjective programming. *Journal of optimization theory and applications, 126*(3), 473-501.

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.

Sangroya, A., Bouchenak, S., Dami, & Serrano, n. (2016). Experience with benchmarking dependability and performance of MapReduce systems. *Perform. Eval., 101*(C), 1-19. doi:10.1016/j.peva.2016.04.001

Savic, D. (2002). Single-objective vs. multiobjective optimisation for integrated decision support. *Integrated Assessment and Decision Support, 1*, 7-12.

Scott, J. (2015). A tale of two clusters: Mesos and YARN. Retrieved from http://radar.oreilly.com/2015/02/a-tale-of-two-clusters-mesos-and-yarn.html

Seada, H., & Deb, K. (2015). *U-nsga-iii: A unified evolutionary optimization procedure for single, multiple, and many objectives: Proof-of-principle results.* Paper presented at the Evolutionary Multi-Criterion Optimization. (pp. 34-49). Springer.

Shankar, D., Lu, X., Wasi-ur-Rahman, M., Islam, N., & Panda, D. K. D. (2014). *A Micro-benchmark Suite for Evaluating Hadoop MapReduce on high-performance networks.* Paper presented at the Workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware. (pp. 19-33). Springer.

Sharma, B. P., Wood, T., & Das, C. R. (2013). *Hybridmr: A hierarchical mapreduce scheduler for hybrid data centers.* Paper presented at the Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on. (pp. 102-111). IEEE.

Sheikhalishahi, M., Wallace, R. M., Grandinetti, L., Vazquez-Poletti, J. L., & Guerriero, F. (2016). A multi-dimensional job scheduling. *Future Generation Computer Systems, 54*, 123-131. doi:http://dx.doi.org/10.1016/j.future.2015.03.014

Shreedhar, M., & Varghese, G. (1996). Efficient fair queuing using deficit round-robin. *Networking, IEEE/ACM Transactions on, 4*(3), 375-385.

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). *The hadoop distributed file system.* Paper presented at the Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. (pp. 1-10). IEEE.

Smanchat, S., & Viriyapant, K. (2015). Taxonomies of workflow scheduling problem and techniques in the cloud. *Future Generation Computer Systems, 52*, 1-12.

Sousa, E., Paul, J., Lari, V., Hannig, F., Teich, J., & Stechele, W. Resource-aware Computer Vision Application on Heterogeneous Multi-tile Architecture. Retrieved from https://www.date-conference.com/files/file/date14/.

Srirama, S. N., Jakovits, P., & Vainikko, E. (2012). Adapting scientific computing problems to clouds using MapReduce. *Future Generation Computer Systems, 28*(1), 184-192.

Su, S., Li, J., Huang, Q., Huang, X., Shuang, K., & Wang, J. (2013). Cost-efficient task scheduling for executing large programs in the cloud. *Parallel Computing, 39*(4–5), 177-188. doi:http://dx.doi.org/10.1016/j.parco.2013.03.002

Tan, J., Meng, X., & Zhang, L. (2012). Delay tails in MapReduce scheduling. *ACM SIGMETRICS Performance Evaluation Review, 40*(1), 5-16.

Tang, Z., Zhou, J., Li, K., & Li, R. (2012). *MTSD: A task scheduling algorithm for MapReduce base on deadline constraints.* Paper presented at the Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International.

Tiwari, N., Sarkar, S., Bellur, U., & Indrawan, M. (2015). Classification Framework of MapReduce Scheduling Algorithms. *ACM Computing Surveys (CSUR), 47*(3), 49.

Tsai, J.-T., Fang, J.-C., & Chou, J.-H. (2013). Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Computers & Operations Research, 40*(12), 3045-3055. doi:http://dx.doi.org/10.1016/j.cor.2013.06.012

Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Seth, S. (2013). *Apache hadoop yarn: Yet another resource negotiator.* Paper presented at the Proceedings of the 4th annual Symposium on Cloud Computing. (p. 5). ACM

Venugopal, S., Buyya, R., & Ramamohanarao, K. (2006). A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computing Surveys (CSUR), 38*(1), 3.

Verma, A., Cherkasova, L., & Campbell, R. H. (2011). *ARIA: automatic resource inference and allocation for mapreduce environments.* Paper presented at the Proceedings of the 8th ACM international conference on Autonomic computing. (pp. 235-244). ACM. (pp. 235-244). ACM.

Villars, R. L., Olofson, C. W., & Eastwood, M. (2011). Big data: What it is and why you should care. *White Paper, IDC*.

Wang, D., Chen, J., & Zhao, W. (2013). A task scheduling algorithm for Hadoop platform. *Journal of Computers, 8*(4), 929-936.

Wang, K., Zhou, X., Li, T., Zhao, D., Lang, M., & Raicu, I. (2014). *Optimizing load balancing and data-locality with data-aware scheduling.* Paper presented at the Big Data (Big Data), 2014 IEEE International Conference on. (pp. 119-128). IEEE.

Wang, L., Zhan, J., Luo, C., Zhu, Y., Yang, Q., He, Y., . . . Zhang, S. (2014). *Bigdatabench: A big data benchmark suite from internet services.* Paper presented at the 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA). (pp. 488-499). IEEE.

Wang, X., Wang, Y., & Cui, Y. (2014). A new multi-objective bi-level programming model for energy and locality aware multi-job scheduling in cloud computing. *Future Generation Computer Systems, 36*, 91-101. doi:http://dx.doi.org/10.1016/j.future.2013.12.004

Wang, Y., & Shi, W. (2014). Budget-driven scheduling algorithms for batches of MapReduce jobs in heterogeneous clouds. *Cloud Computing, IEEE Transactions on, 2*(3), 306-319.

Warneke, D., & Kao, O. (2009). *Nephele: efficient parallel data processing in the cloud.* Paper presented at the Proceedings of the 2nd workshop on many-task computing on grids and supercomputers. (p. 8). ACM.

White, T. (2009). *Hadoop: The Definitive Guide: The Definitive Guide*: O'Reilly Media.

White, T. (2012). *Hadoop: The definitive guide*: " O'Reilly Media, Inc.".

Wirtz, T., & Ge, R. (2011). *Improving mapreduce energy efficiency for computation intensive workloads.* Paper presented at the Green Computing Conference and Workshops (IGCC), 2011 International. (pp. 1-8). IEEE

Wolf, J., Rajan, D., Hildrum, K., Khandekar, R., Kumar, V., Parekh, S., . . . Balmin, A. (2010). Flex: A slot allocation scheduling optimizer for mapreduce workloads *Middleware 2010* (pp. 1-20): Springer.

Wu, L., Garg, S. K., & Buyya, R. (2011). *SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments.* Paper presented at the Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on, (pp. 195-204). IEEE..

Yao, Y., Wang, J., Sheng, B., Tan, C., & Mi, N. (2015). Self-Adjusting Slot Configurations for Homogeneous and Heterogeneous Hadoop Clusters. *IEEE Transactions on Cloud Computing*. doi:DOI 10.1109/TCC.2015.2415802

Yao, Z., Papapanagiotou, I., & Callaway, R. D. (2015). *Multi-Dimensional Scheduling in Cloud Storage Systems.* Paper presented at the International Communications Conference (ICC), (pp. 395-400). IEEE..

Yong, M., Garegrat, N., & Mohan, S. (2009). *Towards a resource aware scheduler in hadoop.* Paper presented at the Proceedings of the 2009 IEEE International Conference on Web Services, Los Angeles, CA, USA.

Yu, J. (2007). QoS-based scheduling of workflows on global grids. Retrieved from https://minerva-access.unimelb.edu.au/handle/11343/39376.

Yu, J., & Buyya, R. (2005). A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing, 3*(3-4), 171-200.

Yu, J., & Buyya, R. (2006). Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming, 14*(3-4), 217-230.

Zaharia, M. (2009). Job scheduling with the fair and capacity schedulers. *Hadoop Summit, 9*.

Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., & Stoica, I. (2010). *Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling.* Paper presented at the Proceedings of the 5th European conference on Computer systems, (pp. 265-278). ACM.

Zaharia, M., Konwinski, A., Joseph, A. D., Katz, R. H., & Stoica, I. (2008). *Improving MapReduce Performance in Heterogeneous Environments.* Paper presented at the OSDI. (Vol. 8, No. 4, p. 7).

Zhang, F., Cao, J., Li, K., Khan, S. U., & Hwang, K. (2014). Multi-objective scheduling of many tasks in cloud platforms. *Future Generation Computer Systems, 37*, 309-320.

Zhang, W., Rajasekaran, S., Wood, T., & Zhu, M. (2014). *Mimp: Deadline and interference aware scheduling of hadoop virtual machines.* Paper presented at the Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on. (pp. 394-403). IEEE.

Zhang, X., Zhong, Z., Feng, S., Tu, B., & Fan, J. (2011). *Improving data locality of MapReduce by scheduling in homogeneous computing environments.* Paper presented at the Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on.

Zhang, Z., Cherkasova, L., Verma, A., & Loo, B. T. (2013). Performance modeling and optimization of deadline-driven pig programs. *ACM Transactions on Autonomous and Adaptive Systems (TAAS), 8*(3), 14.

Zhifeng, X., & Yang, X. (2013). Security and Privacy in Cloud Computing. *Communications Surveys & Tutorials, IEEE, 15*(2), 843-859.

Zitzler, E., Laumanns, M., Thiele, L., Zitzler, E., Zitzler, E., Thiele, L., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm: Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK) Zürich, Switzerland.

# APPENDIX A

## MYCLOUD USAGE

MYREN cloud service has been used to conduct an experiment of the proposed solution. The following are the screenshot of the account and the number of the virtual machines used in this research. The account created from 28/02/2013 to 02/28/2017.


**Figure 6-1: The main homepage of MyRen Cloud service provider**

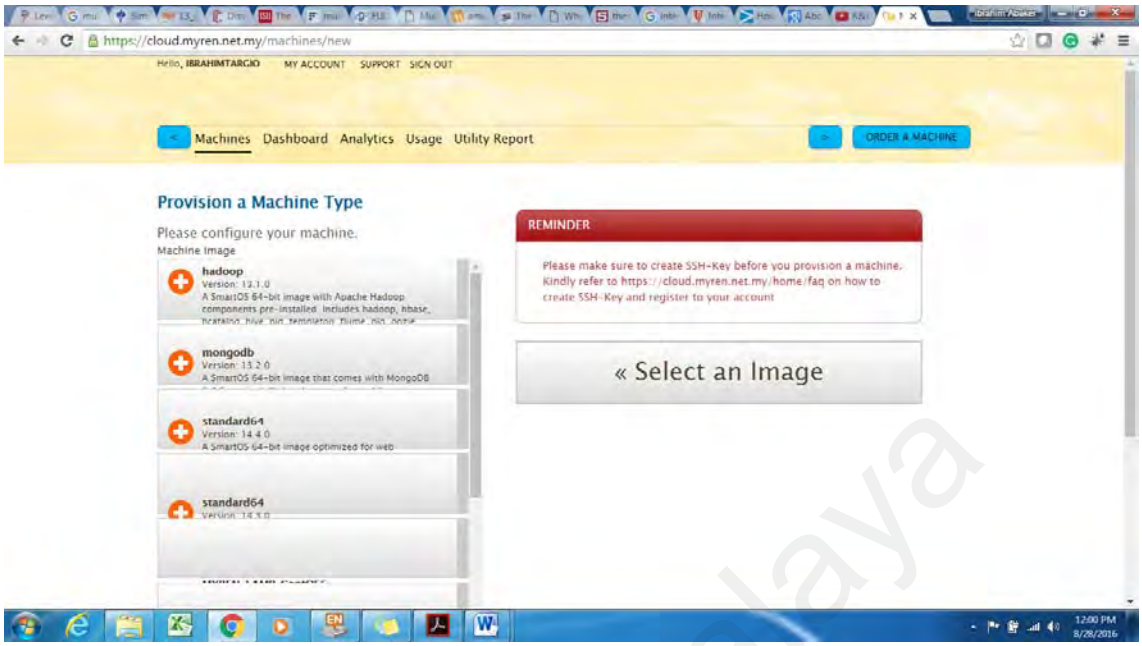



**Figure 6-2: The main Dashboard of login**

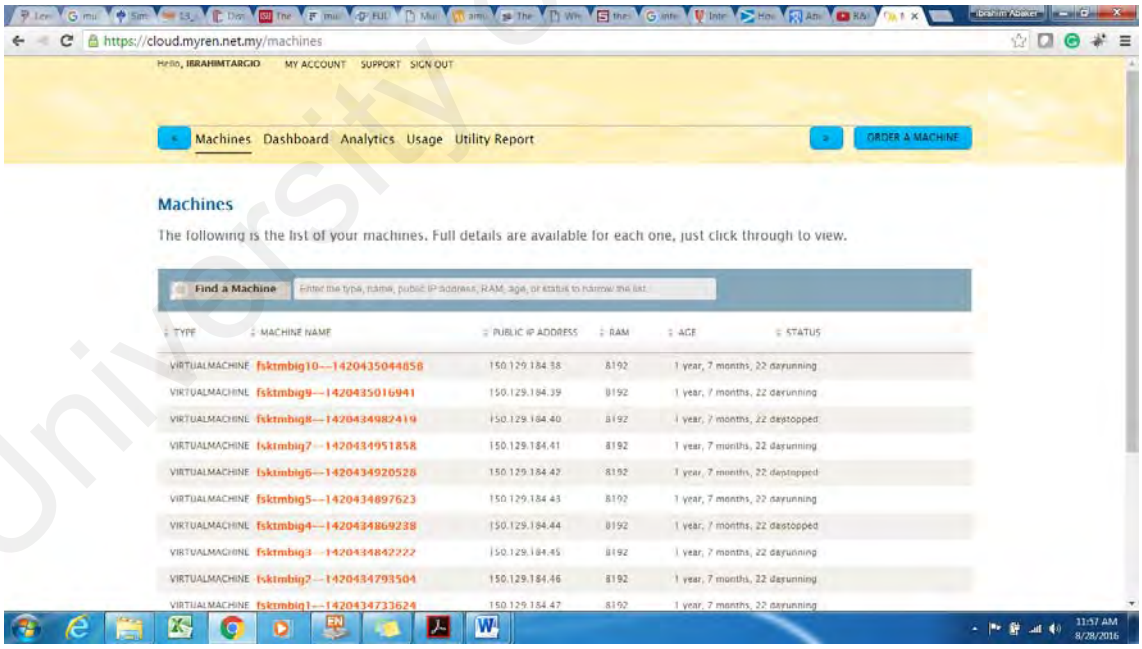



**Figure 6-3: The number of virtual machines provisioned**

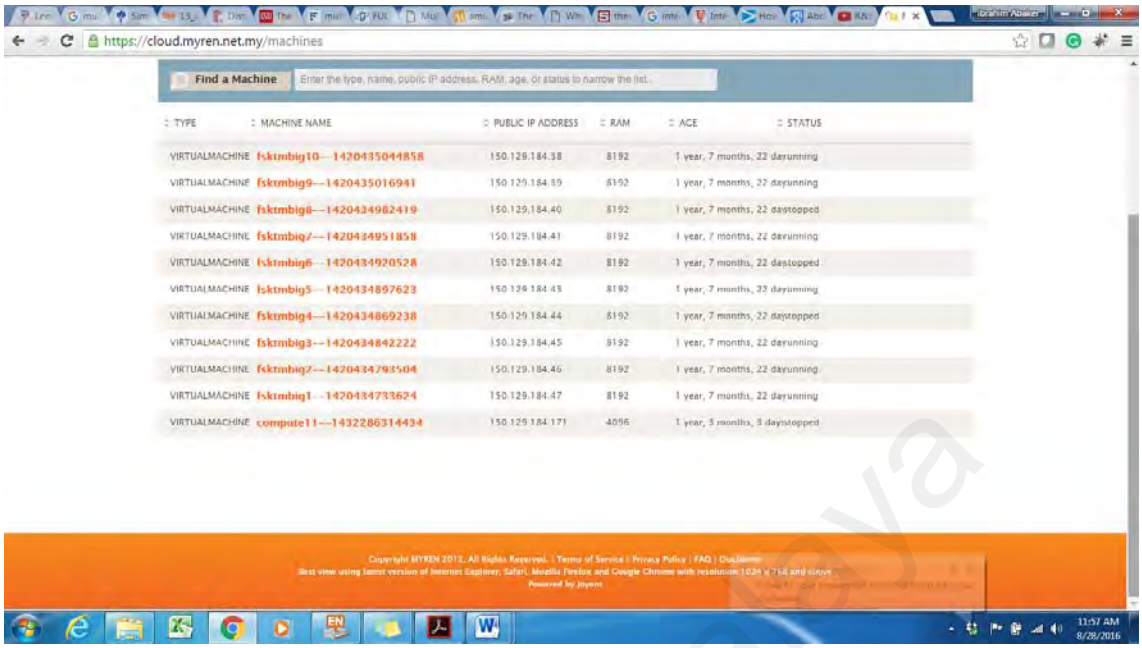



**Figure 6-4: The number of virtual machines provisioned**

## LIST OF PUBLICATIONS AND PAPERS PRESENTED

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of “big data” on cloud computing: Review and open research issues. Information Systems, 47, 98-115.

Hashem, I. A. T., Chang, V., Anuar, N. B., Adewole, K., Yaqoob, I., Gani, A., ... & Chiroma, H. (2016). The role of big data in smart city. International Journal of Information Management, 36(5), 748-758.

Hashem, I. A. T., Anuar, N. B., & Gani, A. (2015). Schedule optimization for big data processing on cloud. Paper presented at the 2nd International Conference on Big Data Analysis and Data Mining, J Data Mining Genomics Proteomics.

Hashem, I. A. T., Anuar, N. B., Gani, A., Yaqoob, I., Xia, F., & Khan, S. U. (2016). MapReduce: Review and open challenges. Scientometrics, 1-34.

Khan, N., Yaqoob, I., Hashem, I. A. T., Inayat, Z., Mahmoud Ali, W. K., Alam, M., ... & Gani, A. (2014). Big data: survey, technologies, opportunities, and challenges. The Scientific World Journal, 2014.

Yaqoob, I., Chang, V., Gani, A., Mokhtar, S., Hashem, I. A. T., Ahmed, E., ... & Khan, S. U. (2016). Information fusion in social big data: Foundations, state-of-the-art, applications, challenges, and future research directions. International Journal of Information Management.

Hashem, I. A. T., Anuar, N. B., Gani, A, Sakariyah, .A .K., Sangaiah, .A .K. 'Multi-objective Scheduling of MapReduce Jobs in Big Data Processing', Multimedia Tools and Applications, (Accepted with minor).