LOW LATENCY FAST DATA COMPUTATION SCHEME FOR MAP REDUCE BASED CLUSTERS

AISHA SHABBIR

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy

School of Computing
Faculty of Engineering
Universiti Teknologi Malaysia

JUNE 2020

DEDICATION

To my parents (late), who dreamt to see their daughter as a PhD Doctorate and serving humanity at her best. Their highly motivating words (said earlier), intrigued me to continue my research in the moments of dejection and almost leaving the assignment. I would like to extend this dedication to my siblings who helped me to fulfil my parents dream.

ACKNOWLEDGEMENT

First of all, I would like to express my deepest gratitude to ALLAH almighty the most affectionate and merciful, for HIS countless blessings upon me. I offer my praises to the most compassionate and glorious personality, Hazrat MUHAMMAD (Peace Be upon Him), who is a role model for the entire world.

I would like to express my immense gratefulness and appreciation to my supervisor **Prof. Ts. Dr. Kamalrulnizam Abu Bakar** for his timely contributions, meticulous supervision and guidance throughout my research journey. It has been a great opportunity to learn a lot from him during the course of this work. To keep my research work in the right direction, the credit of excellent reviews and suggestions should go to my supervisor Prof. Ts. Dr. Kamalrulnizam Abu Bakar and co-supervisor Dr. Raja Zahilah binti Raja Mohd. Radzi. I would like to extend my special thanks and appreciation to my co-supervisor Dr. Raja Zahilah binti Raja Mohd. Radzi for her guidance and for some research grant.

My heartiest thanks to my siblings and my husband Muhammad Siraj for their continuous support and motivation which always lightened my way to success. Without their Duaa, love and encouragement, I would never be able to finish this thesis. Also, I am indebted to my senior and great friend Dr. Tasneem for her continuous support and guidance for the research work and every matter. I would like to thank and acknowledge the support and help of my lab mates, friends and research group fellows. I am grateful to Universiti Teknologi Malaysia and all its staff for their support and help.

ABSTRACT

MapReduce based clusters is an emerging paradigm for big data analytics to scale up and speed up the big data classification, investigation, and processing of the huge volumes, massive and complex data sets. One of the fundamental issues of processing the data in MapReduce clusters is to deal with resource heterogeneity, especially when there is data inter-dependency among the tasks. Secondly, MapReduce runs a job in many phases; the intermediate data traffic and its migration time become a major bottleneck for the computation of jobs which produces a huge intermediate data in the shuffle phase. Further, encountering factors to monitor the critical issue of straggling is necessary because it produces unnecessary delays and poses a serious constraint on the overall performance of the system. Thus, this research aims to provide a low latency fast data computation scheme which introduces three algorithms to handle interdependent task computation among heterogeneous resources, reducing intermediate data traffic with its migration time and monitoring and modelling job straggling factors. This research has developed a Low Latency and Computational Cost based Tasks Scheduling (LLCC-TS) algorithm of interdependent tasks on heterogeneous resources by encountering priority to provide cost-effective resource utilization and reduced makespan. Furthermore, an Aggregation and Partition based Accelerated Intermediate Data Migration (AP-AIDM) algorithm has been presented to reduce the intermediate data traffic and data migration time in the shuffle phase by using aggregators and custom partitioner. Moreover, MapReduce Total Execution Time Prediction (MTETP) scheme for MapReduce job computation with inclusion of the factors which affect the job computation time has been produced using machine learning technique (linear regression) in order to monitor the job straggling and minimize the latency. LLCC-TS algorithm has 66.13%, 22.23%, 43.53%, and 44.74% performance improvement rate over FIFO, improved max-min, SJF and MOS algorithms respectively for makespan time of scheduling of interdependent tasks. The AP-AIDM algorithm scored 66.62% and 48.4% performance improvements in reducing the data migration time over hash basic and conventional aggregation algorithms, respectively. Moreover, an MTETP technique shows the performance improvement in predicting the total job execution time with 20.42% accuracy than the improved HP technique. Thus, the combination of the three algorithms mentioned above provides a low latency fast data computation scheme for MapReduce based clusters.

ABSTRAK

Kluster berasaskan *MapReduce* adalah paradigma yang muncul untuk menganalisis data besar bagi meningkatkan dan mempercepatkan klasifikasi, penyiasatan dan pemprosesan data besar dalam jumlah yang banyak, besar dan kompleks. Salah satu daripada isu asas dalam pemprosesan data besar dalam kluster *MapReduce* adalah menangani kepelbagaian sumber, terutama ketika ada kebergantungan data antara tugasan. Kedua, *MapReduce* menjalankan tugas dalam banyak fasa; trafik data perantaraan dan masa penghijrahannya menjadi hambatan utama untuk pengiraan kerja yang menghasilkan data perantaraan yang banyak dalam fasa perombakan. Seterusnya menghadapi faktor-faktor untuk memantau isu kritikal pertelingkahan adalah perlu kerana ia menimbulkan kelewatan yang tidak perlu dan menimbulkan kekangan yang serius terhadap prestasi keseluruhan sistem. Oleh itu, kajian ini bertujuan untuk menyediakan skema pengiraan data besar kurang tangguhan yang memperkenalkan tiga algoritma untuk menangani perhitungan tugas yang saling bergantung antara sumber-sumber yang pelbagai, mengurangkan trafik data perantaraan dengan masa penghijrahan dan pemantauan dan permodelan faktor-faktor pertelingkahan kerja. Kajian ini telah membangunkan algoritma Penjadualan Tugasan Berasaskan Kurang Tangguhan dan Kos Komputasi (LLCC-TS) bagi tugasan saling bergantung pada sumber yang pelbagai dengan memberi keutamaan untuk menyediakan penggunaan sumber yang menjimatkan kos dan pengurangan masa. Seterusnya, Algoritma Migrasi Data Perantaraan Dipercepat yang Berasaskan Pengumpulan dan Pemisahan telah dibentangkan untuk mengurangkan trafik data perantaraan dan masa pemindahan data dalam fasa rombakan dengan menggunakan pengumpul dan pemisah tersuai. Selain itu, Skema Penjumlahan Masa Ramalan *MapReduce* (MTETP) untuk pengiraan kerja *MapReduce* telah dihasilkan dengan menggunakan teknik pembelajaran mesin (regresi linear) dengan menghadapi faktor-faktor yang mempunyai kesan pada masa pengiraan kerja untuk memantau pertelingkahan kerja dan meminimumkan kelewatan. Algoritma LLCC-TS mempunyai kadar peningkatan prestasi 66.13%, 22.23%, 43.53% dan 44.74% berbanding FIFO, penambahbaikan max-min, SJF dan Algoritma MOS yang ditingkatkan masing-masing untuk jangka masa penjadualan tugasan saling-bergantungan. Algoritma AP-AIDM mencatat peningkatan prestasi 66.62% dan 48.4% dalam mengurangkan masa pemindahan data berbanding algoritma pengumpulan konvensional masing-masing. Selain itu, teknik MTETP menunjukkan peningkatan prestasi dalam meramalkan jumlah masa pelaksanaan kerja dengan ketepatan 20.42% daripada teknik HP yang ditambah baik. Oleh itu, kombinasi tiga algoritma tersebut memberikan skema pengiraan data besar kurang tangguhan untuk kelompok data besar berasaskan *MapReduce*.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A	-	Total number of aggregators
ACC	-	Average Computational Cost
ACT	-	Actual Execution Time
AM	-	Application Manager
B	-	Blocks
CC	-	Computational Cost
CCM	-	Computation Cost Matrix
CF	-	Contributing Factor
CV	-	Cross Validation
DAG	-	Directed Acyclic Graph Google File system
ETC	-	Expected Time of Completion
FARMS	-	Failure-Aware, Retrospective and Multiplicative Speculation
FIFO	-	First In First Out
GFS	-	Google File System
HDFS	-	Hadoop distributed file system
HOD	-	Hadoop On Demand
LATE	-	Longest Approximate Time to End
LT	-	Link Traffic
M	-	Mappers set
MAPE	-	Mean Absolute Percentage Error
MB	-	Megabyte
MM	-	Match Making
MOS	-	Multi-Objective Scheduling
MR	-	MapReduce
MTE	-	Minimum Task Execution time
NN	-	NameNode
NT	-	Network Traffic
P	-	Total number of partitions
PIR	-	Performance Improvement Rate (%)
R	-	Reducer Set

RM	-	Resource Manager
RMSE	-	Root Mean Square Error
SJF	-	Shortest Job First
SAMR	-	Self-Adaptive MapReduce scheduling
SMAC	-	SMAC (Social, Mobile, Analytics, and Cloud)
SUT	-	Start-Up Time
SZ	-	Total data size
Th	-	Throughput
YARN	-	Yet Another Resource Negotiator

LIST OF SYMBOLS

$ccm_{(i,k)}$	-	Element of Computation cost matrix
$d(x, y)$	-	Distance between nodes x and y
$e_{(i,j)}$	-	Element of ETC matrix
I_{data}	-	Input data size
m	-	Total number of tasks
mc_i	-	Binary variable for data transfer from mapper to in-node combiner
m_{ij}	-	Binary variable for data transfer from in-node combiner to aggregator
M_s	-	Machine Set
m_{ij}	-	Variable indicating whether data is processed from mappers
n	-	Total number of machines or processors
N_m	-	Number of map compute nodes
N_m^{slot}	-	Number of map slots
N_r	-	Number of reduce compute nodes
N_r^{slot}	-	Number of reduce slots
N_r^{wav1}	-	Number of reducers used in wave1 execution
$N_r^{wav_n}$	-	Number of reducers used in other waves execution
PP_M	-	Processing Power of the machine
r_{ju}	-	Binary variable for data transfer from aggregator to reducer
TA_i	-	Aggregated traffic of block after aggregation
TA_{P1}^1	-	Partition one of first aggregator
TA_{P2}^1	-	Partition two of first aggregator
$T_{available}$	-	Shortest possible time when Machine is available
T_{ready}	-	Time when machine starts its functionality
T_L	-	Task length
TMC_i	-	Traffic from one mapper to in-node combiner
TM_{ij}	-	Traffic of one mapper (after in-node combiner) to aggregator
TM_{B1}	-	Traffic for one virtual block after in-node combiner to aggregator
TM_B	-	Total network traffic from the Map Layer to Aggregation Layer (sum up from all blocks)
TR_{ju}	-	Traffic for reducer u coming from all aggregators placed in a virtual blocks

TR	-	Total network traffic from the Aggregation Layer to reduce layer
T_m^{total}	-	Map phase computation time
T_{con}^{total}	-	Time contribution by the CF
T_m^{avg}	-	Average map phase execution time
T_r^{avg}	-	Average reduce phase execution time
T_{sh}^{avg}	-	Average shuffle phase execution time
T_{wav1}^{avg}	-	Total average time for first wave
$T_{wav_n}^{avg}$	-	Total average time for other waves
Ts	-	Task set
T_{sh}^{total}	-	Shuffle phase computation time
T_r^{total}	-	Reduce phase computation time
V_{ki}	-	Total volume of the keys from mapper i
Y	-	Estimated dependent variable
β_0	-	Y-intercept
β_1	-	Slope of the regression line and co-efficient for X_1 independent variable
β_2	-	Slope of the regression line and co-efficient for X_2 independent variable

CHAPTER 1

INTRODUCTION

1.1 Overview

The present era has witnessed a dramatic advancement in scientific frontiers. MapReduce based clusters is an emerging paradigm for big data analytics to scale up and speed up the big data classification, investigation, and processing of the huge volumes, massive and complex data sets. A plethora of developments have occurred in digital technologies such as social media and networks, financial transactions, sensor data, business and financial dealings, and person to person communications through digital platforms. These developments resulted in the generation of massive amounts of data and production of such data in several formats like text messages, videos, sound, pictures, social content, XML, and so forth. Such data are growing enormously, leading to difficulties in storing, processing, and analysing them by using traditional databases and conventional tools and techniques. The pace of the evolution of currently integrated applications has posed a great challenge to the researchers to think critically and provide efficient design methodologies to handle such data. Big Data analytics is an emerging technology to extract useful information from substantial volumes and a variety of raw data (Thomas and Leiponen, 2016; Fernandez *et al.*, 2019). Fast data is the application of big data analytics to smaller data sets and it gathers and mine structured and unstructured data so that required action can be taken to get the desired result (Miloslavskaya, 2016).

The International Data Corporation Gens and Predictions claimed that there might be forty folds data growth from the year 2012 to 2020 and expected that it would double for every two years interval as depicted in Figure 1.1 (Gens *et al.*, 2017). This challenge urged the need for processing, storing, and investigating the

large bulk of data in almost all fields with the help of smart and efficient platforms and techniques. Furthermore, reducing time to analyze the growing amount of data, with less cost of the computational resources, is the biggest challenge faced by both researchers and industrialists (Salih *et al.,* 2019). The volume of data production is tremendous, and a significant part of the delivered data is not utilized because of the limited resources to store and process them. It is almost impossible to store all the data created due to the high cost. The processing issue is beginning to see an attainable horizon, yet at the same time has space to evolve. Thus the vital issue isn't to store all of the data produced but to extract meaningful information and process them efficiently with the given resources. The volume of data created is massive, and considerable amounts of the generated data have not been utilized due to the lack of resources to store and process them (Ahmad *et al.,* 2019). The need to store, investigate, and process the complex and huge sets of information-rich data is common to all fields of studies in the present age. For the huge amount of data, effective and efficient schemes and methodologies need to be developed to analyze and extract valuable information hidden within the data.

Several frameworks have been proposed for processing huge volumes of data. Some of the widely used frameworks are MapReduce (Goudarzi, 2017), Dryad (Microsoft, 2004; Isard *et al.,* 2007), Spark (Karau and Warren, 2017; García-Gil *et al.*, 2017), Dremel (Melnik *et al.,* 2010) and Pregel (Agneeswaran, 2014; Zhao *et al.*, 2016; Whang, 2018). The most well-known framework is MapReduce. MapReduce is emerging as an efficient big data processing platform (Kang *et al.,* 2015; Hashem *et al.,* 2018a). It has initially introduced by Google and it was designed for processing huge amounts of data by exploiting the parallelism among a cluster of machines. MapReduce provides an extendable and efficient data processing technique that significantly improves the performance of the massive data-driven applications (Koutroumpis *et al.,* 2017).

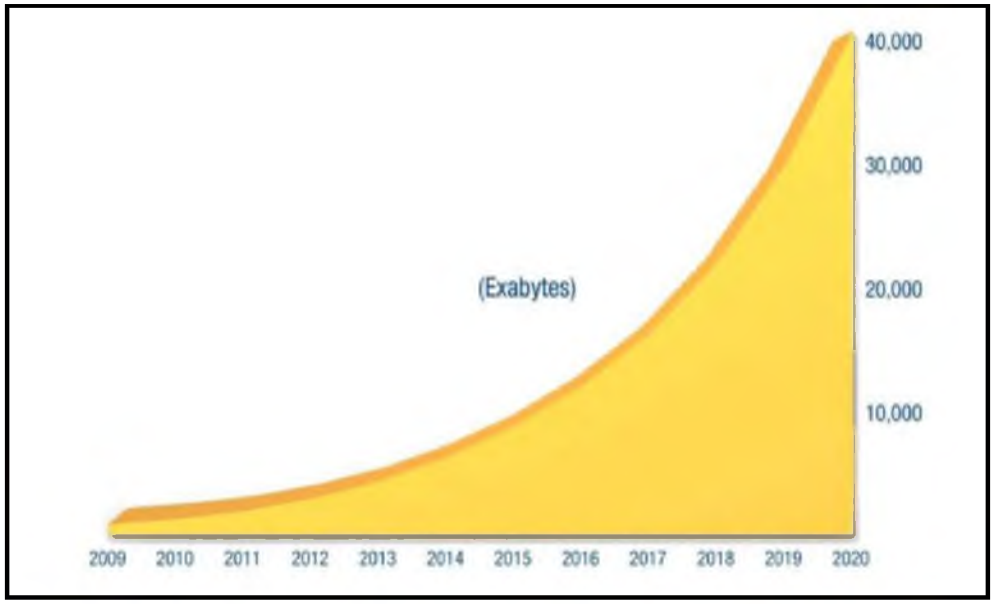

Figure 1.1 Data volume growth versus years (Gens *et al.*, 2017)

1.2 Problem Background

During the last few years, MapReduce has gained attention for data-intensive applications in a variety of fields (Amirian *et al.*, 2017; Espinosa *et al.*, 2019). Researchers from different fields utilized MapReduce to manage their large amounts of the data and their complexity (Hashem *et al.*, 2016). The massive amounts of data stored in a distributed fashion require processing in parallel (Chen and Zhang, 2014), such that new knowledge and innovation can be mined within an acceptable time span. Extracting meaningful and valuable information from huge datasets is important for providing attractive new services and improving the quality of the existing ones (Kambatla *et al.*, 2014). Data processing has been successfully adopted in a number of applications (Kobusińska *et al*, 2018) such as data mining, data analytics, scientific computation, and search engine. However, processing of massive amounts of data has been challenged by these applications because of the complexity of the data that should be processed and the scalability of the underlying algorithms that support such processes (Talia *et al.*, 2019). MapReduce is possibly the most popular framework for processing the existing large-scale data primarily because of

its important features that include scalability, fault tolerance, parallel processing and flexibility (Bonner *et al.,* 2017). Nowadays, MapReduce is used for expressing distributed computations on massive amounts of data. It is mainly based on the parallel processing of the computing infrastructure that exploits parallelism among computing infrastructure to handle many major issues introduced by the increasing use of vast amounts of data (Alabdullah *et al.,* 2018). The default implementation of MapReduce is Hadoop (Apache, 2008a).

The overall process workflow of the MapReduce is shown in Figure and explained as follows:

i. Map Phase: It is the first phase of the MapReduce, where the input data is collected and divided into sub-tasks to process it in parallel. The scheduling of the tasks over the compute mapper units are performed in the Map phase.
ii. Shuffle Phase: It is the intermediate phase of MapReduce. This phase transfers the output data of the mappers to the reduce units for its further processing.
iii. Reduce Phase: In this phase of the MapReduce, all the intermediate data is gathered and data reduction for the final outcome is performed, according to the requirements of the application or the program running over it. This phase also extracts the desired results (Deshai *et al.,* 2019).

For the past few years, big data corporations such as IBM, Google, Amazon, and Microsoft have set foot in cloud computing, and have provided some data processing solutions based on cloud computing services (IBM, 2018). MapReduce in conjunction with cloud computing is emerging as a promising solution to process large amounts of data sets (Géczy, 2014; Kobusińska *et al.*, 2018). MapReduce services in cloud, allows enterprises to process their data without dealing with the complexity of building and managing large installations of MapReduce platforms. Using virtual machines (VMs) and storage hosted in the cloud, enterprises can simply create virtual MapReduce clusters to process their data. A general view of the scenario, when different tasks have been submitted for execution on the given resources is shown in Figure 1.2.

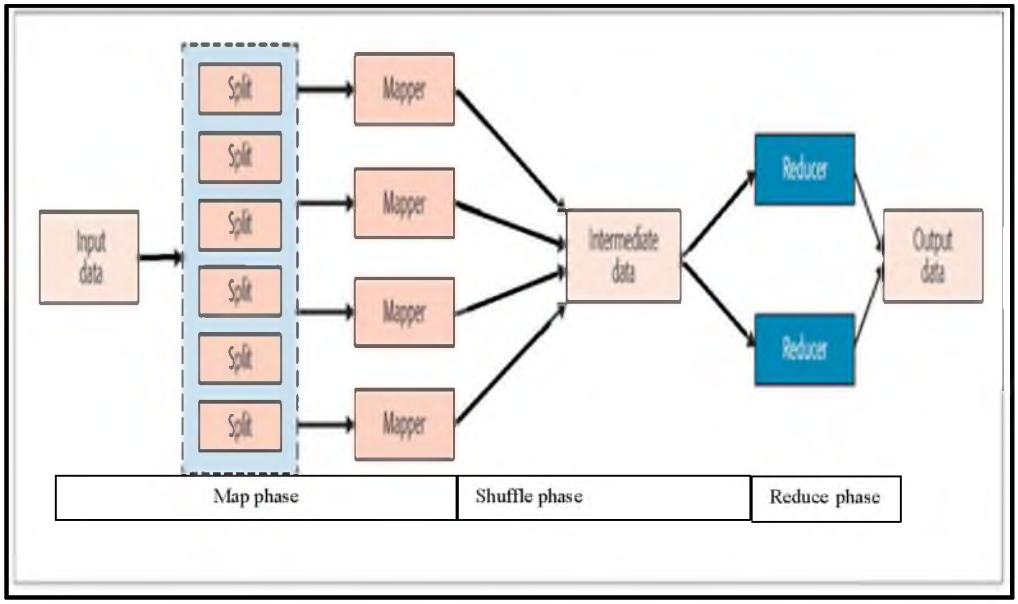

Figure 1.2 MapReduce workflow (Deshai *et al.*, 2019)

The group of machines having similarity for some specification constitutes a cluster. It can be seen from Figure 1.3 that the resources could be homogeneous or heterogeneous sets of machines. The submitted tasks for the execution could be independent or inter-dependent in nature. When different tasks are submitted for computation, then the next step is to assign these tasks to available resources for their computation. It is challenging to proficiently allocate a variety of assets to tasks of diverse nature QoS assurance, minimal makespan and effective resource utilization. Consideration and restriction of input data copies transfer between different phases of the MapReduce are important in the MapReduce framework (Afrati *et al.*, 2015) and thus it can help optimize the communication cost between map and reduce phases. There is a need to establish an efficient mapping scheme that can minimize the communication cost without affecting the performance of a specified task.

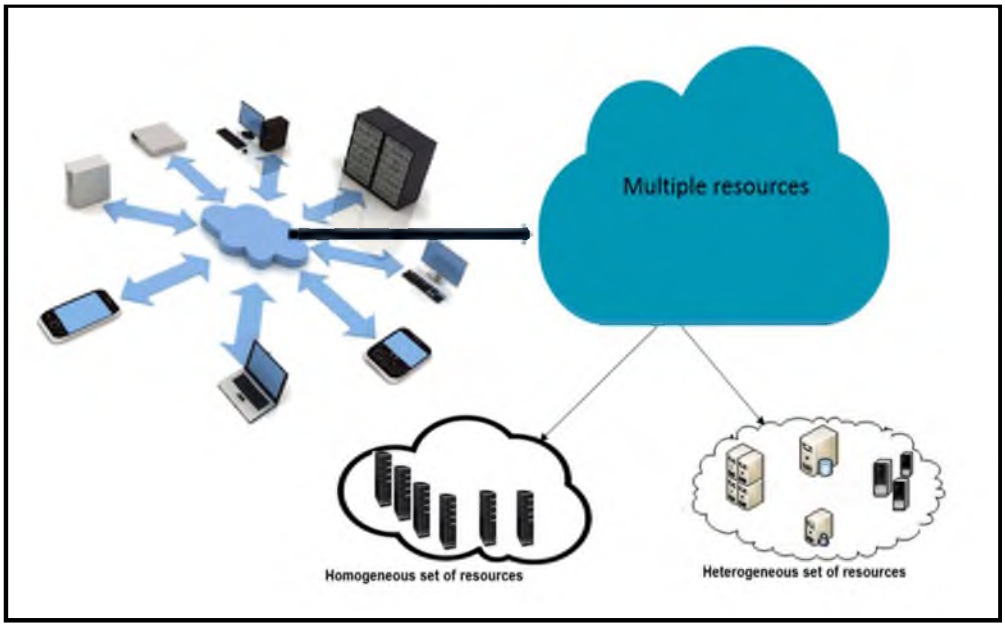

Figure 1.3 Resource distribution for tasks

In MapReduce, intensive disk input/output during the shuffling phase increases the overall execution time, which in turn degrades the performance of overall systems. 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. Research in this area can increase the performance of MapReduce by reducing the shuffle phase time. Moreover, in a distributed framework the impact of a stragglers on the speed of a parallelized processing can be quite significant. Specifically, it decreases resource utilization and increases job completion time. While performing configuration, an inappropriate approach can cause inefficient execution of jobs, which leads to performance degradation (Shi *et al.*, 2016). Creating such algorithms that receive input from the user, understanding the characteristics of underlying hardware. The following sections discussed the issues of fast data computations in MapReduce based clusters in detail.

1.2.1 MapReduce Workflow Scheduling Algorithms for Data Computations

MapReduce has been designed to accommodate large-scale data-intensive workloads running on large single-site homogeneous nodes. For MapReduce, the problem arises when computations have to be commenced across a different set of resources (Pandey and Saini, 2018). Some of the researchers provided certain solutions to extend the horizons of MapReduce for heterogeneous environments i.e., when there are different computing mappers used in terms of its capability to handle the task, but their schemes provide large computational overheads and in-efficient resource utilization. The main cause of in-efficient resource utilization is that tasks are allocated to the resources in a way that some resources (processing units) are overloaded and some are under-utilized (Gaur *et al.*, 2018). In addition to it, their priority criteria worked well either for small jobs or for long types of jobs. The vast piles of data are a consequence of the data from diverse sources. Moreover, it is imperative to proficiently compute diverse nature of tasks for a variety of compute nodes and to satisfy the customer's requirements at an acceptable time and computational cost without penalizing some resources with computational overheads.

Scheduling is one of the processing techniques which deals with the ordering and assignment of tasks to the available resources for execution. In the map phase, the scheduling of the tasks has been commenced. For instance, if some users submit their tasks for processing to a cloud data centre, which has some resources to execute the submitted tasks. A general view of the scenario would be like as shown in Figure 1.4. As can be seen from Figure 1.4, when different users submit their requests to a data centre to accomplish a job. Each user request is considered to be a task in the scenario and by the help of scheduler, will be assigned to a resource (processing machine) available in the data centre. If all tasks have been submitted at same time for the processing to a single data centre. Then, there is a need for a suitable scheduler to process all submitted requests in an appropriate manner with the objective of providing the services in a minimum time period.

If the submitted tasks have some inter-dependency then it is needed to handle these tasks in a proper way. Inappropriate handling of inter-dependent tasks during scheduling will not yield significant results which will eventually lead to the overall system's poor utilization. There is a need of finding an optimal assignment of a variety of tasks to the diverse set of resources (mappers) and it becomes more challenging when there is some inter-dependency among the tasks.

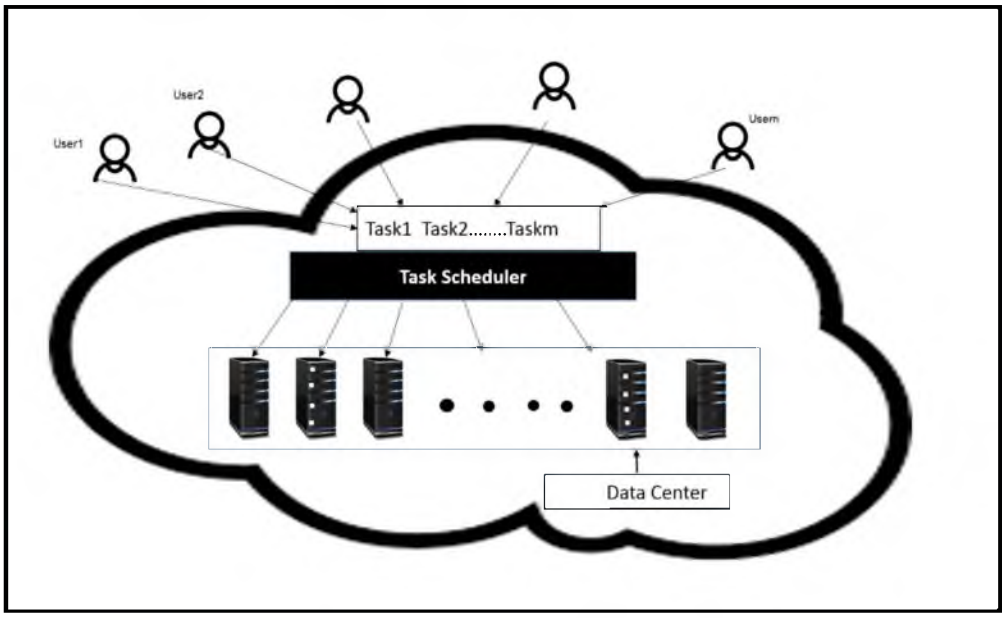

Figure 1.4 General view of the scenario of task submission

Many researchers put forward different scheduling algorithms like first in first out (FIFO). In this scheduler, all tasks are given to a solitary queue, and the task from the head of the queue is scheduled to the first available resource. FIFO the default scheduler of Hadoop MapReduce was having the similar bases of the First Come First Serve (FCFS) scheduler. The upside of the FIFO scheduler is that the algorithm is very simple and clear. However, it mostly brings starvation when the shorter tasks come after the long jobs/task. Thus, a short task may need to wait for a long time until the resource gets free. Another point is that some tasks give different execution times on different resources thus the scheduling of the first task on a second or other available machine may provide better results with minimum

makespan (Apache, 2008b). Many simulators like CloudSim (Calheiros *et al.,* 2010), Hadoop and iFogSim (Gupta *et al.,* 2017) used it as a default scheduler.

A scheduling algorithm named as Chess was proposed by Zacheilas and Kalogeraki (2016) which focuses more to increase the throughput by minimizing the overall cost. Their algorithm worked well for up to the certain addition of the jobs to the pool. However, their algorithm does not appear to produce the same quality of results among the high number of jobs like with the rapid increase in the number of jobs the search space of Chess grows exponentially. Thus, the cluster leads to the higher execution time. Further, Balagoni and Rao (2017) presented a multi-objective scheduling algorithm for heterogeneous environments. Their schemes mainly encountered the two factors i.e. load locality and fairness. Generally, to handle these two factors at one time is a tangible task. Their mechanism distributes the tasks by balancing the load ratio among the whole cloud environment. Unfortunately, their algorithms also among all others ignores the inter-task dependency issue.

The dependency factor was addressed by Tan *et al.* (2014) and their solution has provided an optimization scheme to handle the dependencies between map and reduce tasks and introduced the MapReduce enabled workflow scheduler. Their solution lacks effective resource utilization due to waiting for a long time for the dependent (map) tasks completion time. In addition, this scheduling algorithm fails to work if there is some inter-dependency among the input data side. A multi-target scheduling scheme of MapReduce jobs (MOS) was presented by Hashem *et al.* (2018b). Their scheduling algorithm focused more on the earliest finishing of the individual task and has not assigned priority to tasks. Thus, in this way, it results in higher schedule length (makespan) for the execution of all the map tasks. The scheduling algorithm lacks in the effective utilization of the heterogeneous resources because of ignorance of balancing the whole workload dispersal among the computing nodes.

In light of the aforementioned related works, it is revealed when map tasks computations have to be commenced on a heterogeneous set of resources, the

computational overhead increases with higher value of makespan. As can be deduced from the previously discussed related work, many of the existing scheduling algorithms which are designed for MapReduce sometimes result in in-efficient resource utilization and incurring high makespan and computational cost. In addition, the existing MapReduce scheduling algorithms have no consideration for task interdependencies on the input side (Carrillo, 2017), though the big data tasks are mostly diverse in nature and have dependencies among them. However, there are some studies which have considered dependencies between Map and Reduce tasks (Jlassi, 2015; Tan *et al.,* 2014). Thus, there is a need to provide a solution for the problem of the allocation of the variety of data centre assets to diverse nature of tasks proficiently with minimal makespan, computational cost and effective resource utilization.

1.2.2 Data Traffic Overhead of MapReduce Intermediate Phase Handling Algorithms

MapReduce as indicated by the name has two main phases that is. Map and Reduce. However, between the map and reduce phase, there exists another intermediate phase named as shuffle. In the shuffle phase, the output of mappers is migrated from the mappers to reducers. For the long type of jobs i.e. having a very large size of data sets, the intermediate data traffic produced in the shuffle phase is enormous as shown in Figure .

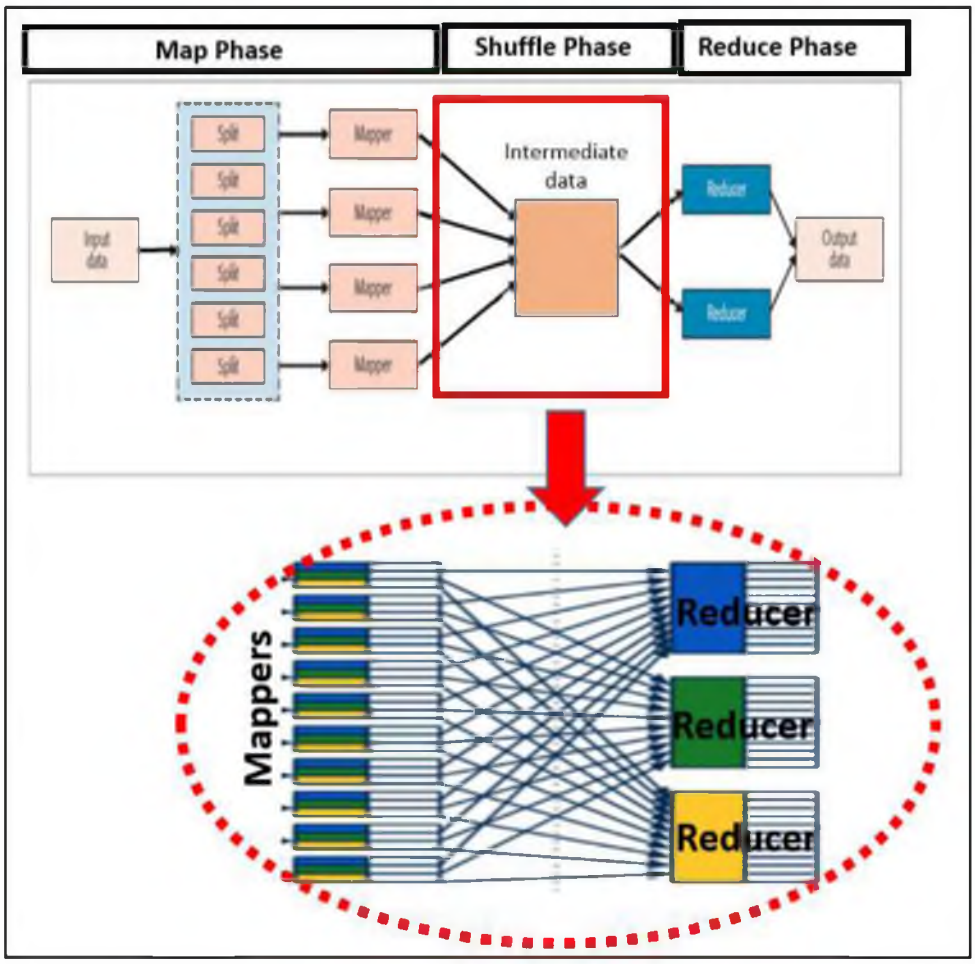

Figure 1.5 MapReduce intermediate traffic problem insight

Further, if all the intermediate data will be forwarded to the reducers directly, there will be huge traffic congestion and it deteriorates the overall MapReduce performance severely. Inappropriate migration scheme for large intermediate data transfer, results in extra migration time. Though several attempts have been made by researchers for the improvement of MapReduce performance, much attention has not been paid to the intermediate data traffic produced in the shuffle phase. Conventionally, distribution of metadata to the reducers was done by the hash function which does not encounter the data size associated with keys and results in high network traffic in the shuffle phase. Encountering and introducing the factors which can improve the shuffle's phase execution time is necessary.

To solve the traffic congestion issue, Ibrahim *et al.* (2010) have built up a reasonableness mindful key segment approach that monitors the circulation of the middle of the keys' frequencies. This algorithm has used a fairness-aware key partition approach that keeps track of the distribution of intermediate keys' frequencies. Thus, it results in excessive burden of key distribution records. Then, some researchers put forward the solutions which used the aggregators for reducing traffic. A specific kind of administration supplier that concentrates on the collection, and redistribution of data by reducing the similar entities are known as data aggregators. Authors Costa *et al.* (2012) used aggregation for the reduction of intermediate data but the scheme limits the use of servers by putting the condition of directly connected servers i.e. it was applicable to only those topologies in which servers have direct linkage with the other servers.

Moreover, Yan *et al.* (2013) have presented a sketch oriented information structure for catching MapReduce key size measurements and produced the groups for the allotment to reduce tasks. The compelling burden adjusting way to deal with information skew was put forward by Hsueh *et al.* (2014). But the both solution proposals were limited to single map task, disregarding the information accumulation openings from various mappers. A shuffle-aware scheduling algorithm has been presented by Ahmad *et al.* (2014) that lessens the measure of intermediate data to be rearranged by some shapes and dimensions to converge to reduce assignments. His work produces certain delays by elongating the traffic in different shapes for the heavy loaded jobs.

To tackle this problem Ke *et al.* (2015) incurred an aggregator placement scheme to reduce network traffic. This scheme has aggregated the mappers output by placing an aggregator for each mapper. Though the approach gave the better results for reducing the intermediate data traffic but at the cost of placement of multiple aggregators and therefore incurs high aggregation cost. In addition, it cascaded the aggregators thus can incur failure of one to many aggregator problem. Further, Lavanya (2016) proposed a solution for reducing the data traffic at the latter part of MapReduce with an Aggregator and Check function. However, the implementation of this scheme failed to work for complicated jobs with multiple requirements as it

needs the reducer's work by means of check function at the shuffle stage for its working. Wu (2018) proposed an incremental data allocation approach to reduce partition skew among reducers on MapReduce. This scheme worked for handling the load balancing among mappers and then reducer but due to creation a large number of micro-partitioning in the mapping phase and then passing of such splitted results for further steps in multiple rounds it increases the latency of job computations in MapReduce.

Based on the previous discussion, it can be concluded that all types of jobs computations in MapReduce based computing clusters follow the workflow as shown in Figure 1.2. In the Shuffle phase, when map phase output has to be transferred to Reduce Phase, the intermediate data migration of huge jobs, i.e. having very large size of data sets, generate high network overhead. Thus, the intermediate data migration of large meta data in the intermediate phase of MapReduce by an inappropriate migration algorithm results in extra migration time. Thus, for huge jobs the intermediate data traffic produced is enormous and its migration becomes a major bottleneck and decreases the performance of big data computations in MapReduce based computing clusters rigorously. However, to resolve this congestion and data migration time issue some aggregation schemes were proposed by researchers but their aggregation solutions produce high aggregation cost and lacked to balance aggregated intermediate data load among reducers. Moreover, sending data over the network without considering the contentious link results in high transmission cost. Accordingly, to transfer the intermediate data load with consideration of aggregation cost and contentious link, in a balanced way to reducers is necessary.

1.2.3 MapReduce Job Straggling Handling Techniques

Achievability of the results at the required time is desirable and important as well. It has been noticed that the performance of a big data computing cluster in terms of total job completion time often reduces by the delay in completion of one or a number of tasks. These tasks are called stragglers. There could be several reasons

for the tasks to be like a straggler. Firstly, the task could be allocated to an inappropriate computational node or it may be the hardware failure on which the task is running or it could be a factor which needs to be carefully tuned before the execution begins.

Some researches put forward some cloning based speculation execution solutions to handle the straggling problem by the detection of slow running tasks. Initially, Google MapReduce i.e. default implementation as Hadoop gave the solution which deploys the duplicates of the tasks near to the job completion (Apache, 2008a). That was pretty simple and easiest to implement but having the drawback that it produces un-required copies. Next, the Longest Approximate Time to End (LATE) scheme was proposed by Zaharia *et al.* (2008) which measures the progress rate for the tasks. It works on the idea of measuring the completion time of the slowest task. Major flaw of this algorithm was that it worked for the slow tasks and was unable to break the different phases of MapReduce during its progression. Another limitation of LATE was to impose restrictions on the backups.

Self-Adaptive scheduling scheme was proposed by Tang *et al.* (2015). It distinguishes the slower and faster tasks merely and does not provide any effective solution to handle the stragglers. Further, a Failure-Aware, Retrospective and Multiplicative Speculation (FARMS) is proposed by Fu *et al.* (2017). FARMS measure the responsiveness of each node. In case of detection of the unresponsive node, it copies the whole set of tasks running on that node to the other node. The major defect of the FARMS is that it replicates the whole tasks while the cause of the un-responsiveness might be one or more tasks not the whole number of tasks running on the given node. Thus, in this way it produces more replication overhead with the wastage of resources. Recently, Xu and Lau (2017) presented an optimization for speculative execution scheme to mitigate stragglers but their proposed scheme posed a limit on the size of the data and a ratio of stragglers just cut down under the threshold which may affect the system performance abnormally.

Herodotus proposed Starfish (Herodotou *et al.,* 2011) which collects the past executed jobs profile information at a fine granularity for job estimation and

automatic optimization. On the top of the Starfish, Herodotou proposed Elasticiser (Herodotou *et al.*, 2011) which provides Hadoop cluster resources in terms of VMs. However, collecting detailed job profile information with a large set of metrics generates an extra overhead, especially for CPU-intensive applications. As a result, Starfish overestimated the execution time of a Hadoop job.

Few researchers attempted to do predictive based speculation like Verma *et al.* (2011) and Chen *et al.* (2014) both tried to model the Hadoop MapReduce performance. The model name provided by Chen *et al.* (2014) was CRESP. Though, their schemes are using too many assumptions and they ignore the impact of the many factors which were encountered by the later studies. The HP model is restricted to a constant number of reduce tasks, whereas CRESP only considers the number of reduce tasks to be equal to the number of reduce slots. It is unrealistic to configure the parametric value as constant always rather it varies depending on the type of application (e.g. CPU intensive, or disk I/O intensive) and user requirements. The improved HP model/scheme presented by Khan *et al.* (2016) appears to be with a better prediction in contrast to others as it provided the modelling of one of crucial factors i.e., the varied number of reducers but it still ignores many factors which have significant impact on job estimation time. IoTDeM, an IoT Big Data-oriented model is proposed by Lu *et al.* (2018) for predicting MapReduce performances using machine learning techniques and Ceph which is a unified distributed storage system. The authors claimed by themselves in their paper that most of the research was done on HDFS (which is basically a core component of Hadoop MapReduce), thus they tried to model with different distributed storage system. The drawback of this approach is that it lost the generality and tied itself to specific application and architecture.

In the light of the previous discussion, it has been observed that for large scale data analysis as part of their core services for tasks such as log analysis, feature extraction or data filtering; Map-Reduce, through its Hadoop implementation, has proven to be an efficient framework. One important challenge when performing such analysis is to predict the performance of individual jobs. In addition, the big data computations faced latency issues and job completion got delayed due to straggling

factors. In order to deal with the stragglers, existing studies extensively use two types of speculative execution solutions for handling the stragglers, i.e., cloning based and prediction based straggler handling techniques. For cloning one, Researchers have proposed some strategies of backing up the slower tasks and speculating the performance of the individual computational nodes. On the other hand, some tried to model the different phase's times to monitor the tasks execution time through prediction phenomenon for handling the stragglers in advance. To enhance the performance of MapReduce, monitoring and handling the factors which could produce straggling is necessary. Thus, accordingly a scheme is required which tackles the contributing factors which could become the straggling cause, at the early stage of job computation.

1.3 Problem Statement

MapReduce based clusters is one of the emerging paradigms which enables the processing of massive volumes of data in parallel with many low-end computing nodes. This thesis addresses the neglected aspects of MapReduce by the existing studies which can significantly improve the data computations in MapReduce based clusters. Many scheduling algorithms were proposed, to improve the performance of big data computation in MapReduce based clusters but these algorithms resulted in longer makespan (scheduling length) and high processing latency leading to inefficient resource utilization and high computational and resource cost. Moreover, these algorithms do not consider the interdependency of tasks. In addition, some researchers attempted to reduce the network overhead created in the middle phase of MapReduce, nevertheless, their solutions generate high data migration time and very high aggregation and transfer cost. Moreover, to reduce the unnecessary processing delays in the MapReduce based clusters, the straggling issue was addressed by many researchers but their schemes does not encounter the straggling created due to inappropriate handling of some contributing factors which could become straggling cause at later stage. Thus, this research addresses aforementioned issues and provides a low latency fast data computation scheme which introduces a combination of

algorithms and technique, to handle inter-dependent task computation among heterogeneous resources, reducing intermediate data traffic with its migration time and monitoring and modelling job straggling factors.

1.4 Research Questions

Based on the discussion provided in Section 1.2. The research questions can be formulated as follows:

i. How to schedule inter-dependent tasks to the diverse set of resources with reduced makespan?
   a) How to handle the task inter-dependency while scheduling the jobs of MapReduce in a heterogeneous resources environment?
   b) How to minimize the latency issues for the task scheduling in map phase?
ii. How to reduce the data migration time and network overhead created for migration of large size intermediate data for the intermediate phase of MapReduce?
iii. How to minimize the big data processing latency in MapReduce due to job straggling?

1.5 Research Aim

This research aims at reducing the latency of data computations in MapReduce based clusters by providing the Low Latency Fast Data Computation (LLFDC) scheme which comprises the scheduling of the inter-dependent tasks in resource heterogeneous environment, reducing the network traffic by shuffle phase partitioning and predicting the completion time for MapReduce job computation to reduce latency and monitor job straggling.

1.6 Research Objectives

The following research objectives have been achieved throughout this research work:

i. To develop a workflow scheduling algorithm for inter-dependent MapReduce tasks for the heterogeneous environment with reduced makespan for the map phase and computational cost.
ii. To improve an intermediate data partitioning algorithm to reduce the data traffic and data migration time in MapReduce intermediate phase.
iii. To improve a time prediction technique for MapReduce job computation to monitor the job straggling and minimize the latency.

1.7 Research Contribution

The contributions of this research are summarized as follow:

i. Low Latency and Computational Cost based Tasks Scheduling (LLCC-TS) algorithm for the effective utilization of heterogeneous resources with minimizing cost and makespan along with the handling of the task interdependencies and improving the processing workflow of MapReduce based clusters.
ii. An Aggregation and Partition Based Accelerated Intermediate Data Migration (AP-AIDM) algorithm for the reduction of data traffic and data migration time in the MapReduce intermediate phase.
iii. MapReduce Total Execution Time Prediction (MTETP) technique for reducing the data processing latency in MapReduce due to job straggling factors.

1.8 Research Scope

The scope of the research covers the following.

i. The scheduling considered for the static set of data jobs with having the inter-dependency among its sub-tasks.
ii. The analysis of the interdependency is out of the scope of this research. The interdependent job cases put assumption on some sub-tasks that will must come after certain tasks followed the base restrictions from the latest literature for the dependent case of jobs.
iii. The nodes failure-related issues for the straggling is out of the research scope.

1.9 Significance of the Study

This research provides the improved performance of data computations in MapReduce based clusters and provision of the distribution strategy for the resources by developing the scheduling algorithms which encounters the interconnection of the jobs and gives the solution for effective resource utilization. The proposed algorithms can work well for both the homogeneous and heterogeneous environment. Moreover, it provides a basis to handle larger types of data sets for certain digital applications like Facebook and Tango. Furthermore, the partitioning scheme is suitable for processing the different types of jobs by reducing the extra overhead caused during the migration of the intermediate data. Thus, it would be beneficial for many of the business cases by processing the data in a less time with less overhead in order to respond the customers in an effective manner. In addition, the detection and resolving the problems of slow processing tasks leads the bigger frameworks for instance the power houses and grid stations to analyse their system's abnormal response in an optimistic way rather than shutting down the whole system.

1.10 Thesis Organization

The rest of the thesis organization is as follows:

Chapter 2 presents an extensive literature review of the study domain namely MapReduce based big data computing clusters. In addition, the most recent available solutions and their limitations are thoroughly examined and presented.

Chapter 3 describes the research methodology adapted to achieve the presented objectives. In addition, it provides an experimental environment used to achieve and verify these research objectives.

Chapter 4 provides the design, implementation and evaluation details of the proposed algorithm i.e. Low Latency and Computational Cost based Tasks Scheduling (LLCC-TS) algorithm for research objective number one. LLCC-TS algorithm is developed to reduce makespan time for the Map phase and computational cost.

Chapter 5 presents the design, implementation and evaluation details of the proposed algorithm i.e. Aggregation and Partition Based Accelerated Intermediate Data Migration (AP-AIDM) algorithm for research objective number two. AP-AIDM algorithm is developed to reduce data migration time and traffic overhead for shuffle phase.

Chapter 6 provides the design, implementation and evaluation details of the proposed technique i.e. MapReduce Total Execution Time Prediction (MTETP) technique for research objective number three. MTETP technique is designed to reduce big data processing latency in MapReduce due to job straggling.

Chapter 7 is the conclusion chapter which comprises the research achievements of this research work. In addition, it provides the limitations and future research directions for this research work.

REFERENCES

Afrati, F., Dolev, S., Sharma, S. and Ullman, J.D. (2015). Meta-MapReduce: A Technique for Reducing Communication in MapReduce Computations. *arXiv preprint arXiv:1508.01171*.

Ahmad, F., Chakradhar, S. T., Raghunathan, A., and Vijaykumar, T. N. (2014). ShuffleWatcher: Shuffle-aware Scheduling in Multi-tenant MapReduce Clusters. In Proceedings of the *USENIX Annual Technical Conference 2014 (USENIX ATC 14)*, 1–13.

Ahmad, R., Ahmad, T., Pal, B. L., and Malviya, S. (2019). Approaches for Semantic Relatedness Computation for Big Data. *SSRN Electronic Journal,* SSRN 3349564.

Alabdullah, B., Beloff, N.and White, M. (2018). Rise of Big Data-Issues and Challenges. In proceedings of the *21st Saudi Computer Society National Computer Conference (NCC),* IEEE, 1-6.

Althebyan, Q., Jararweh, Y., Yaseen, Q., Alqudah, O., and Al-Ayyoub, M. (2015). Evaluating Map Reduce Tasks Scheduling Algorithms Over Cloud Computing Infrastructure. *Concurrency and Computation: Practice and Experience*, 27(18), 5686-5699.

Amirian, P., van Loggerenberg, F., and Lang, T. (2017). Big Data and Big Data Technologies. In Proceedings of the *Big Data in Healthcare*, Springer, Cham, 39-58.

Ananthanarayanan, G., Ghodsi, A., Shenker, S., and Stoica, I. (2014). Effective Straggler Mitigation: Attack of the Clones. In Proceedings of the *10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI}* 13, 185-198.

Ananthanarayanan, G., Kandula, S., Greenberg, A., Stoica, I., Lu, Y., Saha, B., and Harris, E. (2010). Reining in the Outliers in Map-Reduce Clusters using Mantri. *Time*, 265–278.

Apache (2008a) Apache Hadoop, Avaialable Online : http://hadoop.apache.org/

Apache. (2008b). FIFO, Available Online: https://developer.ibm.com/articles/os-hadoop-scheduling/

Apache. (2008c). Fair Scheduler, Available Online: https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html

Apache. (2009). Capicity Scheduler, Available Online: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html

Arabnejad, V., & Bubendorfer, K. (2015). Cost Effective and Deadline Constrained Scientific Workflow Scheduling for Commercial Clouds. In Proceedings of the *14th International Symposium on Network Computing and Applications*, IEEE, 106–113..

Balagoni, Y., and Rajeswara Rao, R. (2017). Locality-Load-Prediction Aware Multi-Objective Task Scheduling in the Heterogeneous Cloud Environment. *Indian Journal of Science and Technology*, 10(9), 1–9.

Bechini, A., Marcelloni, F., and Segatori, A. (2016). A MapReduce Solution for Associative Classification of Big Data. *Information Sciences*, 332(C), 33–55.

Bellare, M., Goldreich, O. and Goldwasser, S. (1994). Incremental Cryptography: The Case of Hashing and Signing. In Proceedings of the *Annual International Cryptology Conference*, Springer, Berlin, 216-233.

Benjelloun, F. Z., and Lahcen, A. A. (2015). Big Data Security: Challenges, Recommendations and Solutions. *Web Services: Concepts, Methodologies, Tools, and Applications,* IGI Global, 25-38.

Bonner, S., Kureshi, I., Brennan, J. and Theodoropoulos, G. (2017). Exploring the Evolution of Big Data Technologies. *Software Architecture for Big Data and the Cloud*, Morgan Kaufmann,253-283.

Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D. and Freund, R. F. (2001). A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. *Journal of Parallel and Distributed Computing*, 61(6), 810–837.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., and Buyya, R. (2010). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and experience*, 41(1), 23-50.

Carrillo, G.E. and Abad, C.L. (2017). Inferring Workflows with Job Dependencies from Distributed Processing Systems Logs. In Proceedings of the *15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, IEEE, 1025-1030.

Chen, J., Wang, D., and Zhao, W. (2013a). A Task Scheduling Algorithm for Hadoop Platform. *Journal of Computers*, 8(4), .929-936.

Chen, K., Powers, J., Guo, S., and Tian, F. (2014). CRESP: Towards Optimal Resource Provisioning for Mapreduce Computing in Public Clouds. *IEEE Transactions on Parallel and Distributed Systems*, 25(6), 1403–1412.

Chen, Q., Liu, C. and Xiao, Z. (2013b). Improving MapReduce Performance Using Smart Speculative Execution Strategy. *IEEE Transactions on Computers*, 63(4), 954-967.

Condie, T., Conway, N., Alvaro, P., Hellerstein, J.M., Gerth, J., Talbot, J., Elmeleegy, K. and Sears, R. (2010). Online Aggregation and Continuous Query Support in Mapreduce. In Proceedings of the *SIGMOD International Conference on Management of Data*, ACM, 1115-1118.

Costa, P., Donnelly, A., Rowstron, A., and O'Shea, G. (2012). Camdoop: Exploiting in-Network Aggregation for Big Data Applications. In Proceedings of the *9th USENIX Conference on Networked Systems Design and Implementation*, USENIX Association, 1–3.

Cox, D. R., Kartsonaki, C., and Keogh, R. H. (2018). Big data: Some Statistical Issues. *Statistics and Probability Letters*, 136, 111–115.

Dahiphale, D., Karve, R., Vasilakos, A. V., Liu, H., Yu, Z., Chhajer, A., Wang, J. and Wang, C. (2014). An Advanced MapReduce: Cloud MapReduce, Enhancements and Applications. *IEEE Transactions on Network and Service Management*, 11(1), 101–115.

Deshai, N., Sekhar, B.V.D.S., Venkataramana, S., Srinivas, K. and Varma, G.P.S. (2019). Big Data Hadoop MapReduce Job Scheduling: A Short Survey. *Information Systems Design and Intelligent Applications*, Springer, 349-365.

DeZyre. (2015). Hadoop Ecosystem Components and Its Architecture. Available Online : https://www.dezyre.com/article/hadoop-ecosystem-components-and-its-architecture/114

Drakos, G. (2018). How to Select the Right Evaluation Metric for Machine Learning Models: Part 1 Regression Metric. *Towards Data Science*, 1–9.

Ekanayake, J., Gunarathne, T., and Qiu, J. (2011). Cloud Technologies for Bioinformatics Applications. *IEEE Transactions on Parallel and Distributed Systems*, 22(6), 998–1011.

Espinosa, J. A., Kaisler, S., Armour, F., and Money, W. (2019). Big Data Redux: New Issues and Challenges Moving Forward. In Proceedings of the *52nd Hawaii International Conference on System Sciences,* 1065-1074.

Feng, D. G., Zhang, M., & Li, H. (2014). Big Data Security and Privacy Protection. *Jisuanji Xuebao/Chinese Journal of Computers*, 37(1), 246–258.

Fernandez, A., Triguero, I., Galar, M., and Herrera, F. (2019). Guest Editorial: Computational Intelligence for Big Data Analytics. *Cognitive Computation*, 11, 329–330.

Foundation, T. A. S. (2008). MapReduce Tutorial. *Available Online:* https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.pdf. 1–43.

Frost, J. (2013). Regression Analysis: How Do I Interpret R-squared and Assess the Goodness-of-Fit? *The Minitab Blog,* 30.

Frost, J. (2016). How to Compare Regression Slopes. *The Minitab Blog*. Available Online : https://blog.minitab.com/blog/adventures-in-statistics-2/how-to-compare-regression-lines-between-different-models

Fu, H., Chen, H., Zhu, Y., and Yu, W. (2017). FARMS: Efficient Mapreduce Speculation for Failure Recovery in Short Jobs. *Parallel Computing*, 61, 68–82.

García-Gil, D., Ramírez-Gallego, S., García, S., and Herrera, F. (2017). A Comparison on Scalability for Batch Big Data Processing on Apache Spark and Apache Flink. *Big Data Analytics*, *2*(1), 1.

Gartner (2013). Gartner. Available Online: https://searchbusinessanalytics.techtarget.com/news/2240162412/New-Gartner-report-dissects-the-hype-around-big-data-technologies

Gaur, M., Minocha, B., and Muttoo, S. K. (2018). A Study of Factors Affecting Mapreduce Scheduling. *Advances in Intelligent Systems and Computing*, Springer Verlag, 654, 275–281.

Géczy, P. (2014). Big Data Characteristics. *The Macrotheme Review*, 3(6), 94–104.

Gemayel, N. (2016). Analyzing Google File System and Hadoop Distributed File System. *Research Journal of Information Technology*, 8(3), 66–74.

Gens, F., Philip, C., and Bill, F. (2017). IDC FutureScape: Worldwide IT Industry 2018 Predictions, Printerfriendly- US41883016.

Goudarzi, M. (2017). Heterogeneous Architectures for Big Data Batch Processing in MapReduce Paradigm. *IEEE Transactions on Big Data*, 5(1), 18–33.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., and Buyya, R. (2017). iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments. *Software: Practice and Experience*, *47*(9), 1275–1296.

Hashem, I. A. T., Anuar, N. B., Marjani, M., Ahmed, E., Chiroma, H., Firdaus, A., Abdullah, M.T., Alotaibi, F., Ali, W.K.M., Yaqoob, I. and Gani, A. (2018a). MapReduce Scheduling Algorithms: A Review. *Journal of Supercomputing*, 1-31.

Hashem, I. A. T., Anuar, N. B., Marjani, M., Gani, A., Sangaiah, A. K., and Sakariyah, A. K. (2018b). Multi-objective Scheduling (MOS) of MapReduce Jobs in Big Data Processing. *Multimedia Tools and Applications*, 77(8), 9979–9994.

Hashem, I. A. T., Anuar, N.B., Gani, A., Yaqoob, I., Xia, F. and Khan, S.U. (2016). MapReduce: Review and open challenges. *Scientometrics*, 109(1), 389-422.

Herodotou, H., Dong, F., and Babu, S. (2011). No One (Cluster) Size Fits All. In Proceedings of the *2nd ACM Symposium on Cloud Computing - SOCC '11*, ACM, New York, 1–14.

Herodotou, H., Lim, H., Luo, G., Borisov, N., and Dong, L. (2011). Starfish : A Self-tuning System for Big Data Analytics. *Cidr*, 11, 261–272.

Hsueh, S. C., Lin, M. Y., and Chiu, Y. C. (2014). A Load-Balanced Mapreduce Algorithm for Blocking-Based Entity-Resolution with Multiple Keys. In Proceedings of the *Research and Practice in Information Technology Series*, Australian Computer Society, 152, 3–9.

Hu, Y., Luo, C., Tang, T., Lou, J., Cai, H., and Li, J. (2010). Peer-to-Peer Aided Live Video Sharing System, *United States Patent,* US 7,733,808 B2.

Hyndman, R. J., and Koehler, A. B. (2006). Another Look at measures of Forecast Accuracy. *International Journal of Forecasting*, 22(4), 679–688.

Ibarra, O. H.,and Kim, C. E. (1977). Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors. *Journal of the ACM*, 24(2), 280–289.

IBM. (2018). Rides the Clouds. Available Online: https://www.thestreet.com/markets/ibm-claws-ahead-in-the-cloud-14656264

Ibrahim, S., Jin, H., Lu, L., Wu, S., He, B., and Qi, L. (2010). LEEN: Locality/Fairness-Aware Key Partitioning for MapReduce in the Cloud. In Proceedings of the Second International Conference on Cloud Computing Technology and Science, IEEE,17-24.

Ijaz, S., and Munir, E. U. (2018). MOPT: List-Based Heuristic for Scheduling Workflows in Cloud Environment. *The Journal of Supercomputing*, 75(7), 3740-3768.

Investopedia. (2019). R-Squared Definition. Available Online : https://www.investopedia.com/terms/r/r-squared.asp

Isard, M., Budiu, M., Yu, Y., Birrell, A., and Fetterly, D. (2007). Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. *Operating Systems Review (ACM)*, 59–72.

Jeffrey, D., and Sanjay, G. (2008). MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1), 107–113.

Jin, H., Yang, X., Sun, X. H., and Raicu, I. (2012). ADAPT: Availability-Aware Mapreduce Data Placement for Non-Dedicated Distributed Computing. In Proceedings of the *International Conference on Distributed Computing Systems*, 516–525.

JJ. (2016). MAE and RMSE — Which Metric is Better? – Human in a Machine World – Medium. Available Online : https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

Jlassi, A., Martineau, P. and Tkindt, V. (2015). Offline Scheduling of Map and Reduce Tasks on Hadoop Systems. Available Online : https://hal.archives-ouvertes.fr/hal-01324994

Kalavri, V., Brundza, V. and Vlassov, V.(2013). Block Sampling: Efficient Accurate Online Aggregation in Mapreduce. In Proceedings of the *5th International Conference on Cloud Computing Technology and Science*, IEEE, 250-257.

Kang, S. J., Lee, S. Y., and Lee, K. M. (2015). Performance Comparison of OpenMP, MPI, and MapReduce in Practical Problems. *Advances in Multimedia*, 1-10.

Karau, H., and Warren, R. (2017). High Performance Spark. *Best Practices for Scaling and Optimizing, Apache Spark*, 1-198.

Karagöz, E. (2014). *Incremental Hash Functions* (Doctoral Dissertation, Bilkent University).

Ke, H., Li, P., Guo, S., and Stojmenovic, I. (2015). Aggregation on the Fly: Reducing Traffic for Big Data in the Cloud. *IEEE Network*, 29(5), 17–23.

Khan, M., Jin, Y., Li, M., Xiang, Y., and Jiang, C. (2016). Hadoop Performance Modeling for Job Estimation and Resource Provisioning. *IEEE Transactions on Parallel and Distributed Systems*, 27(2), 441–454.

Kobusińska, A., Leung, C., Hsu, C. H., Raghavendra, S., and Chang, V. (2018). Emerging Trends, Issues and Challenges in Internet of Things, Big Data and Cloud Computing. *Future Generation Computer Systems*, 87, 416–419.

Kokilavani, T., and Amalarethinam, D. I. G. (2011). Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing. *International Journal of Computer Applications*, 20(2), 43-49.

Konjaang, J. K., Maipan-uku, J. Y., and Kubuga, K. K. (2016). An Efficient Max-Min Resource Allocator and Task Scheduling Algorithm in Cloud Computing Environment, *arXiv preprint arXiv:1611.08864*.

Koutroumpis, P., Leiponen, A., and Thomas, L. (2017). The (Unfulfilled) Potential of Data Marketplaces. *ETLA Working Papers* 2420.

Lavanya, S .R., and Sharma, A. (2016). Traffic Analysis in MapReduce, *International Journal of Computer Science and Mobile Computing,* 5(5), 765-770.

Lee, H., Kim, Y.W. and Kim, K.Y. (2018). Implement of MapReduce-based Big Data Processing Scheme for Reducing Big Data Processing Delay Time and Store Data. *Journal of the Korea Convergence Society*, 9(10), 13–19.

Lin, J., and Dyer, C. (2010). Data-Intensive Text Processing with MapReduce. *Synthesis Lectures on Human Language Technologies*, 3(1), 1-177.

Liroz-gistau, M., Pacitti, E., and Valduriez, P. (2013). MR-Part : Minimizing Data Transfers Between Mappers and Reducers in MapReduce. *Bases de Données Avancées*. (Bda), 1–16.

Lomotey, R. K., and Deters, R. (2014). Towards Knowledge Discovery in Big Data. In Proceedings of the *8th International Symposium on Service Oriented System Engineering, SOSE 2014*, IEEE, 181–191.

Lu, Z., Wang, N., Wu, J., and Qiu, M. (2018). IoTDeM: An IoT Big Data-oriented MapReduce Performance Prediction Extended Model in Multiple Edge Clouds. *Journal of Parallel and Distributed Computing*, 118, 316–327.

Luo, T., Zhu, Y., Wu, W., Xu, Y., and Du, D. Z. (2017). Online Makespan Minimization in MapReduce-like Systems with Complex Reduce Tasks. *Optimization Letters*, 11(2), 271–277.

Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., and Vassilakis, T. (2010). Dremel: Interactive Analysis of Web-Scale Datasets. In Proceedings of the *36th International Conference on Very Large Data Bases*, 330–339.

Microsoft. (2004). Dryad - Microsoft Research. Available Online : http://research.microsoft.com/en-us/projects/dryad/

Miloslavskaya, N. and Tolstoy, A. (2016). Application of Big Data, Fast Data, and Data Lake Concepts to Information Security Issues. In Proceedings of the *4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW),* IEEE,148-153.

Narayanapppa, M. T., Channabasamma, A., and Hegadi, R. S. (2018). Need of Hadoop and Map Reduce for Processing and Managing Big Data. *Web Services: Concepts, Methodologies, Tools, and Applications*, IGI Global 1588-1600.

Neelakandan, S., Divyabharathi, S., Rahini, S., and Vijayalakshmi, G. (2016). Large Scale Optimization to Minimize Network Traffic Using Mapreduce in Big Data Applications. In Proceedings of the *2016 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC)*, IEEE, 193–199.

Nita, M.-C., Pop, F., Voicu, C., Dobre, C., and Xhafa, F. (2015). MOMTH: Multi-Objective Scheduling Algorithm Of Many Tasks in Hadoop. *Cluster Computing*, 18(3), 1011–1024.

Oussous, A., Benjelloun, F. Z., Ait Lahcen, A., and Belfkih, S. (2018). Big Data Technologies: A Survey. *Journal of King Saud University - Computer and Information Sciences*, 30, 431–448.

Pandey, V., and Saini, P. (2018). How Heterogeneity Affects the Design of Hadoop MapReduce Schedulers: A State-of-the-Art Survey and Challenges. *Big Data*, 6, 72–95.

Patel, G., Mehta, R., and Bhoi, U. (2015). Enhanced Load Balanced Min-min Algorithm for Static Meta Task Scheduling in Cloud Computing. *Procedia Computer Science*, 57, 545–553.

Phinney, M., Lander, S., Spencer, M., & Shyu, C.-R. (2016). Cartesian Operations on Distributed Datasets Using Virtual Partitioning. In Proceedings of the *Second International Conference on Big Data Computing Service and Applications (BigDataService)*, IEEE, 1–9.

Prajapati, V. (2013). Big Data analytics with R and Hadoop : set up an integrated infrastructure of R and Hadoop to turn your data analytics into Big Data analytics. Packt Publishing. Available Online : https://dl.acm.org/doi/book/10.5555/2578622

Rashmi, S., and Basu, A. (2016). Deadline constrained Cost Effective Workflow Scheduler for Hadoop clusters in Cloud Datacenter. In Proceedings of the *International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, IEEE, 409-415).

Rasooli, A., and Down, D. G. (2014). COSHH: A Classification and Optimization based Scheduler for Heterogeneous Hadoop Systems. *Future Generation Computer Systems*, 36(36), 1–15.

Ren, X., Ananthanarayanan, G., Wierman, A., and Yu, M. (2015). Hopper: Decentralized Speculation-Aware Cluster Scheduling at Scale. In Proceedings of the *Special Interest Group on Data Communication - SIGCOMM '15*, ACM, New York, 45(4), 379–392.

Salih, F. I., Ismail, S. A., Hamed, M. M., Mohd Yusop, O., Azmi, A., and Mohd Azmi, N. F. (2019). Data Quality Issues in Big Data: A Review. *Advances in Intelligent Systems and Computing*, 843, 105–116.

Saraladevi, B., Pazhaniraja, N., Paul, P. V., Basha, M. S. S., and Dhavachelvan, P. (2015). Big Data and Hadoop-A Study in Security Perspective. *Procedia Computer Science,* 50, .596-601.

Shekhar, S., & Xiong, H. (2008). RMS Error. In *Encyclopedia of GIS* , 967–967. Available Online : http://statweb.stanford.edu/~susan/courses/s60/split/node60.html

Shi, L., Wang, Z., Yu, W. and Meng, X. (2017). A Case Study of Tuning MapReduce for Efficient Bioinformatics in the Cloud. *Parallel Computing*, 61, .83-95.

Sindhu, S., and Mukherjee, S. (2011). Efficient Task Scheduling Algorithms for Cloud Computing Environment. In Proceedings of the *International Conference on High Performance Architecture and Grid Computing*, Springer, Berlin, 79-83.

Singh, A. K., Shafique, M., Kumar, A., and Henkel, J. (2013). Mapping on Multi/Many-Core Systems. In Proceedings of the *50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, IEEE, 1-10.

Srinivas Agneeswaran, V. (2014). Big Data Analytics Beyond Hadoop: Real-Time Applications with Storm, Spark. Available Online : https://www.semanticscholar.org/paper/Big-Data-Analytics-Beyond-Hadoop%3A-Real-Time-with-Agneeswaran/c2ffbcd8c26f49670afde6b4b080c097d37f1c43

Stewllwagen, E. (2019). Welcome to Forecast Pro - Software for sales forecasting, inventory planning, demand planning, S&OP and collaborative planning. Available Online : http://www.forecastpro.com/Trends/forecasting101August2011.html

Taillard, J., Philip, P., and Bioulac, B. (1999). Morningness/Eveningness and the Need for Sleep. *Journal of Sleep Research*, 8(4), 291–295.

Talia, D. (2019). A View of Programming Scalable Data Analysis: from Clouds to Exascale. *Journal of Cloud Computing: Advances, Systems and Applications*, 8(4), 1-16.

Tan, J., Wang, Y., Yu, W. and Zhang, L. (2014). Non-work-Conserving Effects in MapReduce: Diffusion Limit and Criticality. *ACM SIGMETRICS Performance Evaluation Review*, 42(1), 181-192.

Tang, Z., Jiang, L., Zhou, J., Li, K., and Li, K. (2015). A Self-Adaptive Scheduling Algorithm for Reduce Start Time. *Future Generation Computer Systems*, 43(44), 51–60.

Tang, Z., Liu, M., Ammar, A., Li, K., and Li, K. (2016). An Optimized Mapreduce Workflow Scheduling Algorithm for Heterogeneous Computing. *The Journal of Supercomputing*, 72(6), 2059–2079.

Theme, H. (2019). RMSE: Root Mean Square Error - Statistics How To. Available Online : https://www.forecastpro.com/Trends/forecasting101August2011.html

Thomas, A., Krishnalal, G. and Raj, V.J. (2015). Credit Based Scheduling Algorithm in Cloud Computing Environment. *Procedia Computer Science*, 46, 913-920.

Thomas, L. D. W., and Leiponen, A. (2016). Big Data Commercialization. *IEEE Engineering Management Review*, 44(2), 74–90.

Tiwari, N., Sarkar, S., Bellur, U., and Indrawan, M. (2015). Classification Framework of MapReduce Scheduling Algorithms. *ACM Computing Surveys*, 47(3), 1–38.

Tsai, M.-Y., Chiang, P.-F., Chang, Y.-J., and Wang, W.-J. (2011). Heuristic Scheduling Strategies for Linear-Dependent and Independent Jobs on Heterogeneous Grids. In Proceedings of the *International Conference on Grid and Distributed Computing,* Springer, Berlin, 496-505.

Varoquaux, G. (2018). Cross-Validation Failure: Small Sample Sizes Leadto Large Error Bars. *Neuroimage*, 180, 68–77.

Verma, A., Cherkasova, L., and Campbell, R. H. (2011). Resource Provisioning Framework for MapReduce Jobs with Performance Goals. In Proceedings of the *International Conference on Distributed Systems Platforms and Open Distributed Processing*, Springer, Berlin, 165-186.

Wang, K., Zhou, X., Li, T., Zhao, D., Lang, M., and Raicu, I. (2014). Optimizing Load Balancing and Data-Locality with Data-Aware Scheduling. In Proceedings of the *International Conference on Big Data (Big Data)*, IEEE, 119–128.

Whang, K. Y. (2018). Recent Trends of Big Data Platforms and Applications. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11157 LNCS, 10–11.

Wu, D., Dai, Q., Liu, J., Li, B. and Wang, W. (2019). Deep Incremental Hashing Network for Efficient Image Retrieval. In Proceedings of the *Computer Vision and Pattern Recognition Conference on*, IEEE, 9069-9077.

Xu, H., and Lau, W. C. (2017). Optimization for Speculative Execution in Big Data Processing Clusters. *IEEE Transactions on Parallel and Distributed Systems*, 28(2), 530–545.

Yadwadkar, N.J., Hariharan, B., Gonzalez, J.E. and Katz, R. (2016). Multi-task Learning For Straggler Avoiding Predictive Job Scheduling. *The Journal of Machine Learning Research*, 17(1), 3692-3728.

Yan, W., Xue, Y., and Malin, B. (2013). Scalable and Robust Key Group Size Estimation for Reducer Load Balancing in MapReduce. In Proceedings of the *International Conference on Big Data*, IEEE, 156–162.

Yu, J. H., and Zhou, Z. M. (2019). Components and Development in Big Data system: A Survey. *Journal of Electronic Science and Technology*, 17(1), 51–72.

Zacheilas, N., & Kalogeraki, V. (2016). ChEsS: Cost-Effective Scheduling Across Multiple Heterogeneous Mapreduce Clusters. In Proceedings of the *International Conference on Autonomic Computing (ICAC)*, IEEE, 65–74.

Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R.H. and Stoica, I. (2008). Improving MapReduce Performance In Heterogeneous Environments. *Osdi, Static.Usenix.Org,* 8(4), 1-7.

Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S. and Stoica, I. (2010). Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling. In Proceedings of the *5th European conference on Computer systems*, 265-278.

Zhao, J., Tao, J.,and Streit, A. (2016). Enabling Collaborative MapReduce on the Cloud with a Single-Sign-on Mechanism. *Computing*, 98(1–2), 55–72.

Zhao, X., Chen, Y., Xiao, C., Ishikawa, Y., and Tang, J. (2016). Frequent Subgraph Mining Based on Pregel. *Computer Journal*, 59(8), 1113–1128.