Resource and Performance Modelling of Hadoop Clusters Using Machine Learning

by

Hassan Tariq

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2020

Abstract

There is a huge and rapidly increasing amount of data being generated by social media, mobile applications and sensing devices. Big data is the term usually used to describe such data and is described in terms of the 3Vs - volume, variety and velocity. In order to process and mine such a massive amount of data, several approaches and platforms have been developed such as Hadoop. Hadoop is a popular open source distributed and parallel computing framework. It has a large number of configurable parameters which can be set before the execution of jobs to optimize the resource utilization and execution time of the clusters. These parameters have a significant impact on system resources and execution time. Optimizing the performance of a Hadoop cluster by tuning such a large number of parameters is a tedious task. Most current big data modeling approaches do not include the complex interaction between configuration parameters and the cluster environment changes such as use of different datasets or types of query. This makes it difficult to predict for example the execution time of a job or resource utilization of a cluster. Other attributes include configuration parameters, the structure of query, the dataset, number of nodes and the infrastructure used.

Our first main objective was to design reliable experiments to understand the relationship between attributes. Before designing and implementing the actual experiment we applied Hazard and Operability (HAZOP) analysis to identify operational hazards. These hazards can affect normal working of cluster and execution of Hadoop jobs. This brainstorming activity improved the design and implementation of our experiments by improving the internal validity of the experiments. It also helped us to identify the considerations that must be taken into account for reliable results. After implementing our design, we characterized the relationship between different Hadoop configuration parameters, network and system performance measures.

Our second main objective was to investigate the use of machine learning to model and predict the resource utilization and execution time of Hadoop jobs. Resource utilization and execution time of Hadoop jobs are affected by different attributes such as configuration parameters and structure of query. In order to estimate or predict either qualitatively or quantitatively the level of resource utilization and execution time, it is important to understand the impact of different combinations of these Hadoop job attributes. You could conduct experiments with many different combinations of parameters to uncover this but it is very difficult to run such a large number of jobs with different combinations of Hadoop job attributes and then interpret the data manually. It is very difficult to extract patterns from the data and give a model that can generalize for an unseen scenario. In order to automate the process of data extraction and modeling the complex behavior of different attributes of Hadoop job machine learning was used. Our decision tree based approach enabled us to systematically discover significant patterns in data. Our results showed that the decision tree models constructed for different resources and execution time were informative and robust. They were able to generalize over a wide range of minor and major environmental changes such as change in dataset, cluster size and infrastructure such as Amazon EC2. Moreover, the use of different correlation and regression techniques, such as M5P, Pearson's correlation and k-means clustering, confirmed our findings and provided further insight into the relationship of different attributes and with each other. M5P is a classification and regression technique that predicted the functional relationships among different job attributes. The use of k-means clustering allowed us to see the experimental runs that shows similar resource utilization and execution time. Statistical significance tests, were used to validate the significance of changes in results of different experimental runs, also showed the effectiveness of our resource and performance modelling and prediction method.

Acknowledgments

I would like to express my gratitude to my supervisor Associate professor Dr. Ian Welch for his continuous support. I feel extremely privileged and proud to work with him. He is the best teacher, best supervisor, and best person in the whole world. I would also like to acknowledge the guidance provided by Dr Harith Al-Sahaf. He always provided in-depth and useful insight for the different problems during my PhD.

I would like to thank administrative staff of the school especially Patricia Stein. She is one of the most cooperative and student friendly person in the whole faculty. She has always given me right advice at the right time. I would never ever forget her valuable contribution in my life.

In the end I would like to thank my wife who back in Pakistan fulfilled the tedious task to bring up three young kids. She has always been a big support. I would like to thank my mother and father to constantly pray for me and pushing me to get there.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Big data is a term coined in 1990s represents data of volume, velocity and variety. Large volumes of data is being generated at unprecedented rate from variety of sources [1]. For example, in health and medical sciences where data is mostly unstructured and can be from variety of sources as well such as CT scans, X-Rays, claims, documents and machinery [2]. In businesses, the interaction and preferences of customers can be analyzed to reduce costs and get better recommendations [3]. The use of technological trends like Internet of Things, Cloud Computing and easy access to smart devices are the main contributors to generate big data [4]. In social life, data is being analyzed to understand the user behavior and interactions such as Facebook, Twitter, and LinkedIn [5]. Behind the scene, powerful systems and distributed computing are supporting cleaning, processing, analyzing, securing and granular access to massively evolving data [6,7].

Traditional data mining methods are not able to handle and analyze the big data in timely and cost effectively manner [8,9]. Web and social media alone is producing an overwhelming amount of data on daily basis which makes current big data infrastructures to be improved and optimized for different data types [10] [11] [12]. There have been a number of efforts to make current big data technologies more efficient, scalable and robust. These include improving load balancing and process scheduling capabilities across different environments [13] [14] [15] [16] [17] [18], improving security of data [19] [20] [21], optimized utilization of available data storage [22] [23], making of scalable and applied cloud technologies and developing performance matrices for distributed file systems [24–28]. Distributed computing paradigms, such as peers to peers, clusters, grids, and the cloud platforms, have been used for distributed data mining [29, 30]. Clusters are the system or group of systems that functions as one processing resource. The systems in clusters are usually linked with high-speed networks and are a suitable platform to perform distributed data mining tasks. In 2004, Google launched Map Reduce paradigm, which has become the most popular platform for distributed big data mining platforms [31]. It has two basic functions, Map and Reduce. Map generates key-value pairs on the basis of user defined functions and the Reduce phase groups them together to compute the output.

Apache Hadoop is an open source distributed and parallel processing framework serving many of big data applications. Hadoop is designed to reduce low performance and complexity encountered while processing and analyzing traditional Big data technologies. It relieves considerable communication load from network and servers [32]. Hadoop consists of its own filesystem named Hadoop Distributed File System (HDFS) and a MapReduce engine. HDFS employs a Master/slave architecture for both distributed computation and distributed storage. HDFS clusters consists of a single NameNode, a Master server that manages the file system namespace and regulates access to files by clients, and many DataNodes that manage storage issues along with Task Tracker and Job Tracker [33]. Copies of the same data exists in different slave nodes ensuring a fault tolerant working. This phenomena is called data replication [34]. The Job Tracker receives the job from the user and splits it into parts. The architecture involves storing data as blocks which can scale up to Gigabytes and Terabytes in the DataNodes [35].

1.1 Problem Statement

Optimizing different resources and performance of Hadoop cluster faces many challenges. These challenges motivated us to set our research objective. Following section highlights some of the challenges.

Hadoop cluster experimental setup design and configuration parameters tuning:

To optimize the performance of Hadoop cluster, we need to tune large number of configuration parameters [34]. Existing work has reported the impact of change in block size [18] and trying different replication strategies [26] to enhance throughput of the system. Each of these parameters can take more than one value. The tuning of Hadoop parameters not only require understanding of the system internals and knowledge of workload but also hardware characteristics of the cluster. Insufficient execution times and under utilization of resources often results due to misconfiguration of parameters. These configuration parameters may interfere each other in a complex way [36].

Furthermore, the performance of Hadoop is greatly dependent on the nature of application and is also influenced by the cluster design choices and inherent system characteristics [37]. So, to benchmark the performance of Hadoop, there have been some efforts to measure performance on different storage devices [38] and emerging hardware resources like Knights Landing clusters [39], characterization of cost-effectiveness on deploying big data systems under different hardware and software capabilities [40], and the impact of geographical diversity [41]. However, these experiments are often not repeatable or reproducible. Also the impact of confounding factors is not taken into account which can also impact when validating the system in external environment [42] [43]. So there are two important perspectives that need to be answered while tuning configuration parameters as mentioned by [44];

- *Which parameters to configure that can optimize the performance of Hadoop cluster?*
- *Analyzing how its performance and resource utilization is affected?*

It is necessary to understand which factors are important in designing and running experiments on Hadoop cluster and which confounding factors exist that can impact the performance and resource utilization of Hadoop cluster. We applied an experimental design approach, Hazard and Operability (HAZOP), to see the confounding factors in our Hadoop cluster. It was originally proposed by Lawley and was used to review the design of processes in chemical plants [45]. It not only identified the risks associated with a process but also highlighted the possible causes and potential consequences [46]. So, it is a structured way to correct and prevent possible causes of potential risk in the system by taking timely actions [47,48]. It was used in various studies such as in health sciences [49], identifying risk in waste pickers in Brazil [50], accident prevention system [51], risk analysis of radioactive storage tanks [52], human-robot interaction [53], risks assessment of bio-pharmaceutical industry [54], road safety measures [55]. This ensured the internal validity of the system while designing a robust experimental methodology. The method mainly relied on using the guide words in combination with process parameters. This highlighted the deviations of the process from intended normal process [56].

Modelling resource and performance based on job-characteristics:

Due to the large number of parameters involved, the prediction of execution time and resource utilization of Hadoop jobs is a difficult task. Heterogeneity of infrastructure, hardware/software errors and resource contention, stragglers can also occur. Traditional white-box based modeling approaches requires the good understanding of the system's internals such as operating system, workload and hardware.

Machine learning (ML) is a black-box technique which has the advantages of being flexible, robust and simpler to build [57]. The process involves training computers for the ability to automatically extract important information from data. The purpose is to search or predict for associations or patterns in data [58]. It is a multidisciplinary field of science incorporating computer science, statistics, information theory and artificial intelligence [59]. Machine learning has been used to model: energy utilization of clusters [60] and performance prediction of jobs [61]. There is also some work reported to optimize the use of resources utilization [62], but the potential of using machine learning to model characteristics of Hadoop job for diverse system resources is still largely an unexplored area of research. To solve this issue, we used machine learning approaches like decision trees (DT).

DT is a supervised learning technique used for the modelling of diverse problems such as fault detection [63–66], speech recognition [67], prediction of system vulnerabilities [68], mining gene expression data [69], modelling diabetes and asthma based on neighborhood [70], modelling macroeconomics factors in air-traffic [71], predicting obesity based on body mass index [72]. DT is one of the most commonly used data mining tool because they are flexible, robust and easy to build and interpret [73–75]. It can work with variety of data formats and predicts the relationships of important confounding attributes in the form of tree [76]. DT gives non-parametric models and enables us to simultaneously model large number of variables [77].

Correlation of factors affecting resource utilization of Hadoop clusters:

Execution time and resource utilization of Hadoop jobs is affected by important configuration parameters like block size, replication factor and number of mappers, along with structure of query, choice of dataset and cluster size. Correlation is a measure of how closely a set of variables are associated [78]. It are usually expressed in the form of a number between 0 and 1, where 0 is not associated and 1 is strongly correlated. Correlation has been applied but not limited to calculate the association between: disease and genes in health sciences [79], factors affecting vehicle-based air pollution [80], rainfall and twitter response data [81].

DT provided a good insight into how different factors are linked to resource utilization prediction. But DT sometimes become unstable and insensitive due to stochastic nature of data [82]. Also, DT does not provides functional relationship between the factors affecting resource utilization and execution time of Hadoop cluster jobs. M5P is a Classification and Regression Tree (CART) method, to find the linear relationship between confounding factors [83]. It also has the advantage of being able to handle continuous variables and missing data. The leaves of the tree are multiple linear regression models [84]. It has been used in various studies such as estimating hybrid energy consumption [85], maximum power point trackers [86], prediction of wind power [87,88], modelling and predicting traffic accidents duration on urban freeway [89], predicting concrete strength [90], prediction of mechanical properties of leaf [91], predict precipitation index [92].

Clustering is an unsupervised machine learning technique used to classify unlabelled objects into same group or cluster [93, 94]. K-mean clustering algorithm is widely used method due to its simplicity and fast convergence [95,96]. The main objective is to minimise the within-group sum of squares [97]. Data objects gets associated with the centroid closest to them [98, 99]. It has been used in various studies such as to classify seismic wave arrival times [100], to locate the earthquake sources [101], kidney disease detection [102], deriving hydro-power reservoir's operational rules [103], detecting geochmemical anomaly [104], rice yield estimation [105], fault diagnosis [106], image segmentation [107].

1.2 Research goal and objectives

This section describes motivation, overall research goal, research question and objectives of the thesis.

The design of a robust experimental methodology includes but not limited to the number of nodes, design of the cluster, data size, type of input, number of experiments, parameter inter-relationship, statistical significance of the results, the dataset used [14, 15, 108–112]. Absence of robust methodology can be a major factor in non-reproducibility of the results at some later stage by same author or someone else [42, 113–116]. HAZOP is one such method that can be applied to the experimental design [117]. This could increase the reliability of experiments by ensuring both internal and external validity of results [42, 113–116]. Simulation modelling techniques, such as trace driven, event driven, execution driven, complete system simulation and software profiling, and analytical modelling techniques, such as probabilistic models, queuing model, Markov models and Petri Net models, are often computationally expensive and less interpretable [118]. While experimental approach such as performance measurement, often includes hardware/ software monitoring, workload characterization, capacity planing and benchmarking, are more interpretable, less expensive and easy to build [119].

To optimize Hadoop job performance and resource utilization, we should carefully select from an extensive list of 200 parameters [18]. Other attributes such as job input, dataset, cluster size and infrastructure can also be used to model its performance and resource utilization [18] [26] [14] [15] [62] [120]. The inter-related impact of these attributes can be hard to predict manually. Automated procedures such as machine learning can be effectively used to model such complex relationship [60]. Such a correlation between attributes can be useful to cluster designers and managers to optimize its performance and resource utilisation [61] [62].

The overall research goal is to **model and predict Hadoop job performance and resource utilization**. To achieve this we designed the following research questions:

1. **What quantification can be done for the configuration parameters affecting Hadoop job performance and resource utilization?**
2. **What impact the changes in infrastructure and job type has on Hadoop job performance and resource utilization?**
3. **In what way machine learning can be applied to model Hadoop job performance and resource utilization?**

We used different machine learning techniques such as decision trees, M5P trees and k-mean clustering. More specifically this research addressed the following sub-objectives:

Objective 1: Design a robust experimental methodology to characterize the factors affecting Hadoop performance This objective highlighted different aspects of designing and running experiments on Hadoop clusters for its performance characterization. Some important questions in this regard were:

- What confounding factors exist in designing experiments on Hadoop cluster?

- which possible hazards can cause deviations from normal functioning?
- What are the possible consequences caused by these deviations?
- What are the severity and likelihood of that hazard to take place?
- What actions can be taken to mitigate the affect of these hazards?

Objective 2: Empirical analysis of resource utilization and execution time to characterize the factors affecting Hadoop jobs This thesis highlighted different aspects of Hadoop job to optimize it for a particular resource and dataset. Some of the important questions were:

- What Hadoop parameters are important?
- What values of these parameters have been reported in literature and documentation?
- What impact each parameter has on CPU usage, disk usage, memory usage, network usage and overall execution time?
- Does different Hadoop job inputs have impact on resource utilization provided the dataset remains the same?
- Are there any confounding factors affecting the performance and resource utilization of cluster.

Objective 3: Modelling and prediction of resource utilization for Hadoop jobs using machine learning.

This sub-objective results in the development of a methodology to build models for Hadoop clusters running large datasets. Some of the questions we tried to answer were:

- What machine learning-based technique we can use to build a model for our Hadoop jobs?
- Can these models predict resource utilization for a set of chosen values?
- Can job input or dataset selection and cluster size can become part of model building process?

- How generalizable these models can be for unseen datasets and different cluster sizes?

Objective 4: Correlation of features affecting Hadoop job performance.

This sub-objective highlighted important interactions or relationships between Hadoop configuration parameters, dataset or query structure and different cluster sizes. Some of the important questions were:

- How can we model such a complex feature space into a single function?
- Which features are important for under or over-utilization of a particular resource such as CPU usage?
- What is strength of correlation and direction of correlation between each Hadoop job feature and its performance measure?
- Which job configuration setups shows similar performance and resource utilization?

1.3 Contributions

This section highlights the main contributions made by this thesis. The contributions are as follows.

- Insights into designing and implementing reliable Hadoop experiments. HAZOP was applied to Hadoop cluster design for experiments highlighting the important confounding factors that can affect performance and resource utilization of Hadoop jobs. By applying HAZOP, we identified the considerations that must be taken into account for reliable results. This activity also saved time when we encountered some deviations from either normal working of cluster or running of jobs on cluster because it speeds up troubleshooting.
- A performance characterisation of Hadoop jobs allowing identification of the attributes of Hadoop job that have significant effect on the utilization

of different resources and execution time. Some of these features included block size, replication factor, number of mappers and query structure. This characterisation will help users of Hadoop cluster understand the impact of different values of these features on CPU usage, disk usage, memory usage, network usage and execution time. Our results highlighted the complexity of problem of tuning Hadoop clusters due to the large number of configurable parameters are involved. Each of these parameters can possibly take a range of values which made the problem even complex due to possible combinations of resulting parameters values. Also, different elements of queries significantly impacted on overall performance and resource utilization of Hadoop cluster.

- A machine learning-based modeling and prediction approach to estimate the resource utilization of Hadoop cluster. We discretized the performance and resource utilization of the cluster into high usage and low usage. This enabled us to build models of each of the performance measures by using decision trees. This thesis also contributed to model and predict resource utilization for datasets of different sizes and shapes. The models were built on the basis of a single dataset with six different query structures and then used to predict the performance and resource utilization when used different datasets. This showed that our models have good generalizability for unseen data.

- An evaluation of the prediction accuracy of a model built using our machine learning techniques. To evaluate the approach we changed the number of nodes on the cluster, the dataset used to run the Hadoop jobs and changed the infrastructure as well. To see the generalizability of our approach over changes in infrastructure, we configured a Hadoop cluster on Amazon EC2. We applied the model built from our previous experiments and found that it showed good prediction accuracy. The correlation between different features of Hadoop job and resource utilization was also investigated. Some features contributed more towards the under or over utilization of a particular resource for example CPU or overall execution time. We used M5P trees to see how different factors are related into functional relationship. The strength and direction of their association was also calculated using Pearson's correlation technique. K-means clustering technique was used to investigate the experimental runs showing similar performance behavior. The M5P trees helped us to understand the inter-related impact of different attributes while k-means clustering helped us to how different experimental runs were related.

1.4 Structure of the Thesis

The remainder of this thesis is organized as follows.

- **Chapter 2- Background and Literature Review**

  This chapter introduces the background information necessary to understand the concepts presented in this thesis. It highlights different components of Hadoop ecosystem. It also gives a brief overview of Hive and its functionality. This chapter also covers some of the relevant literature published in recent years and helps in identifying the problem.

- **Chapter 3- Application of HAZard and OPerability analysis for the design of Hadoop cluster**

  This chapter focus on applying Hazard and Operability analysis (HAZOP) to see potential factors that can hinder or affect the working to hardware, running of experiments and collection of measurements. This chapter describes how we can ensure the internal validity of Hadoop experiments. This also highlighted the important deviations, their possible consequences, the severity and likelihood of occurring those hazards, and possible mitigations to improve designing and running of our experiments on Hadoop clusters.

- **Chapter 4- Characterization of Hadoop performance and resource utilization by parameter tuning and job input**

  This chapter focuses on understanding the factors that can affect the Hadoop performance. Potentially important configuration parameters used in our experiments were identified from literature and Hadoop documentation. This chapter also describes the design and implementation of our experimental setup which served as platform to run all of our current and future experiments. We discussed the datasets we used and the performance measures taken into account as well. We also discussed what different values of each configuration parameter has been used in our experiments and how these impact on system resources and execution time. This chapter also highlights the complexity of large number of different combinations of parameter space.

[Published in Tariq, H., Welch, I. and Al-Sahaf, H., 2018, November. An Investigation of Hadoop Parameters in SDN-enabled Clusters. In 2018 12th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (pp. 1-9). IEEE.]

- **Chapter 5- Modelling and prediction of resource utilization of Hadoop cluster using Machine Learning**

  This chapter describes our method to model and predict resource utilization of Hadoop cluster. It discusses our experimental design and setup. It also covers the use of machine learning-based method to build model and then predict the resource utilization for different inputs, and highlights the important features participating towards modelling of each of the system resource and thus predicting its severity of utilization.

  [Published in Tariq, H., Al-Sahaf, H. and Welch, I., 2019, December. Modelling and Prediction of Resource Utilization of Hadoop Clusters: A Machine Learning Approach. In Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (pp. 93-100).]

- **Chapter 6- Modelling and prediction of resource utilization of Hadoop cluster for different datasets, cluster sizes and infrastructure**

  This chapter discusses the impact of environmental changes such as using different dataset and adding or removing one node in the existing cluster. The chapter also discusses the generalizability of model generated from previous experiments to predict resource utilization when using different datasets. It also describes the similarities and differences between different cluster size for the same job.

It also presents the association or correlation between different Hadoop parameters, inputs or queries, and cluster size. A tree-based regression model is used to give multiple linear regression models. The strength and direction of correlation was also calculated for each feature of Hadoop job and resource. This highlights the relative importance or influence of each feature to optimize Hadoop jobs to predict a particular performance measure.

[Partially published in Tariq, H., Al-Sahaf, H. and Welch, I., 2019, December. Modelling and Prediction of Resource Utilization of Hadoop Clusters: A Machine Learning Approach. In Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (pp. 93-100).]

- **Chapter 7- Contributions and Future Work**

  This chapter summarizes the results and highlights the contributions derived from different phases of the thesis. It also discusses the possible future directions of the thesis.

Chapter 2

Background and literature Review

This chapter presents the research background of the thesis and followed by a discussion of the related works that supports the motivations of the thesis. It starts with a general introduction of Hadoop. Section 2.1.1 discussed different components of the Hadoop ecosystem including Hadoop distributed file system (HDFS), Yet Another Resource Negotiator (YARN) and the programming framework MapReduce. It also highlights the important characteristics of HDFS. Section 2.1.2 highlights some important configuration parameters and their brief description. In Section 2.1.3 we have discussed general introduction to Hive and its components. Software-defined networking (SDN) has been introduced in Section 2.1.4. Its architecture has been discussed to give reader general idea about the working of SDN. Subsequently, we discussed related work in Section 2.2 to motivate the research works reported in this thesis. To the end of, the chapter summarizes the discussions on related works to make connections to following contribution chapters (i.e. Chapters 3-6).

2.1 Background

Big data analytics is a fact growing area focused on collecting, processing and analyzing the multi-scale, and multi-source data. The main objective is to discover the patterns, find the correlations and extract the information to get some useful insight that may improve decision making in common businesses of life [121]. Machine learning based approaches are usually being used to accomplish this task of mining data. Both supervised and unsupervised learning techniques have been applied for this purpose which are computationally expensive because of the serial implementation of algorithms [122]. Parallel and distributed computing approaches have been in use to process large volumes of data [123]. Several technologies are making use of parallelism in computing such as grids, clusters etc [124–126]. But these technologies often do trade-offs between performance, cost, failure management, maintenance, usability and data recovery.

Apache Hadoop was developed by Dough Cutting and Mike Cafarella in 2005 [127]. Hadoop is a general framework serving lots of big data applications like machine learning, crawling etc. Hadoop is not a single application, infact it is an ecosystem comprising of different components working together such as Hive, Pig, Giraph, Mahout and Cascading. Hadoop distributed file system (HDFS) is built on top of Hadoop ecosystem and is a core part of this framework [128,129]. Various research is underway to comprehend the areas which require special investigation [130]. Areas such as scheduling, data locality, map reduce optimization, multiple sequence alignment, straggler problem and speculative execution, shuffle and sort optimization are under active study [131].

Using Hadoop for distributed processing of big data as compared to other platforms, such as Spark [132], has many advantages. Firstly Hadoop is highly fault-tolerant because of replication of data block across different nodes of the cluster [133–135]. Secondly, data security in Hadoop is fine-grained [136–138]. Instead of copying the data into memory, Hadoop performs processing of data where it is actually stored, thus it relieves network and severs of considerable communication load [32].

2.1.1 Components of Hadoop

The Hadoop Common is the foundation of Hadoop. It consists of basic services such as abstraction of operating system and its filesystem. Hadoop common also provides Java Archive (JAR) files to start and run jobs on Hadoop. [139] Other components of Hadoop ecosystem are Hadoop Distributed File System

(HDFS), MapReduce programming framework and Yet Another Resource Negotiator (YARN).

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is one of the most popular open-source platform for the storage of Big Data or large volumes of data [129]. HDFS is a distributed filesystem that helps to run the basic hardware components. It has been reported to store as large as the data of size 100PB on a cluster [140]. It is designed to run on commodity hardware and has many similarities with existing distributed file systems. Its main feature are:

1. Suitable for the distributed storage and processing.
2. Command-line based interface to interact with HDFS.
3. Streaming access to data
4. File permissions and authentication
5. Built-in servers for NameNode and DataNode.
6. Portability across heterogeneous hardware and software platforms

Characteristics of HDFS The following paragraphs discusses the characteristics of Hadoop distributed file system. It is important to understand how different components work and what is their inter-linkage while the job is being executed. These characteristics of HDFS are the core features of the Hadoop ecosystem as well. Overall, they make the Hadoop data processing fault-tolerant and robust.

NameNode and DataNodes HDFS has a Master/slave architecture. An HDFS cluster consists of a single NameNode, a Master server that manages the file system namespace and manages access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

The NameNode and DataNode are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java lanGauge; any machine that supports Java can run the NameNode or the DataNode software. Usage of the highly portable Java lanGauge means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNode software. The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case.

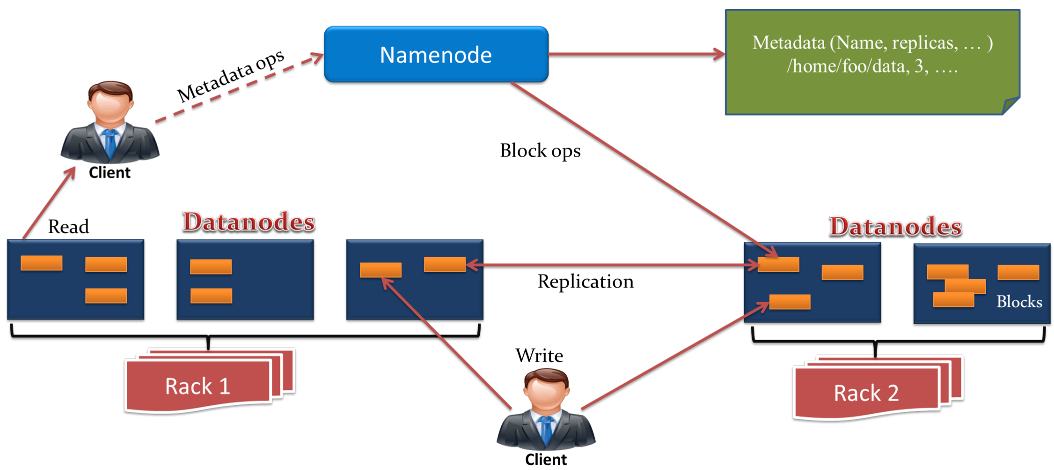

Figure 2.1: HDFS architecture

Data replication HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once and have strictly one writer at any time.

The NameNode makes all decisions regarding replication of blocks. It periodically receives a Heartbeat and a Blockreport from each of the DataNodes in the cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly. A Blockreport contains a list of all blocks on a DataNode.

Replica placement The placement of replicas is critical to HDFS reliability and performance. Optimizing replica placement distinguishes HDFS from most other distributed file systems. This is a feature that needs lots of tuning and experience. The purpose of a rack-aware replica placement policy is to improve data reliability, availability, and network bandwidth utilization. The current implementation for the replica placement policy is a first effort in this direction. The short-term goals of implementing this policy are to validate it on production systems, learn more about its behavior, and build a foundation to test and research more sophisticated policies.

Large HDFS instances run on a cluster of computers that commonly spread across many racks. Communication between two nodes in different racks has to go through switches. In most cases, network bandwidth between machines in the same rack is greater than network bandwidth between machines in different racks.

The NameNode determines the rack id each DataNode belongs to via the process outlined in Hadoop Rack Awareness. A simple but non-optimal policy is to place replicas on unique racks. This prevents losing data when an entire rack fails and allows use of bandwidth from multiple racks when reading data. This policy evenly distributes replicas in the cluster which makes it easy to balance load on component failure. However, this policy increases the cost of writes because a write needs to transfer blocks to multiple racks.

For the common case, when the replication factor is three, HDFS's placement policy is to put one replica on one node in the local rack, another on a node in

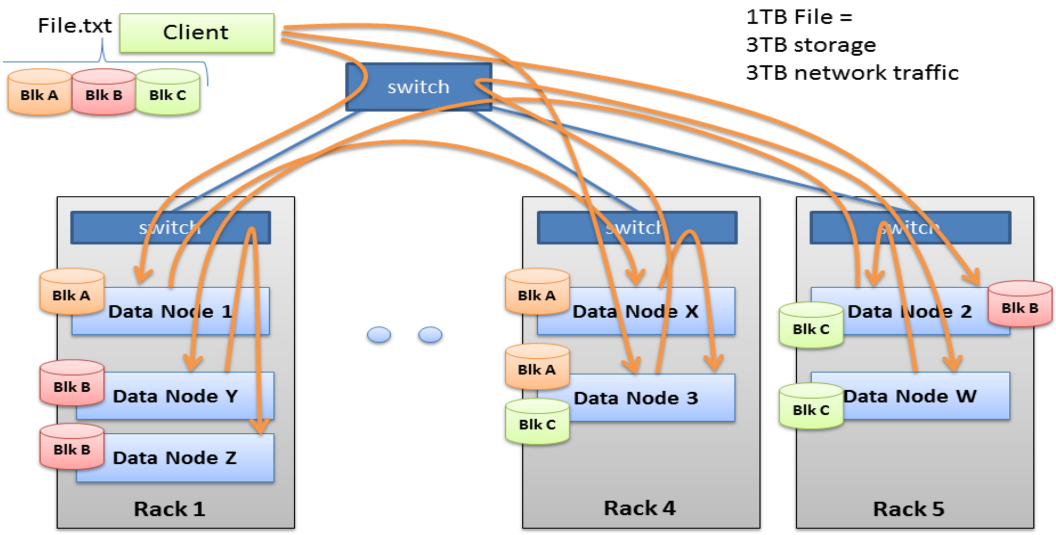

Figure 2.2: HDFS architecture

a different (remote) rack, and the last on a different node in the same remote rack. This policy cuts the inter-rack write traffic which generally improves write performance. The chance of rack failure is far less than that of node failure; this policy does not impact data reliability and availability guarantees. However, it does reduce the aggregate network bandwidth used when reading data since a block is placed in only two unique racks rather than three. With this policy, the replicas of a file do not evenly distribute across the racks. One third of replicas are on one node, two thirds of replicas are on one rack, and the other third are evenly distributed across the remaining racks. This policy improves write performance without compromising data reliability or read performance.

Replica selection To minimize global bandwidth consumption and read latency, HDFS tries to satisfy a read request from a replica that is closest to the reader. If there exists a replica on the same rack as the reader node, then that replica is preferred to satisfy the read request. If HDFS cluster spans multiple data centers, then a replica that is resident in the local data center is preferred over any remote replica.

The Persistence of File System Metadata The HDFS namespace is stored by the NameNode. The NameNode uses a transaction log called the EditLog to persistently record every change that occurs to file system metadata. For example, creating a new file in HDFS causes the NameNode to insert a record into the EditLog indicating this. Similarly, changing the replication factor of a file causes a new record to be inserted into the EditLog. The NameNode uses a file in its local host OS file system to store the EditLog. The entire file system namespace, including the mapping of blocks to files and file system properties, is stored in a file called the FsImage. The FsImage is stored as a file in the NameNode's local file system too.

The NameNode keeps an image of the entire file system namespace and file Blockmap in memory. This key metadata item is designed to be compact, such that a NameNode with 4 GB of RAM is plenty to support a huge number of files and directories. When the NameNode starts up, it reads the FsImage and EditLog from disk, applies all the transactions from the EditLog to the in-memory representation of the FsImage, and flushes out this new version into a new FsImage on disk. It can then truncate the old EditLog because its transactions have been applied to the persistent FsImage. This process is called a checkpoint. In the current implementation, a checkpoint only occurs when the NameNode starts up. Work is in progress to support periodic checkpointing in the near future.

The DataNode stores HDFS data in files in its local file system. The DataNode has no knowledge about HDFS files. It stores each block of HDFS data in a separate file in its local file system. The DataNode does not create all files in the same directory. Instead, it uses a heuristic to determine the optimal number of files per directory and creates subdirectories appropriately. It is not optimal to create all local files in the same directory because the local file system might not be able to efficiently support a huge number of files in a single directory. When a DataNode starts up, it scans through its local file system, generates a list of all HDFS data blocks that correspond to each of these local files and sends this report to the NameNode: this is the Blockreport.

The Communication Protocols All HDFS communication protocols are layered on top of the TCP/IP protocol. A client establishes a connection to a configurable TCP port on the NameNode machine. It talks the ClientProtocol with the NameNode. The DataNodes talk to the NameNode using the DataNode Protocol. A Remote Procedure Call (RPC) abstraction wraps both the Client Protocol and the DataNode Protocol. By design, the NameNode never initiates any RPCs. Instead, it only responds to RPC requests issued by DataNodes or clients.

MapReduce Paradigm

It is the other core part of Hadoop framework which is originally developed by Google [141]. Several public and hybrid clouds deploy MapReduce paradigm. Amazon Elastic MaprReduce, Microsoft Azure [142,143], IBM's Clue cloud and Google App Engine [144, 145] are some of the public clouds that enable users to perform MapReduce-based computations [146]. It is cost effective as users don't have to consider physical infrastructure or software installations.

It performs distributed processing of the data in two steps; map and reduce. During map task, input dataset is being converted into key/value pair which are scalar transformations. Output of map tasks is grouped by keys and then sorted and divided to simpler tasks. Usually, map tasks can be sub-divided into *four* distinct phases.

1. *Compute :* It involves the application of map function to each key-value pair

2. *Collect :* The processed key-value pairs are stored in a map output buffer.

3. *Sort :* It takes place between collect and spill phases of map task. The key-value pairs are sorted at this stage.

4. *Spill :* The output buffer empties all of its content to a file on local disk. This process in being initiated when the map output buffer is full.

The list of keys are then combined and presented to the Reduce task which performs operations on values arrays for each key. In the end the matching keys from map task are combined and results are being collected.

Yet Another Resource Negotiator (YARN)

YARN is a distributed application management framework introduced in Hadoop 2.0 [130]. It enhanced parallelism and resource management along with better scalability. It works on top of HDFS and also handles real-time interactive processing and batch-processing of data. It is designed in a way that it is compatible with MapReduce API [145].

There are two separate functionalities of YARN. First is to allocate and manage the resources across the cluster which is also called as Resource Manager daemon. Second, is the Application Master daemon which is a framework to schedule and monitor tasks using task Tracker daemons at the cluster nodes. In short, YARN ensures the life cycle management of all applications executed in the cluster [147].

2.1.2 Important Hadoop configuration parameters

The previous section highlighted different components of the Hadoop framework. This section highlights some of the important configuration parameters. The important of these parameters has been demonstrated in several studies in the literature. We are describing them briefly here to give general idea of their functionality and impact over cluster performing and resources utilization. We choose to discuss these and ignoring all other parameters as these are commonly discussed parameters while deployment, optimization and estimation of Hadoop job performance. We have used different values of these parameters for executing different jobs. Different experimental setups reported in thesis refer to different combinations of values of these parameters.

Slot value is used to show the capacity of a cluster node to accommodate the task. Usually each node has pre-defined number of slots but one can also configure the slot as map or reduce slot according to his convenience and requirement. The value of slot is usually being set with heuristic numbers without considering the characteristics of the job. This can affect the optimal performance of cluster for various jobs.

Number of mappers

How long are our mappers running for? If they are only running for a few seconds on average, then we should see if there's a way to have fewer mappers and make them all run longer, a minute or so, as a rule of thumb. To what extent this is possibly depending on the input format we are using [127].

Number of Reducers

For maximum performance, the number of reducers should be slightly less than the number of reduce slots in the cluster. This allows the reducers to finish in one wave and fully utilizes the cluster during the reduce phase [127].

Block size

A disk has a block size, which is the minimum amount of data that it can read or write. HDFS, too, has the concept of a block, but it is a much larger unit-64 MB by default. Like in a filesystem for a single disk, files in HDFS are broken into block-sized chunks, which are stored as independent units [26].

Replication factor

To ensure against corrupted blocks and disk and machine failure, each block is replicated to a small number of physically separate machines (typically three). If a block becomes unavailable, a copy can be read from another location in a way that is transparent to the client. A block that is no longer available due to corruption or machine failure can be replicated from its alternative locations to other live machines to bring the replication factor back to the normal level [15].

2.1.3 Hive

This section gives general introduction to Hive which is an important and integral part for running of our experiments on Hadoop framework. Hive is a data warehousing infrastructure that is built on top of Hadoop [148]. It facilitates data summarization, analysis and query in distributed Hadoop clusters [149]. We can query the data by using SQL-like lanGauge called HiveQL. The queries can be run by using Hive shell, JDBC or ODBC. Hive uses MapReduce framework to breakdown HiveQL statements for parallel processing on Hadoop cluster [150]. We can use other execution engines as well, for example, Tez, Spark [151]. A general overview of Hive-Hadoop setup is shown in Figure 2.3. The main components of Hive and functionality is as follows:

- **External interfaces -** Hive facilitates the access through web user interface, command line interface (CLI) and through application programming interfaces (API) like ODBC and JDBC.

- **Metastore -** It is the main catalog of Hive metadata. It has a separate storage system to store Hive metadata and provides service to other components of Hive for metastore access. It not only stores metadata of Hive tables, such as their schema and location, but also stores partitions in a relational database.

- **Driver -** This component receives queries from external interfaces and manages the life cycle of HiveQL statements during compilation, optimization and execution. The functions of its main components are:
  - **Compiler** first receives the query from Hive driver where it is parsed and type checked. A semantic analysis is also performed here by the help of schema in metastore.
  - **Optimizer** makes the logical planning of the job. The plan is directed acrlyic graph (DAG) pf MapReduce and HDFS tasks.
  - **Execution Engine** submits the MapReduce jobs to Hadoop for execution. this process initiates once compilation and optimization has been completed successfully.

2.1.4 Software-defined Networking (SDN)

This section gives general overview of SDN and its architecture. We used SDN instead of using normal switch as it gives more control to the user and we can

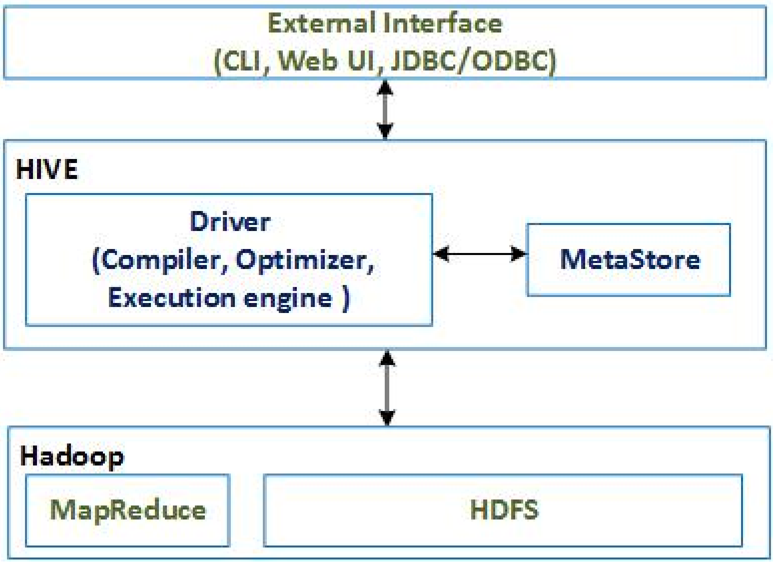

Figure 2.3: Hive architecture

store desired network measurements in a configured database. SDN is a networking architecture that separates the control layer from forwarding layer of the network. It simplified the network control, innovation through network programmability and management. It helped easy monitoring of networks, designing of fault tolerant networking and open protocols and interfaces [152] [153].

As shown in Figure 2.4, the framework of SDN consists of two APIs and three planes. The data plane comprises flow forwarding devices like routers and switches. The control plane makes the decision and data plane forwards the traffic on the basis of those decisions. The control plane have controllers or severs which decide on the traffic flow forwarding strategies. The application plane consists of network applications like access control lists (ACLs) and firewalls. These three planes are connected by two APIs: southbound API and northbound API. The southbound API is the interface between control plane and data plane. The controller can send flow forwarding decisions to switches by using southbound API. The northbound API is the network programming interface used to to design and implement new and innovative network protocols and applications [154].

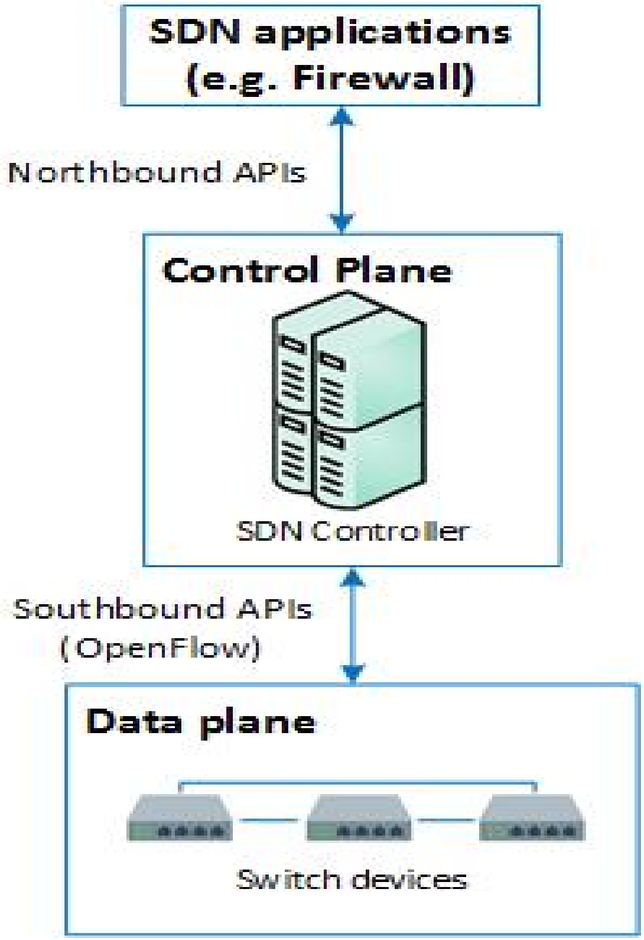

Figure 2.4: SDN architecture

2.2 Related work

In this section, we introduced the research works that are closely related to this thesis regarding their pros and cons to better motivate the research of this thesis. The section is organized in correspondence to research objectives stated in Section 1.2.

Performance is an important factor that has to be taken into account in the development, designing, configuration and tuning of a computer system. Computer performance evaluation can be divided into three broad areas, namely simulation, analytical and measurement [155]. Simulation performance modeling involves techniques like trace driven, event driven, execution driven, complete system simulation and software profiling. Analytical modeling includes probabilistic models, queuing model, Markov models and Petri Net models etc. Performance measurement involves the hardware/ software monitoring,workload characterization techniques, capacity planing and benchmarking [118]. In the literature, several approaches has been mentioned to improve Hadoop performance by parameter tuning.

2.2.1 Configuration parameters based optimization

Different phases of map task include compute, collect, sort,spill,combine and merge-spill. All phases except spill and merge-spill are CPU intensive. So, the number of maximum map tasks running for a particular MapReduce application on a node is called slot value. The slot value has an impact over CPU utilization of cluster. Kamal and Freeh showed that a single value for maximum number of map tasks, running concurrently, is not applicable for different types of applications [110]. They configured two 6 node clusters: first consist of IBM PowerPC machines and the second consist of x86 machine. The experiments were run by using six applications from PUMA suite [156] and seven variants of *terasort*. PUMA is a benchmarking suite developed by Purdue university. The number of map tasks were varied with an increment of 8 from 16 to 64 maps per slot. The workload of 150 and 300 GB were used to run experiments on both of the clusters. They measured the performance of thirteen different applications that have a wide range of IO and CPU characteristics. So no one value is suitable for all of the applications.

They introduced Hadoop counters that measured the CPU usage per-map task IO throughput. The applications formed three distinct regions when the values of the two measures, for TaskTracker of one node, were normalized and plotted against each other. The CPU utilization was on x-axis and IO throughput on y-axis. The applications in each regions were categorized as IO intensive, CPU intensive and balanced. Base upon their personal experience to define that 30% of the region should be IO intensive, 60% should be balanced region and 10% should be CPU intensive. Each regions has a certain range of values representing the number of map tasks giving best performance for that application.

percentage of data So when they have chosen the application specific number of map task, the CPU performance degradation was only 1% as compared to 7% in case of single value for all applications.

Wang *et al.*, [109] suggested a n approach to achieve efficient and fair slot configuration for Hadoop cluster. The slot indicates the capacity of each node of the cluster to accommodate map and reduce tasks. This value can be kept static during the lifespan of the cluster. They adopted two approaches to adjust slot value. First approach was to decide the number of map and reduce tasks before launching cluster and Fair scheduler was used to schedule job. Second approach included, adjusting slot values dynamically by allowing the change of a slot in an online manner to either a map slot or reduce slot. This dynamic approach involves calculating the difference between expected and actual map slot. if the aexpectation is higher than the free slots will be assigned as a map slots. They compared the results for static and dynamic slot configuration with different workloads. They also presented a new fairness calculation scheme to achieve minimum makespans without degrading it. Makespan is the total time to complete processing of all jobs [157, 158]. The experiments were run on Amazon EC2 platform. They built two clusters of 10 nodes and 20 nodes with slot size of 4. Six PUMA benchmarks were used to evaluate the performance of their proposed systems. The workloads were 4 GB and 8GB and the datasets used were wikilink data and movie rating data. For static slot configuration three different settings were tested while for dynamic slot configuration two workloads were tested i.e. simple and mixed workloads. For simple workloads ten jobs for each application was submitted to cluster with an interval of 2 seconds. The results showed that fair scheduler performed well in most of the experiments. For mixed workloads, five sets of workloads were designed. Each set consisted of 12 different combinations of applications and workloads. Two sets of workload were created from initial three mixed work loads. these two sets represented map-intensive and reduce-intensive workloads. Their results showed that a fair scheduler with 2:2 ration represents the best makespan. They compared their results with traditional Hadoop schedulers like FIFO and capacity scheduler.

Yao *et al.*, stated that static configuration throughout the lifespan of cluster, can result in low system resources utilization and lengthy completion times [111]. Each node in Hadoop cluster can take up to a certain number of map or reduce tasks. This value can be set fixed to manipulate the cluster's maximum parallelization capacity. They allocated map and reduce tasks dynamically during the execution of the Hadoop jobs. At the end of each task Hadoop needs to decide that how many map or reduce tasks are left? and whether a slot is available to be assigned. They first estimated the workload of each node for concurrently running jobs. They fixed the maximum number of map and reduce such that the number of map tasks does not go beyond a certain value at a time. The experiments were run on Amazon EC2 platform. They built cluster of 5 nodes and 20 nodes with slot size of 4. They have run the experiments by using four Purdue MapReduce Benchmark Suite. The workloads were 10 GB and 7 GB and the datasets used were wikilink data and movie rating data. A simple and mixed workload was designed to evaluate the performance of their method. They improved completion time of a batch of MapReduce jobs by adjusting mad and reduce slot ratio on the basis of an estimated workload. The scheme was evaluated on a cluster at Amazon EC2 and demonstrated 28% reduction in makespans and 20% increase in resource utilization.

Block size is an important configuration parameter which directly affects the Hadoop distributed file system. Krishna *et al.,* [26] demonstrated that performance of HDFS can be evaluated by using read and write operations. In order to do so, they performed read and write operations of small and large files by using *data* size of 1GB, 2GB, 4GB and 8GB. A five node cluster was built for experimentation purpose with Ubuntu installed on it. The performance was measured on the basis of overall execution time. In order to make a comparison the execution times for the same data size was compared. For each data size, number of files and the *file* size was altered. File size started from 1MB and is doubled up to 1024MB in each of their experiments. Their results showed that the performance of read and write operations of HDFS remained poor when the file size was less than the default block size (64MB) and the performance increased when the file size is greater than the default block size.

Bansal *et al.,* [159] studied the impact of different Hadoop parameters on the performance of Hadoop jobs. They studied the impact of block sizes of 64MB, 128MB, 256MB and 384MB on the overall and average execution time of the job. They used 10Gbps uplink and 1 Gbps lateral ports for the connection of their 1 Master and 4 node cluster. For performance evaluation, they used Purdue MapReduce benchamrk suite (PUMA). They experimented with different scheduling algorithms in Hadoop yarn such as fair sharing, capacity scheduling and first in first out(FIFO), to show that cache size, memory and spill memory along with the nature of job submitted can impact the execution time. They didn't report the number of times experiment repeated to ensure that their findings are repeatable and reproducible.

Xueyuan *et al.,* [112] studied the impact of Hadoop configurations on the performance of Hadoop jobs. They altered memory related settings of Hadoop in docker environment and observed the resource utilization for different settings. They used Hadoop 2.7.1 and Ubuntu 14.04.03 LTS was configured by using docker version 1.9.1 on a HP server machine having 24 Intel Xeon cores and 1TB hard drive. Each docker container acted as a cluster node. Some of the important memory related parameters they used in their experiments are: *MapReduce.map.memory.mb*, *MapReduce.reduce.memory.mb* and *yarn.nodemanager.resource.memory-mb* etc. *Eighteen* different values of these parameters have been used to run WordCount and TeraSort. They measure the CPU and memory footprints of the system. Their results showed that appropriate customization of the parameters can improve the performance of the Hadoop. The performance can be improved by increasing memory allocated to the nodes as compared to default memory parameter settings. they have also stated that their method can only achieve better results if the memory is the bottleneck of the nods, otherwise it can cause performance degradation of CPU.

Nascimento *et al.,* [160] investigated the impact of different Hadoop configuration parameters while running Hadoop-based Exact Diameter Algorithm (HEDA). They used twitter and Internet research lab (IRL) datasets and monitored the CPU usage, memory usage, network traffic and the execution time for different cluster sizes. They used Ganglia for cluster monitoring purpose [161]. Their results showed that the choice of parameters, on clusters of different nodes, has improved the run-time of HEDA by 80 percent. The results for cluster sizes of 1, 2 4 and 8 nodes running job maps 6, 12, 24 and 48 were reported in their studies. They used replication factor 1 and 3 and block sizes of 64MB, 128MB and 256MB for all sizes of clusters. However, their work lacks the relationship between different Hadoop parameter setting and system, memory and execution time. Furthermore, they did not reported the number of times they repeated the experiments to ensure experimental rigour.

Apart from parameter tuning, some researchers also tried to optimize the performance by modifying current Hadoop ecosystem. One such example is modifying replica placement strategy to enhance resource utilization.

Qu *et al.*, [14] has presented dynamic replica adjustment strategy(DRS) based on Markov Model. It relies on the data access frequency and data access patterns. Here each file in the system is treated as a state. Firstly file access times for each file is calculated in a given period of time to see how frequent a file is being accessed. Each file is replicated according to the access frequency calculated based on Markov model. Whenever a new file is added into the system it is assigned a non-state entity. But as this new file is accessed over a certain number of times, it will be taken as a state in the system. Similarly if any file has not been accessed over a certain period of time, it is added to non-state space. The replica number of file *i* can be computed as

$$m_j = \lceil R \times N \times \pi_j \rceil \tag{9}$$

where R is the replication factor and N is the number of files in system.

Parameters	**Value**
Replication	3
Nodes	5
Racks	3
Performance metrics	Avg. job time

Table 2.1: Parameters used

Qu *et al.*, [14] demonstrated that replicas can be equally distributed across *n* racks. When a file is to be placed on a particular rack, its adjacent file is also placed at that rack. In this way related files are placed on the same rack involving reduced data transmission between nodes or racks. In this strategy, first average number of replicas are calculated for each rack based on markov model using formula.

$$m_i = \frac{c_i}{r} \tag{2.1}$$

where c_i is the number of replication number of file i and r is the number of racks. The replication number of a file i on each rack is S_i. If there are no replica of file i placed on the rack where file j is placed, then place replica over this rack otherwise move forward. This process is repeated until all replicas are uniformly distributed.

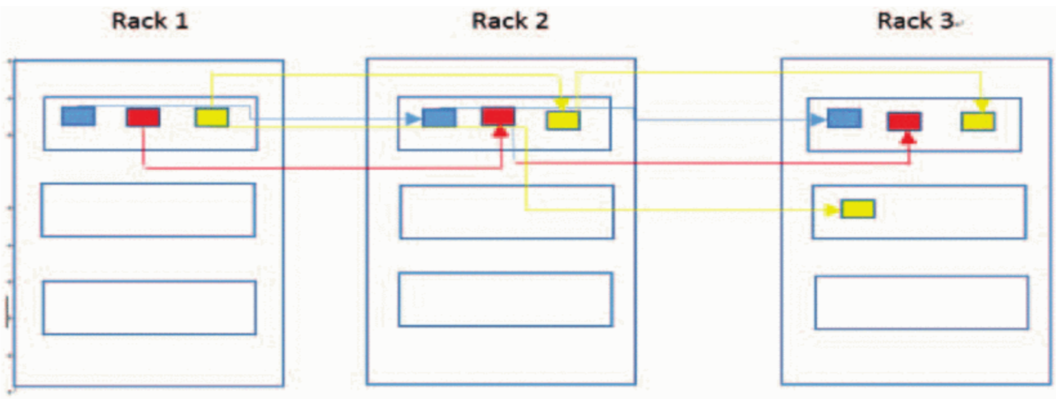

Figure 2.5: Homogeneous Replica Placement Strategy

Dai *et al.*, [15] demonstrated the even distribution of the replicas in HDFS using partition replica placement strategy(PRPP). They designed a theoretical simulation to for this purpose. This involves the division of all available nodes into three sections. Two sections will have 1/3 of the nodes (S2) while one section will have 2/3 of the nodes (S1). This is called as section formation phase followed by replication phase. Replication phase involves the distribution of replicas to all nodes. First a table is constructed for each section, where a column corresponds to replica assignment of one node. It starts assigning the replicas that are supposed to be on the same rack from left to right and top to bottom. This will be further randomized so that no nodes have the same replica assignment. After distributing the first two replicas to nodes in S1,they distributed the remaining one replica among nodes of S2. This type of replica placement strategy is only suitable for homogeneous clusters where each node has same computing capabilities. It eliminated the use of default load balancing utility of HDFS.

Parameters	Value
No. of Blocks	8000
Replication	3
Nodes	600
Racks	30,45
Performance metrics	min, max, mean, variance and std. deviation

Table 2.2: Parameters used

Zheng and Shen, addressed the issue of low resource utilization and efficiency problem for Hadoop [62]. They proposed a data-parallel processing framework to horizontally scale parallelism of task execution. This means that more sub-tasks have been added to the usual process of map-reduce [162, 163]. The data management of current distributed file system is relatively coarse grained due to big block split which can lead to coarse granularity of data processing. So, it is important to get a fine grained data processing strategy to improve resource utilization of cluster. Their overall strategy consists of three main aspects: Sub-block, Task, and Pre-shuffle. In the sub-block step, they further split the block to reduce data management granularity which can impact on performance and scalability of HDFS. The sub-blocks position is same as that of block so it can easily be recognized by the map task without increasing load on NameNode. This can cause serious disk performance degradation if the number of sub-blocks are too many.

Then in the task section, based on sub-block,they improved the parallelism of map/reduce task execution. They used multiple threads(called sub-task) to make full use of resources as each thread can process one or more sub-blocks. The speculative tasks which can result in under-utilization of resources and long execution times, can easily be handled by sub-task strategy as it will not hold up the entire block. They implemented their idea by using Apache Hadoop 1.2.1.In the experimental environment, we build Hadoop cluster with five servers, one for JobTracker and NameNode, the others for compute nodes. Each server has two 6 core Intel Xeon CPUs (E5645, 2.4GHz, with Hyper-Threading Technology), 64GB RAM and 300GB hard disk. All servers are installed Ubuntu Server 12.04 with Sun JVM version 1.6.0_26. They used Wordcount, Sort and Kmeans to run experiment to see scalability, performance under multi-CPUs and resource utilization. Their experiments shows that they optimize the cluster resources and also speed up the completion time.

Mathiya and Desai [120] demonstrated that the performance of Apache Hadoop Yet Another Resource Negotiator (YARN) depends upon a large number of parameters. They classified the Hadoop YARN parameter configuration settings into different categories. The main classes were CPU-related, memory-related and I/O-related parameters. A parameter was assigned to one of these classes if it can directly affect the performance of CPU, memory or I/O of the system. One of the drawbacks of their study includes the lack of any experimental design to justify that these parameters do affect the class they were put into.

Summary

The summary of literature review is shown in Table 2.3. Current studies showed that optimal performance of Hadoop depends on several characteristics of Hadoop job. Researchers have tried to optimize or estimate the resource utilization or execution time of Hadoop jobs by improving job scheduling, slot value assignment, block size alteration etc. But careful analysis of the current studies also highlighted some gaps for the future work.

Firstly, the lack of robust experimental design is not present in most of the studies. This includes the lack consistency of workloads in consecutive experiments to be compared, not reporting the datasets they used in their studies, and the application they used to run the these experiments.

Secondly, not reporting the input of the experiments or number of times the experiments has been reported is another important gap in the existing work. Hadoop jobs might be resource extensive or time consuming but both of these aspects are important for good experimental design. Ignoring these can introduce scientific bias into the results which can give misleading and unreliable results.

Thirdly, another important problem raised because of these shortcomings is that the studies do report the percentage of increase or decrease in performance or resource utilization but they did not report the statistical significance of this percentage difference. This is very important aspect of the scientific experiments as a very prominent difference in percentage may not be significant statistically and this can impact the validity of the results achieved.

Table 2.3: Summary of literature review for Hadoop optimization.

Paper	**Performance measure**	**Objective**	Setup	Exp. design details	Dataset	Application	Parameter value	Datasize	Number of experiments	Input	Statistical Significance
Kamal and Freeh (2014)	CPU usage	Modeling application specific behavior	2 x 6 nodes	Not Defined	wikipedia	PUMA, teraSort	Yes	Yes	Not defined	Not defined	X
Wang *et al* (2014)	Completion time	Modeling resource utilization	Amazon EC2 10 nodes, 20 nodes	Not Defined	Wikilink data, movie rating	PUMA	4 virtual cores	Defined	Not defined	Not defined	X
Yao *et al* (2013)	Completion time, resource utilization(slots)	Improve scheduling of jobs	Amzaon EC2 5 node	Not Defined	Wikilink data, movie rating	PUMA	5 virtual cores	Defined	Not defined	Not defined	X
Krishna *et al* (2014)	execution time	Modeling HDFS read and write operations	5 nodes	Not Defined	Not defined	DFSIO	Defined	Defined	Not defned	Not defined	X
Bansal *et al* (2014)	Overall and average execution time	Improve scheduling of jobs	5 node	Not Defined	Not defined	PUMA	Not defined	Not defined	Not defined	Not defined	X
Xueyuan *et al* (2016)	CPU and memory usage	Modeling memory utilization	Docker	Not defined	Not defined	Not defined	WordCount, TeraSort	Defined	Not defined	Not defined	Not defined
Nascimento *et al* (2017	CPU, memory and network usage, execution time	Modeling application specific behavior	9 node	Not Defined	Twitter	HEDA	Defined	Not defined	Not defined	Not defined	X
Qu *et al* (2016)	Time and disk usage	Optimize HDFS	Simulation	Not Defined	Not defined	Not defined	Not defined	Defined	Not defined	Not defined	X
Dai *et al* (2016)	Disk usage	Optimize HDFS	Theoretical simulation	No	Not defined	Not defined	Not defined	Not defined	X	X	X
Zheng and Shen (2014)	Disk usage	Optimize HDFS	5 node	Partial	Not defined	Wordcount, TeraSort, Kmeans	Defined	Yes	X	X	X
Mthiya and Desai (2015)	Memory usage, CPU usage and I/O	Model YARN parameters	Not defined	Not defined	Not defined	Not defined	Default	X	Not defined	Not defined	X

2.2.2 Job characteristics-based optimization

Correlation analysis is carried out to measure the relationship between two variables or quantities [164,165]. A high level of correlation means that the relationship between the two variables is very strong while a low correlation means that the variables are not related or less related [166]. Correlation analysis is a statistical method usually measured as correlation coefficient [167,168]. The value of correaltion coefficient ranges between -1.0 to $+1.0$. Anything around ± 1.0 is considered as strongly correlated [169]. The difference between the plus and minus symbols is the direction of correlation or relationship. A $+$ sign means that both variables or quantities are directly related. If one quantity increases, the other quantity also increases. While $-$ sign shows that if the value of one variable increases, the value of other variable decreases accordingly [170]. From machine learning point of view, this correlation or relationship can be represented in the form a linear regression model [171]. Linear regression was first proposed by Sir Francis Galton and is a mathematical model which shows the functional relationship between multiple variables [172,173].

Hadoop job is a complex working of different frameworks, resources and hardwares working together to achieve parallelism and distributed computing. So, overall cluster setup or cluster environment largely affects the execution of job. Cluster environment includes nature of application, type of input, system configuration settings and hardware, workload etc. It is important in defining its performance and resource utilization. Following paragraphs are some work reported to in the literature which shows that Hadoop performance and resource can be optimized or correlated to different characteristics of Hadoop cluster.

Nabavinejad and Goudarzi [174] proposed Smart Configuration Selection (SCS) method to choose a set of sample parameter configuration for virtual machines to run MapReduce job. They used Pearson correlation coefficient [175,176] or Kendall rank correlation coefficient [177,178] to select the most effective configurations. The recommended configurations were tested on Amazon EC2 and Microsoft Azure and showed less estimation error (11.58%) as compared to random configuration selection (19.72%). They used different benchmark applications, like WordCount and TeraSort, to see the difference between different configurations. Their results also showed that when this SCS is used with makespan minimizing algorithm, it can reduce the execution time upto 36% as compared to random selection.

Li *et at.*, [179] proposed a tool which calculated the mathematical relationship between configuration parameters and system performance. This tool help to predict the relationship which suits best for a particular hardware. They monitored CPU usage, disk usage and network usage every second on a nine node cluster. The experiments were run with predicted and default configurations using TeraSort. The running time of the predicted parameter settings was ten times faster than the default settings.

Cluster workload is correlated with the overall performance of the cluster. Moraveji *et al.*, [18] investigated how to load shared resources of a cluster of servers with data-intensive applications so that throughput degradation never falls below a certain limit. First they calculated the throughput of a single workload on a single physical server. They considered file size and file operation request size for the performance characterization of throughput. Their results showed that larger request size (RS) always gives higher throughput. Other parameters like last-level cache (LLC), system file cache and disk cache also effect the workloads greatly. They extended their work by putting multiple workloads on a single server which takes into the consideration file size, request size and number of concurrent workloads. Their results showed that throughput always degrades when the amount of data exceeds LLC capacity. Other contributing factors can be processor execution engine, file system cache, disk cache and disk bandwidth. In another experiment they calculated multiple workloads on multiple servers where they performed consolidation so that throughput is not degraded more than 50%.

On the basis of their experiments they designed a greedy algorithm to solve the consolidation problem. This algorithm minimizes the throughput degradation of consolidated workloads when distributed among a number of physical servers. They compared their algorithm results with brute-force technique which overloaded the schedulers to show their designed algorithm was better.

Cluster hardware/resource profile affects the performance of Hadoop job as shown in the work reported by Mohan *et al.,* [41]. They compared the read performance of triple replication storage in HDFS with Reed-Soloman erasure coded storage module available in RAID module of HDFS in various cluster distribution across diverse geographical locations across Australia on NeCTAR reseach cloud. They tried to understand the performance of various storage codes over distributed data across geo-diverse clusters. Two types of experiment setups were used, in one setup performance comparison between default HDFS replication and Hadoop's erasure coding was carried out and in other the comparison of Hadoop erasure code with local reconstruction codes in XORBAS Hadoop. Five clusters with varying geographical locations were used for the experiments. The default block size of 64MB was maintained in all of the experiments along with a file size of 640MB. Their results showed that erasure codes didn't degrade the performance and XORBAS Hadoop does not perform well as reported in other studies. Their study emphasized the importance of understanding the scheduling and block placement policies of Hadoop.

Resource available impacts the performance and resource utilization of of Hadoop jobs as demonstrated in work done by Zhu *et al.* [180]. They proposed a automatic configuration tuning framework for general systems. Their system used divide-and-diverge method to estimate the resource requirement. This resource utilization along with application specific workload requirement was then used to auto-tune configuration parameters. They tested this proposed architecture on Tomcat system on virtual machines, Cassandra and MySQL. Their results showed that their framework significantly improved the throughput of all of the systems.

Number of nodes or size of cluster is also an important contributing factor affecting job execution over Hadoop cluster as reported by Pal and Agrawal [108]. They reported their work on HDFS and MapReduce to investigate the relationship between memory usage by number of mappers and reducers. Their results showed that memory usage increases as the number of nodes of the cluster increases. They also demonstrated that cluster deployment can be problematic when we use WebUI to deploy cluster on Ubuntu.

Summary

A summary of literature review demonstrating the relationship of cluster environment for optimization of Hadoop performance is given in Table 2.4. Hadoop ecosystem is complex combination of different modules or frameworks working together. In order to achieve optimal and fair performance and resource utilization, we have to carefully measure and analyze different factors constituting the cluster or job environment. These environmental factors include but not limited to nature of application, type of input, configuration settings hardware etc. In addition to the shortcomings identified in Section 2.2.1, the work reviewed in current section showed following research gap. Most of the studies focused on optimizing one aspect of either the Hadoop ecosystem or Hadoop job. But It will be interesting to investigate what impact it has when multiple parameters have been optimized using many different values at a time. This will highlight the important relationship or correlation between the parameters. This will also provide a good insight into Hadoop cluster performance and resource utilization under these changes.

Table 2.4: Summary of literature review showing correlation of cluster environment with Hadoop optimization.

Paper	**Hadoop component**	**Objective**	**Setup**	**Dataset**	**Apps**	**Input**	**Stat. significance**	**No. of exp.**	**Parameter inter-relationship**
Nabavinejad and Goudarzi(2019)	Parameter configurations	Performance estimation on VM	VM 2 nodes	Defined	PUMA	Not defined	Not defined	Not defined	X
Li *et al.*, (2014)	Parameter configuration	Modeling relationship between system performance and parameters	9nodes	Defined	TeraSort	Not defined	Not defined	Not defined	X
Moraveji *et al.*, (2012)	Throughput	Modeling workload	2 servers	Not defined	Not defined	TestDFSIO	Not defined	Defined	X
Mohan *et a.*, (2015)	HDFS	Modeling HDFS	Cloud cluster	simulated	WorkCount	Not defined	Not defined	Not defined	X
Zhu *et al.*, (2017)	Configuration parameters	Resource requirement prediction	6 nodes	defined	Huawei's cloud	Not defined	Not defined	Not defined	Yes
Pal and Agarwal (2014)	Parameter tuning for memory usage	Modeling cluster deployment	5 node	defined	Not defined	Not defined	Not defined	Not defined	X

2.2.3 Machine learning based performance optimization

In this section, we have reviewed literature relevant to Hadoop optimization for resource and performance modeling. The main focus of this review is to highlight what machine learning techniques have been used and what was the purpose of their use. From purpose we meant to say which aspect of Hadoop ecosystem have been optimized and how it improved the job execution. There are two important objectives of using machine learning techniques, first one is the modelling of one or few Job characteristics and second is prediction based on the models built. All machine learning techniques are used to build models which can help to understand the patterns within a data. This model is later on tested on an unseen data which is is not part of the model building process. This process is called prediction. The percentage of correct prediction gives us a measure of goodness of the model built. We divided the current work into two broader categories, the performance modelling and performance prediction. Performance prediction usually involves both modeling and prediction processes.

Performance modelling

Kim and Kim [181] build a cooperative data and simulation modelling approach for performance analysis of the Hadoop system. Their modelling approach involved a workload model, which describe the application and disk Input/Output (I/O) processes, and a system model which involved the operations of Hadoop Distributed File System (HDFS) and MapReduce. They used Artificial Neural Networks (ANNs) for data modeling and Discrete Event System Specification (DEVS) for simulation modelling. Root mean square error was calculated to compare the prediction accuracy of real results with simulation results, and the results showed that cooperative modeling approach can achieve promising results as compared to individual data modeling and simulation modeling approaches.

Wang *et al.* [182] reported a method to tune the configuration of a large number parameters in Apache spark by using different machine learning techniques. They defined the performance of Hadoop by using equation.

$$Perf = F(p, d, r, c)$$

Where *Perf* denotes the performance of the Spark platform, *p* denotes the user's application, *d* denotes the input data, *r* denotes the platform's resources, and *c* denotes the configuration parameters of Spark platform, *F*is a function of *Perf* about *p*,*d*,*r*, and *c*. In their experiments, they only focused on configuration parameters *c*, to optimize the performance. *p*,*d*, *r* remained fixed in their experiments.They trained their model on random sampling on parameter space and used Decision tree (C5.0), logistic regression, Support vector machines (SVM) and the artificial neural network. Their results showed that decision tree model can give best computational performance and prediction accuracy for both binary and multi-classification.

Khan *et al.* [57] reported modeling of job time estimation in Hadoop clusters. They used historical records to employ the Locally Weighted Linear Regression (LWLR) technique to estimate the execution time of a job. They calculated resource provisioning by the Lagrange Multiplier technique. They evaluated their model on an in-house Hadoop cluster and Amazon Elastic Compute Cloud (EC2). Their experimental results showed up to 95.51% accuracy in estimating the deadline of Hadoop job.

Yigitbasi *et al.* [183] demonstrated a machine learning based auto-tune method for MapReduce applications. They run the experiments on two different clusters. They collected data of two different workloads on both clusters and then used different multiple Linear regression Models to build models. The have used root mean square error (RMSE), R^2 Statistics and basic statistics to compare the results from different workloads. They used support vector regression (SVR), M5Trees and ANN models to compute the accuracy and computational performance for different workloads. Their SVR model showed comparable or better prediction accuracy than other cost-based auto-tuner like Starfish.

Ataie *et al.*, proposed a design-time hybrid approach to build analytical model (AM) and machine learning(ML) models to estimate MapReduce jobs execution time in Hadoop clusters [61]. Firstly, they proposed an AM based on queueing networks to model the execution of jobs. The results obtained are exploited as analytical data which is used to train ML model. In order to create a more accurate performance predictor, new data from the operational system has been fed into the ML model. The limitations of their work includes the use of synthetic data to train the model initially. This is because real data would have noisy data which will not give accurate predictions. So they used an incremental and iterative strategy to introduce real data into the learning system. They have used linear regression, polynomial regression and Guassian support vector regression to train on feature set which includes number of map and reduce tasks, average and maximum values of map execution time, average and maximum values of reduce execution times, average and maximum of shuffling times, dataset size and number of cores. Their experiments have been performed on CINECA, the Italian super-computing center. PICO, the Big Data cluster available at CINECA, is composed of 74 nodes, each of them boasting two Intel Xeon 10-core 2670 v2@2.5GHz, with 128 GB of RAM. Out of the 74 nodes, 66 are available for computation. The dataset used for running the experiments has been generated using the TPC-DS benchmark data generator. Their results show that the hybrid approach performed 21% better than pure ML approach.

Foo *et al.*, applied machine learning approach combining evolutionary NN and sensitivity analysis(SA) to model the energy prediction of the cloud data center [60]. They used genetic algorithms for feature selection by utilizing SA as guide.The feature subset, along with the NN architecture, is represented by a structurally-inclusive encoding scheme in the form of a chromosome matrix. A population of chromosomes is maintained through the genetic process of crossover and mutation. The chromosome's fitness is evaluated at every generation to determine its survival in the next generation. The algorithm "prunes" away connections between the neurons to de-emphasize a particular input neuron's contribution to the NN s output should such an input feature causes the chromosome to have a weak fitness. The eventual NN is a network with reduced complexity. This leads to a model with better generalization and at lower computational cost. A complex NN has high computational cost and the tendency to overfit. The proposed evolutionary NN combined with SA helps to extract the key factors impacting the cloud data center energy performance. A Hadoop cluster with Hadoop Distributed File System (HDFS) and MapReduce stack (Facebook Apache Hadoop version 0.20.1), is set up to perform the experiments. The software stack is installed over 6 x HP Proliant DL360P and DL380P Gen8 servers, consisting of 120 cores housed within a single rack. Each server is equipped with 64 GB memory, dual socket 6-core Intel(R) Xeon(R) CPU E5-2667 @ 2.90GHz with hyper-threading technology. The Hadoop cluster comprises of 1 x NameNode, 1 x secondary NameNode and 4 x datanodes. All nodes are installed with CentOS 6.5 and running on bare-metal hardware without hypervisor or virtualization.

A mixture of MapReduce jobs, in the form of the WordCount application and Sort application are executed during the experiments. The Hadoop MapReduce counters such as the Map file byte read, the Reduce file byte write and etc. are extracted using the build-in Hadoop web admin user interfaces (UIs). The counters can be access via the HDFS NameNode admin at port 50070 and the MapReduce Job tracker admin at port 50030. The other counters such as the CPU and memory utilization and network IO are collection using Ganglia, an open source monitoring system. The power consumption data is collected using the Raritan intelligent Power Distribution Unit (iPDU), through which the servers' power supply are connected into. For sensitivity analysis they have used partial derivative method and Weights method. Their results indicate that file size, time taken to complete the task and resources assigned to job has high impact over energy consumption.

Xu *et al.*, presented a deep learning approach for speculative execution and replication of deadline critical jobs [184]. They used machine learning approach called deep neural network (DNN) to solve straggler migration problem. A Deep neural network, modeled loosely after the human brain, consists of an input layer, hidden layers, and an output layer. The input layer takes inputs from a dataset, and passes data to the following hidden layer. Each hidden layer, which is not directly exposed to the input data, consists of multiple neurons. A neuron is a computational unit that takes weighted inputs and generates an output using an activation function. The output layer takes results from hidden layers, and produces a value or a vector of values [185].

For a job with deadline, Xu *et al.,* proposed a metric called probability of completion before deadline (PoCD) to quantify the likelihood of straggler migration strategy to meet job execution deadlines [184]. Their model consist of two deep neural networks, first for the task execution time estimation and second one for determining the number of extra attempts to launch for each straggler. Their method relies on task progress score, time of estimating the task execution, amount of bytes read and written on HDFS and CPU execution time.Their cloud testbed consist of 80 nodes. Each node has 8 vCPUs and 2GB memory. The physical servers are connected to a Gigabit Ethernet switch and the link bandwidth is 1 Gbps. WordCount with a 1.2GB workload for WordCount from Wikipedia. They compared their results with speculative-resume with quantitative analysis strategy which is a model-based optimization method. Their results show that their DNN-based method is not able to accurately estimate execution time of straggler as compared to model-based strategy, but it performs better than Hadoop-S and Hadoop-NS in terms of PoCD and cost.

Performance prediction

Berral *et al.,* described the ALOJA project and different tools provided this framework [40]. This is a Hadoop based long-term collaborative project between Barcelona Supercomputing Center and Microsoft to characterize the cost-effectiveness of Big data deployments along with their run-time performance. The project has created an open repository featuring 40,000 Hadoop jobs execution and performance details. This tool compares the software and hardware configurations for deploying cost-effective Hadoop deployments.

The framework has been tested over a high-end cluster of 2 to 16 nodes per execution, 64GB of RAM machines; 6 SATA2 SSD drives as RAID-0 storage (1.6GB/s read, 1GB/s write), and local 3TB SATA HDD storage per node; network interfaces with 4 Gigabit Ethernet and 2 FRD InfiniBand ports per node providing a bandwidth peak up to 56Gbps, and InfiniBand switching , and a Cloud IaaS with Microsoft Azure environment using A7 instances using 3 to 48 data-nodes per execution plus a head-node, with 8 cores and 56GB of RAM;

mounting up to 16 remote volumes limited to 500 IOPS. In order to enhance the knowledge discovery capabilities they used ML algorithms like Regression trees, Nearest neighbors, FFA Neural Networks, and polynomial regression. They showed that resulting models can forecast or predict the execution behaviors and times for new configurations and hardware choices. Their work has limitation of not exploring the relationship of configuration and hardware attributes.

Ling *et al.*, [186] predicted the execution time of two Hadoop applications under different input size. They used linear regression model and error correction neural network model to train the model and predict the total execution time. They deployed 5-node cluster with MapReduce framework. The hardware specifications of the cluster are: HP with two Intel(R) Xeon(R) CPU E5649 processors, 2.5GHz, 64bit, 32GB memory and 4TB disk. Hadoop versions 2.3.0 was deployed and real-time network traffic was collected and stored in HDFS. In order to evaluate the results, they used *mean relative error*, *mean absolute percentage error* and RMSE*root mean square error*. RMSE is given below,

$$\text{RMSE} = \sqrt{\frac{\Sigma_{i=1}^{M}(actual^{(i)} - predicted^{(i)})^2}{M}} \tag{5}$$

They also calculated PRED(25) to see the fitness of model. This is given below,

$$\text{PRED}(25) = \frac{m}{M} \tag{7}$$

where m is the length of dataset where MRE is less than o.25. The model is a better fit if the value of PRED(25) is close to 1.0. Their results showed that error correction neural network model gives 2% better prediction of overall execution time under different input sizes as compared to regression model.

Summary

This section has presented the work related to modelling and prediction of resource and performance using machine learning methods. There are several gaps highlighted from this review as shown in Table 2.5. Most of the studies has not taken into account the complex relationship between Hadoop job characteristics for modeling and prediction. The reason can be the non-flexibility of the modeling method they choose, the dataset for modeling might have missing values or different data-types which can lead non-robustness of the technique. This is very important because a model might have considered type or size of workload but how it can efficiently predict the performance or resource utilization of a Hadoop job when several important aspect has not been included in this process. For example the type and structure of input, hardware profile or configuration parameters.

Table 2.5: Summary of literature review for machine learning based Hadoop optimization.

Paper	Features included in the model building process						
	Config. parameters	**System resources**	**Execution time**	**Input/ User application**	**Hardware**	**Workload**	**Relative importance of features**
kim and kim (2017)			✓		✓	✓	
Wang *et al.*, (2016)	✓		✓				
Khan *et al.*, (2016)			✓				
Yigitbasi *et al.* (2013)						✓	
Ataie *et al.*, (2016)					✓		
Foo *et al.*, (2016)		CPU, memory, disk and network				✓	✓
Xu *et al.*, (2017)		CPU	✓				
Barrel *et al.* (2015)	Block size, number of replicas, and mappers		✓		✓	✓	
Ling *et al.*, (2016)				✓			

2.3 Chapter Summary

This chapter provides a comprehensive literature review, including fundamental concepts related to Hadoop and its ecosystem with a focus on Hadoop distributed file system and Map-Reduce programming framework. The working of Hadoop components and the important factors affecting the performance and resource utilization has also been discussed. Besides, in order to motivate the research of the thesis, this chapter has also discussed and analyzed the shortcomings in the related literature.

Some of the general challenges for modeling resource and performance utilization of Hadoop cluster are: a) ensuring the internal validity while designing the Hadoop cluster and experiment, b) estimating the correlation or interdependence of Hadoop job characteristics, c) improving the models to make prediction more generalizable for different cluster environmental changes. Based on the literature review, the specific limitations are:

1. There has been experimental evaluations to benchmark Hadoop performance for different storage devices [38] and emerging hardware resources like KNL [39], characterization of cost-effectiveness on deploying Big-Data systems under different hardware and software capabilities [40] and the impact of geographical diversity [41], but none of them has focused on problems such as repeatability or validity. The current studies lack the design of a robust experimental methodology. Not only that it is important to mention the experimental setup but also the experimental design details as well. Experiment methodology includes but not limited to the number of nodes, design of the cluster, data size, type of input, number of experiments, parameter inter-relationship, statistical significance of the results, the dataset used etc. Absence of robust methodology is a major factor in non-reproducibility of the results at some later stage by same author or someone else [42]. To the best of our knowledge, some of the studies did not report the details of dataset used [14, 15, 26, 62], applications used to run the experiments [14, 15, 108–112] and size of data [15, 159], number of experiments run [14, 15, 18, 26, 62, 109–111, 174], defining input [120].

2. There is an extensive list of more than 80 parameters that play an important role in determining the task performance [34]. Some limited work has been reported for identifying the impact of change in block size [18] and trying different replication strategies [26] to enhance throughput of the system. To the best of our knowledge, the inter-related impact of these parameters is yet to be investigated [18] [26] [14] [15] [62] [120] [109] [110] [111] [108]. The statistical significance of the results validate the differences in the performance of the two experiments. However, most studies did not report this [18] [26] [174] [14] [15] [62] [120] [108].

3. Traditional approaches of Hadoop performance characterization and parameter tuning include manual trial and error or white-box modeling methods. Hadoop has large number of parameters and prior knowledge of its internal structure is also required along with the cluster hardware profile, the type of input given and the dataset used. Machine learning models are black-box modeling methods which has the advantage of being robust, flexible and simpler to build. There is some limited work reported using machine learning, black-box techniques, to model and predict complex Hadoop performance characteristics. Researchers have used machine learning to model the problem like energy utilization of clusters [60] and performance prediction of jobs [61]. There is also some work reported to optimize the use of resources utilization [62]. However, to the best of our knowledge current literature has no work reporting the complex relationship between different Hadoop parameters [183] [61] [60], resource utilization [186] [183] [61] [60] [57] [181], job characteristics [40] [184] [183] [61] [60] [57] [181] and system's layout such as number of nodes [182] [186] [184] [183].

Chapter 3

Application of HAZard and OPerability analysis for the design of Hadoop cluster

3.1 Introduction

The goal of this chapter is to apply Hazard and operability (HAZOP) analysis to the design of Hadoop cluster experiments. HAZOP is a brainstorming activity ensured the internal validity of the system and represented a robust experimental design. The process outlined the detail analysis of the confounding elements that need to be considered while designing the Hadoop cluster, thus ensuring the smooth design and running of experiments. These confounding elements can seriously cause operational hazards which can not only affect the normal working of the cluster and execution of Hadoop jobs but can also introduce errors into measurements if not countered. Lack of such experimental design often results in fragile experiments, biased measurements and wastage of time.

3.1.1 Chapter goal

The objective of this chapter was to investigate the following research questions.

- **What factors and confounding factors were important for the design of**

Hadoop cluster? and How?

3.1.2 Chapter organization

Section 3.2 presents the introduction to Hazard and Operability (HAZOP) analysis, and discusses the reason as to why using HAZOP in our experimental design is important. Section 3.4 presents the application of HAZOP for Hadoop cluster deployment. Section 3.5 presents the overall summary of the application of HAZOP for Hadoop.

3.2 Background

Our aim in this work is to achieve reliable experiments. One aspect of this is how to achieve consistent results. This can also be thought of as internal validity which concerns the extent to which a relationship between a dependent and independent variable actually exists and is not obscured by confounding factors [187,188]. Experiments may produce inaccurate conclusion because the designers have not properly controlled design and experiment factors which may affect the final result. For instance, having an unreliable memory usage measurements because of some background service is running.

In computer science experiments were used for theory falsification [189], investigation of assumptions [190] and developing theory based upon observations [191]. But as pointed out by some researchers, the reliability of experiments or internal validity was compromised while designing the experiments in computer science [42,113–116]. This lack of reliability or internal validity was due to absence of control in experimental design [42,113–116].

HAZard and OPerability (HAZOP) is a structured and systematic approach involving thorough analysis of a system, its processes and components by experts to identify if any system or environment attributes in which the system operates, may result in deviation from the design intent [117]. HAZOP provides a methodology to design reliable and replicable experiments by controlling all variables and hypothetical scenarios in an experimental setup, allowing the validation of experiment's internal validity by predicting the relationship between dependent and independent variables.

In their work, Maxion et. al. [192, 193] investigated how to design experiments that identify potential confounds and appropriately control or mitigate them. In their case study approach, they use their experiments on keystroke biometrics for user identification to illustrate the difficulties in implementing reliable measurement in the presence of confounds.

Mansoori et. al., applied HAZOP methodology to a case study of network security experiments [43]. Their aim was to ensure reliability of the measurements by finding the confounding factors in the experiments. Their approach involved the case study of IP tracking where they defined the client honeypot as apparatus, web pages as subjects and request as stimuli.

Seifert et. al. [194] performed a hazard and operability study and identified hazards and bias which could be introduced into experimental studies using state-based high interaction client honeypot. Their study used experiments on real-world datasets demonstrated that subjects of studies the input dataset and its source of collection can have significant effect on a measurement study. Increased number of malicious web sites in datasets gathered from adult web sites and top level domains were observed. Their repeated measurement study on the number of malicious web sites in the .nz domain over an eight months period, revealed the impact of retrieval time on the outcome of a measurement study as significant increase and decrease in the number of detected malicious web sites were observed.

The aim of a HAZOP study was to identify risk and hazards into the operability of the system functions. Originally found in chemical engineering, we are applying it to the design of Hadoop cluster experiments [195, 196]. In our context, risks and hazards in a quantitative computer science experiment could translate into increased risk of an unexpected and biased variable which would impact the internal validity of the experiments. High internal validity was desirable because it indicates that confounds coming from the experiment itself were considered and dealt with in order to ensure we are measuring what we believe we are measuring. Confounds in this context were variables that cause hazards to the reliability of validity. External validity, on the other hand, is the degree of certainty in which it could be argued that the conclusion of the study will hold true and can be used to make predictions about a larger population. High external validity in an experiment warrants a high degree of certainty about the generalizability of the findings. For example, that the findings about the particular Hadoop cluster setup will generalise to similar Hadoop cluster setup but different number of nodes.

Absence of robust methodology was a major factor in non-reproducibility of the results at some later stage by same author or someone else [42]. For example while characterizing and optimizing performance of Hadoop ecosystem, some of the studies did not report the details of dataset used [15, 62], some did not mention the size of data [15, 159] or number of experiments run [18, 174] or testing the results for their statistical significance.

Our approach was inspired by the need for some form of guidance to designers of reliable experiments on Hadoop clusters. We would like to provide Hadoop-specific guidance than is found in the work by Maxion et. al. and Mansoori et. al that applies to cybersecurity experiments. This not only identified the potential hazards in our experimental design but will also highlighted the possible actions that can be taken to counter these operational problems.

3.3 Hazard and Operability analysis

We adopted HAZOP as a methodology for reviewing the design of our Hadoop experiments. This met the needs identified above because it was neither too prescriptive nor specific. The following approach was based upon work on applying HAZOP to cybersecurity experiments [197]. Based upon the reviews of the literature a HAZOP study involves the following steps:

1. Firstly, the methodology of the study was defined. It described how the experiment was meant to be conducted. The objective was to outline different events that were taking place in our experiment. We might be interested in observing and measuring some of these events. The high-level view of the experimental design was analyzed to investigate any confounding factors. Parameters of experimental methodology such as apparatus, stimuli and subjects need to described in details [195, 196].

2. Possible hazards were defined by applying the pre-selected guide words to the lower-level view of the experiment. These hazards were called as deviations as they altered the usual working of the experiments. It was not feasible to apply all the guide words to every component or process. At this stage the objective was to take care of all possible deviations.

3. The possible consequences, as a result of some deviation, were defined. This helped us make an assessment whether actions need to be taken to minimise or mitigate the deviation.

4. We could eliminate or minimize the impact of hazard by taking appropriate action [198]. This largely depended on the severity and likelihood of the hazard. If necessary actions were not taken, it is a threat to validity of the experiments.

This can be understood by an example of designing a Hadoop cluster-based experiment. If the cluster CPU measurement shows that only Master nodes is being utilized while running an job on bid dataset, The possible hazards could be the incorrect configuration of the cluster, incorrect configuration of switch, or coding error. The consequences could be the incorrect measurement, incorrect configuration on the switch or Hadoop framework. We could take actions to check for the code first, check whether the cluster nodes were configured correctly and can communicate with each other through password-less SSH.

Features of the HAZOP approach mentioned in this chapter may vary compared to the original HAZOP model used in chemical engineering and was adapted to suit the requirement of the study. Our method largely resembles Mansori *et. al.* work, that applied HAZOP to cyber security experiments to find confounding variables [43]. This practice was also accepted by many other studies applying HAZOP to other fields [199, 200]. For instance, having a team leader and a group of specialists from other fields was not deemed necessary for this research and was not followed. Our methodology followed a similar approach in analysis to the research performed by Seifert et. al. [194]. HAZOP methodology in this thesis was mostly focused on identifying the confounding factors in our experiments that may result in erroneous results or experiment failures. The main components of our HAZOP study included:

1. Study Node: Specific points in the design where the deviations are studied.

2. Guide words: Short words suggesting deviations from the correct state of the system . The standard keywords of HAZOP include No, More, Less, As Well As, Part Of, Reverse, and Other Than. Other keywords such as Early, Late, Before and After were considered to be derivatives of "Other Than" keyword to apply more specific meanings to a deviation [201]. The application of the guide words results in hypothetical deviations related to that component or process (Table 3.1). Some guide words should be interpreted broadly as different form of these words may be more appropriate to the process or artifact to which they were being applied. Guide words used in our research were:

3. Deviation: Combined with guide words, they define a more descriptive departure from the design intent of the component or process.

4. Cause: Described the potential cause which would result in the deviation.

5. Severity (Denoted by S, 1=Low, 2=Medium, 3=High): Expressed the potential hazard and threat if the deviation occurs. If it was not possible to entirely eliminate a hazard, then it should be minimised in respect to its severity and likelihood [198].

6. Likelihood (Denoted by L, 1=Low, 2=Medium, 3=High): Expressed the possibility of the deviation happening within the study node. The focus of our HAZOP study was to identify experimental scenarios with high and medium likelihood of occurrence and proposes measures to mitigate or significantly reduce their effects. The "cutoff" points represented the line below which the threats to validity of final data and result were so low and can be neglected [198].

7. Consequence: Described the potential results if the deviation occurs.

8. Required Action: Identified any measures to be taken to either remove the cause or eliminate the consequence.

Table 3.1: Guide words and their interpretations in our study

Guide words	Meaning
NO	No part of design intention is achieved
More	Quantitative increase in a parameter or variable
Less	Quantitative decrease in a parameter or variable
Other than	Compared to design something completely different happened
Unreliable	A value of measurement is not the same when measured again in the same context as it was measure before.
Part of	Qualitative decrease/ modification
As well as	Qualitative increase/ modification
Wrong	Wrong signal or action
Biased	Systematic error across a set of measurements usually introduced due to way the sampling done or the measurement process itself

Applying a HAZOP methodology to a repeated measurement experiments, we questioned every component of the experimental design including data input, datasets, processes, cluster node hardware, Hadoop framework and measurements used in the interaction. These lower-level design components and their associated hazards were each be discussed in detail in the following sections followed by HAZOP tables. It was a good way to explore the relationship between risk, severity, likelihood and required actions in a HAZOP study. In our study of HAZOP, we focused on finding confounding factors which might introduced any hazard into the study such as wrong measurements.

3.4 HAZOP for Hadoop

As Hadoop is a multi-node cluster which involves many different elements of hardware and software working together. These individual elements needed to be integrated and configured in a specific way. As designing a Hadoop cluster required good and carefully use of skills and experience, it was very important to look for potential operational hazards that can cause deviate the cluster from its normal functionality. Below is the detailed HAZOP analysis of the Hadoop cluster which was not only helpful for this study, as it helped us finding the confounds in our experiments, but also for the future researchers who want to design their own Hadoop cluster and look for the solutions of the different operational hazards they face.

3.4.1 Apparatus (Hadoop cluster)

Multi-node Hadoop clusters may vary in their design and capabilities. However, all share some basic components that are discussed below:

- Cluster nodes: These were the physical cluster resources responsible for storing and processing data.
- Switch : Different systems of the cluster were connected by either a physical or virtual networking device such as switch. All of the communication was done through this.
- Hadoop framework: It was the software responsible for taking care of all cluster job related tasks.

The hazards introduced due to failures of different components of the cluster and the actions taken to mitigate their impact are listed in table 3.2.

Cluster nodes

These were the computer systems that were connected through some networking device. When configured using Hadoop these different systems act like a single storage and processing resource. The data might be physically distributed unevenly over different systems but as a cluster the data is distributed in a way that if one of the system is turned off or fails, the job is still being completed.

A Hadoop cluster has Master-slave architecture. One system was designated as Master or NameNode while all other systems were slaves or datanodes. The Master or NameNode was of the central importance regarding the configuration of Hadoop cluster. It recorded the IP address of all of the slave nodes. It was connected to datanodes using password-less secure shell (SSH) [202] protocol. NameNode kept the track of all of the available resources. It also kept track of the status of jobs running on different datanodes. As Hadoop has its own file system, the NameNode kept the record of how and where different blocks of data were being stored over the cluster resources.

The job tracker present at Master node keeps track of the different tasks of same job executed at datanodes. A job was being divided and subdivided by using a MapReduce framework. The individual tasks were assigned and performed at datanodes. These tasks were monitored by using task tracker modules present at datanodes. Datanodes or slave nodes were responsible for storing different blocks of same data.

Bias might potentially be introduced into our study when setting up the cluster nodes can affect the internal validity of the system. There were two important aspects of cluster nodes configuration. First was hardware and second was software. It was important to know the hardware profile of a multi-node cluster. If different nodes have different CPU's memory or storage capacities the cluster is heterogeneous, but if all nodes have same hardware profile, the cluster is called as homogeneous cluster. In our study, we setup homogeneous cluster as we want to investigate how different jobs used different resources because the nodes of the cluster. From software perspective, we have to take care of the operating system installed on the individual nodes. Each node must have same operating system as Hadoop configuration largely depends upon the operating system it is being installed [203].

Some of the common hazards that could affect the job execution on cluster nodes were:

- There was no cluster node present that could execute the Hadoop job. Possible causes were that cluster hardware was being replaced, it was being turned off or damaged. Another possible reason could be the absence of Java and Java run time environment.

- When the cluster size was increased by adding one extra node. This was important hazard for utilization of all resources because once we add any extra node in already configured Hadoop cluster, the new cluster node need to have same configuration in various XML files. Secondly, the default Hadoop distributed file system need to be reformatted the file system of whole cluster.

- When a cluster node was being removed from an already configured cluster, the Hadoop distributed file system need to be formatted again. This would make the Hadoop framework aware of the available resources and thus resource and job performance modeling could be accurately predicted.

Switch

The switch had central importance in the execution of Hadoop jobs. All the communication between different nodes of the cluster was via a switch. We used a SDN-configured switch while designing our Hadoop cluster. The control plane and data plane of such a switch were separated. The controller to switch, Faucet, was installed and configured on Master node. This enabled us to monitor packets transferred in or out of the different nodes of cluster. Gauge was an integral part of OpenFlow switch which fetches the port statistics of the switch and store it in a database. We used InfluxDB to store network traffic data for each of our experiments. All of these network related measurements could be seen by using a data visualization dashboard like grafana.

Some of the common hazards caused due to deviations of normal functionality of switch were:

- There was no switch that allowed connection between the host systems. Possible causes could be that switch might be turned off, broken or unplugged.

- Another possible hazard was the replacement of switch which could also result in no communication between the systems. This hazard takes place when the new switch has not been configured to allow the communication between devices.

- Wrong switch supplied by the vendor or product was not as per specifications could also be an operational hazard in Hadoop cluster which could be avoided by cross checking the ordered product along with features it provides.

Hadoop framework

We configured framework to distribute and manage data on cluster hardware resources and then execute job using built-in programming paradigms like MapReduce. For convenience we did use many add-ons on top of basic Hadoop framework such as Hive. We configured Hive to store and query structured data on Hadoop framework. The installation of correct version of Hadoop was also important. If all nodes of the cluster did not have same Hadoop version installed and configured on it, the multi-node cluster cannot be configured correctly and it will result in under utilization of some resources.

There are many pre-requisites for the installation of Hadoop framework such as enabling and installation of Java and SSH on each and every node of the cluster. Some of the deviations were as under:

- The Hadoop framework was not present. Possible causes of this deviations could be the complete absence of Hadoop framework on system, Java was not installed and configured on cluster, Hadoop configurations and environment variables were missing.
- Another possible deviation was that part of Hadoop framework was missing. Possible cause of this behavior could be that some files of the Hadoop framework were missing.
- Installation of wrong framework was another important deviation. Possible causes could be the downloading and installation of wrong version of framework and wrong configuration.

Study Nodes	Guide word	Deviation	Possible causes	S	L	Consequences	Required action
Switch	No	No connection to the host	Switch might be turned off	2	1	No communication between devices	Check that switch is powered on
			Plugged off	2	1	No communication between devices	Check that switch is plugged in
			Broken	2	1	No communication between devices	check physical integrity of the switch
	Other than	Switch has been replaced	Switch not configured	3	2	No communication between devices	configure the switch and test that communication is taking place between cluster nodes
	Wrong	Wrong switch connected	Vendor supplied wrong product	3	1	Cluster cannot be setup	Check the product serial number and the services it supports or ask the vendor
Cluster node	No	No cluster present	Cluster hardware damaged	3	1	Cluster will not be accessible	Check physical integrity of the hardware
			Cluster nodes replaced	2	1	Cluster will not be accessible	Check the systems configured for Hadoop cluster. For any new system configure it first to be included in the cluster
			Cluster nodes turned off	2	1	Cluster will not be accessible	Check the power supply to the cluster is working
	As well as	Cluster node hardware upgraded	Replaced hardware component of a node with different component	3	1	Different processing at different nodes, affecting the overall performance and resource measurement prediction	Restrict physical access to cluster nodes and always tag the product and maintain the inventory

	Part of	Cluster node hardware downgraded	Replaced hardware component of a node with a component of less capability	3	1	Different processing at different nodes, affecting the performance measurement prediction	Tag and test the hardware before including into cluster, restrict access to the system
Hadoop framework	No	Hadoop is not present	Java not installed	3	2	Hadoop does not respond	Check the installation and configuration of Java on all nodes of the cluster, check java version
			SSH not installed and tested	3	2	Job does not run	Check SSH is installed, configured and tested on all nodes of the system
			Hadoop configuration and environmental variables not defined	3	2	Hadoop does not respond and job does not run	Check environment configuration files of Hadoop and operating system
	Part of	Hadoop not running	Installation files missing	3	1	Hadoop does not start or jobs not executed	check that all necessary files are there, download and replace the files
	Wrong	Wrong framework	Hadoop configured wrongly	3	2	Jobs are not executed, not distributed	check that Hadoop environment and configuration files for correctness
			wrong version of framework	3	2	Jobs are not executed, not distributed	check the pre-requisites for framework, or you have downloaded correct framework from website

Table 3.2: Hazards that could be introduced by Different components of the cluster and the actions required to mitigate or minimize the impact

3.4.2 Subjects (Performance and Resource measures)

Modeling performance and resource utilization of Hadoop cluster was the primary objective of this thesis. So, identifying any hazard that could stop, modify or alter the collections of performance and resource utilization measures was very important. The measures we were collecting are network traffic measures (both packets_in and packets _out), CPU usage, disk usage, memory usage and execution time. The possible hazards and the actions taken to minimize their impact were discussed below and were presented in Table 3.3.

For network traffic measurement, it was very important that the switch was installed and configured correctly. As the switch was SDN-enabled, so the correct installation and configuration of Gauge and InfluxDB were very important. Gauge fetches the network statistics from the switch for each port of the cluster and InfluxDb served the purpose of storing those network statistics. Some of the hazards that could result in the deviation from the normal collection of network traffic measurement were:

- There were no traffic measurements recorded after the job was completed. The possible causes of this hazard were; influxDb might not be installed correctly, Faucet or Gauge might not be configured, or licence for the SDN switch had been expired, or the presence of a wrong switch in the cluster.

- Another important measurement is the collection of incorrect traffic measurements. Possible causes for this could be some problem with the switch or the device was not as specified by vendor.

For measurements of system utilization such as CPU usage, disk usage and memory usage, it is important that all the physical hardware be correct and functioning as expected. Any malfunctioned component could affect the operability and performance of the whole cluster. These system performance measurements were collected during the execution of Hadoop job at each node of the cluster. Once the job was finished the measures were stored in a single file at a designated location on cluster. The hazards caused due to deviations were discussed below.

- Having no measurement recorded at the end of each job or experiment was one of the hazards. It could be caused due to the connection failure between the cluster nodes, job was not submitted, the scripts running the job and collecting these measures had error or bug or some libraries need to be installed on the system such as psutil library of the python.

- Another hazard was the collection of wrong system performance measurements. The possible causes were the presence of coding errors in scripts.

- Unreliable measurements were the most important hazard for such an experiment. The possible causes of this hazard were: job might be terminated earlier, some background application or service was being activated or running and a component hardware was being changed.

- Recording the system performance earlier than expected was a very serious hazard. Possible cause was faulty script writing.

- Similar to early measurement, late measurement was also a hazard. It was also caused by faulty script writing.

The execution time was an important performance measure for different jobs executed over Hadoop cluster. The overall execution time of a job indicated the impact of particular configuration parameter. It also helped to differentiate between the performance of Hadoop jobs when querying differently over same dataset or executing same query structure for different datasets. The overall execution time was recorded by scripting in our experiments. Some important hazards were discussed below:

- Having no execution time recorded was an operational hazard in this experiment. The possible reason could be the presence of coding errors in the scripts or supporting libraries were not installed.

- When same job was showing more or less execution time than expected, it was an important hazard for our experiment as it makes the measurement unreliable. The possible causes could be the running of some extra services or background application while executing that job.

- If the execution time recorded was wrong, the data analysis could be performed. Possible reason for this kind of hazard could be the errors in script, or using incorrect library.

Study Nodes	Guide word	Deviation	Possible causes	S	L	Consequences	Required action
System performance measurements	No	No measurement	Scripts has errors	3	2	No performance measurements collected	Check the code written for any errors or bugs and test it
			Libraries are not installed	2	3	No performance measurements collected	check whether the any function or module used in scripts require the installation of libraries
			Job has not been submitted to framework	2	2	No performance measurements collected	Check that Hadoop is up and running and it is executing jobs
			No communication between cluster nodes	3	1	No performance measurements collected	test the ssh communication between the cluster nodes
	wrong	Wrong measurements collected	Scripts has errors	3	2	Results not interpretable	Check for any logical error in the script
	Unreliable	Unreliable measurements for same experimental setup	Job terminated early	3	2	Results not reliable or repeatable	check that the job has not completed, cluster is online and all services are running
		for same experiment	additional application or service running	2	1	Results not reproducible	check for the firewall settings of the system or any other services running over the cluster
			component hardware is changed	2	1	Results not reproducible	Check for any changes in cluster hardware, restrict access to cluster, test and tag the components
	Early	Measurement recorded before the end of job	Measurements are taken at particular time, Faulty script	3	1	Results not reliable	Check the code for possible logical errors

	Late	Measurement recorded after some time	Measurements are taken at particular time, Faulty script	3	1	Results not reliable	Check the code for possible logical errors
Network traffic measurements	No	No traffic measurements	InfluxDb not configured, Faucet and Gauge not configured, licence expired	3	2	No packets and bytes transferred data available for analysis	
	Wrong	Wrong traffic measurements	Switch is not as specified by vendor	3	1	Results are not reproducible	Check the device integrity, specifications and functionalities
Execution time	No	No time recorded	Errors in script	3	2	No job execution time recorded	Check the code for possible logical errors
			libraries not installed	3	2	No job execution time	Refer to the lanGauge library requirement
			job not running	3	1	No job execution time	Check all services of Hadoop are up and running, or any errors during job submission into the Hadoop framework
	Unreliable	Execution time longer than expected	Extra services or background applications running	3	1	False execution time measurement	Check for any firewall settings, or any unwanted service running on the system
	Wrong	Execution time is wrong	Libraries missing	2	1	False execution time measurement	Check that correct library or supporting module has been installed
			script has logical error	3	1	False execution time measurement	check for any logical error in programming, test the program first

Table 3.3: Hazards that could be introduced into subjects (performance and resource measures) and actions taken to mitigate their effects

3.4.3 Stimuli (Hadoop job)

Once the cluster had a fully functional Hadoop framework installed on it, the parallel and distributed processing of data was initiated by submitting a Hadoop job. A Hadoop job had several characteristics which could affect the overall performance and resource utilization of cluster. Some of the key hazards were discussed below:

Job input

Hadoop job input was an important element of a job. As our experiments were executed on a Hadoop-Hive setup, the input was given in the form of SQL-like query. The query format was known as HIVEQL and it initiated a Hadoop job. Query is mostly a SELECT statement with different clauses in it. We used ORDER BY clause and GROUP BY clause with varying number of columns in each clause. Some of the operability-related hazards of input were:

- No Hadoop job started as we input the query to initiate the job. The possible reason could be the absence of Hadoop framework on the system, or another job was in progress, or Hadoop not configured correctly or out of memory or system resources.
- Wrong input could also lead to an operational hazard. The possible cause could be the incorrect syntax of Hive query.

Dataset

A Hadoop job always required some dataset to process. We can use any dataset of our choice to run Hadoop job and monitor system performance and resource utilization. Using Hive required a structured dataset to be uploaded into the HDFS. The dataset was stored in the form of table and we could use Hive query to start a MapReduce job. Common hazards affecting validity of our experiments were:

- If the dataset was absent the Hadoop job would not be executed. Possible reason could be that the dataset file was not uploaded into the HDFS.

- Using an incorrect dataset was another important hazard. It would result in termination of Hadoop job every time the input was given. Possible causes could be the use of a wrong dataset file or referring to wrong Hive table while designing the query.

- Operability of Hadoop job could also be compromised when a part of dataset is missing. Possible causes were that the raw data included in the system might included unwanted characters or some missing characters.

Configuration parameters

A Hadoop job was controlled and affected by configuration parameters. These configuration parameters were defined before the beginning of a job. If not, the Hadoop framework assumed the default values of these parameters. An important hazard related to configuration parameters was that wrong value selection can affect the overall job execution. Possible cause could be the wrong scale of value for an important parameters in XML files such as assigning 500 to replication factor could increase the size of the originally stored data in HDFS by 500 times, or assigning 3 to block size will result in very small blocks of data and thus affecting overall performance and resource utilization of cluster.

3.5 Summary

We applied HAZOP to identify many hazards in our study. We also highlighted the possible actions one could take when encountered with such operational problems. It was recommended to conduct such analysis before the design and implementation of actual experiments as it ensured the internal validity of the experiments. There were numerous efforts in the literature that showed the importance of such analysis for any process or setup. But currently, we haven't came across any such analysis for the design of Hadoop cluster.

Study Nodes	Guide word	Deviation	Possible causes	S	L	Consequences	Required action
Job input	No	No job started	Already running a job	1	2	Job not executed	check whether there is another job running
			Out of resources	3	2	Job not executed	check for the correctness of configuration parameters, or increase the cluster hardware capacity
			Hadoop not configured	3	2	Job not executed	Check for the Hadoop environment variables, or installation requirements are fulfilled
			Dataset not present	2	1	Job not executed	Check for the data present on HDFS
			Referring to other dataset	2	2	Job not executed	Check the syntax of job input for correctness
	Wrong	wrong job input given	Input syntax not followed	2	2	Job does not start	Check for the correctness of syntax for input
			Dataset not present	2	1	Job terminated early	Check whether data is present in the HDFS and accessible by Hive
			Referring to wrong dataset	2	2	Job terminated early	Check fro the correctness of name of dataset
Dataset	No	Dataset is absent	Dataset file not loaded	2	1	Job will never execute	Check whether the dataset is being uploaded into HDFS
	Wrong	incorrect dataset used	Dataset was incorrectly downloaded	3	1	Incorrect experimental design, job will terminate	Check for the correctness of dataset file

	Part of	Dataset has missing values	Dataset pre-processing has not been done	3	2	Job will terminate, wrong output, wrong performance measurements and execution time	Pre-process the dataset by using some tool to get rid of any missing or unwanted characters
	As well as	Additional wrong entries	Dataset pre-processing has not been done	2	2	Job will terminate, wrong output, wrong performance measurements and execution time	visualize the data to verify structural integrity
Configuration parameters	Wrong	Wrong configuration parameters values	wrong scale of value	3	2	Over-utilization or under-utilization of resources	See the documentation of framework to understand the functionality and impact of the parameters.

Table 3.4: Hazards that could be introduced by stimuli and the actions taken to mitigate their impact

Chapter 4

Characterization of Hadoop performance and resource utilization by parameter tuning and job input

4.1 Introduction

The previous chapter used risk and hazard analysis of our experimental design to help us improve it by identifying hazards to mitigate. This chapter investigates the empirical relationship between different Hadoop parameters, network and system performance measures. The parameters taken into account were block size, replication factor and number of mappers. The reason to choose these parameters was that they are the most commonly discussed configuration parameters in the literature. The type of job can also impact the performance (overall execution time) and resource utilization of cluster and it is an area which is wide open to explore. Hence, we have considered the query structure while estimating the cluster performance and utilization behavior. In order to measure the impact of each parameter, we analyzed packets in, packets out, CPU usage, disk usage, memory usage and execution time of different job.

4.1.1 Chapter goals

The goals of this chapter are to investigate the following research questions.

1. **What quantification can be done for configuration parameters affecting Hadoop job performance and resource utilization?**

2. **What impact the change in query has on utilization of different resources such as CPU usage, disk usage, memory usage, network usage and execution time?**

4.1.2 Chapter organization

Section 4.3 presents our design of our experiments including the dataset and experimental setup, Section 4.4 discusses the results of our experiments in order to characterize Hadoop performance and resource utilization. Section 4.4.1 discuss the results of the experimental to highlight the impact of changing block size on different performance measures. Section 4.4.2 presents the results to show impact of changing replication factor. Section 4.4.3 highlights the impact of changing number of mappers on different performance measures. Section 4.4.4 shows the importance of query structure and its impact on resource utilization and execution time of Hadoop jobs. Section 4.5 shows the summary of this section.

4.2 Methodology

This section describes the method for characterization of Hadoop performance and resource utilization by tuning parameters. The method consisted of three phases which are described below:

4.2.1 Phase 1: Hadoop job specification

Phase 1.1: We *selected the Hadoop parameters* to vary, such as like replication, block size, and number of mappers, and decided the values of these parameters to be used in our experiments. We designed five set of experiments to further our investigation.

1. *Experiment 1:* involved using four different block sizes i.e. 128MB, 256MB, 512MB and 1024MB. The number of replicas were *two* and number of mappers were *five*.

2. *Experiment 2:* involved using two different number of replicas i.e. *two* and *three*. The block size for both experimental runs was kept constant to 256MB and number of mappers were *five*.

3. *Experiment 3:* involved the running experiments for four different number of mappers i.e. 5, 10,15 and 20. The block size was kept constant to 128MB and number of replicas to *two* for all experimental runs.

4. *Experiment 4:* involved running the experiments to see the impact of the number of columns in `ORDER BY` clause of Hive query. Three queries were designed with *three*, *two*, and *one* number of column. Other configuration parameters were kept constant. The block size was 128MB, number of replicas was *two* and number of mappers was *five* for all experimental runs.

5. *Experiment 5:* involved running the experiments for different the number of aggregate functions in `SELECT` clause and the number of columns in `GROUP BY` clause of Hive query. Three queries were designed with *one*, *two*, and *three* number of aggregate functions. Other configuration parameters were kept constant. The block size was 128MB, number of replicas was *two* and number of mappers was *five* for all experimental runs.

Phase 1.2: This phase involved the *selection of dataset* to run Hadoop job. All experimental runs used Berkeley Earth Surface Temperature dataset.

Phase 1.3: The next step is *specifying the query* to decide the structure of query like how many columns to select and which clauses to be included. We used Hive-based query to run our experiments. Our query have varying number of clauses in `Select` clause, `Order By` clause and `Group By` clause as shown below. For experiment 1-3 query 1 was used to execute jobs, for experiment 4 queries 1-3 were used while for experiment 5 queries 4-6 were executed. The queries are described below:

```
Query 1: SELECT * FROM table_name ORDER BY col_1,col_2,col_3;
Query 2: SELECT * FROM table_name ORDER BY col,col_2;
Query 3: SELECT * FROM table_name ORDER BY col_1;
Query 4: SELECT aggregate_function_1 FROM table_name
        GROUP BY column_1,column_2;
Query 5: SELECT aggregate_function_1,aggregate_function_2
        FROM table_name GROUP BY column_1,column_2;
Query 6: SELECT aggregate_function_1,aggregate_function_2,
        aggregate_function_3 FROM table_name GROUP BY column_1;
```

Phase 1.4: Lastly, we decided upon the *performance metric* to measure like packets_in, packets_out, CPU usage, disk usage, memory usage and execution time, and *number of iterations* to run for each combination of the parameter settings. Each experiment was repeated 30 times.

4.2.2 Phase 2: Data collection

This phase involved automatic *collection* of performance metric measures during each run of Hadoop job. The Master node was connected to slave nodes via an OpenFlow switch. The associated Software-defined Networking (SDN) controllers were Faucet (operating as a learning switch) and Gauge (Monitoring port statistics). InfluxDB is a database optimized for storing large number of small measurements such as packets_in and Grafana is a monitoring dashboard used to display InfluxDB measures.

During each run of the experiment, we collected packets_in, packets_out, CPU usage, disk usage and memory usage with an interval of 10 seconds for entire duration of Hadoop job. For packets_in and packets_out data, we used InfluxDB to fetch data for ports connected to cluster nodes. For CPU usage, disk usage and memory usage data, we used python scripts to monitor these system statistics at each of the node.

4.2.3 Phase 3: Analysis

We calculated the mean resource usage and execution time for each node of the experimental setup. The boxplot analysis was used to get further insight into the results.

4.3 Experimental design

This sections described the experiments run to characterise the impact of changing important configuration parameters. We discussed the implementation and design of our experimental setup and Hadoop cluster in Section 4.3.2. We discussed the results of our experiments along with statistical significance of these results in Section 4.4. The following section describe datasets used in proposed method and experimental setup.

4.3.1 Datasets

The Berkeley Earth Surface Temperature Study combined 1.6 billion temperature reports from 16 pre-existing archives. The dataset had six attributes and almost 10 million records. The dataset is publicly available at `https://data.opendatasoft.com/explore/dataset/`.

4.3.2 Experimental setup

We built a *four* node cluster locally. Each node had Intel(R) Core(TM) i7-7700 CPU@3.6GHz,500GB disk space, 8GB RAM running Ubuntu 16.04 LTS. We configured the cluster for Hadoop 2.9.0 and Hive 2.3.3 on Master node. We used Hive-based query to run our experiments. During each run of experiment we collected packets_in, packets_out, CPU usage percentage, disk usage percentage and memory usage with an interval of 10 second. Each experiment was repeated 30 times.

As shown in Figure 4.1, Master node was connected to slave nodes via an

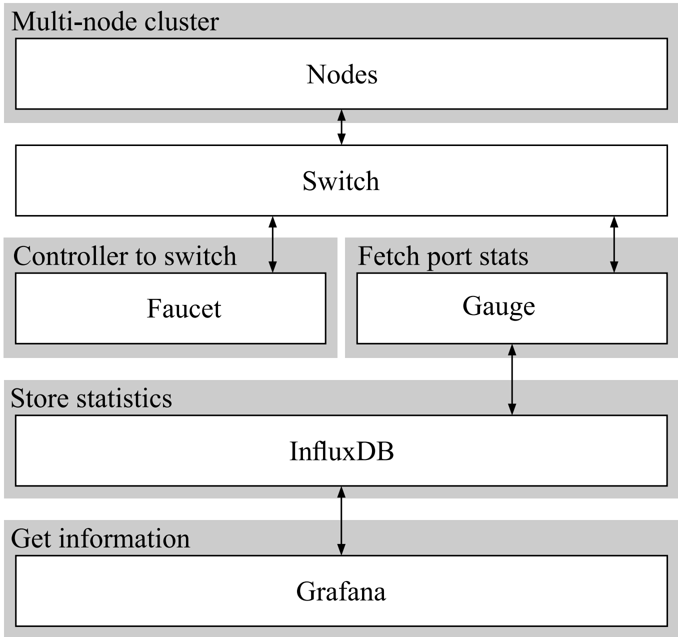

Figure 4.1: Experimental design

OpenFlow switch. The associated SDN controllers are Faucet (operating as a learning switch) and Gauge (Monitoring port statistics). Both Faucet and gauge were the version 1.6.8 and installed from source [204]. InfluxDB is a database optimized for storing large number of small measurements, Prometheus is a tool allowing extraction of information about network state from Faucet and Grafana is a monitoring dashboard used to display InfluxDB measures.

For statistical significance, we conducted two sample t-test. For each of the experiments, we tested following null and alternate hypothesis with $\alpha = 0.05$.

H_0 = Means are equal

H_1 = Means are not equal

4.4 Results and discussion

This section describes our results and discuss the results in details. Impact of different Hadoop parameters on our performance measures has been analyzed, such as the impact of Block size, impact of replication factor and impact of input data.

4.4.1 Impact of Block size

In this particular setup we analyzed the impact of changing HDFS block size on network and system metrics. We used four different block sizes of 128MB, 256MB,512MB and 1024MB. The replication factor was 2 for all the experiments and each experiment was repeated 30 times. Number of mappers and reducers were set to default value, which were 5 and 1 respectively. The Hive-based query was used to run experiment which was, *Select * from table_name order by column_1, column_2, column_3;*

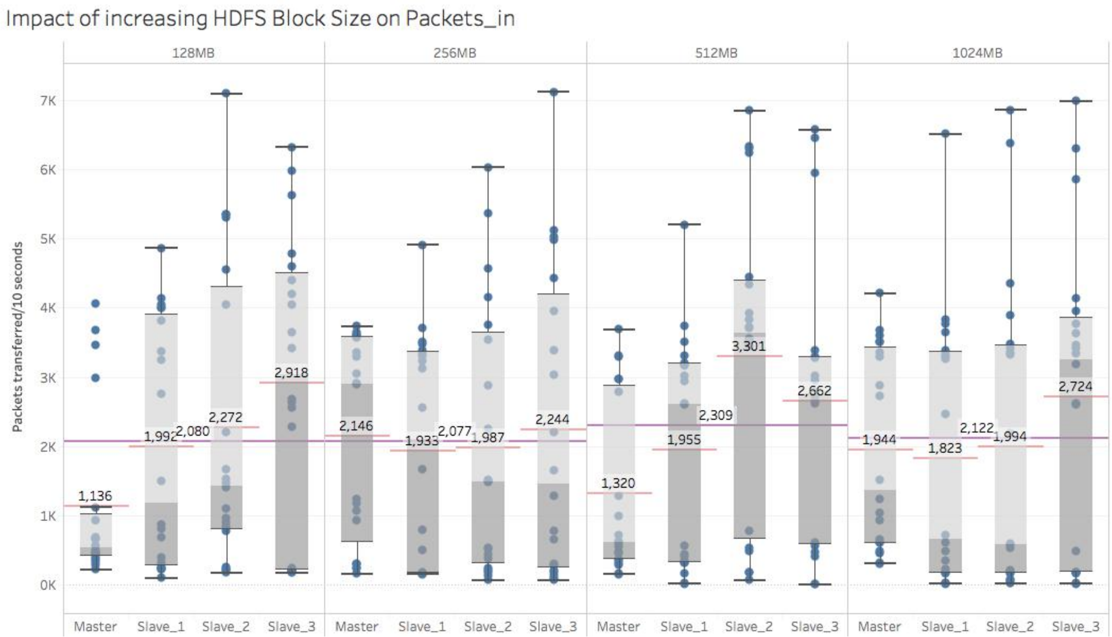

Figure 4.2: Box plot and average packets_in data for 30 runs for different block sizes.

Figure 4.2, show the impact of increasing HDFS block size on packets_in. There was a slight increase in the overall average packets transferred when we increased the block size from 128MB to 256MB. This increase in overall average continued as we increased the block size to 512MB. But as we further increased the block size to 1024MB, the overall average dropped down. For individual nodes, the Master node showed a smaller mean than the overall mean in all of the experiments except for the block size 256MB, where it was higher than the overall mean. Slave-1 consistently showed similar individual means as that of overall mean across all experiments. Slave-2 showed similar behavior as that of slave-1 except for 512MB where the individual node mean was much higher as compared to the overall mean. Slave-3 showed higher individual mean than the overall mean in all of the experiments.

Figure A.1, show the results from packets_out. Block sizes 128MB and 256MB showed similar overall averages. There was a slight increase in overall mean for 512MB block size and then there was slight decrease for block size 1024MB. For individual nodes, Master node showed higher mean than overall mean in all of the experiments. All of the slave nodes showed less than overall mean in all of the experiments except slave_2 for 512MB.

Figure A.2, showed the CPU percentage usage for different block sizes. Overall mean was similar for 128MB, 256MB and 512MB block sizes. It increased for the block size 1024MB. Mean of individual nodes showed that the Master node and slave-2 always performed higher CPU usage than other nodes and the overall mean as well. Slave-1 and slave-3 always showed lower than the overall mean in all of the experiments.

Disk percentage usage as shown in Figure A.3. As we increased block size from 128MB to 256MB, there was a decrease in overall and Master node disk usage. Disk usage gradually increased as we further increased the block size to 512MB AND 1024MB.

Virtual memory usage is shown in the Figure A.4. Memory usage showed similar pattern as that of CPU usage for overall and individual node means. Master node showed much higher memory usages as compared to other nodes. Slave-2 also showed higher than overall average memory usage. Slaves 1 and 3 showed very low memory usage in all of the experiments.

Figure A.5, showed the impact of increasing block size on execution time of the job for 30 runs of the experiments. As we doubled the block size from 128MB to 256MB, there was an increase in execution time as well. But as we further increased the block size to 512MB, the execution time lowered a bit and then increased again for block size 1024MB.

Table 4.1: Statistical significance t-test of impact of increasing block size

Packets_in	128MB	256MB	512MB
256MB	=		
512MB	=	=	
1024MB	=	=	=
Packets_out			
256MB	=		
512MB	=	=	
1024MB	=	=	=
CPU usage			
256MB	=		
512MB	-	+	
1024MB	+	-	+
Disk usage			
256MB	-		
512MB	+	+	
1024MB	+	+	+
Memory usage			
256MB	-		
512MB	-	+	
1024MB	-	-	+
Execution time			
256MB	=		
512MB	=	=	
1024MB	+	=	=

Summary We evaluated the impact of HDFS block size by considering network and system metrics. Increasing block size from 128MB to 256MB, directly impacted the overall execution time while inversely impacted all other system and network parameters. Increasing the block size to 256Mb resulted in increase of packets_in, packets_out and disk usage and decrease in CPU usage, memory usage and execution time. Taking the block size further to 1024MB directly impacted packets_in, packets_out, CPU usage, disk usage and execution time.

When we analyzed change in block size from 256MB to 512MB, it positively impacted packets_in, packets_out and disk usage only. While the change from 256MB to 1024MB, packets_in, packets_out, CPU, disk, memory usages and execution time were directly impacted. For the change from 512MB to 1024MB, there was an increase in all system metrics along with execution time. Table 4.1 showed the results from student t-test, the change in CPU usage, disk usage and memory usage were significant as compared to default 128MB except 128MB to 256MB. For execution time, only the change from 128MB to 1024MB was significant.

4.4.2 Impact of replication factor

In this particular setup we analyzed the impact of changing HDFS replication factor on packet_in, packet_out, CPU usage percentage, disk usage percentage, virtual memory usage and execution time. We used two different replications factors i.e., 2 and 3. The block size was kept 256MB for both set the experiments and each experiment was repeated 30 times. Number of mappers and reducers were set to default value, which were 5 and 1 respectively. The Hive-based query was used to run experiment which was, *Select * from table_name order by column_1, column_2, column_3;*

Figure A.6, showed the impact of increasing replication factor from 2 to 3. There was a definite increase in overall mean packets transfer for replication factor 3. Individual nodes showed mean close to the overall mean. For factor-2, Master node and slave-3 has showed higher than overall mean while for factor-3 only slave-1 showed much higher mean packets transfer as that of other nodes.

Figure A.7 showed the packets_out data of the two experiments. Unlike packets_in, packets_out of factor-3 showed much higher overall mean as compared to factor-2. Master node in both of the experiments showed much higher mean that overall mean. Salve-1 showed ± 500 mean as compared to overall mean in both of the experiments. Slave-2 showed similar or lower mean packets transfer than overall while slave-3 showed lower mean in both of the experiments.

Both CPU usage and virtual memory usages are shown in the Figure A.8

and Figure A.9 respectively. Both of these showed similar patterns. Master and slave-2 nodes had higher than overall means in both of the experiments. Slave-1 and slave-3 had much lower mean usage as compared to overall means. For CPU usage , overall means were similar while for memory usage the there was a slight decrease for factor-3.

Figure A.10, showed the disk percentage usage for both of the experiments. There was a decease in overall mean for factor-3. The master node also showed this pattern and showed a great deal of decrease in disk usage for factor-3. All of the slave nodes performed much less than overall mean disk usage.

The execution times of the two experiments were shown in the Figure 4.3. AS we increased the replication factor from 2 to 3, there was a prominent decrease in overall time.

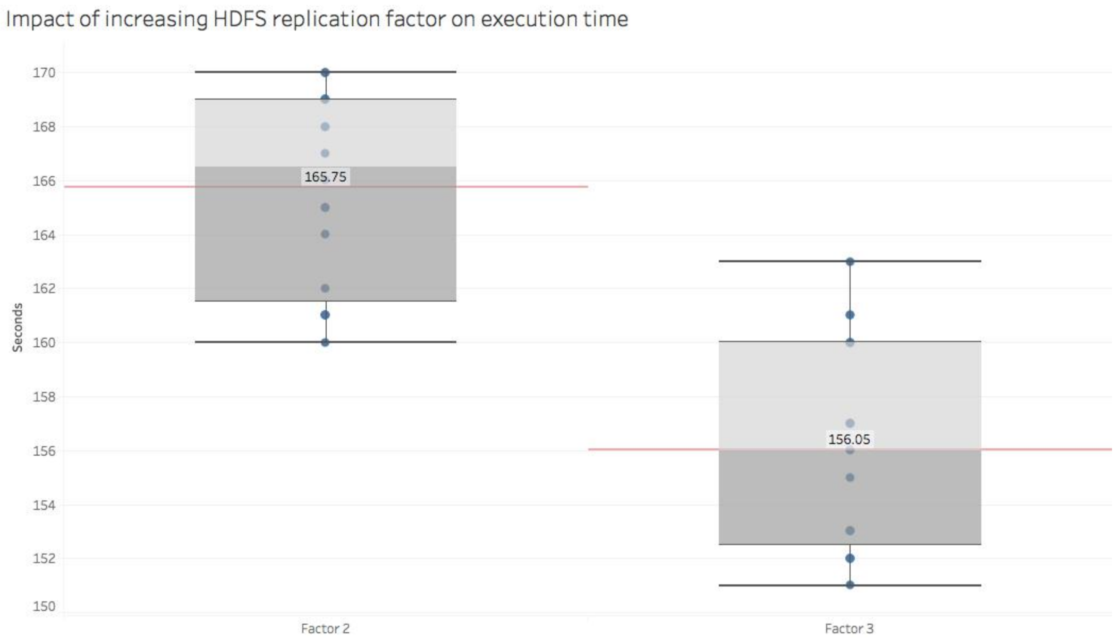

Figure 4.3: Box plot and average execution for 30 runs for different replication factors

Summary In this particular experiment, we observed the impact of changing HDFS replication factor on packets_in, packets_out, CPU usage, disk usage, memory usage and execution time. The statistical significance t-test as shown

Table 4.2: Statistical significance of the impact of replication factor using t-test

	Packets_in	**Packets_out**	**CPU usage**	**Disk usage**	**Memory usage**	**Execution time**
	Factor-2	Factor-2	Factor-2	Factor-2	Factor-2	Factor-2
Factor-3	+	+	+	-	-	-

in Table 4.2, the change was significant for all of the metrics. The change in replication factor increased the packets_in, packets_out and CPU usage, while decreased the disk usage, memory usage and execution time.

4.4.3 Impact of mappers

In this particular experiment, we analyzed the impact of changing number of mappers in HDFS on packet_in, packet_out, CPU usage percentage, disk usage percentage, virtual memory usage and execution time. We used four different number of map tasks i.e., 5, 10, 15, 20 mappers. The block size was kept 256MB and replication factor was 2 for all experiments and each experiment was repeated 30 times. Number of reducers were set to default value. The Hive-based query was used to run experiment which was, *Select * from table_name order by column_1, column_2, column_3;*

Figure A.11, showed the impact of increasing the number of mappers on packets_in. For default settings, 5-mappers were utilized in each run. The overall mean of the average number of packets transferred for the default number of mappers was the least among all set of experiments. This overall mean increased as we doubled the number of mappers to 10 but then slightly lowered as the increase became three to four fold of the original number. Here, the job run and dataset used were held constant between all experimental runs, so there was very little chance of any other factor contributing towards this variability.

When we analyze individual node packet_in data across each of the experiments, we found that Master node received highest amount of data for default number of mappers which gradually decreased as the number of mappers increased. Also, the master node mean was close to the overall average for the 10,15 and 20-mappers. Slave-1 seemed to receive lower number of packets that overall in all of the experiments except when 15-mappers used, where it was slightly above the overall average. Slave-2 and slave-3 were variable in all of the experiments. For default, both received less average data than overall average. But increasing the number of mappers resulted in a higher average packet_in transferred than overall mean for either of them.

Figure A.12, showed similar patterns for the overall mean of average across all experiments. The default number of mappers was associated with the lowest number of data transferred out of each data node. The overall mean of average packets_out data increased as the number of mappers increased from five to ten and then it slightly decreased as the number of mappers increased to 15 and 20. The master node showed the highest number of average packets_out in all of the experiments and it was consistently higher than the overall average as well. The reason for this difference might be the presence of YARN on Master node which allocated and scheduled the map tasks on slave nodes. We observed that slave-1 and slave-3 have transferred less data than the overall average packets_out data in all of the experiments. Slave-2 consistently showed greater or equal mean than overall mean in all experiments except in case of 15 mappers. The reason for that could be the fact that in this particular experiment slave-1 and slave-3 showed much higher packets_out than in other experiments.

The Impact of increasing the number of mappers on CPU usage percentage is shown in Figure A.13. The overall mean of each experiment increased slightly with the increase of number of mappers. The master node showed the highest mean CPU usage in all the experiment and it was almost double to that of overall mean. Another important factor was the overall lower mean CPU usage of slave nodes across all experiments except slave-2 which showed the mean CPU usage higher or equal than that of overall average CPU percentage usage.

The disk usage percentage is shown in Figure A.14. The results showed higher disk usage percentage on Master node because we are using Hive-based query which run on master node. Increasing number of mappers resulted in gradual increase in mean disk usage percentage of each of the nodes and also overall mean of experiments.

Figure A.15, showed the impact of increasing number of mappers on virtual memory usage. For individual nodes, the master node showed highest mean virtual memory usage in all of the experiments. Slave-2 showed almost the similar mean memory usage as that of overall average memory usage. Slave-1 and slave-3 showed the lowest mean memory usage in all of the experimental runs.

Figure 4.4, showed the overall average execution time for each of the experiment. Increasing number of mappers resulted in decreasing the mean execution time. Also less number of mappers resulted in mean execution times with greater standard deviations. The experiment with 20-mappers showed more consistent and least standard deviation among mean execution times.

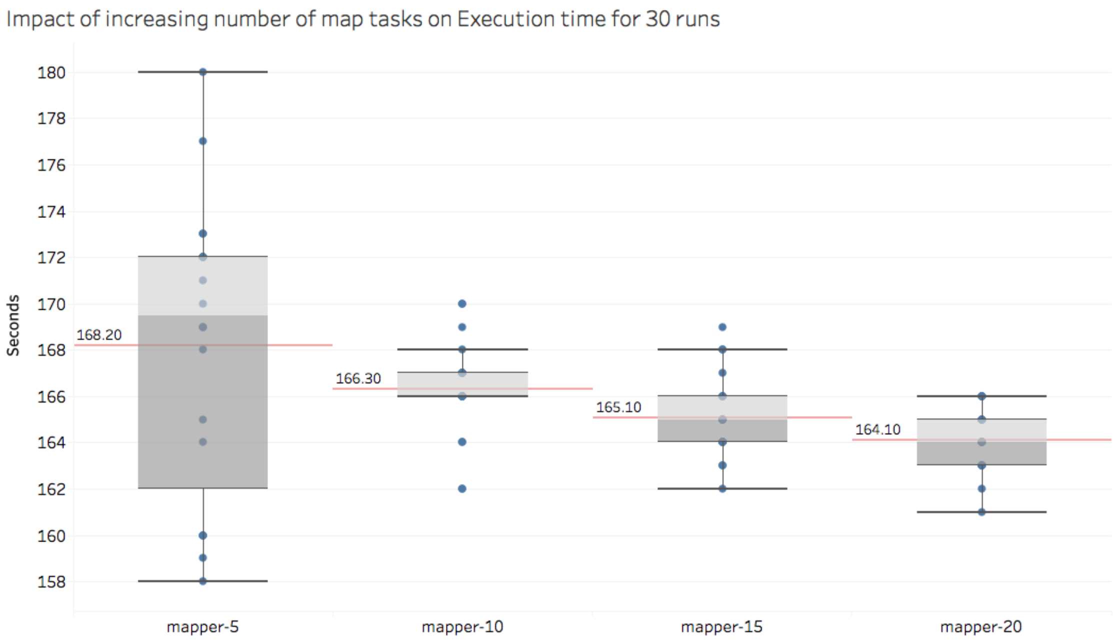

Figure 4.4: Box plot and average execution time for 30 runs for different number of mappers

Summary This experiment entailed the impact of explicitly defining number of mappers in our experiment and observe the change on our metrics. For all comparisons across the experiments involving 5, 10, 15 and 20 number of mappers, there was a gradual decrease in memory usage and execution time and gradual increase in CPU usage and disk usage.Packets_in and packets_out showed an increase first and then a slight decrease in case of 15 mappers, followed by increase again for 20 mappers.

Statistical t-test were used to analyze the significance of the changes to CPU

Table 4.3: Statistical significance of the results for the impact of number of mappers using t-test

Packets_in	5-mappers	10-mappers	15-mappers
10-mappers	+		
15-mappers	+	=	
20-mappers	+	=	=
Packets_out			
10-mappers	+		
15-mappers	+	=	
20-mappers	+	=	=
CPU usage			
10-mappers	+		
15-mappers	+	+	
20-mappers	+	+	+
Disk usage			
10-mappers	+		
15-mappers	+	+	
20-mappers	+	+	-
Memory usage			
10-mappers	-		
15-mappers	-	-	
20-mappers	-	-	+
Execution time			
10-mappers	-		
15-mappers	-	-	
20-mappers	-	-	-

usage, disk usage, memory usage, network usage and execution time as shown in Table 4.3. We found that changes in packets_in and packets_out when comparing 5 mappers with all other experimental runs, were significant. While for CPU usage, disk usage and memory usage changing the number of mappers always resulted in significant impact.

4.4.4 Impact of Query

In our experimental setup, the Hadoop job is executed when we input a Hive query. Hive query has SQL-like syntax and we can use `SELECT` statements to query the data. The queries could have different number of columns on `SELECT` clause, `ORDER BY` clause or `GROUP BY` clause. Therefore, due to the difference in syntax of query, the Hadoop jobs executed and affected the resource utilization and execution times differently. This section showed our experimental runs to highlight the impact of query on performance and resource utilization. At this stage, we only considered `SELECT` queries with `ORDER BY` clause and `GROUP BY` clause.

Number of columns in order by clause

In this particular experiment setup, we analyzed the impact of changing query structure on packet_in, packet_out, CPU usage percentage, disk usage percentage, virtual memory usage and execution time. The block size was kept as 256MB and the replication factor was set to 2 for all experiments and each experiment was repeated 30 times. The number of mappers and reducers were set to five. The Hive-based query was used to run each experiment. Three different queries that vary the number of columns in the `ORDER BY` clause were designed to see the impact of this parameter on our selected measures. The queries are:

1. *Select * from table_name order by column_1, column_2, column_3;*

2. *Select * from table_name order by column_1, column_2;*

3. *Select * from table_name order by column_1;*

Figure A.16, showed the impact of using different number of parameters in query on packets_in data. The results showed gradual increase in average packets transferred from query-1 to query-3. For individual nodes, the master node showed gradual decrease in data transfer into the system as we reduce the number of columns in `ORDER BY` clause from three to one. In all of the experiments, the master node showed higher mean than overall average except query-3 which had ordering by one column. For query-1, all of the slave nodes showed lower mean packets_in than overall average. For query-2, the all slaves except slave-3 showed lower mean than overall mean. For query-3, all slave nodes except slave-2 demonstrated higher mean than overall mean.

Packets_out results are shown in the Figure A.17. The overall average packets transferred out showed the same pattern as that of the packet_in. Master node showed greater mean than overall average in all of the experiments and there was gradual increase in individual mean from query-1 to query-3. Slave-1 and slave-3 also showed a gradual increase in mean from queries 1 to 3. slave-2 showed lower mean than overall mean in all of the experiments except query-1.

Figure 4.5 showed the CPU usage percentage for the impact of change of query structure. There was a slight decrease in overall average CPU usage percentage from query 1 to 3. As most of the processing and data storage takes place at Master node, it showed similar pattern as that of overall mean. Slave-1 and slave-3 has showed much lower CPU usages as compared to overall CPU usage and other nodes as well. Slave-2 has performed very close to overall mean in all of the experiments.

Disk usage percentage results were presented in Figure A.18. As we decreased

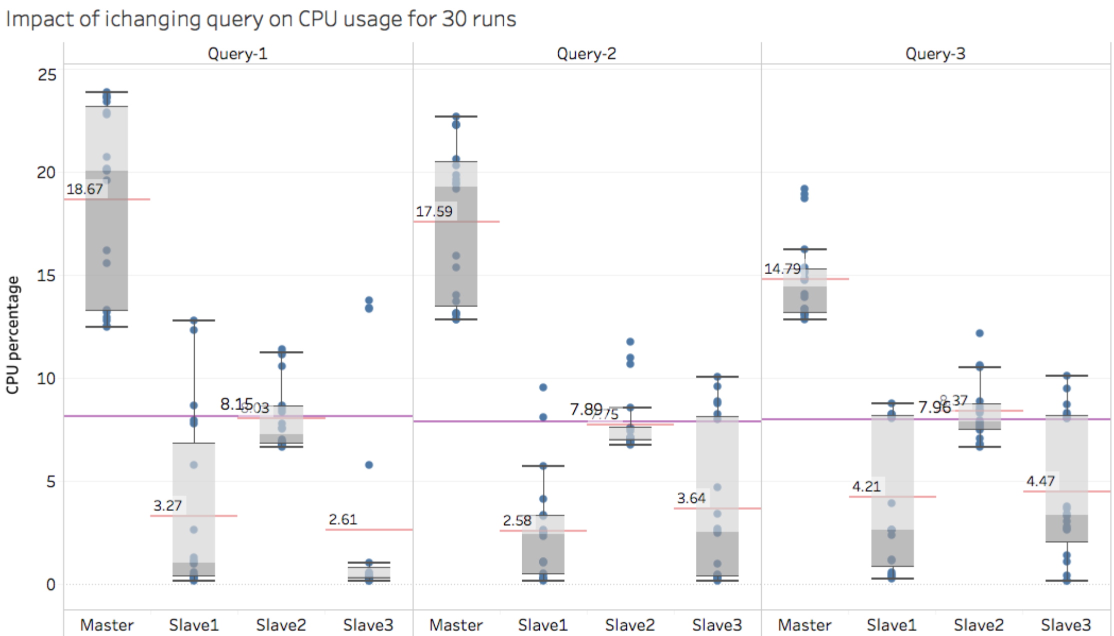

Figure 4.5: Box plot and average CPU usage percentage data for 30 runs for different queries

the number of order by columns from 1 to 3, there was a gradual increase in overall and Master node mean disk usage percentages. All of the slave nodes,

Table 4.4: Statistical significance of the number of columns in order by using t-test

Packets_in	Query-1	Query-2
Query-2	+	
Query-3	+	=
Packets_out		
Query-2	+	
Query-3	+	=
CPU usage		
Query-2	-	
Query-3	-	=
Disk usage		
Query-2	+	
Query-3	+	+
Memory usage		
Query-2	-	
Query-3	-	=
Execution Time		
Query-2	=	
Query-3	=	=

showed consistently low disk usages as compared to the overall mean disk usage.

Virtual memory usage were presented in the Figure A.19. The impact of changing the query structure showed that there is gradual and slight decrease in overall mean and mean memory usage of individual nodes. Slave-2 showed memory usage similar to that of the overall mean for all of the experiments.

Figure A.20, showed the execution time for all three set of experiments of query change. As we decreased the number of order by columns there was a decrease in execution time. The possible reason for this could be the decrease in complexity of query in `ORDER BY` clause. The reason that query-3 showed much lower execution times was that it has only one parameter in `ORDER BY` clause as compared to query-1 and query-2 which has 3 and 2 parameters in the clause.

Summary The results showed that reduction in number of columns order by columns, gradually increased the packets_in, packets_out, disk usage and CPU usage, while gradually decreased the memory usage and execution time. We used a statistical t-test (Table 4.4) to determine the significance of changes in system and network performance measures. We found significant differences in mean when comparing query-1 to other queries except execution time. But if we compare the query-2 and query-3, only the change in disk usage was significant.

Number of aggregate functions

In this particular experiment, we analyzed the impact of changing query structure on packet_in, packet_out, CPU usage percentage, disk usage percentage, virtual memory usage and execution time. The block size was kept 256MB and replication factor was 2 for all experiments and each experiment was repeated 30 times. Number of mappers and reducers were set to default value. The Hive-based query was used to run experiment. The three different queries used vary in the number of columns in `GROUP BY` clause and designed to explore the impact on our selected metrics. In addition to `GROUP BY` clause, Query-1 had one aggregate function, query-2 had two aggregate functions and query 3 has more than 2 aggregate functions to see the impact of query structure.

1. *Select function_1(column_1) from table_name group by column_1, column_2;*
2. *Select function_1(column_1),function_1(column_2) from table_name group by column_1, column_2;*
3. *Select function_1(column_1),function_2(column_2), function_3(column_1) from table_name group by column_1, column_2;*

Figure A.21 showed the impact number of the aggregate functions in query on packets_in data. The overall mean packets transfer decreased as we increased the number of aggregate functions from 1 to 3. Master node showed the highest mean activity in all of the experiments and it was higher than overall mean as well. All the individual slave nodes showed smaller means than the overall mean except slave-2 which showed higher than the overall mean for query-1.

Figure A.22 showed the results from packets_out data. For overall average, the results showed similar pattern as that of packets_in. For individual node's mean, the master node showed the lowest mean packets transferred out data in all of the experiments. All of the slave nodes showed higher than the overall mean packets_out transferred for all queries. Slave-1 showed the highest activity in query-1 and query-3.

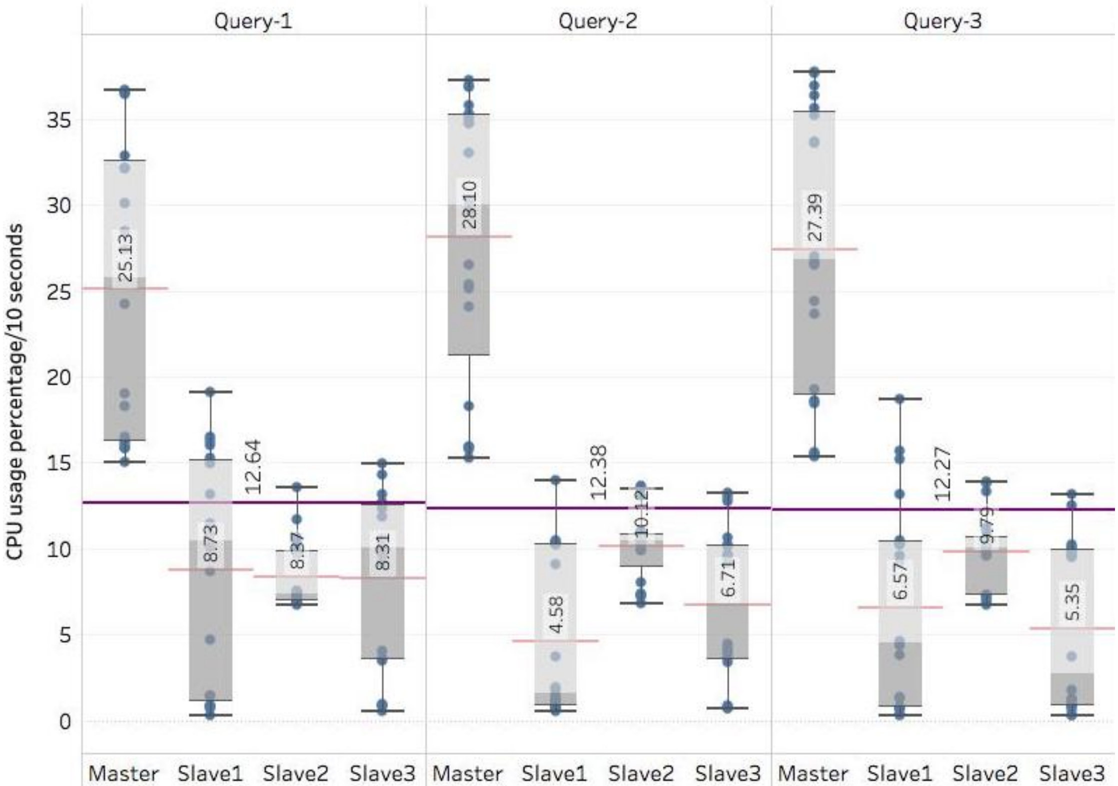

Figure 4.6: Box plot and average CPU usage percentage data for 30 runs for different aggregate functions in queries

The CPU usage percentage results were shown in Figure 4.6. The overall mean in all of the experiments remained similar approximately. The master node showed the highest CPU usage in all of the cases. The mean CPU usage of slave nodes was always less than the overall mean.

Disk usage percentages were shown in Figure A.23. Both overall and individual mean have similar values across all of the experiments. Disk usage utilization of the Master node was the highest and most consistent across all of

Table 4.5: Statistical significance of the number of aggregate functions by using t-test

Packets_in	Query-1	Query-2
Query-2	-	
Query-3	-	=
Packets_out		
Query-2	-	
Query-3	-	=
CPU usage		
Query-2	-	
Query-3	-	=
Disk usage		
Query-2	=	
Query-3	-	-
Memory usage		
Query-2	+	
Query-3	=	-
Execution Time		
Query-2	+	
Query-3	=	-

the experiments while the disk usages of slave nodes remained constant across each of the experiments. Figure A.24, showed the impact of varying the number of aggregate functions on virtual memory usage. For all of the queries, the Master node showed maximum memory usage with query-2 being the highest among all. Query-2, which had two aggregate functions, showed greater overall average as compared to query-1 and query-3. Slaves 1 and 3 showed the much lower mean memory usage as compared to the other nodes. Slave-2 had mean memory usage close to the overall mean.

Figure 4.7, showed the impact of aggregate functions on execution times. Query-1 with one aggregate function had lesser execution time as compared to query-2. For Query-3, which had more than five aggregate functions, the execution time dropped as compared to all other experiments.

Summary The results showed that increasing the number of aggregate functions in the query has slightly decreased packets_in, packets_out and CPU usage

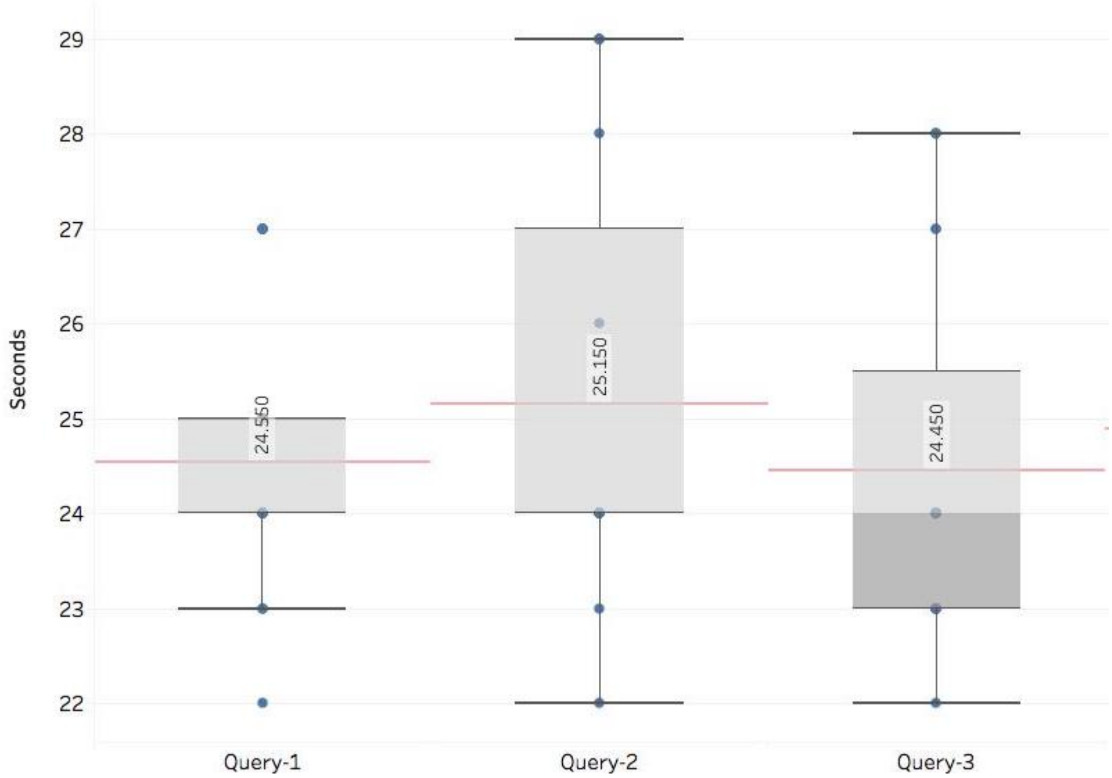

Figure 4.7: Box plot and average execution time for 30 runs for different aggregate functions in queries

from query 1 to query-3. As compared to all previous experiments, disk usage remained constantly high among all experiments except query-3. Virtual memory and execution time has slightly increased for query-2 and then dropped down again for query-3.

The statistical t-test showed that all changes from query-1 to query-2 are significant except disk usage as shown in Table 4.5. For query-1 to query-3, packets_in, packets_out CPU and disk usages were significant. While changes in execution time and CPU usage were significant when comparing query-1 to query-3, Another prominent results was the significant results for memory usage, disk usage and execution time when comparing query-2 with query-3 which showed that even a slightest change in query structure such as adding an aggregate function had impacted on system performance.

4.5 Chapter Summary

If we summarize the results reported in this chapter, there are many conclusions drawn from the current set of experiments which are as follows:

- Changing Hadoop configuration parameters like *Block size, replication and number of mappers* had an impact on system resources and overall execution time of job. Choosing suitable value of a parameter was important but we should always consider the inter-related affect between different parameters.

- Query the same dataset differently showed that it also impact the system resource utilization and execution time. So when we optimize a Hadoop job for a particular system or cluster of computers, we must consider not only the configuration parameters for HDFS, MapReduce but also how we are querying the data.

- The parameters were configured manually, one at a time. But to see the impact of these parameters on the system performance measures it was impossible to run experiments for all possible combinations of parameters by updating the configuration files manually. so, there should be an automatic procedure to study all possible combinations of parameters and query structures.

- As it was impossible to manually find combined effect of different features(configuration parameters and query structure), we have to adopt some statistical or mathematical modelling process like machine learning to find patterns and correlations between the parameter space and query structure.

Chapter 5

Modelling and prediction of resource utilization of Hadoop cluster using Machine Learning

5.1 Introduction

The purpose of this chapter is to investigate the use of machine learning approach to model and predict the resource utilization and execution time of Hadoop jobs. As we discussed in Chapter 4, resource utilization and execution time of Hadoop jobs are affected by different attributes such as configuration parameters and structure of query. These confounding attributes can take a range of values. So, in order to estimate or predict either qualitatively or quantitatively the correct level of resource utilization and execution time, it was important to understand the how different combinations of these Hadoop job attributes can affect overall performance and resources. It was very difficult to run such a large number of jobs with different combinations of Hadoop job attributes and then interpret the data manually. To manually extract patterns from the data and give a model that can generalize for an unseen scenario is a tedious and impossible task in such a a situation. In order to automate the process of data extraction and modeling the complex behavior of different attributes of Hadoop job machine learning can be used. Machine learning (ML) is a collection of statistical techniques that automate the process of data-driven modeling. This enabled us to program systematic discovery of significant patterns in data [205].

5.1.1 Chapter goals

The objective of this chapter is to investigate the following research questions.

1. **Whether machine learning approaches can be used to model the resource utilization of our cluster for different Hadoop configurations?and How?**

2. **Can we get a visualizable and easy-to-interpret machine learning-based model to predict resource utilization and execution time? and How?**

5.1.2 Chapter organization

Section 5.2 presents the brief introduction of commonly used ML approaches such as k-nearest neighbors (KNN), Naive bayes (NB), Random Forests (RF), Decision trees (DTs) and Support vector machine (SVM). Section 5.3 presents the details of our proposed machine learning-based method. Section 5.4 presents the experiment design including the experimental setup, dataset and preliminary experiments. Section 5.5 discusses the results from different experiments. Section 5.6 presents the overall summary of the chapter.

5.2 Machine learning approaches

This section gives a brief overview of different machine learning approaches we used in our initial experiments. These algorithms include decision tree, random forest, support vector machines, naive bayes and k-nearest neighbors method.

k-nearest neighbors (KNN) is an algorithm that can be used when all the values of attributes are continuous. It can also be modified to deal with categorical attributes. The method can be used to predict the class lable of unseen instances using the closest instance or instances [206]. Random Forests (RF) is an ensemble method of classification where multiple decision trees are being generated [207]. The term *random* refers to the building of multiple decision trees from each random sample of data and then selecting best trees. Support vector machine (SVM) is a classification algorithm and model. It is computationally expensive and mathematically complex [208]. For a given labeled training data, the algorithm categorizes the new examples on the basis of an optimal hyperplane [209].

Naive bayes (NB) is a classification algorithm which assumes that for a probability of a given classification the effect of the value of one attribute is independent of the values of the other attributes [206]. Decision trees (DTs) are the class of machine learning (ML) algorithms used for supervised regression and classification tasks. It is one of the most commonly used ML technique [119,210,211].

Every decision tree is composed of three types of nodes: *root* node, *intermediate* nodes and *leaf* nodes. An input variable is associated with each node. More than one edge is coming out of each node that represents the possible values that node can take. A leaf node always represents the value based on the values given from the input variable in the path running from root node to leaf node. DTs always starts at root node and end on a leaf node. The tree does not converge at any point, it always splits its way out as the nodes are processed [212].

Some of the advantages of the decision tree algorithm are:

- It gives interpretable model representations like like some binary tree graph [119].

- It can handle numerical, categorical and binary variables, so no data preprocessing needed in most cases [210].

- It performs well with reasonable amount of computing, so they show scalable computation [211].

- It has the ability to deal with irrelevant inputs [212].

- They are quite tolerant when it comes to dealing with missing value [213].

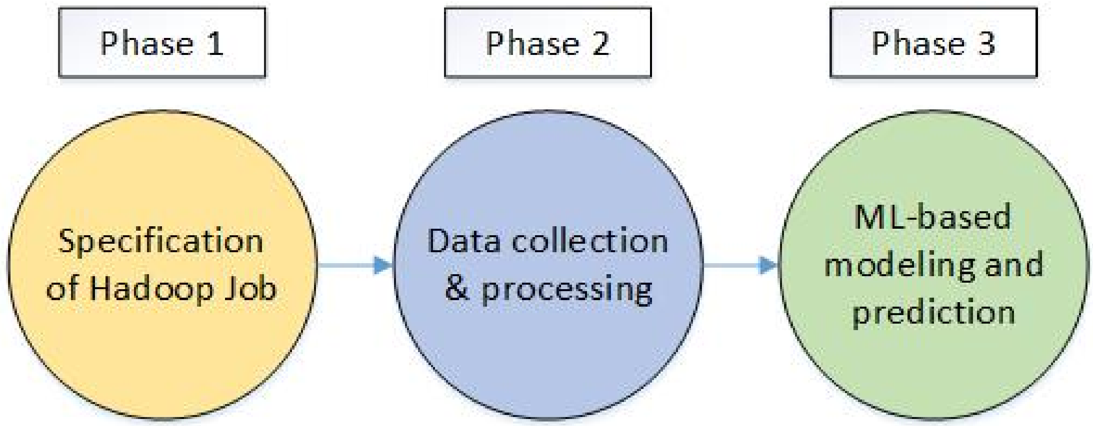

Figure 5.1: Different phases of our method.

5.3 Methodology

Our method for modeling and predicting resource utilization of a Hadoop cluster is described in this section. The method consist of three phases and each phase was divided into sub-phases as shown in Figure 5.1.

5.3.1 Phase 1: specification of Hadoop jobs

Phase 1.1: This phase involved the *selection of dataset* to run Hadoop job. The experiment included running Hive-Hadoop jobs using six different query structures on the Berkeley Earth Surface Temperature dataset.

Phase 1.2: The next step was *specifying the query* to decide the structure of query like how many columns to select and which clauses to be included. We used Hive-based query to run our experiments. Hive is a data warehouse software that facilitates SQL-based querying of large datasets residing in HDFS storage [214]. Our query have varying number of clauses in `Select` clause, `Order By` clause and `Group By` clause as shown below.

```
SELECT * FROM table_name ORDER BY col_1,col_2,col_3;
SELECT * FROM table_name ORDER BY col,col_2;
SELECT * FROM table_name ORDER BY col_1;
SELECT aggregate_function_1 FROM table_name
```

```
        GROUP BY column_1,column_2;
SELECT aggregate_function_1,aggregate_function_2
        FROM table_name GROUP BY column_1,column_2;
SELECT aggregate_function_1,aggregate_function_2, aggregate_function_3
        FROM table_name GROUP BY column_1;
```

Phase 1.3: Next, we *selected the Hadoop parameters* to vary, such as like replication, block size, and number of mappers, and decided the values of these parameters to be used in our experiments.

Phase 1.4: Lastly, we decided upon the *performance metric* to measure like network traffic, CPU usage, disk usage, memory usage and execution time, and *number of iterations* to run for each combination of the parameter settings. Each experiment was repeated 30 times.

5.3.2 Phase 2: data collection and processing

Phase 2.1: This phase involved automatic *collection* of performance measures during each run of Hadoop job. The Master node was connected to slave nodes via an OpenFlow switch. The associated Software-defined Networking (SDN) controllers were Faucet (operating as a learning switch) and Gauge (Monitoring port statistics). InfluxDB is a database optimized for storing large number of small measurements such as packets_in and Grafana is a monitoring dashboard used to display InfluxDB measures.

During each run of the experiment, we collected packets_in, packets_out, CPU usage, disk usage and memory usage with an interval of 10 seconds for entire duration of Hadoop job.

For packets_in and packets_out data, we used InfluxDB to fetch data for ports connected to cluster nodes. For CPU usage, disk usage and memory usage data, we used python scripts to monitor these system statistics at each of the node.

Phase 2.2: Here, we processed the collected data. We calculated the mean resource usage for each run of the experiment,e.g., memory usage and execution time. We created datasets, one for each of our performance measures, with the following feature: block size, number of mappers, number of replicas, percentage of columns selected, number of columns in order by clause, number of aggregate functions and number of columns in group by clause. While including structure of query into our feature space, we take into account the percentage of columns selected rather than actual number of columns. This was because different datasets have different numbers of columns and stating only the number of columns does not tell us the complexity of the dataset itself.

$$Column\_percentage = \frac{No.\ of\ columns\ selected}{Total\ number\ of\ columns} \tag{5.1}$$

Phase 2.3: Then we performed *median-based discretization* of the data. Discretization is a process of converting continuous attribute values into a finite set of intervals with minimal loss of information [215]. The median measure was used instead of the mean because a single very high or very low value can disturb the mid point of the data. If a value was lower than median resource usage, the observation was labeled as low usage, otherwise high usage.

5.3.3 Phase 3: ML-based modeling and prediction

The third phase was the usage of machine learning to model the dataset created in the previous phase. The model was built by running six different queries over the earth surface temperature dataset. We used a decision tree [216] (J48) algorithm from WEKA [217] and performed 10-fold cross-validation. We chose to use a DTs because of the simplicity of the model itself and because it allowed us to automate the process of discovering important relationships between different features of Hadoop jobs. DTs also had another advantage, decision trees built interpretable models which can handle complex interactions, such as the relationship between various parameters and query structures. For each of our performance measures, one tree was generated. We extracted decision rules from our decision trees for each performance measure. Each rule was extracted by traversing the tree from leaves to root node. For example, in Figure 5.2, if we traverse back from leaf nodes indicating high usage just after number of replicas, the rule will be CPU usage was high if the percentage of columns selected > 0.5, number of mappers > 15 and replica > 2. Each leaf node had two numbers associated with it, for example, Low (2160.0/1.0). The first number 2160.0 represent the number of observations used for training end up at this path and the second number 1.0 represent the number of observations incorrectly classified, i.e., full under the other class.

5.4 Experiment Design

This section describes our experimental setup and datasets used in our experiments. Figure5.3 shows the experimental design of our system. When a job was submitted to the cluster, all the communication took place through the switch. As the switch had Faucet (OpenFlow controller) [204] and we could monitor the flow of traffic passing through different ports of the switch. Gauge was used to pull statistics from switch and store them in InfluxDB at a definite interval. From InfluxDB [218] we visualized the *packets_in* and *packets_out* data by using a visualization dashboard named Grafana [219].

5.4.1 Experimental setup

A cluster comprising *four* nodes was built to run the experiments in this study. The cluster was only used for our experiments, there was no concurrent use of the cluster. Each node had Intel(R) Core(TM) i7-7700 CPU@3.6GHz, 500GB disk space, 8GB RAM running Ubuntu 16.04 LTS. We configured the cluster for Hadoop version 2.9.0 and Hive version 2.3.3 on the Master node. Both Faucet and Gauge were the version 1.6.8 and installed from source [204].

5.4.2 Dataset

The Berkeley Earth Surface Temperature Study combined 1.6 billion temperature reports from 16 pre-existing archives. The dataset had six attributes and

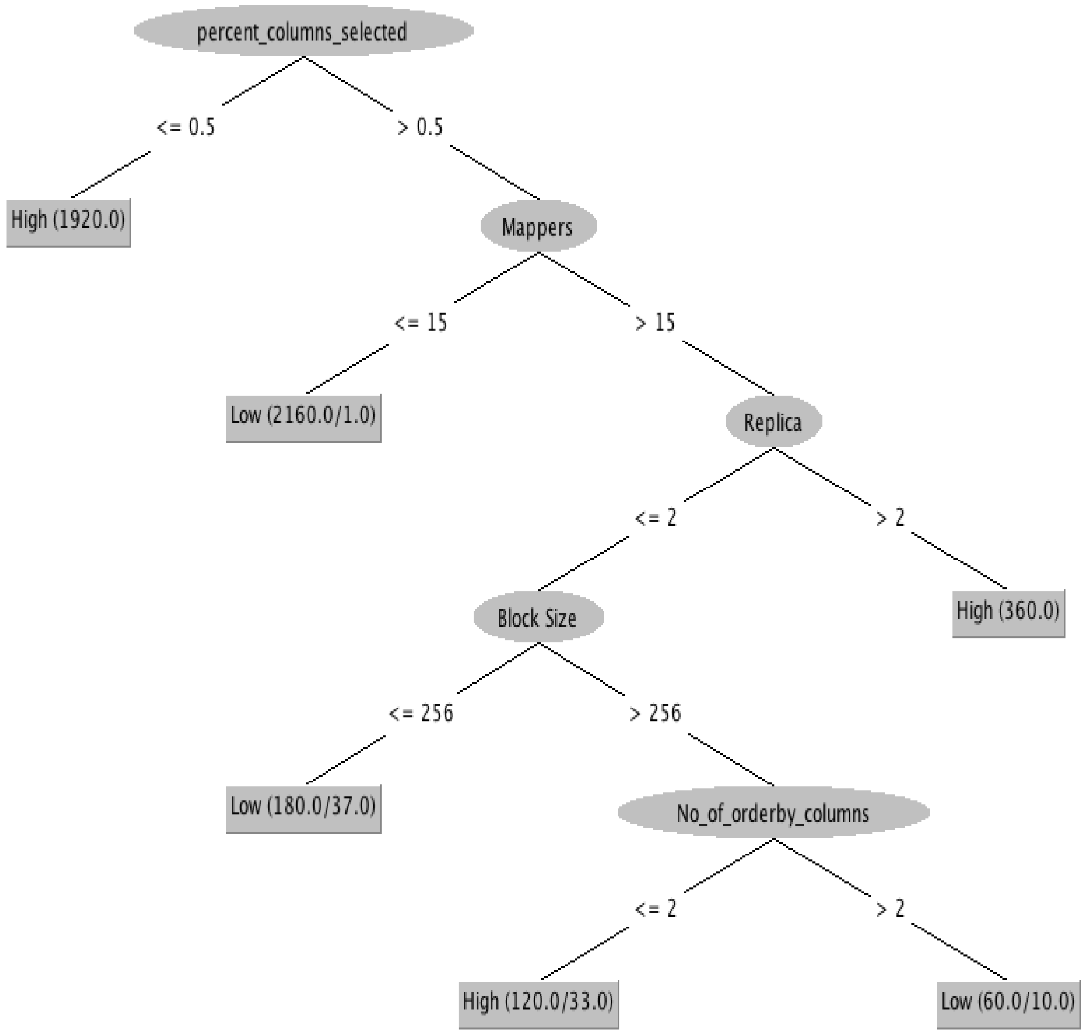

Figure 5.2: Decision tree for CPU usage.

almost 10 million records. The size of the file was 650 MB [1].

5.4.3 Preliminary experiment

This section gave the overview of the preliminary experiment that we setup to understand how we can apply machine learning approaches over existing

[1]available at `https://data.opendatasoft.com/explore/dataset/`

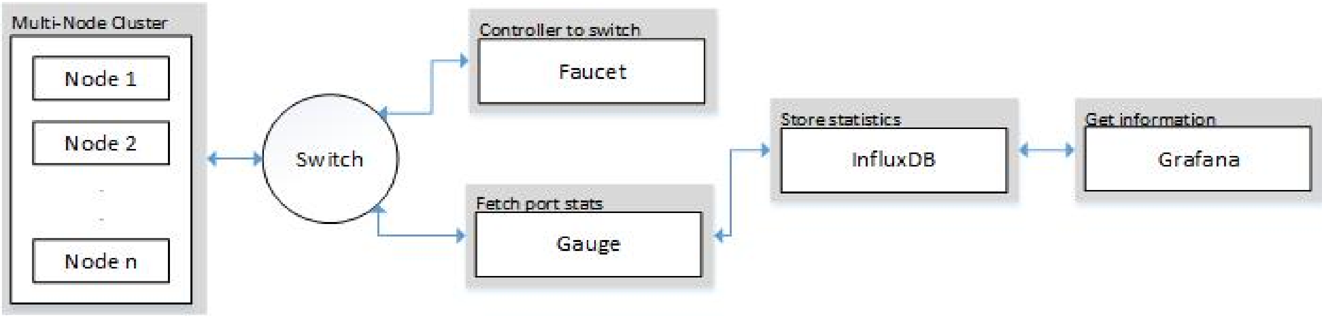

Figure 5.3: Experimental Setup.

dataset. The *dataset* here was the performance measure that was recorded in previous experiment. The details of experimental setup is given in Table 5.1.

In our experimental method, we kept decision tree as baseline method. The classification accuracies of other methods were compared to our baseline method to see whether the results are significantly different or not.

Table 5.1: Experimental settings for the classification of different Hadoop block sizes.

Input:	
Block Size	128, 256, 512, 1024
Replication	2
Number of mappers	5
Query	Q2
Output	Packets_in
Experiments repetition	30
Task	Classification
Algorithms	Naive Bayes, KNN, SVM, DT,Random Forest
Method	10-fold cross Validation
Measurement	Accuracy
Statistical test	Two-tailed T-TEST

5.5 Results and discussions

This section present and describe the results from our experiments. Apart from using decision tree models we have used descriptive statistics such as mean and boxplot to get further insight into the results. The feature appearing at the root node is the most important feature in the node as it appeared in all of the decision rules and correspond to the best predictor.

We built boxplots to analyze the underlying pattern. Apart from showing different elements of boxplots we also plotted mean at different levels of the plot. Figure 5.4 shows the meaning of different elements of the Figure . Mean have been calculated on each level to distinguish the relationship between each scenario, block of settings and overall mean.

5.5.1 Preliminary results

This section provides the results of the preliminary experiment as outlined in Section 5.4.3. As we can see in our results that both decision tree and random forest method showed very good results. The advantages of decision trees outlined in Section 5.2 we decided to use it in further experiments to model and predict the resource utilization of Hadoop cluster which is part of our proposed methodology outlined in Section 5.3.

Table 5.2: Classification of Hadoop Block size using machine learning algorithms

Dataset	**Decision Tree**	**Naive Bayes**	**SVM**	**KNN**	**Random Forest**
CPU usage	100±0.0	97.42±4.38	97.5±3.84	95.67 ±5.86^{+}	99.83 ±1.17
Disk usage	100±0.0	100±0.0	100±0.0	100 ±0.0	100 ±0.0
Packets_in	98.92±3.48	94.17±6.86^{+}	95 ±5.68^{+}	97.58±4.15	98.33 ±3.55
Packets_out	93.67±7.22	67.75±15.38^{+}	73.17±13.59^{+}	95.08 ±5.93	96.92±5.12
Memory usage	99.75±2.50	88.33 ±9.33^{+}	89.42 ±8.45^{+}	96.75 ±5.54	99.75 ±1.86

5.5.2 CPU usage

For *CPU usage*, the most important feature contributing towards the model was the *percentage of columns selected*. number of mappers (M) and Replication factor or number of replica(R) in order by clause were the two other important feature as shown in the Figure 5.2.

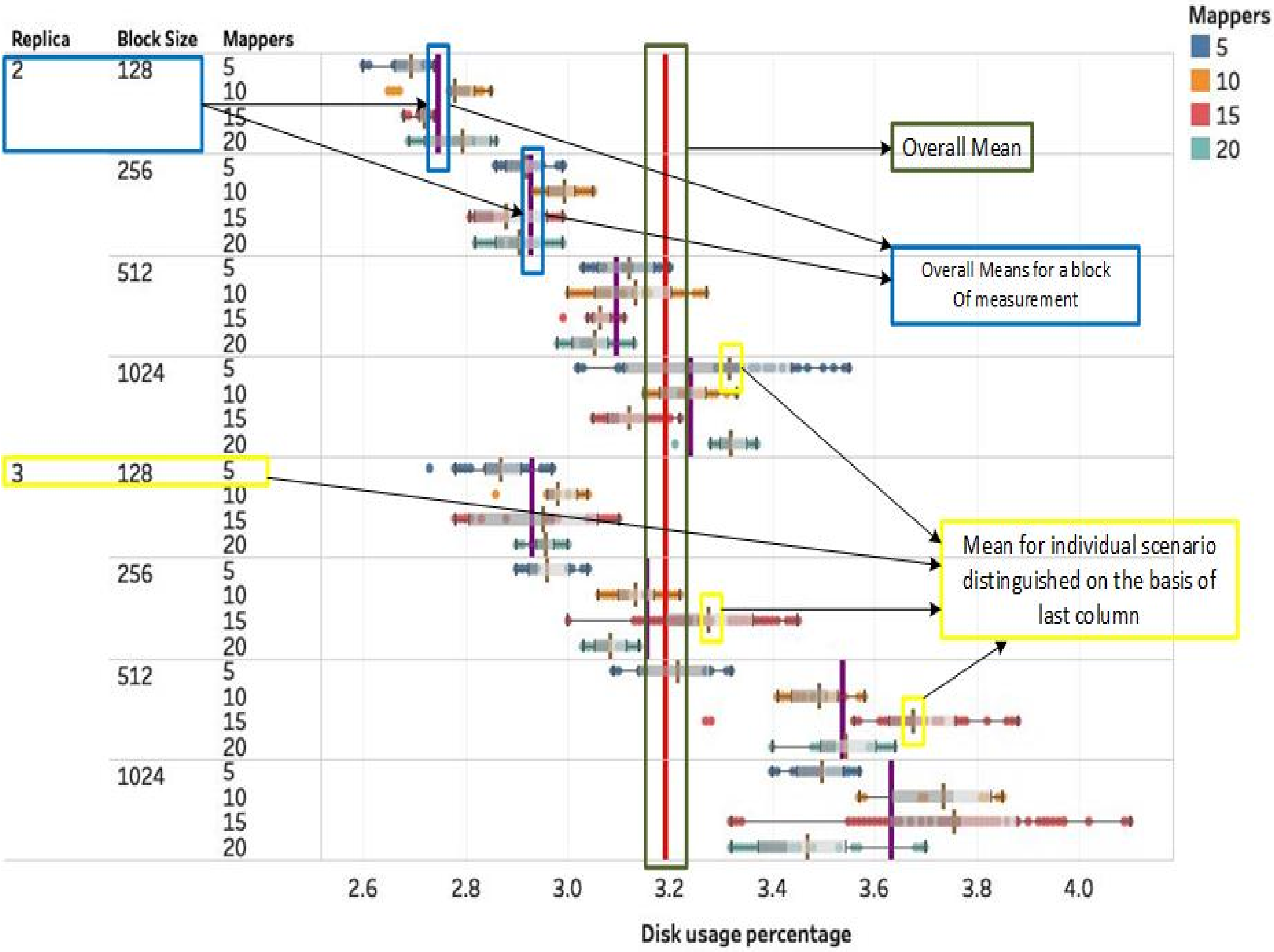

Figure 5.4: Description of different elements of the Figure .

Figure 5.5 showed the detailed analysis of impact of different Hadoop parameter configurations on CPU usage. The red ref line in the Figure showed the overall mean and the small purple lines showed the mean of a certain block of features. The results showed that number of mappers followed a certain pattern along with number of replicas and block size. As the number of mappers was small, the replica mean was smaller than the overall mean of all experiments.

The local mean gradually increased as the number of mappers increased from 5 to 20. The number of replica CPU usage mean became greater than the overall mean when the number of mappers becomes greater than 10. There was a small variation between the local means for each of the replicas. This was also demonstrated in our decision tree at level two, where the tree predicted that when the number of mappers will be greater than 15 and number of replica is greater than 2, the CPU usage will be high.

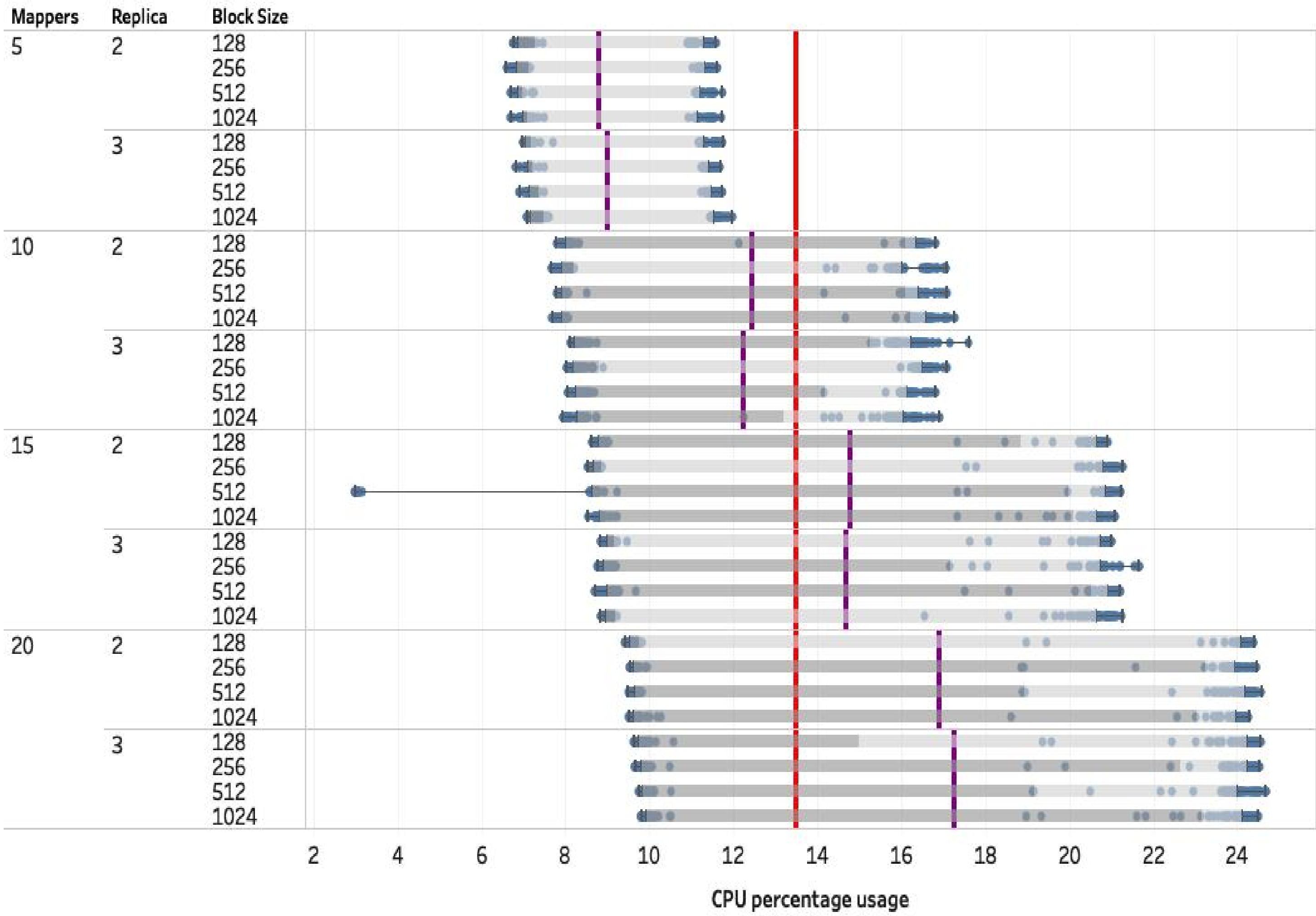

Figure 5.5: Boxplot of different values of block size, number of replica and number of mappers for CPU usage.

Our decision tree results also showed the importance of percentage of columns selected as the root node of the tree and it also predicted that if it was less than 50% the CPU usage will be high. Figure B.9 showed the impact of different elements of query structure over CPU usage. The Figure showed that percentage of columns selected (N) had a very distinguishing impact over CPU usage. The results showed that smaller percentage of columns selected gave higher CPU usage. If the percentage of columns selected remained less than 50 percent, the CPU utilization remained greater than the overall CPU usage. On the other hand if N was greater than 50% the CPU utilization was less than the overall mean CPU usage. Another important factor was the presence of number of columns in group by clause. Higher number of group by clause contributed towards higher CPU usage.

Figure B.4 showed the impact of percentage of columns selected (N), number of mappers and block size over CPU usage. The small value of N, as discussed before, resulted in higher CPU usage. However, number of mappers could also impact the CPU usage and when it was combined with N. The results suggested that smaller number of mappers could result in small mean CPU usage. Our decision tree model also predicted that lower number of mappers could result in lower CPU usage. If $N < 50\%$ and number of mappers greater than 5, CPU usage was higher than overall mean CPU usage. There were a great number of outliers in the boxplots. But when $N > 50$ % the CPU usage always remained less than overall mean CPU usage. Also the difference between local means was very small and there were also very less number of outliers in the chart as shown in Figure B.5.

Figure B.6 showed the interaction between different query structures, block size and CPU usage. The results suggest that there was not much difference in the pattern of CPU usage as we have seen in Figure B.9. As shown in Figure B.8, when we compared the relationship between number of replicas, different query structures and CPU usage the results showed that it did not add any distinguishing pattern in our chart. The decision tree prediction also showed that block size and number of columns in order by clause add very little to the CPU resource utilization.

When we analyzed the impact of different query structures and number of mappers on CPU utilization the results showed different pattern as shown in

Figure B.7. As we can see the CPU usage increased as number of mappers increased from 5 to 20. The analysis also suggested that the presence of order by clause in the query showed compact CPU usage values while group by clause gives very dispersed CPU usage values with high standard deviations.

Rules extracted from decision trees suggested that high CPU usage will result if:

- Number of columns selected (N) ≤ 3
- number of replica (R) > 2, Number of Mappers (M) > 15 and N > 3

Some rules suggesting CPU usage lower than the median CPU usage are:

- M ≤ 15 and N > 3
- B ≤ 256MB, R ≤ 2, M > 15 and N > 3

5.5.3 Disk usage

For *disk usage* model, block size (B) was the most important feature. Other important features were the number of replicas, percentage of columns selected (N), the number of columns in order by clause or the number of order by columns (O) and the number of mappers as shown in Figure B.10.

The decision tree model suggested that larger the value of both block size and number of replica higher the mean disk usage. Figure B.11 show the impact of Hadoop configuration parameters on disk usage. Block size and number of replicas had the most distinguishing effect among all. Local mean disk usage gradually increased as the Block size increased. Local mean disk usage for the Block size when the number of replica was *three* , was higher than the local mean disk usage for the Block size when the number of replica was *two*. Another important behavior was that the higher value of number of replicas and block size always resulted in local mean usage higher than the overall mean usage, for example when the number of replica was equal to *three* and block size was greater than 512 then the mean CPU usage was extremely high. Also, in each of these cases when the number of mappers is equal to 15, the standard deviation and mean disk usage was much higher than the other cases.

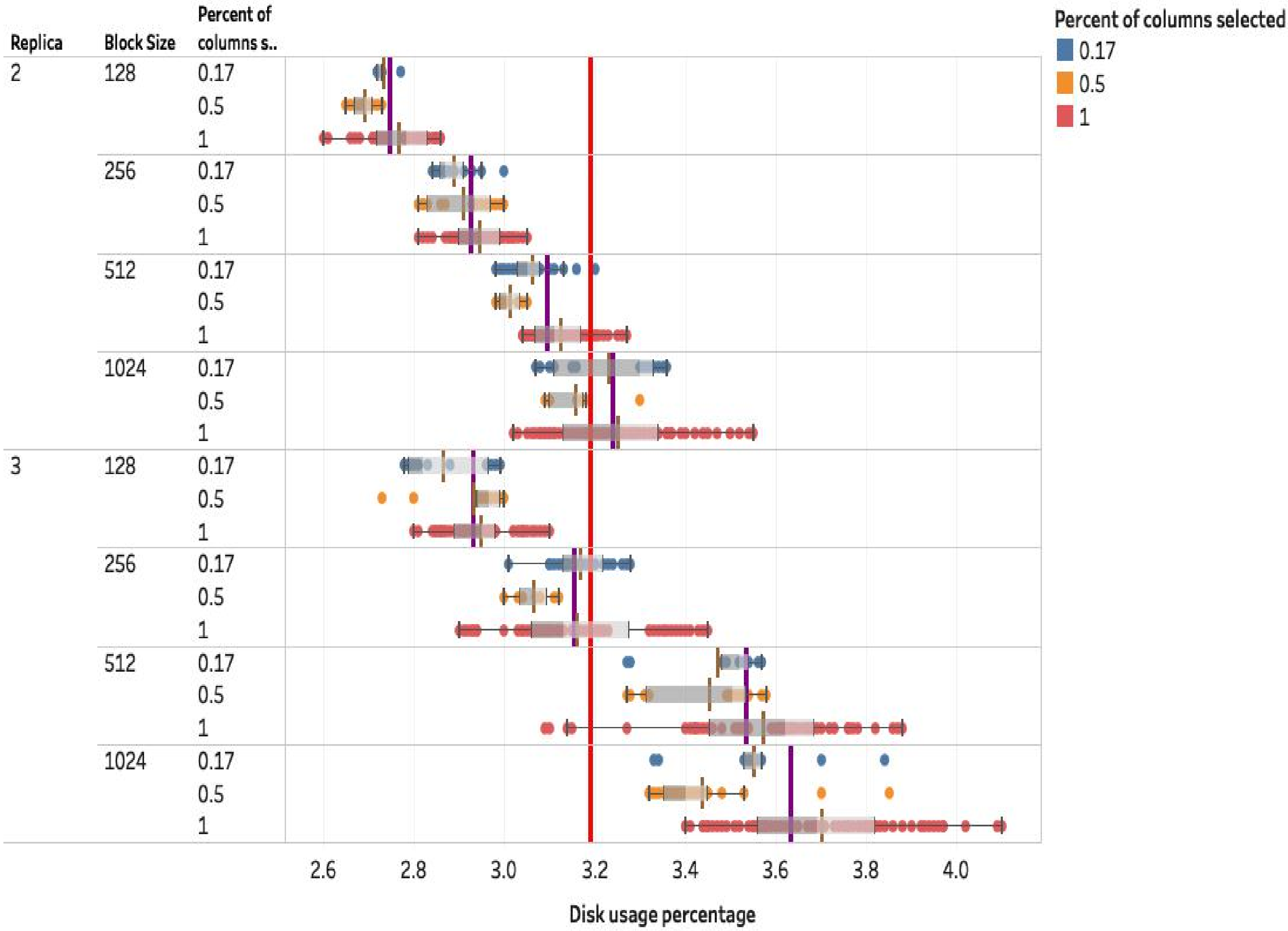

Figure 5.6: Boxplot of number of replicas, block size and percentage of columns selected for disk usage.

Another prediction of our model was the high disk usage for high percentage of columns which increased as the block size and number of replica value increased. Figure 5.6 showed the impact of number of replicas, block size and percentage of columns selected. The results showed that it follows the similar pattern as that of shown in Figure B.11. However, when we selected 100 percent of the columns the disk usage was not only higher than the local mean disk usage but also the standard deviation was much higher than other cases of N.

Figure B.15 show the behavior of disk usage for different query structures used in our experiments. The results suggested that when percentage of columns selected is less than or equal to 50% and a group by clause is present, the mean disk usage will always be less than the overall mean disk usage. The results also showed that mean disk usage is higher than the overall mean disk usage when the number of columns in order by clause was small. Our decision tree model also suggested that lower percentage of columns selected can lead to lower disk usage.

Figure B.14 show the behavior of number of replicas and query structure for disk usage. As discussed previously, number of replicas showed distinguishing patterns in our disk usage results. We can see that mean disk usage was always less than overall mean disk usage when the number of replicas was equal to *two*. It was also observed that not only means, medians but also the standard deviations were high when the number of replicas was equal to *three* and query has order by clause. For higher number of replicas value the disk usage increases up to a certain number of order by clause columns, for example in our results it is *two*.

Figure B.12 show the impact of query structure and block size over disk usage. The pattern for query structure was similar to of that discussed previously. Adding block size to the analysis has resulted in further break down of the distinguishing factors which was evident in our decision tree model as well as the block size appears to be the root node. Increasing the Block size gradually from 128MB to 1024MB showed gradual increase of individual block mean disk usage as highlighted in the Figure with brown lines. If the block size is greater than 512, the mean disk usage was greater than the local mean disk usage. Another important observation was the higher standard deviation for the block sizes 512 and 1024.

Figure B.13 demonstrated the patterns when query structure and number of mappers were taken into account. When the percentage of columns selected was 100% and the number of columns in order by clause is high, graphs show a large number of outliers resulting in lower median of disk usage for the number of mappers greater than or equal to 20.

For high disk usage, some of the decision tree rules were;

- B >512 and R ≤ 2
- B ≤ 128,R > 2 and M $> 10 \leq 15$

.

For low disk usage,

- B $<= 128$, R > 2 M < 10
- B > 512,R$<= 2$, M $<= 5$ O<=2

5.5.4 Memory usage

The number of mappers was the most important feature in the model built on *memory usage* measure. Other important features included block size and percentage of columns selected as shown in Figure B.1.

Our model predicted that greater number of mappers results in higher memory usage. Number of mappers were the initial phase of MapReduce framework and increasing the number of mappers results in more memory to be occupied. This behavior of configuration parameters and their impact on memory usage was also shown in Figure 5.7. Block size and the number of mappers seemed to have a distinguishing impact on memory usage. Mean memory usage increased gradually as the number of mappers increased. Mostly if the number of mappers were greater than or equal to 15 the mean and median of the memory usage remained greater than the overall mean memory usage. Another important observations was when the block size became greater than 256, the individual mean memory usage for smaller number of replicas always remained less than mean memory usage of mappers. For Block size of 1024, the boxplot showed less number of outliers as compared to other block sizes. Figure B.21 show the impact of different elements of query structure over memory usage. The percentage of columns selected is the most distinguishable feature here. The less percentage of columns selected and group by clause resulted in mean and median of memory usage to be higher than the overall mean memory usage as predicted by our decision tree model as well. The results also

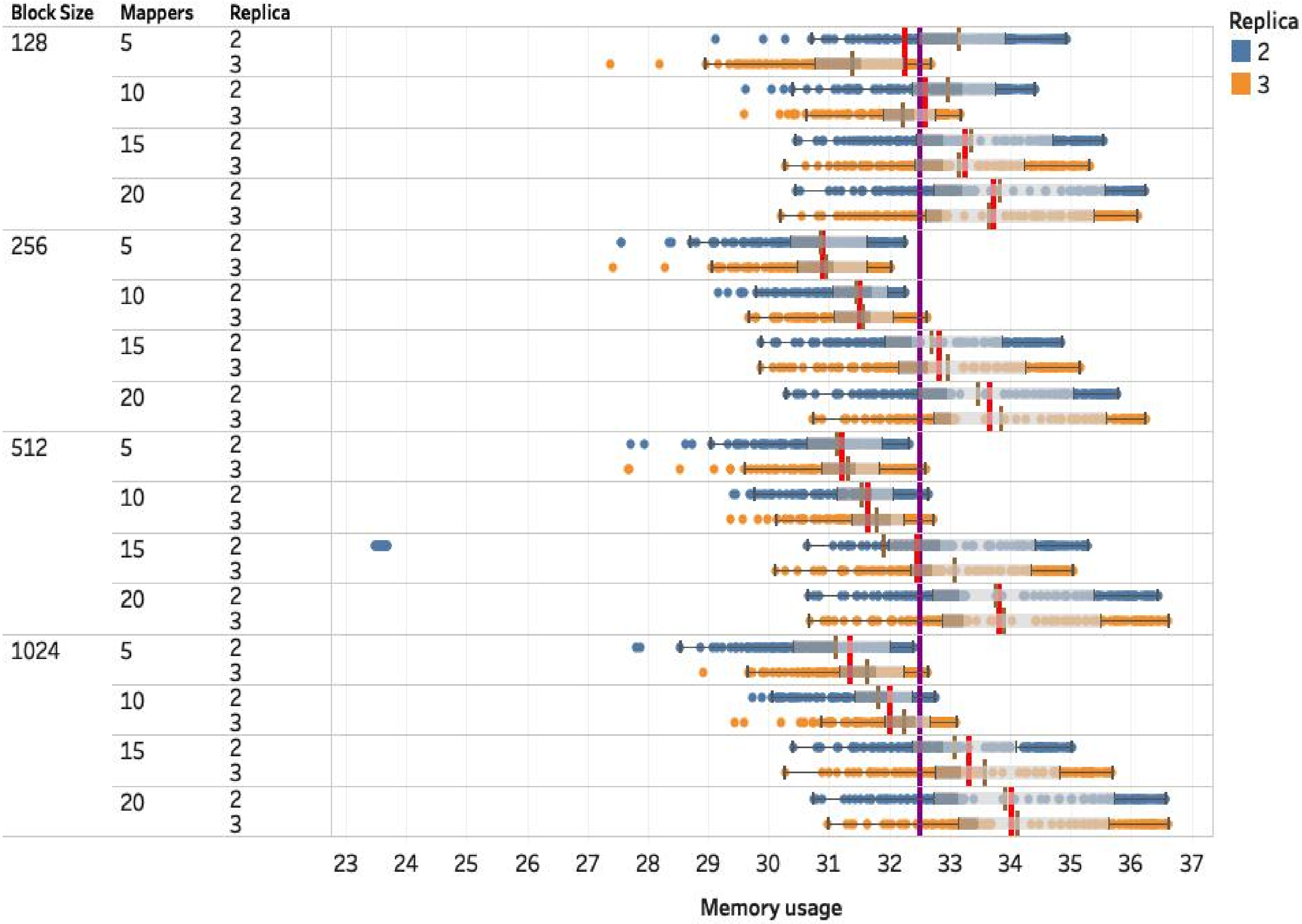

Figure 5.7: Impact of Hadoop configuration parameters on mean memory usage.

suggested that presence of order by clause and selecting all the columns in the data demonstrates lower mean and median of memory usage, as compared to overall memory usage. Another observation was the presence of large number of outliers for the boxplots showing order by clause.

We added number of replicas to understand the underlying pattern as shown in Figure B.20. The boxplot graph showed that for columns selected less than 50% both of the replicas showed similar memory usage. But when selected all of the columns, the number of replicas with value three showed smaller standard deviation in memory usage.

Higher number of mappers and less percentage of columns selected was predicted to result high memory usage in our decision tree model. However, when we included the number of mappers and the query structure together to see the behavior of memory usage, the pattern was quite different as shown in Figure B.19. There was a gradual increase in the mean memory usage of the system as we increased the number of mappers from 5 to 10. The difference between individual means was high when the percentage of columns selected was less than 50% and there was *group by* clause present in the query structure. The pattern was same in the query structure with order by clause but the difference between means was much smaller. Mostly, the median of the memory usage was less than or close to overall mean of memory usage when there was *order by* clause present.

Figure B.18 show the impact of query structure and block size over memory utilization of the system. The presence of *group by* clause and percentage of columns less than 50% demonstrated the memory usage with the greater standard deviations. The block size breakdown over the query structure showed that memory usage decreases as the block size approaches 256. However, the mean memory usage gradually increased as the block size increased from 256 to 1024. The only exception here is due to the presence of outliers in the boxplot which impacts the mean when N was 100% and O was 1 and block size was 512.

We also analysed the behavior of percentage of columns selected, number of mappers and block size over memory usage as shown in Figure B.16. The boxplot show that there was a cascading pattern followed by the increase of number of mappers for N $\leq$ 50%. The mean and median memory usage, for number of mappers greater than 10, was higher than the overall memory usage of all experiments. Also, the difference between these local means was quite significant. Moreover, when the percentage of columns selected is 100% both mean and median memory usage of mappers was less than or equal to the overall mean memory usage.

Figure B.17 show the interaction between percentage of columns selected, replicas and number of mappers. Mean memory usage for number of mappers gradually increased with increasing number of mappers. When the percentage of columns selected is less than 50% and number of mappers greater than 10, the mean and median memory usage remained higher than the overall mean memory usage.

Some of the decision rules for high memory usage were:

- M > 10 and N ≤ 3
- B > 256 MB, R ≤ 2, M $> 10 \leq 15$, N > 3 and O > 1

For low memory usage some of the decision rules were;

- B ≤ 128 MB, R > 2 and M ≤ 5
- B $> 128 \leq 256$, R ≤ 2, M $> 10 \leq 15$ N > 3 and O > 1

5.5.5 Network usage

For network usage, percentage of columns selected and number of replicas were the two important features. Other important features were number of mappers, number of orderby columns and block size as shown in Figure B.2.

Our model predicted that lower percentage of columns selected can result in lower network traffic. Figure 5.8 show the impact of different elements of query structure over network usage. Firstly, the presence of group by clause and percentage of columns selected less than or equal to 50% can result in mean network usage less than the overall mean network usage. Secondly, the presence of group by clause resulted in a boxplot with a large number of outliers contributing toward high means. On the other hand, the presence of order by clause resulted in higher network usage. The mean network usage slightly decreased as the number of columns in the order by clause increased. Also, there were almost no outliers in the boxplots for order by clause.

The interaction of configuration parameters with network usage is shown in Figure B.24. Firstly, number of mappers show a gradual increase in mean network usage up to 15. The mean network usage dropped down as the number

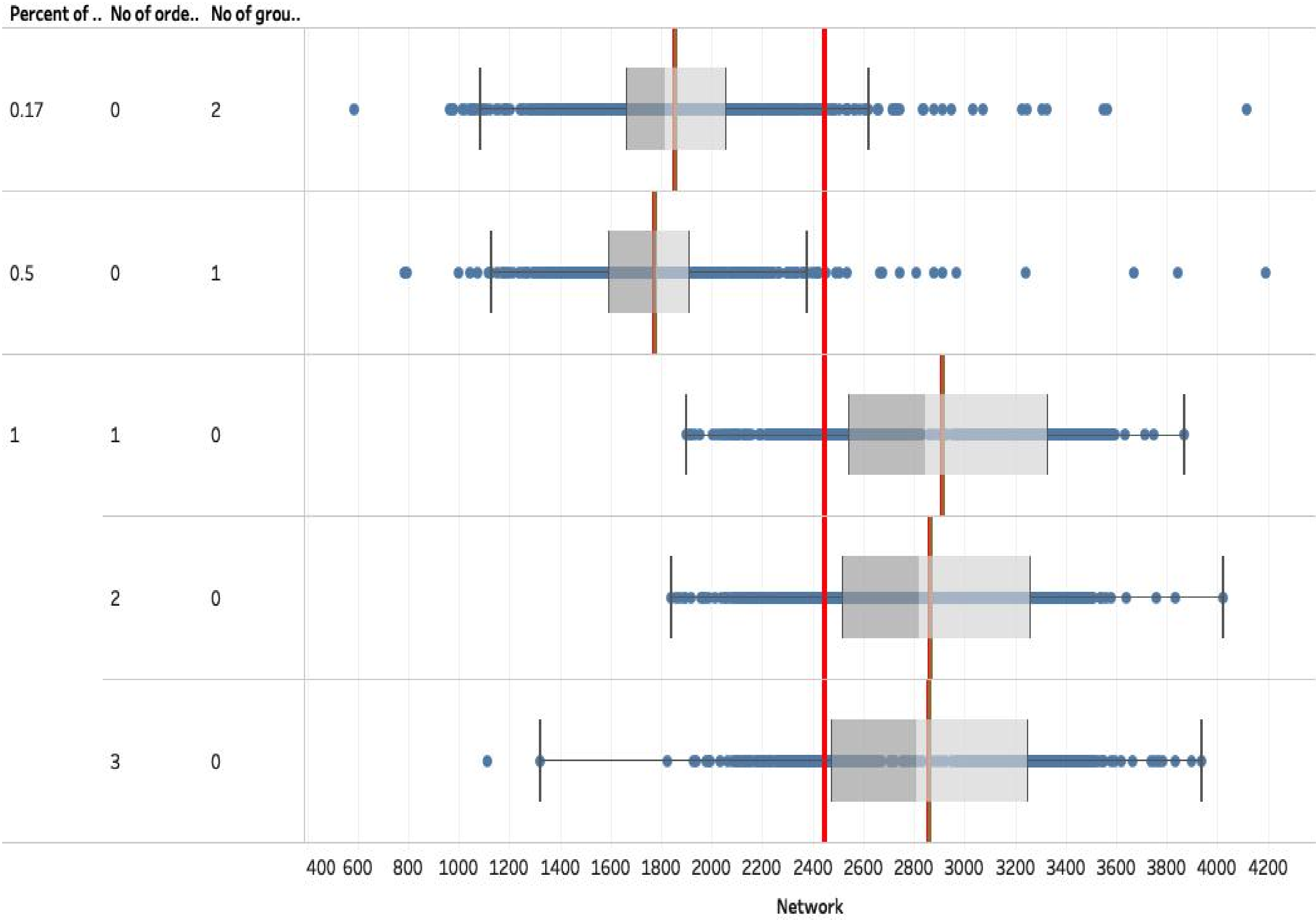

Figure 5.8: Boxplot of different elements of query structure for network usage

of mappers became greater than 15. Secondly, the number of replicas show less mean than the overall mean network usage when the value was *two* and greater mean when the number of replica was *three*.

Figure B.23 show the network usage behavior when number of replicas, number of mappers and number of columns in order by clause (O) are taken into account. The mean network usage, when O was equal to zero or when group by clause was present, was always less than the overall mean network usage. On the other hand the, the mean network usage for lower number of order by clause columns was high most of times and this gradually decreased as the order by columns increased to *three*. Another interesting observation was the impact of number of mappers, where mean usage increased gradually as the number of mappers increased to 15. For replicas equal to two, the mean network utilization was always less than the overall mean network usage.

Another interesting observation was the behavior demonstrated when we compared percentage of columns selected, number of replicas and number of mappers as shown in Figure B.22. The mean network usage for number of replicas was less than the overall mean network usage when the columns selected was less than or equal to 50%. For all columns selected, the mean network usage was higher than mean network usage where number of replicas with value three showed the highest mean usage.

Decision rules predicted that network usage will be higher if:

- N > 3, R > 2, M > 10 and O > 2
- B $> 128 \leq 256$ MB, R ≤ 2, M > 10, N > 3 and O $>\leq 2$

Lower network usage was predicted if:

- B ≤ 256, R ≤ 2, N > 3, M ≤ 5 and O > 1
- B$>$ 128 MB, R ≤ 2, N > 3, M $> 5 \leq 10$ and O > 2

5.5.6 Execution time

Percentage of columns selected was the most important feature in execution time model as shown in Figure B.3. Number of mappers and number of *order by* columns were the other important features.

Figure B.27 showed the impact of configuration parameters on execution time for queries with group by clause. Number of mappers seemed to have a very distinguishing impact over execution time. The mean execution time was less than the overall execution time for the number of mappers less than *ten*. As the number of mappers gone beyond 10, the mean execution time became greater than overall mean execution time. Another important observation was that for number of mappers less than or equal to 10, lower number of replica showed mean time less than the block size mean time. This trend become reverse as the number of mappers becomes greater than 10 until 20. Similar patterns were demonstrated when we analyzed the results from order by clause as shown in Figure B.28. The key difference between the two queries has been that for order by clause the mean for number of replicas was much closer to the overall mean.

Figure B.34 showed the impact of query structure over execution time. The purple line represent the overall mean while green line represents the overall median of execution time of all experiments. The mean execution time for queries with group by clause over the same data was much less than the mean execution time of the queries with order by clause. One common observation was that increasing columns in order by clause can result in higher time. This was very much evident in the Figure B.32 where we added number of replicas into the analysis to further distinguish the underlying patterns. The mean time for three order by columns, when all of the data was selected, was always greater than the overall mean execution time.

Figure B.31 showed the behavior of order by clause and number of mappers over execution time. There was a gradual decrease in execution time as the number of mappers increases from 5 to 10. But as the number of mappers increased from 10 to 20, the mean time increased gradually. Similar patterns were evident in the queries with group by clause as shown in Figure B.33. Figure B.29 and Figure B.30 showed the impact of query structure and block size over execution time. Block size did not seem to be very discriminatory in this scenario. All of the block sizes show similar mean execution time as that of shown in Figure B.34.

Lower execution times were predicted by the following decision rules:

- B $> 256 \leq 512$ MB, R ≤ 2, M $> 10 \leq 15$ N > 3 and O ≤ 1.
- N ≤ 3

Higher execution times were predicted by the following rules:

- M ≤ 10 and N > 3 or M > 15 and N > 3

- M $> 10 \leq 15$, N > 3 and O > 1

5.6 Summary

From the results above we can summarize the following conclusions:

- Percentage of columns selected, number of mappers and number of replicas has greatest impact over utilization of different resources in our Hadoop cluster. Choosing different values could result in high or low resource usage. Our decision tree models of different resources showed that apart from different values of Hadoop parameters, it was also important that how we query the data. In case of disk usage, block size has the greatest impact and it should also be given importance if we have limited storage resources while designing clusters and submitting huge number of jobs.

- Our proposed methodology was very effective in building models which can predict the utilization of resources and execution time. It was due to the robustness of decision tree model which could take data in variety of formats and also handles the missing values very well. The decision tree models were easy to visualize and also showed very good generalizability while taking into account large and diverse attributes.

- In order to see the prediction ability or generalizability of our decision tree model we needed to run further experiment to see what happens when we change the dataset.

- Also, until now we only used the cluster with fixed size, It would also be worth observing the generalizability or prediction ability of our decision tree model for environmental changes like changing the cluster size.

Chapter 6

Modelling and prediction of resource utilization for different datasets, cluster sizes and infrastructure

6.1 Introduction

Chapter 4 showed that different attributes of Hadoop jobs affected the overall resource utilization and execution time of Hadoop jobs. Our decision tree model (Chapter 5) showed how different attributes were linked to over and under utilization of Hadoop resources. The decision tree based modeling was constructed using the data from different experiments. These experiments used the same dataset, same cluster configuration and same infrastructure. It was important to investigate that whether our decision tree models were robust enough so that they could predict the resource utilization and execution time when we change the cluster size or dataset, etc.

6.1.1 Chapter goals

The objective of this chapter is to investigate the following research questions.

1. **What impact the change in dataset, cluster size and infrastructure has on Hadoop job performance and resource utilization?**

2. **Whether cluster job performance and resource utilization models, built previously, can generalize for the changes in cluster size, datasets an infrastructure?**

6.1.2 Chapter organization

Section 6.2 details the methodology proposed to run our experiments. Section 6.3 represent the experimental setups and datasets used to run our experiments. Section 6.4.1 discussed the results of the experiments run to highlight the impact of using different datasets on resource utilization and execution time of Hadoop jobs. Section 6.4.2 represents the importance and affect of changing cluster size for our performance measures. It also highlights the robustness of our previously built decision tree model and how different attributes are correlated with one another. Section 6.4.3 show that running the same experiments on different infrastructure can greatly impact the overall performance of Hadoop jobs.

6.2 Methodology

The method to perform our experiments is presented in this section. Our method consists of *four* phases.

Phase 1: This phase involved updating the Hadoop job environment. We designed three set of experiments to further our investigation.

1. *Experiment 1 :* involved using different datasets to run on the same Hadoop cluster. We executed same query on each of the dataset.

2. *Experiment 2 :* involved running the same query after updating cluster size. Previously, we used *four* cluster, but here we updated the cluster size by adding or removing one node from the existing cluster. Thus five node and three node Hadoop clusters were configured for this experiment.

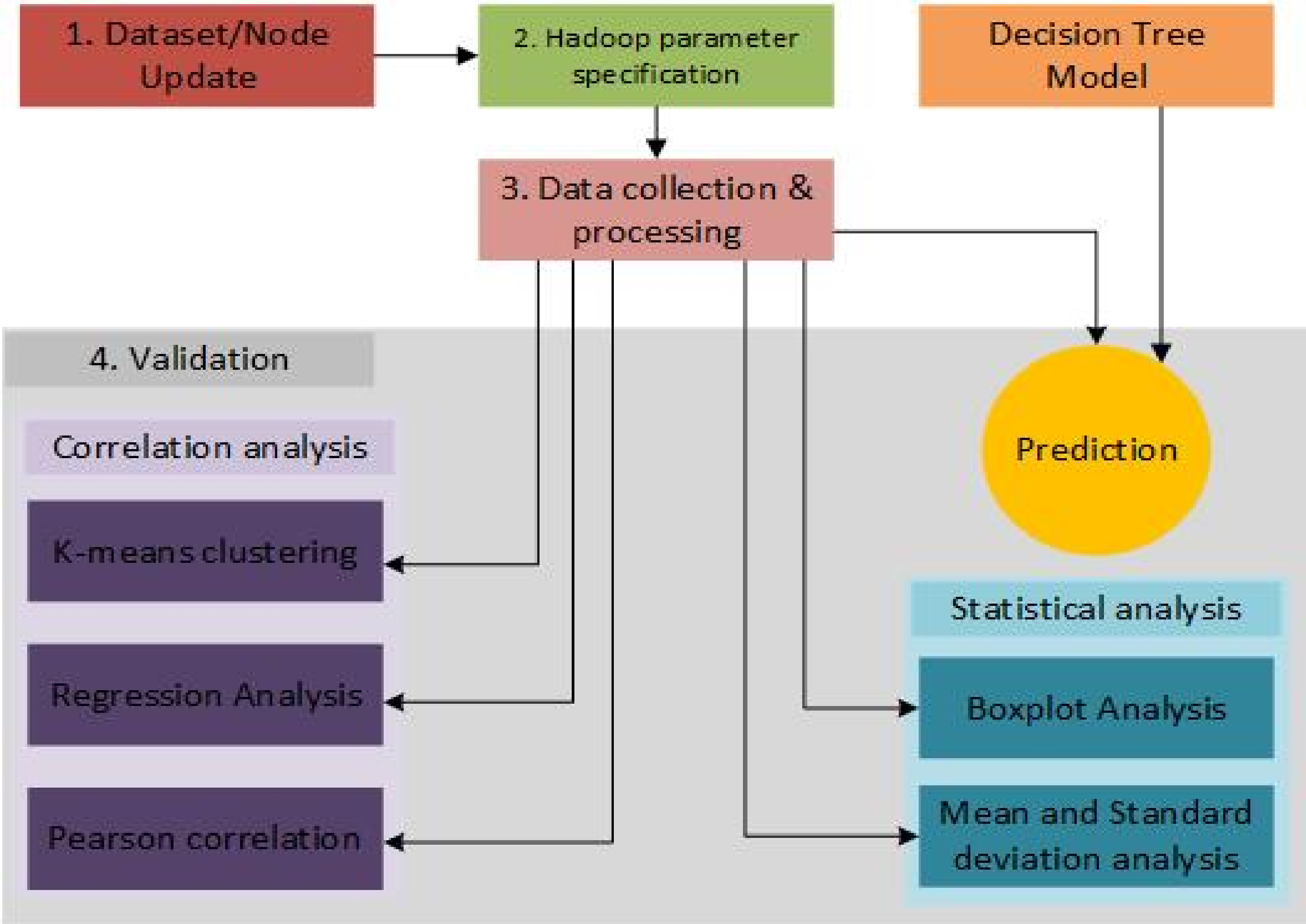

Figure 6.1: Different phases of our method.

3. *Experiment 3 :* involved the running the same experiments on a completely different infrastructure. For this purpose we used Amazon Elastic Compute Cloud (EC2) [220] to design Hadoop cluster.

Phase 2: This phase involved *selection of query*. For *experiment 1*, we used query 1. For *experiment 2 and 3*, we used both query 1 and query 2.

```
Query 1: SELECT * FROM table_name
         ORDER BY column_1,column_2,column_3;
Query 2: SELECT * FROM table_name
```

```
ORDER BY column_1,column_2;
```

Phase 2.2: For all three experiment, this phase was same as that of phase 1.3 in 5.3. **Phase 2.4:** For experiment 1 and 2, this phase was same as that of phase 1.4 in 5.3. But for experiment 3, we only monitored overall execution time of a job.

Phase 3: All of the phases and sub-phases were same as that of phase 2 in 5.3.

Phase 4: This phase involved the validation of results by using different approaches to answer our research questions.

Phase 4.1: Prediction The decision rules were extracted from decision tree model built in phase 3 of 5.3. The decision rules were used to predict the outcome of both of our experiments for all of our performance measures such as CPU usage, disk usage, memory usage, network usage and overall execution time. Firstly, we calculated the percentage of rules that correctly predicted the outcome of both of our experiments. This was done by calculating the number of rules correctly predicting the outcome. This showed the prediction or generalizability of the constructed decision tree models for unseen data.

Secondly, we calculated the percentage of data/instances that were true positives for a given decision rule. In order to do that we simply filtered the number of instances in each dataset that satisfy the conditions of that rule. The true positives were calculated by using the criteria; true positive = total number of positives - false negative. This gave the generalizability and prediction ability of an individual rule for unseen data.

Phase 4.2: Statistical analysis This phase involved analysing the results of different performance measures by applying statistical approaches. We applied *descriptive statistics* like mean and standard deviations. For experiment 1, we compared the overall and individual experimental settings to see the underlying pattern. For experiment 2, we compared the overall and individual settings mean and standard deviations for different cluster sizes. We used the boxplot [221,222] analysis to get further insight into our results.

Phase 4.3: Correlation analysis This phase involved finding relationship or association between different attributes for a particular performance measure of a job or experiment, for example, Pearson correlation [223]. Another important objective of this phase was to investigate the attributes of Hadoop job exhibiting similar effects.

Cluster analysis was performed to group data into groups (clusters) in a way that the data in the same group are more similar to each other than other groups [224]. We used k-means clustering algorithm to do this, where $k = 2$. Our clustering algorithm was based on Lloyd's algorithm with squared Euclidean distances to compute the k-means clustering for each k [225]. This value was chosen to categorize our attributes into two broader categories, such as low and high, as we did in our decision tree model. This was an automated process of finding the set of experimental settings which are correlated [221]. The Calinski-Harabasz criterion (CHC) [226] was used to assess the quality of cluster. The greater the value of this ratio, the more cohesive the clusters (low within-cluster variance) and the more distinct/separate the individual clusters (high between-cluster variance). It was defined as

$$CHC = \frac{SS_B}{SS_W} \times \frac{N-k}{k-1} \tag{5.1}$$

where SS_B is the overall between-cluster variance, SS_W is the overall within-cluster variance , k is the number of cluster, and N the number of observations.

The output of the clustering was analyzed by between-group sum of square, within-group sum of squares, number of observations in each cluster. Analysis of variance (ANOVA) [227] was performed on clustering results to see variations between and within observations that was partitioned into clusters. The ANOVA was computed for each variable (datasets or cluster size here) to see which of them were most effective for distinguishing clusters. We mainly computed *F-statistics* and *p-value*. The larger the F-statistics value, the better the corresponding variable was in distinguishing between the two clusters and the lower the p-value was the more significant difference was present between the elements of the two clusters.

Regression or linear regression model was used to predict a well defined and functional relationship between one dependent and one or more independent variables. M5P tree [217] was a classification and regression tree (CART) based on M5 tree [216]. It had the advantage of being able to deal with categorical and continuous variables along with handling the missing values as well. The tree combines conventional decision tree structure and linear regression models at the leaves. The main idea of the tree was to maximize the standard deviation reduction (SDR) [228]. The SDR was defines as

$$SDR = sd(T) - \sum_{i} \frac{|T_i|}{|T|} \times sd(T_i) \tag{5.2}$$

where T was the set of cases, T_i was the ith subset of cases that result from the tree splitting based on a set of variables (attributes), *sd(T)* was the standard deviation of T, and *sd(T_i)* was the standard deviation of T_i as a measure of error.

6.3 Experimental design

This section outlined the experiment setup and datasets used in our experiments. The experiment design was same as that of Section 5.4.

6.3.1 Experimental setup

The experiment setup was same for all of the experiments except for *experiment 2*, where number of nodes was changed in the cluster. The setup is outlined in Section 5.4.1.

For experiment 1, we used three datasets: NOAA, Airbnb and NES. The experimental setup was kept the same as that of 5.4.1. The Hive query used to run all experiments is :

```
SELECT *
FROM table_name
ORDER BY column_1,column_2,column_3;
```

For *experiment 2*, the existing cluster with *four* has been up-sized to five-nodes and then downsized to three-nodes. The experimental setup was the same as that of 4.1. Two different queries were run, on three cluster setups, using the Berkeley Earth Surface Temperature dataset.

```
Query 1: SELECT *
         FROM table_name
         ORDER BY column_1,column_2;

Query 2: SELECT *
         FROM table_name
         ORDER BY column_1,column_2,column_3;
```

For *experiment 3*, we designed a 4 node cluster at Amazon EC2. Each node was a t2.large instance with 2 vCPUs, 8.0 GB memory and 50 GB of storage. The experiment setup did not involve the use of SDN-enabled switch. The Berkeley Earth Surface Temperature dataset was used to run experiments, using the two queries below.

```
Query 1: SELECT *
         FROM table_name
         ORDER BY column_1,column_2;

Query 2: SELECT *
         FROM table_name
         ORDER BY column_1,column_2,column_3;
```

6.3.2 Datasets

1. The Berkeley Earth Surface Temperature (EST) Study combined 1.6 billion temperature reports from 16 pre-existing archives. Dataset had six attributes and almost 10 million records. The size of the file was 650 MB [1].

2. Airbnb dataset had 8,348,173 records and is published by Inside Airbnb. It had a total of 6 attributes and file size was 3.2 GB [2].

[1]available at `https://data.opendatasoft.com/explore/dataset/`
[2]available at `http://insideairbnb.com/get-the-data.html`

3. Historical National Oceanic and Atmospheric Administration(NOAA) includes daily land surface observations from around the world. The dataset was developed to meet the needs of climate analysis and monitoring studies that require data at a sub-monthly time resolution (e.g., assessments of the frequency of heavy rainfall, heat wave duration, etc.) [229]. File size of the dataset was 2.4 GB. Dataset had ten attributes and almost 37 million records [3].

4. National Current Employment Statistics (NES) dataset was published by Bureau of labor Statistics. The dataset comprised of a monthly survey of 147,000 businesses and government agencies, representing 634,000 worksites. The dataset had 21 attributes, 7,512,728 records and file size was 1.9 GB [3].

6.4 Results and discussion

This sections present the results from our experiments.

6.4.1 Impact of change in dataset

This section discuss the results of different resource utilization and execution time for change of datasets.

CPU usage

Figure C.1 showed the impact of number of mappers and block size over different datasets. NOAA showed the lowest overall mean CPU usage while Airbnb showed the highest overall mean CPU usage. The results showed that mean CPU usage increased as the number of mappers increased. The mean CPU usage of number of mappers remained less than overall mean when the number of mappers were lower than or equal to 10. For the datasets with large number

[3] available at `https://data.opendatasoft.com/explore/dataset/`

of columns, this cascading pattern of increasing mean with the increasing number of mappers was more evident. While for the datasets having lower number of columns the mappers mean was almost equal to the overall mean CPU usage. Another important observation was the high CPU usage in case of Airbnb dataset which was due the presence of lengthy text fields in the datasets.

Figure C.2 showed the behavior of CPU usage when we taken into account all configuration parameters. The results showed that there was a gradual increase in the mean CPU usage with increasing order of number of mappers and replica. keeping everything constant, lower value of number of replicas always resulted in lower CPU usage. For Airbnb dataset the number of replicas with value *two* always gave mean CPU usage lower than overall mean usage.

Figure C.3 showed the impact of number of replicas and block size on CPU usage of different datasets. The number of replicas with larger value showed mean CPU usage greater than the overall mean. However, the difference between overall mean and number of replicas mean was very small. Also, the range of CPU usage was small when the number of columns were small. So, NES dataset had the highest range of values suggesting higher standard deviations for higher number of columns. Figure C.4 showed the impact of number of mappers and number of replicas with CPU usage. The pattern was the same as discussed previously. Number of mappers showed a very discriminatory behavior among its different values. The gradual increase in mean CPU usage was shown in the graph for both NOAA and NES datasets.

Figure C.5 showed the difference in mean CPU usage for different datasets. The results showed that there was a gradual increase in mean CPU usage as the number of mappers increased. Also higher number of replica resulted in higher mean CPU usage. However, for Airbnb dataset, higher number of replicas resulted in mean CPU usage higher than overall mean CPU usage. Figure C.6 showed the comparison of standard deviations for different datasets. The results showed that Airbnb dataset had the highest standard deviation among all.

Figure6.2 showed the clustering of data into two clusters. The results showed that there was a difference in mean CPU usage when the number of mappers were less than or equal to 10 and when number of mappers become greater than 10. Both clusters received equal number of entries. The variables can clearly be segregated in to two distinct clusters. As we can see in the Figures as well that cluster 1 was clearly distinct with respect to the overall mean of the data. The clustering model also suggested that EST dataset showed the highest CPU in cluster 1 while Airbnb showed highest CPU usage in cluster 2.

F-statistics results suggested that NES was the best variable (dataset) that distinguishes between the two clusters. Other datasets that showed very good F-statistics results are EST and NOAA. All of these three datasets had *p-value* less than significance level (0.05) which suggested that the difference between means was significant. We also observed that the between-group sum of squares was higher than within-group sum of square for the two clusters. This suggested that individual measures in each clusters were closer while the values assigned to each cluster were quite distinct and showed distinctly similar CPU usage in different datasets.

For high CPU usage, 100% of the rules were able to predict the CPU usage for NOAA, Aribnb and NES datasets. The rule $R > 2$, $M > 15$ *and* $N > 0.5$, which comprises of the 24.84% of the total number of instances for NOAA, 42.65% for Airbnb and 50% for NES.

For low CPU usage 33% of the rules were true for the NOAA and NES datasets, while 100% for Airbnb. An example of decision rule is M ≤ 15 and N > 0.5 which makes the 100% of the instances for NOAA dataset, 70.85% for Airbnb and 99.79% for the NES.

Individual rules and their prediction percentage for different datasets were given in Section C.1.1. We compared the percentage difference in high and low CPU usage for the three datasets. For NOAA dataset, high CPU usage was 23.89% higher than the low CPU usage. For Airbnb the difference was 14% and for NES the difference was 35.65%.

The *correlation analysis* by using M5P regression model showed a very good correlation with value of 0.8 and root mean square error (RMSE) of the model is low. Linear regression model (LM14), LM1, LM6 and LM13 covered high number of observations and lower RMSE value suggesting how close they were to the line of best fit. As shown in Figure C.7, number of mappers, number of columns in dataset and number of replicas were the most contributing attributes

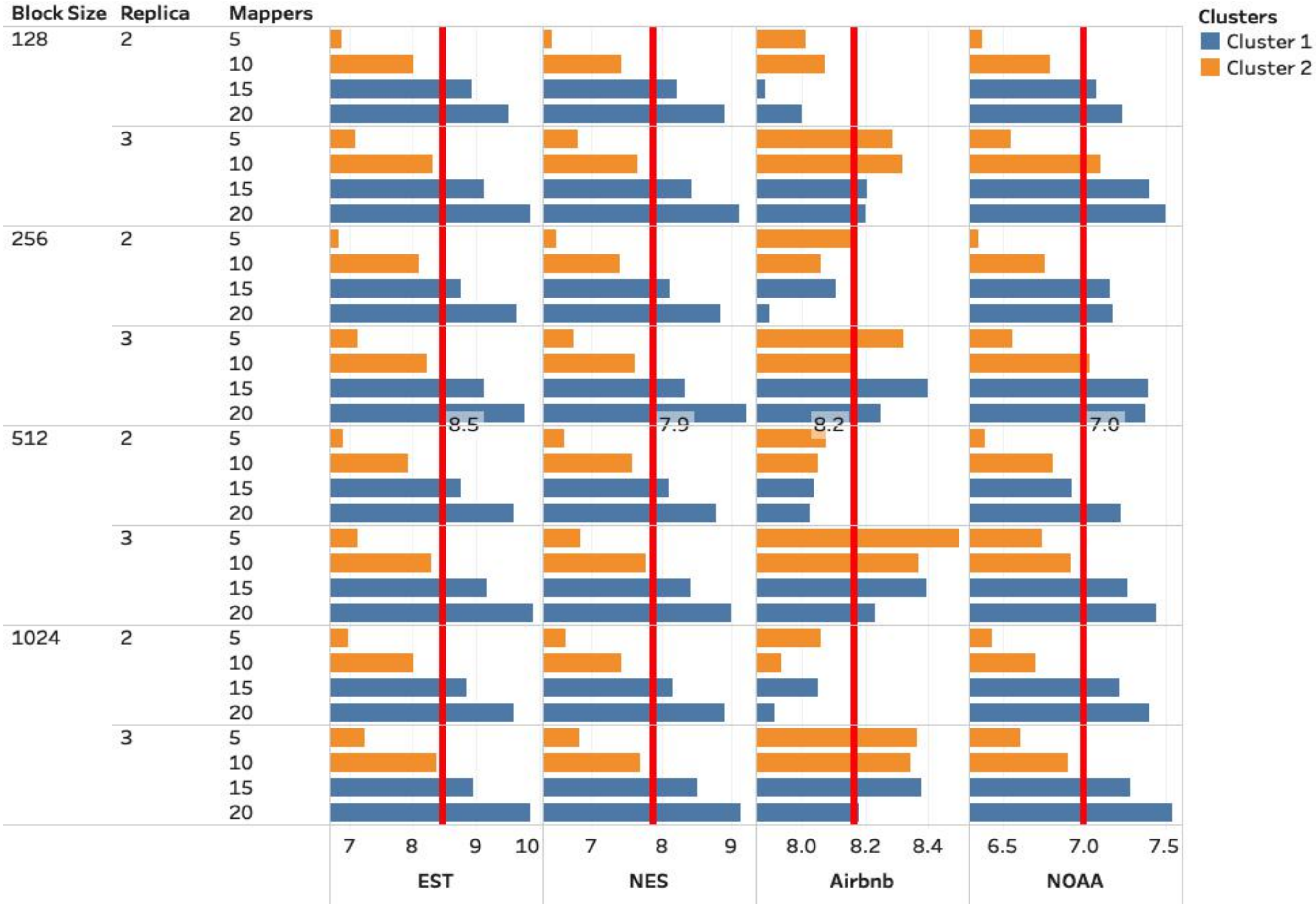

Figure 6.2: Result of k-mean clustering for CPU of different datasets.

contributing towards the overall CPU usage. The linear models were presented in Section C.1.2 and Linear Model(LM) 1 is shown below.

```
CPU usage =
        -0.0181 * No_of_columns
    - 0 * Block_size
    + 0.0109 * Replica
    + 0.0004 * Mappers
    + 0.1795
```

The *Pearson correlation coefficient* was calculated for the CPU usage data of different datasets. The results showed that number of mappers had moderate relationship with CPU usage while number of columns, number of replicas had weak relationships with CPU usage. The association between number of mappers and number of replicas with CPU showed positive correlation while number of columns showed negative correlation with CPU usage.

Disk usage

Figure C.8 showed the behavior of block size and number of replicas for disk usage. The mean usage increased gradually with the increase of block size for the NES and Airbnb datasets, but for NOAA the mean usage increased up to block size of 512 and then drops. Another pattern was that the mean disk usage when the number of replicas was equal to two was always less than when the number of replicas was equal to three for block size up to 512.

Figure C.9 showed the impact of block size and number of mappers on disk usage. The mean disk usage increased as the block size increased. The block size less than or equal to 256 kept the mean disk usage less than the overall mean disk usage. For the datasets having large text columns like Airbnb, increased number of mappers mean disk usage shows cascading increasing or decreasing pattern.

FigureC.10 showed the impact of different values of Hadoop configuration parameters on disk usage. The results showed that experimental runs with low block size and low number of replicas can result in lower mean disk usage than the overall mean disk usage. Also, the higher number of replicas and block size could always result in mean disk usage higher than the overall mean disk usage.

Figure C.11 showed the comparison of means for disk usage of different datasets. The results showed that there was a gradual increase in disk usage with the increase in block size. The EST dataset showed the lowest overall mean disk usage while NES showed the highest mean disk usage followed by the Airbnb. The mean usage was less than overall mean as the block size was less than or equal to 512MB. The block size greater than 512 MB showed the highest disk usage among all other values of block size for most of the datasets.

Figure C.12 showed the comparison of standard deviations of the different datasets. The EST dataset showed the lowest standard deviation while Airbnb showed the highest standard deviation followed by NOAA dataset. This suggested that the datasets having textual data or more number of fields can result in higher mean disk usage.

Figure 6.3 showed the output of k-mean clustering applied on disk usage data for different datasets. The results suggested that block size play a vital role in disk usage. Block size starting from 128MB to 512MB was clustered into one cluster. Although block size of 512 MB seemed to be a value which can result in over-utilization of resources if other Hadoop parameters were not chosen carefully. As we can see block size of 512MB, the number of replicas with value 3 and higher number of mappers results in mean disk usage higher than overall mean disk usage for most of the datasets. For the value above 512MB, the disk usage was high most of the times and this was placed in cluster 2 by the algorithm. The results also suggested that cluster 1 received 32% of the data while cluster 2 got 25% of the data. For both clusters, NES showed the highest mean disk usage followed by Airbnb while EST showed the lowest mean disk usage.

The *F-statistics* demonstrated that NES was the variable or dataset that distinguishes the two clusters, followed Airbnb and EST datasets. The significance test value suggested that the difference between means of the two clusters for these three datasets, is also significant. The between-group sum of squares and within-group sum of squares were very similar. So, the analysis suggested that although the results were significant for NES, Airbnb and EST datasets but the cluster means were very close to the overall mean.

For high disk usage 77.77% of the rules were true for NOAA dataset, and 66.67% rules were true for Airbnb and NES datasets. For example, the prediction percentage of decision rule, B > 256, R > 2, for different datasets was: 55.62% for NOAA dataset, 48.43% for Airbnb dataset and 56.13% for NES dataset.

For low disk usage, 42.85% rules were true for NOAA and Airbnb datasets, and 57.14% rules were true for NES dataset. The individual prediction percentage of decision rules has been calculated, for example, B $<= 256$, R $<= 2$, which was 38.74% for NOAA dataset, 44.86% for Airbnb dataset and 50.10% for NES

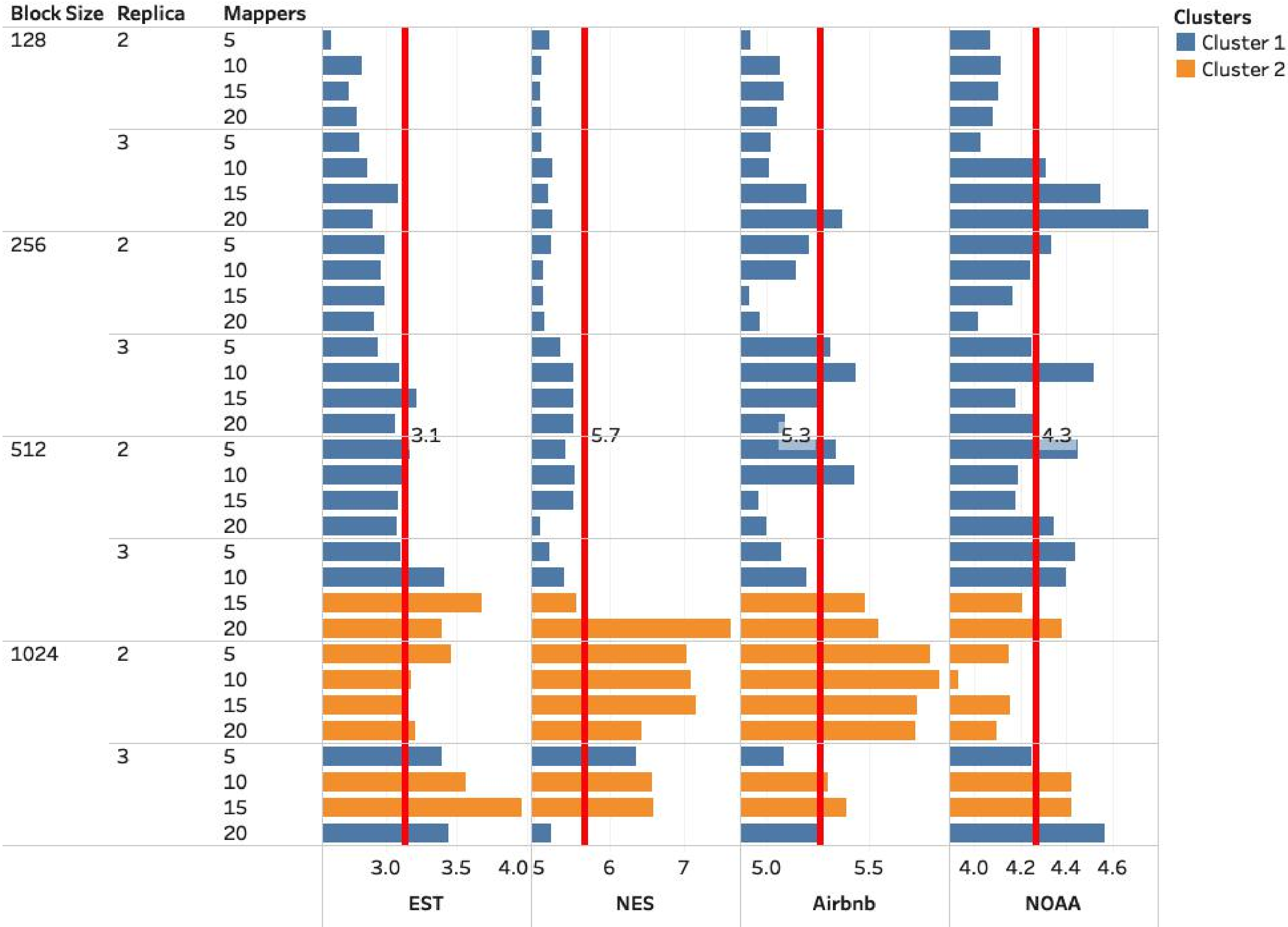

Figure 6.3: Result of k-mean clustering for disk usage of different datasets.

dataset.

Individual rules and their prediction percentage for different datasets was given in Section C.2.1. The percentage difference between high and low usage for disk utilization was : NOAA 18.2%, Airbnb 18.42% and NES was 34.59%.

The *correlation analysis* showed that there is high correlation (0.86) between Hadoop parameters and number of columns in different datasets is demonstrated by the regression M5P tree as shown in Figure C.13. The block size was the most important attribute for disk usage followed by number of columns in the dataset and replica. Linear models (LM) 1, 5, 6, 16 19 are some of the examples with higher number of observations and lower RMSE. A large number of linear models (35) were suggested by the M5P tree, representing the complexity of the situation. These models were listed in Section C.2.2.

The *Pearson correlation coefficient* calculated for the data showed that all of the attributes have positive correlation with disk usage. Number of columns in dataset showed moderate level association with disk usage while block size showed weak association with disk usage.

Memory usage

FigureC.14 showed the mean memory usage for the four datasets. The results showed that EST had the lowest overall memory usage while Airbnb and NOAA has the highest memory usages. For datasets having numeric fields, like EST and NES, lower number of mappers resulted in mean memory usage lower than the overall mean memory usage. But for number of mappers greater than ten, the mean memory usage was greater than the overall mean usage.

Figure C.15 showed the standard deviation for memory usage. The EST dataset showed highest value among all dataset while NES and NOAA showed the lowest values of standard deviation.

Figure C.16 showed the comparison of three datasets for block sizes, number of mappers and replica. As a general trend, less number of mappers showed lesser mean memory usage for numeric datasets like NES and NOAA. But for Airbnb dataset, the mean memory usage was similar or higher to that of the overall mean memory usage. Also, there were a large number of outliers indicated in the boxplots of NOAA and NES.

Figure 6.4 showed the output of k-mean clustering algorithm applied on memory usage data of all the datasets. The results showed that both block size and number of mappers play an important role in distinguishing the underlying pattern for memory usage. When the number of mappers were less than or equal to 10, the mean memory usage was less than overall mean memory usage. For most of the datasets, the number of mappers greater than 10 and block size higher than 512 MB resulted in mean memory usage higher than the overall mean memory usage. The number of observations assigned to cluster 1 were 53% as compare to the cluster 2 which received 47% of all of the observations. For both clusters NOAA datasets showed highest mean memory usage followed by Airbnb and NES while EST showed the lowest mean memory usage in both of the clusters.

The *F-Statistics* and *p-value* showed that NES was the dataset able to significantly distinguish between two clusters. Other datasets showed very small F-statistics and p-values suggesting that the distinguishing ability between clusters was very low. The between-group was smaller than within-group sum of squares which shows that the mean of each cluster was very close to the overall mean memory usage.

For low memory usage, 71.42% of the rules were found to be true for all datasets. A decision rule was: B > 128 MB and M $<= 5$ which was 27.25% of the instances for NOAA dataset, 20.87% for Airbnb and 29.79% for NES dataset.

High memory usage, 100% rules were found true for all datasets. The prediction percentage of a decision rule, M > 15 and N > 3, for different datasets was: NOAA's 27% and Airbnb's 49.27% of the total instances.

Individual rules and their prediction percentage for different datasets was given in Section C.3.1. The percentage difference between high and low memory usage for NOAA dataset is 7.8%, for Airbnb 7.2% and for NES 10.79%.

Figure C.17 showed the M5P tree to demonstrate the correlation between memory usage and Hadoop job attributes for different datasets. The tree predicted 22 linear models as shown in Section C.3.2. Number of mappers and block size were the two most important attributes associated with different levels of memory usages for different datasets.

The *Pearson correlation* coefficient calculated for memory usage showed that number of columns in dataset and number of replicas had negative correlation with memory usage while block size and number of mappers have positive correlations. Number of columns in dataset and number of mappers had weak correlation while number of replicas and block size showed very weak correlation.

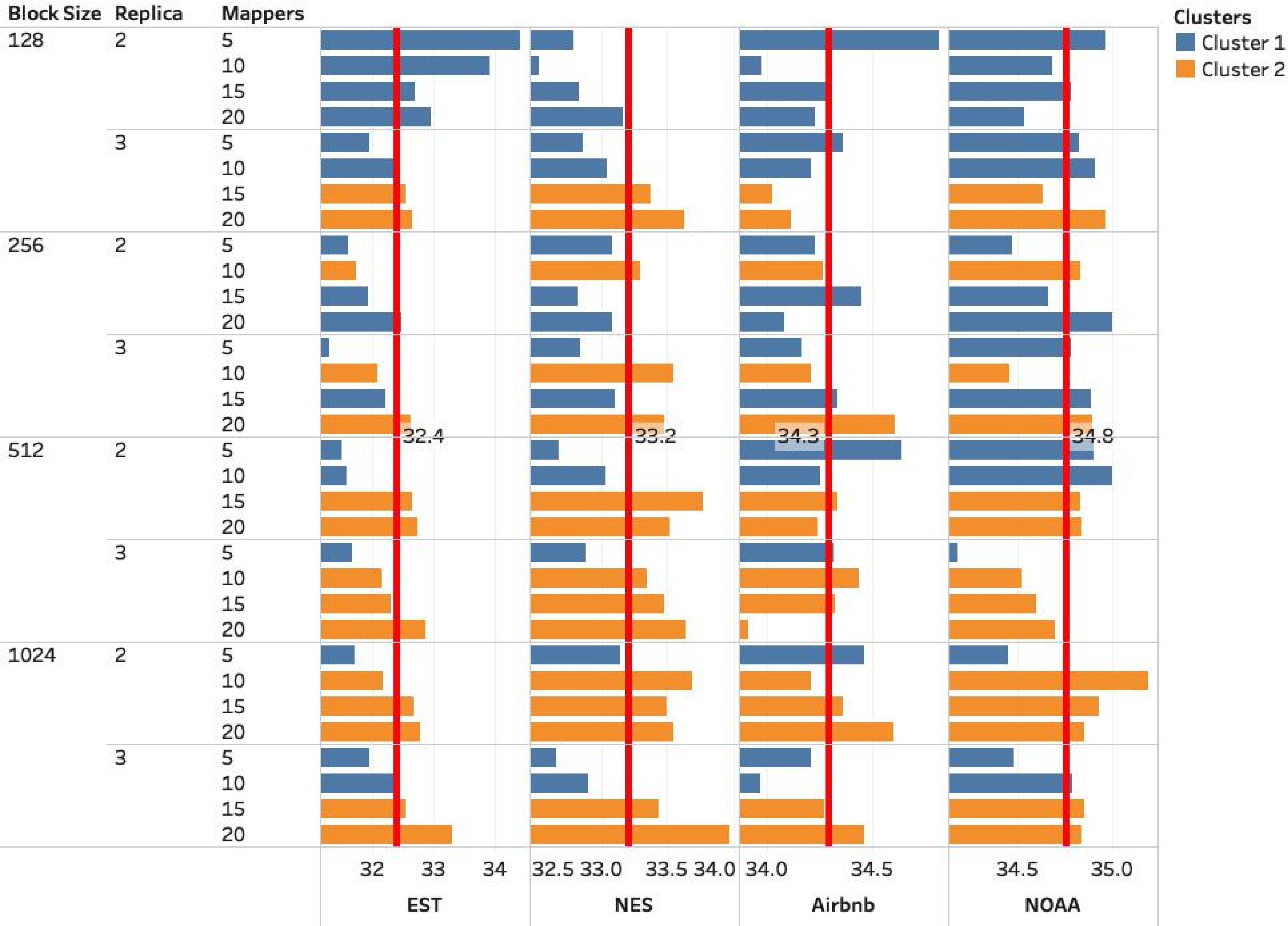

Figure 6.4: Result of k-mean clustering for memory usage of different datasets.

Network usage

FigureC.18 showed the comparison of means for the four datasets. The EST dataset had the lowest network usage while Airbnb had the highest mean network usage followed by the NES and NOAA datasets. Number of replica could have a significant impact on the overall mean network usage. Lower number of replicas resulted in lower network usage than the overall mean network usage in each of our experiments. The results also suggested that datasets with more text columns, like Airbnb, and more number of columns, like NES, can result in higher network usage.

Figure C.19 showed the standard deviations of each of experiment. The results demonstrated that more number of columns and huge text fields in the datasets result in higher standard deviations.

Figure C.20 showed different Hadoop parameters for network usage of NOAA, Airbnb and NES. The Figure showed that the mean and median of number of replicas with lower value was always less than the overall mean network usage.

Figure 6.5 showed the output of k-mean clustering algorithm for network usage data of different datasets. The results showed that less number of replicas has mean network usage less than overall mean network usage while higher number of replicas always resulted in mean network usage higher than the overall mean network usage. Both the clusters received equal number of instances. For both of the clusters Airbnb demonstrated the highest mean network usage, followed by NES. The between-group sum of square was very high which shows that the individual means were not closer to the overall mean and the two cluster groups were significantly distinct.

The *F-statistics* value for Airbnb was the highest followed by EST, NES and NOAA. This showed that Airbnb significantly distinguishes the values in two clusters. The *p-value* also far less than the significance value, which shows that the difference between the means of the two clusters was highly significant. The lower within-group sum of square suggest that the values in each cluster were very close to the cluster mean.

For high network usage, 72% of the rules were true for NES dataset, while for both Airbnb and NOAA datasets 50% of the rules are true. An example of decision rule was R > 2 and N > 0.5, which is 91.04% for NOAA dataset, 98.75% for Airbnb dataset and 97.29% for NES dataset.

For low network usage, 100% of the rules were true for all of the datasets. Individual rules and their prediction percentage for different datasets is given in Section C.4.1. The percentage difference between high and low network usage for NOAA dataset is 72.9%, for Airbnb dataset it was 49.8% and for NES dataset it was 73.66%.

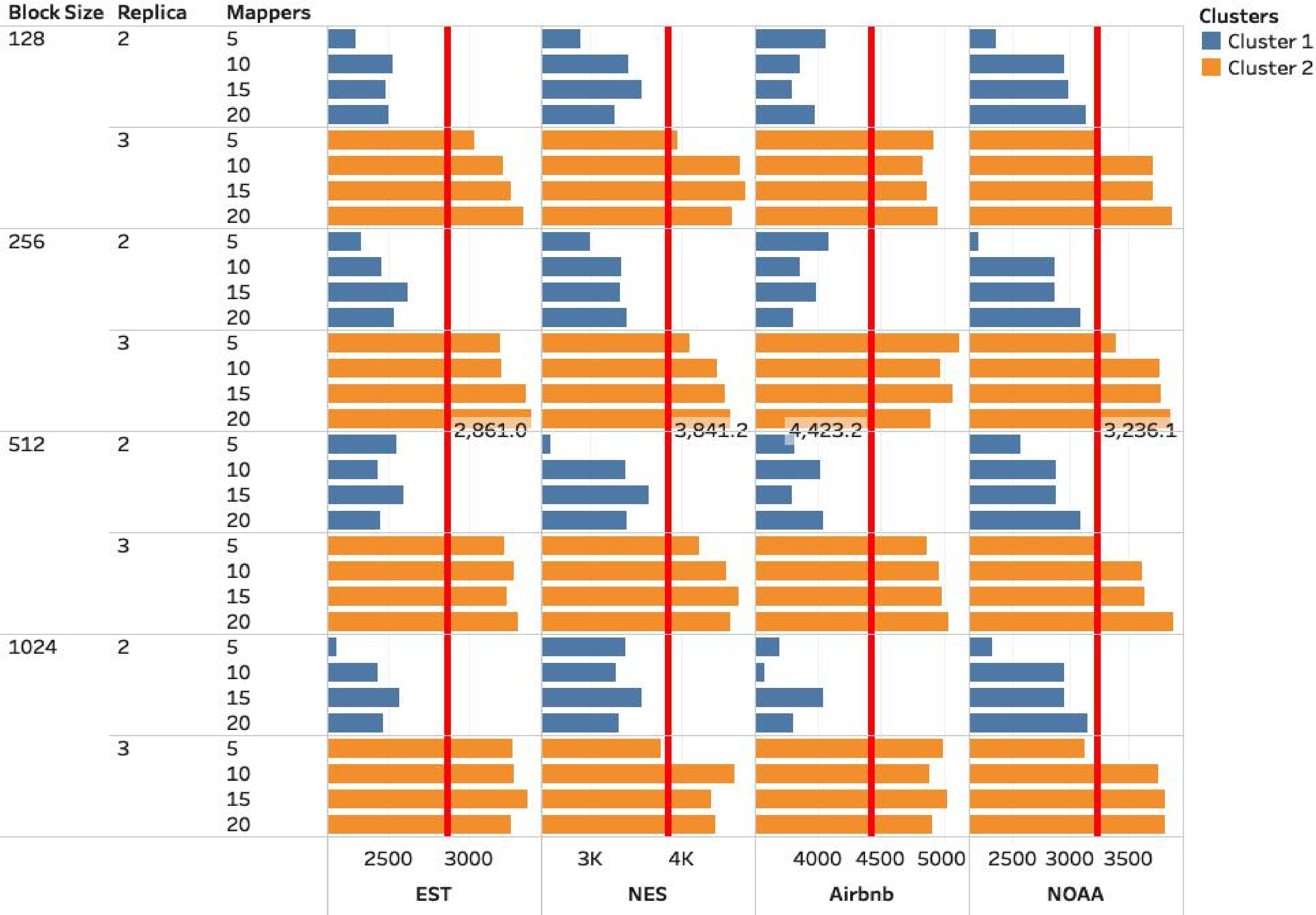

Figure 6.5: Result of k-mean clustering for network usage of different datasets.

Figure C.21 showed the M5P tree for network usage data of different datasets. The correlation coefficient was 0.84, which was high and 12 linear models were predicted for the data. Number of replicas was the most important attribute for network usage, followed by number of columns and number of mappers. Linear models are being presented in Section C.4.2 and LM 12 encompasses 37.5% of the total number of observations with a very low RMSE.

The *Pearson correlation* coefficient for network usage of different datasets showed that number of columns, number of replicas and number of mappers had positive correlation with network usage while block size have negative correlation. While analyzing the strength of the relationship, number of replicas had moderately strong relationship while number of mappers and number of columns had weak associations with memory usage. Block size had very weak association with memory usage.

Execution time

Figure C.22 showed the mean execution time for four datasets used in our experiments. The mean execution time increased as the number of mappers increases for EST. But for the other three datasets the less number of mappers demonstrated mean execution time higher than the overall mean execution time. The results also showed that as the dataset fields become more complex the execution time increased, for example the NOAA datasets had large number of coordinates and Airbnb dataset had more textual data.

Figure C.23 showed the standard deviations of the execution times for different datasets. The results showed that EST has the least standard deviation while Airbnb and NOAA had the highest standard deviations.

Figure C.24, Figure C.25 and Figure C.26 showed different comparisons of block size, number of mappers and replica. The results showed that lower number of mappers resulted in higher execution times for NOAA and Airbnb dataset. But for Airbnb dataset the mean execution time was very much similar to overall mean execution time.

Figure 6.6 showed the output of the k-mean clustering applied on execution times of the datasets. Number of mappers seemed to be playing a very distinct role in clustering the instances into two groups. The cluster 1 received almost 44% instances while cluster 2 got 56% of the values. For cluster 1, EST showed the highest mean execution time followed by Airbnb. For cluster 2, the NES dataset showed the highest mean execution time followed by Airbnb and NOAA.

The *F-statistics* showed that EST was the best dataset, which was able to distinguish between two clusters followed by NES and NOAA. the *p-value* also suggested that EST, NES and NOAA datasets showed significant difference between means of the two clusters. The between-group sum of square was less than withing-group sum of square. This suggested that the means of each clusters were close to the overall mean.

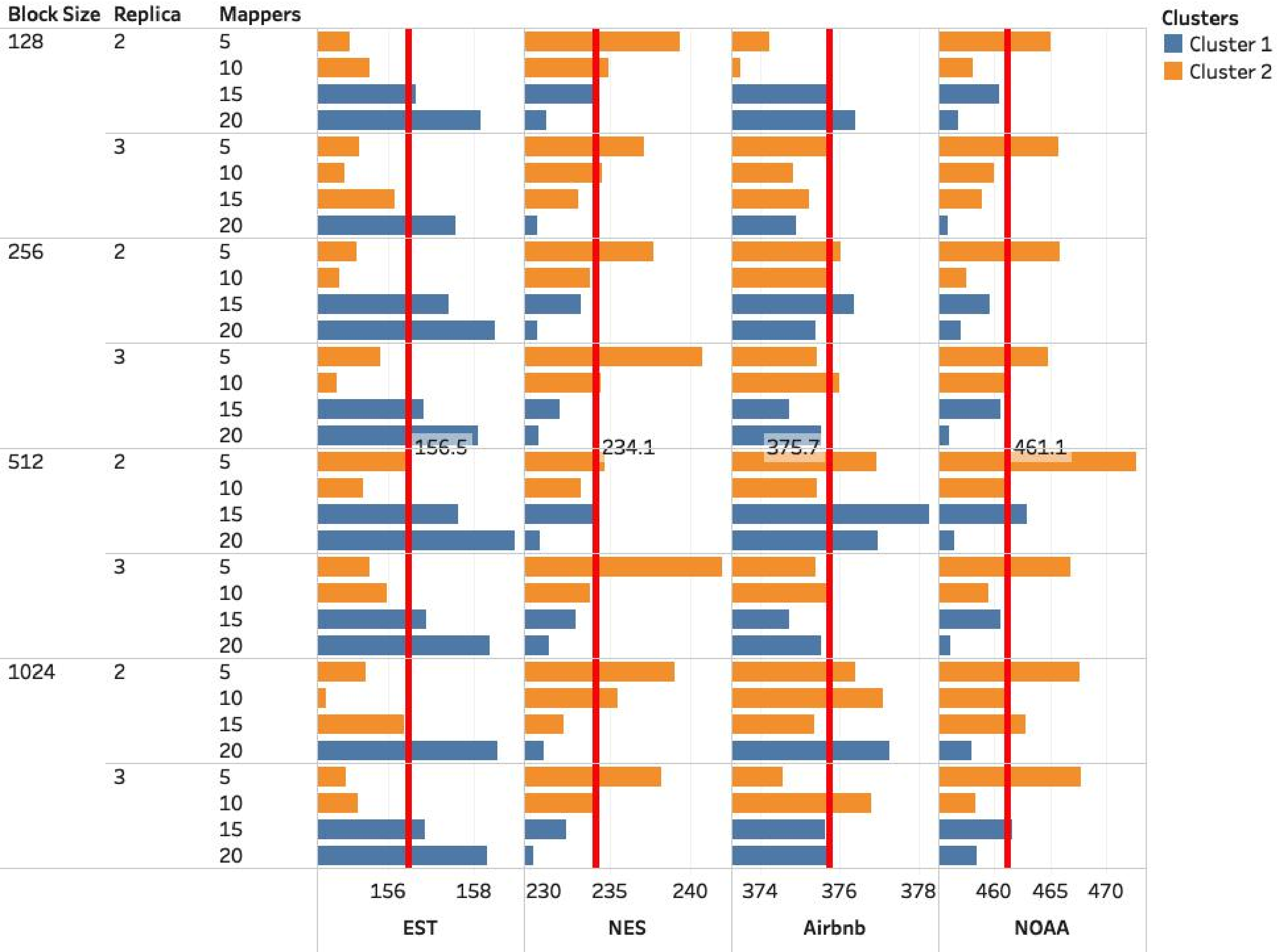

Figure 6.6: Result of k-mean clustering for execution time of different datasets.

For high execution time, the following rules have been found to be true for NOAA, Airbnb and NES datasets, which are 100% of the total rule set. A decision rule was N > 3 and M $<= 10$, which is 58.54% for NOAA dataset, 49.34%

for Airbnb dataset and 67.39% for NES dataset of the total number of instances.

Individual rules and their prediction percentage for different datasets was given in Section C.5.1. The percentage difference between high and low execution times for NOAA dataset is 8.64%, for Airbnb dataset is 7.2% and for NES dataset is 13.43%

The correlation analysis for overall time data for different datasets was shown in Figure C.27. A total of 3 linear models were predicted with a correlation coefficient of 0.36. Number of columns in dataset and number of mappers are the only attributes which contributed towards the correlation and prediction of outcome. the linear modes are presented in Section C.5.2 where LM1 covers 50% of the observations.

The *Pearson correlation* coefficient for the overall execution time showed that number of columns in the datasets, number of replicas and number of mappers have negative correlation with execution time while block size has positive correlation. The number of columns showed weak relationship while all other attributes showed very weak associations with execution time.

Table 6.1: Mean and standard deviation of CPU usage, disk usage, memory usage, network usage and execution time for high and low usages of different datasets. ($\bar{x} \pm$ s)

	Low Usage			**High usage**		
Datasets	**NOAA**	**Airbnb**	**NES**	**NOAA**	**Airbnb**	**NES**
Network Usage	2085 $\pm$ 215	3419 $\pm$ 206	2530 $\pm$ 298	3966 $\pm$ 71	5210 $\pm$ 157	4828 $\pm$ 164
CPU usage	6.2 $\pm$ 0.02	7.84 $\pm$ 0.03	6.4 $\pm$ 0.06	7.6 $\pm$ 0.13	8.68 $\pm$ 0.14	9.2 $\pm$ 0.11
Disk Usage	3.9$\pm$ 0.023	4.88$\pm$ 0.01	5.06 $\pm$ 0.02	4.7$\pm$ 0.024	5.88 $\pm$ 0.06	7.6 $\pm$ 0.11
Memory Usage	33.28 $\pm$ 0.25	33.92 $\pm$ 0.16	31.24 $\pm$ 0.2	35.47 $\pm$ 0.08	35.11 $\pm$ 0.06	34.12 $\pm$ 0.09
Execution Time	447.6$\pm$ 1.54	367.13 $\pm$ 0.89	223.1 $\pm$1.62	481.8 $\pm$ 1.74	387.46 $\pm$ 1.92	248.2 $\pm$ 1.9

Summary

From the results above we can summarize the following:

- We outlined two approaches to analyze our results, the *first* approach was the analysis by using descriptive statistics and building different graphical representations like boxplots and plotting means and standard deviations,

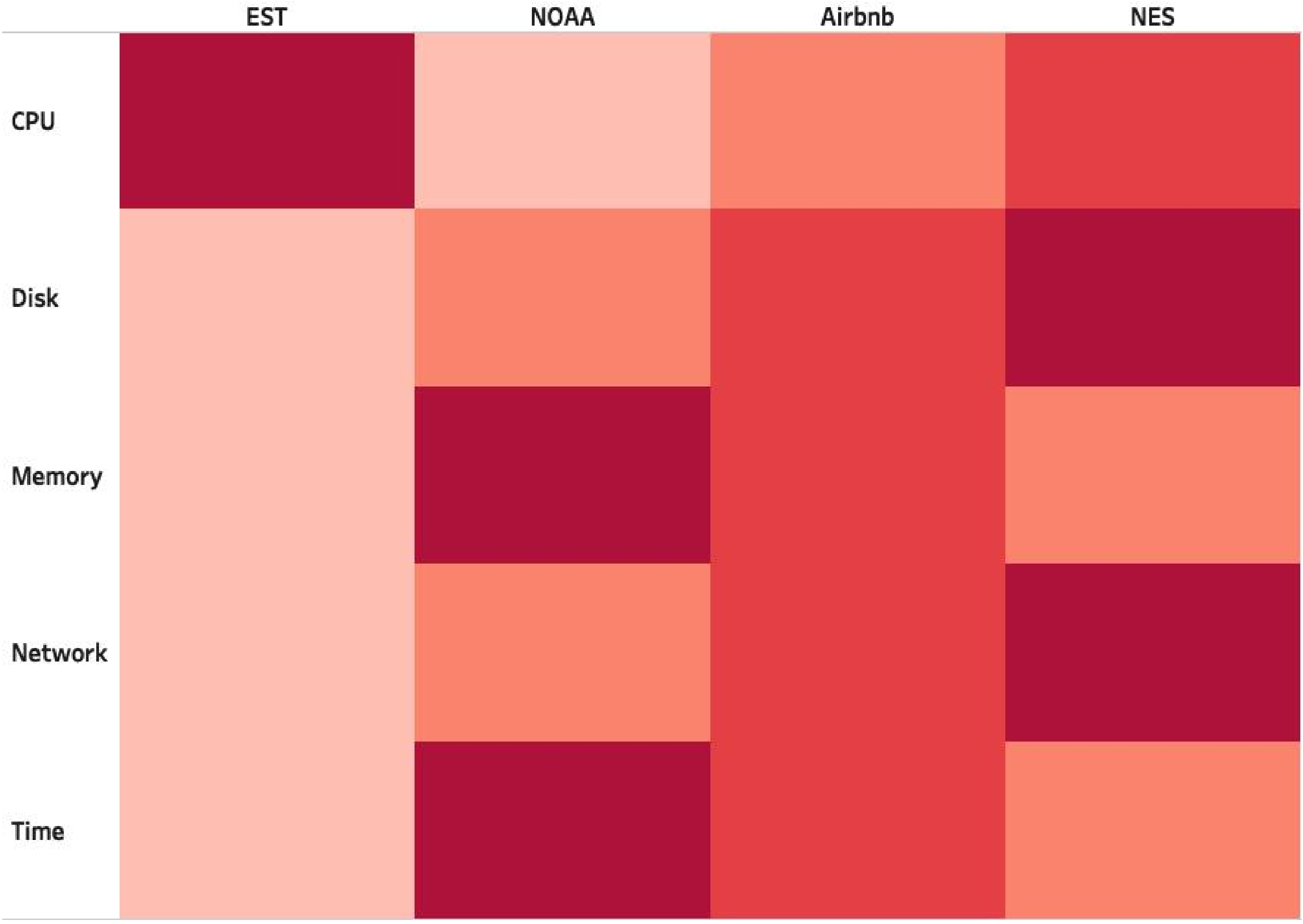

Figure 6.7: Heat map of resources utilization for different datasets

the *second* approach involved the use of automated process of clustering the data into two clusters by using *k-mean clustering algorithm*. The results showed that different values of number of mappers could greatly impact overall CPU usage, memory usage of Hadoop cluster and execution times of the Hadoop job. While different values of replica could affect disk usage and network usage of the cluster. Block size value had the highest impact over disk usage of Hadoop cluster. These results were mostly aligned with our previous findings in Chapter 5.

- We performed t-tests to check the statistical significance of the difference between highest and lowest usage of our selected performance metrics. This showed the effectiveness of the our method to give us the Hadoop job attributes with desired resource utilization and execution time. We found that all of our metrics disk usage, network usage, memory usage and execution time were statistically significant for both NOAA and Airbnb datasets. The results were shown in table 6.1.

- The decision tree build in previous experiment, Chapter 5, can generalize over the current experiments very well. On average 75% of the decision rules were able to correctly predict the resource utilization for NOAA, Aribnb and NES datasets. Although decision tree models are able to generalize well over unseen data from different datasets, the results also highlighted the impact a dataset can have on resource utilization.

- The correlation analysis was done to see the association of different attributes to each other. The analysis not only showed the strength of relationship but also the direction of relationship. Number of columns in datasets and number of mappers are the attributes, strongly correlated with the resource utilization and execution time of Hadoop jobs. Number of replicas was moderately correlated to the resource utilization, but it was strongly correlated with network usage. The block size showed weak correlation with most of the performance measure for different datasets.

- The overall comparison of resource utilization for different datasets was shown in Figure 6.7, where the lighter color represents lower resource utilization and the darker color represents the higher resource utilization. The EST dataset showed the lowest resource usage among the four datasets compared. While the NES datasets showed the highest overall resource utilization followed by Airbnb. This demonstrates that the datasets with higher number of columns and lengthy textual columns could greatly impact the overall system performance.

- So, apart from taking into consideration the importance of query structure and the different values of Hadoop parameter configurations (as discussed in Chapter 4 and Chapter 5), the choice of dataset was also important while designing and implementing a Hadoop cluster. Some of the important features of the datasets which should be taken into account were: the number of columns in the dataset table and type of data in the columns.

6.4.2 Impact of change in cluster size

This section discuss the results of different resource utilization and execution time for change in number of nodes in cluster. We executed our experiments after adding and removing one node in the existing *four* node cluster.

CPU usage

For five node cluster, our model predicted 75% of the outcome correctly while for three node cluster 60% of outcome predicted correctly.

FigureD.1 showed the overall mean CPU usage for cluster of different sizes. When we increased the number of nodes from *four* to *five*, overall mean CPU usage decreased by 16.67% and when we decreased the number of nodes from *four* to *three*, the overall mean CPU percentage usage was increased by 10.64%. If we look at the percentage change in decreasing cluster size from five nodes to three nodes, the mean CPU usage increased by 25.53%.

The analysis of standard deviation of CPU usage showed that increasing number of nodes resulted in low mean CPU usage but the standard deviation has increased significantly. While downsizing of the cluster from *four* nodes to three nodes keeps the standard deviation of CPU usage similar as shown in Figure D.2.

Figure D.3 showed the boxplot for each of the experiments run on different cluster sizes. The results showed many interesting patterns in our experiments. Firstly, decreasing the cluster size from five nodes to three nodes results in greater median CPU usage. Secondly increasing block size, number of replicas and number of mappers has a great impact on overall CPU usage. The results showed that higher block size, higher replication factor and higher number of mappers always result in higher CPU usages provided the size of the cluster remains the same. Decreasing the number of nodes in the cluster put more pressure on CPU usage. Same amount of work has to be done by lower number of CPU which resulted in significant increase in CPU usage. So increasing or decreasing number of nodes is a major environmental change in the system and it impacted overall CPU utilization.

Figure 6.8 showed the output of the k-mean clustering performed for CPU usage of different cluster sizes. The results showed that mean CPU usage increased with the increase of number of mappers. Also, the mean CPU usage remained less than the overall mean CPU usage when the number of mappers were less than or equal to 10. As soon as the number of mappers become greater than 10, the mean CPU usage become higher than the overall mean CPU usage. Both clusters received equal number of instances. For both clusters, the three node cluster showed highest mean CPU usage followed by *four* nodes and five nodes cluster. The between-group sum of square was higher than within-group sum of square which suggests that both of the k-mean clusters were not close to overall mean and their boundaries were very distinct apart from each other as well.

The *F-statistics* results showed that four node Hadoop cluster was dataset with highest distinguishing ability between the clusters. The *p-value* for the results showed that the difference between two clusters was statistically significant for all of the three Hadoop cluster sizes.

The *correlation analysis* by using M5P tree showed that the model had very high correlation coefficient with a value of 0.97. Number of mappers and number of nodes seemed to have a strong correlation with CPU usage for all of the experiments. Seven Linear models were predicted for the data as shown in FigD.1, and LM 1, LM2 and LM7 encompasses more than 83% of the observations as shown in Figure D.4. These models showed 0% RMSE, which suggests that they were the best fit lines in this correlation. The equation of LM1, which alone covered more than 50% of the observations is given below;

```
CPU usage =
-0.0013 * Nodes
+ 0.0003 * Replica
```

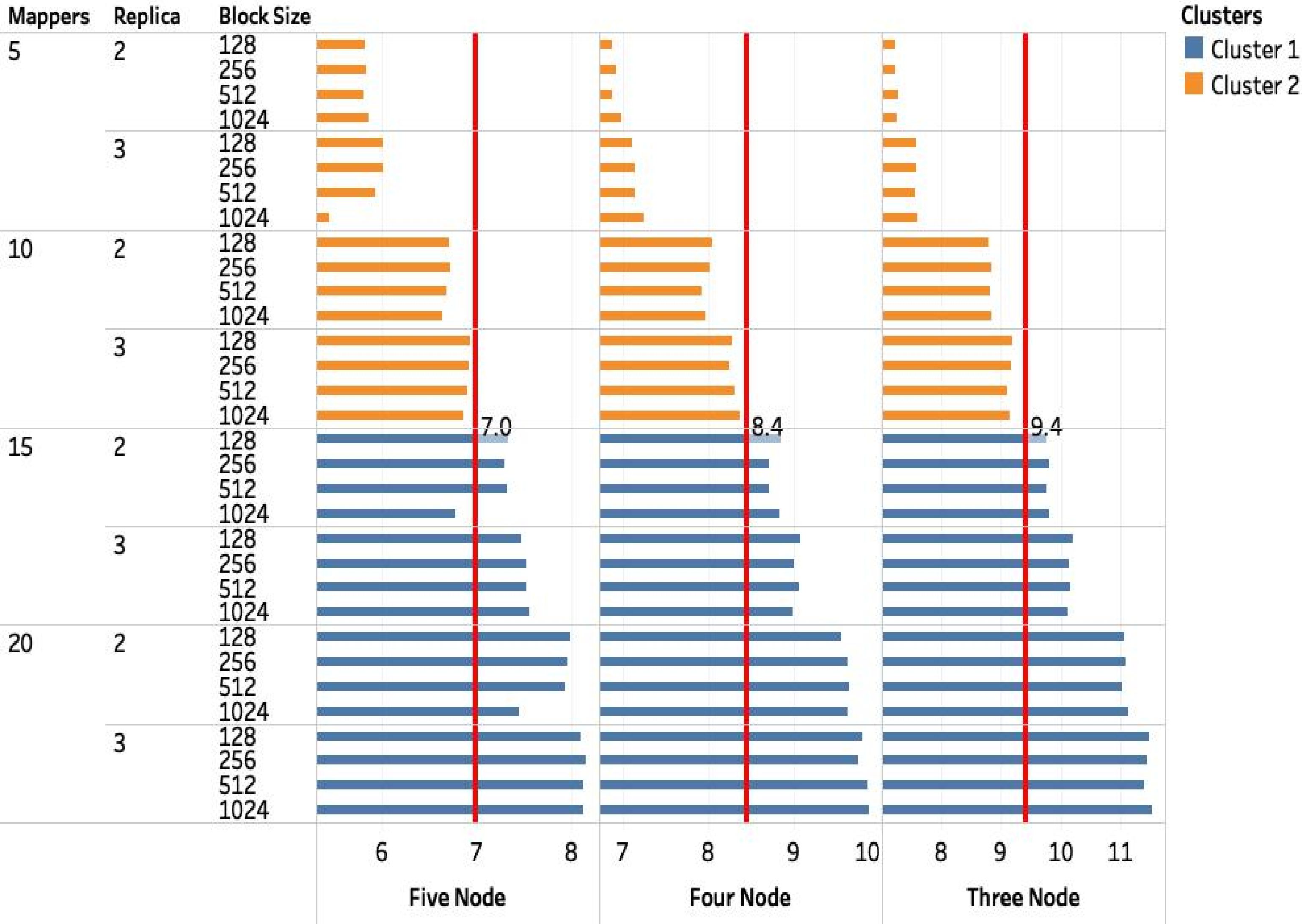

Figure 6.8: Result of k-mean clustering for CPU usage for cluster of different sizes.

```
+ 0.0002 * Mappers
+ 0.003
```

The *Pearson correlation* coefficient for CPU usage clusters with different nodes revealed that number of replicas and number of mappers had positive correlation with CPU usage while block size and number of nodes have negative correlation. The number of mappers and number of nodes demonstrated moderately strong association with CPU usage while number of replicas and block size showed very weak association.

Disk usage

Our model was able to predict 70% of the resource utilization correctly for both three nodes and five node cluster.

Figure D.5 showed the mean disk usage for the change in cluster size. When we increased the number of nodes from *four* to five the mean disk usage has increased by 27.9% and when we downsized the cluster from *four* nodes to three nodes the mean disk usage has increased by 39.20%. However decreasing the number of nodes from five to three nodes suggests that there was an overall increase in mean disk usage by 15.68%.

Figure D.6 showed the standard deviations of the cluster of different sizes. It shows that three node cluster had the highest standard deviation for disk usage percentage followed by five node cluster and *four* nodes cluster. The results also showed that smaller block size gives higher standard deviation in disk usage on the cluster with three nodes while larger block size gives higher standard deviation for cluster with five nodes. It seemed like the Hadoop parameters we chose and the dataset we used to run the jobs has an impact on disk usage which resulted in very low disk usage for the *four* nodes cluster.

Figure D.7, showed the boxplot of the disk usage for three different cluster sizes. A common pattern in all of these experiments was that lower block size results in lower disk usage compared to overall mean disk usage. As the block size increased, individual mean disk usage also tends to increased than overall mean disk usage.

Figure 6.9 showed the output of the k-mean clustering algorithm for the three Hadoop clusters with different number of nodes. The results showed that most of the times higher block size demonstrated mean disk usage higher than the overall mean disk usage. For all three Hadoop clusters, lower disk usage along with lower number of replicas and number of mappers resulted in mean disk usage lower than the overall mean disk usage. The clustering algorithm clustered almost 72% instances into one cluster and remaining 28% into other cluster. For both clusters, the three node cluster showed highest mean disk usage followed by five node and *four* nodes Hadoop cluster. The between-group sum of squares was less than within-group sum of square which suggested that although the cluster means were distinct but their boundaries may not be very distinct.

The *F-statistics* value for the Five node Hadoop cluster, followed by *four* nodes and three nodes, was the highest which shows that it has played greater role in distinguishing the between two clusters. The *p-value* suggests that the difference between the two clusters was statistically significant for the three Hadoop setups.

Figure D.8 showed the M5P regression tree for correlation between Hadoop parameters an cluster size. The correlation coefficient, 0.94, was very high suggesting that the model was good in predicting the disk usage and 12 linear models were predicted for different associations of attributes as shown in D.2. The number of nodes was the most important attribute contributing towards disk usages of the system. Block size was the other important attribute for disk utilization of different cluster sizes. Linear models with the most number of observations and lower RMSE value include LM1 and LM12.

The *Pearson correlation* coefficient for disk usage showed that number of nodes had negative correlation with disk usage while all of the Hadoop configuration parameters showed positive correlation with disk usage. The number of nodes and number of mappers demonstrated moderate association with disk usage. Block size showed a weak association while number of replicas showed very weak association with disk usage.

Memory usage

Our model was able to correctly predict 42% of the memory usage of five node cluster while 50% of the three node cluster.

Figure D.9 showed the mean memory usage for three clusters of different sizes. When we increased the number of node from *four* to five, overall mean memory usage decreased 20.54%. When we downsized the cluster from *four* node to three node the memory usage has decreased by 7.2%. if we calculate

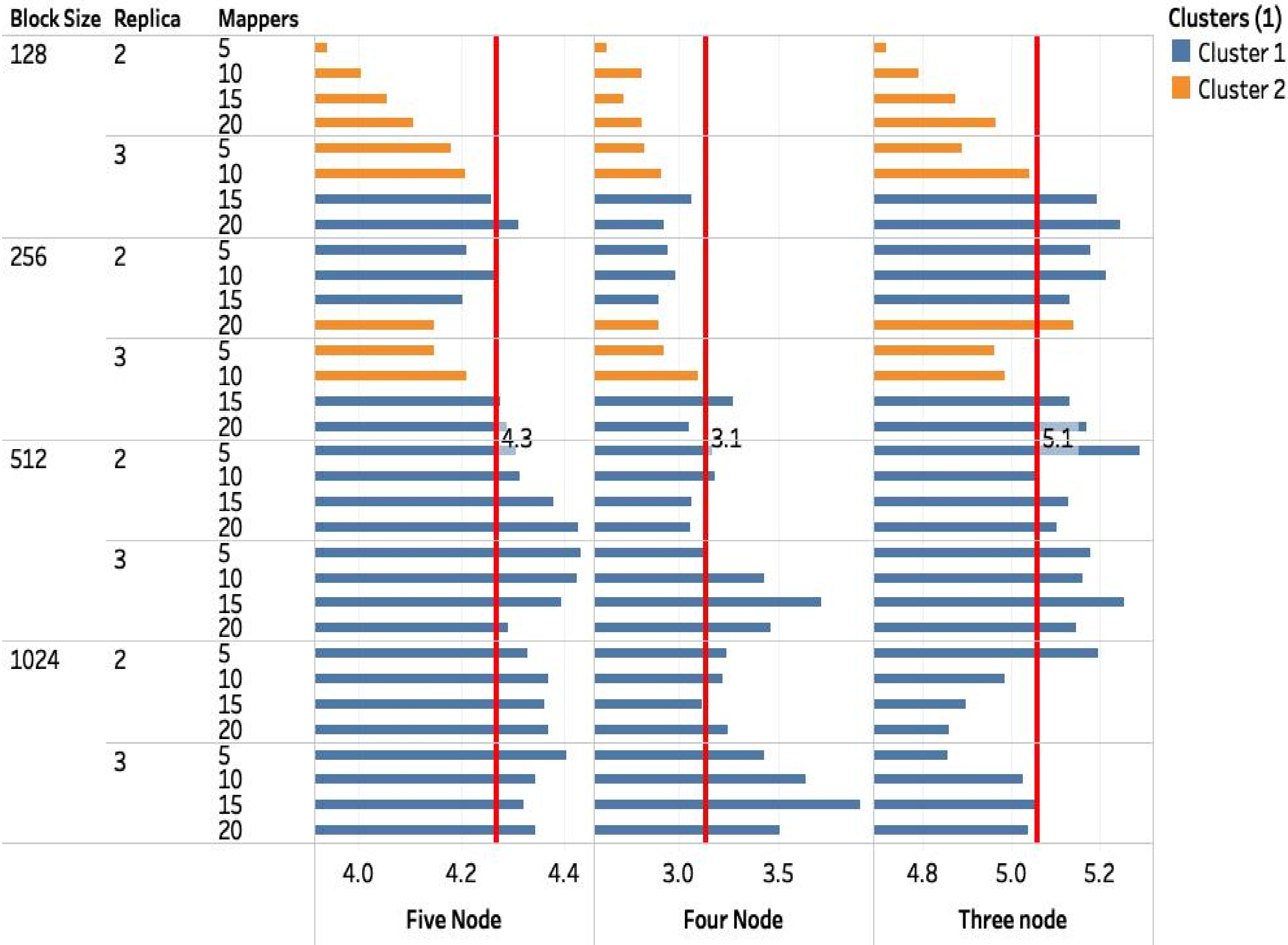

Figure 6.9: Result of k-mean clustering for disk usage for cluster of different sizes.

the percentage increased in mean memory usage for downsizing the cluster by two nodes, that was from five node and three node, there was an increase of 14.37%.

Figure D.10 showed the comparison of standard deviations of the five nodes, *four* nodes and three node cluster with the same dataset. Although, five node cluster showed the least mean memory usage but it had the highest overall standard deviation. The *four* nodes cluster had the highest mean memory usage but it had the lowest overall standard deviation. The percentage difference between the two was 20.37%.

Figure D.11 showed the boxplot of the memory usage of the three cluster sizes. The results showed that number of mappers had the most distinguishing effect on the cluster of different sizes. Lower number of mappers resulted in lower memory usage as compared to overall memory usage. As the number of mappers increased above 10, mean memory usage remained higher than overall memory usage. Another interesting observation was that when block size was 1024, the standard deviation of mean memory usage of the five node cluster was most of the times high.

Figure 6.10 showed the clustering of memory usage data for three different Hadoop clusters. The results demonstrated that the number of mappers played a vital role in memory usage of Hadoop clusters. The number of mappers lower than or equal to 10 mostly resulted in mean memory usage less than the overall mean memory usage. The cluster group 1 received 47% of the instances while cluster group 2 received 53% of the instances. For both groups, the *four* nodes Hadoop cluster showed the highest mean memory usage. The between-group sum of square was less than within-group sum of squares, which demonstrate that the cluster boundaries were not distinct enough and their centers were a bit close to the overall mean memory usage.

The *F-statistics* showed that the three node Hadoop cluster was the variable with highest distinguishing ability between the means of two k-mean clusters. The *p-value* for the three Hadoop setups suggests that the difference between the means of the two clusters was statistically significant only for three node and five node Hadoop cluster.

The *correlation analysis* for memory usage data of different cluster sizes is shown in Figure D.12. The model tree predicted eight linear models, shown in Section D.3, with a high correlation coefficient of 0.78. LM1 and LM8 covered highest number of observations (66.67%) with 0% RMSE demonstrating it to be equivalent to the best fit line. The number of nodes was the most important attribute in predicting the association between attributes and memory usage. The number of mappers was the other important attribute in this regard.

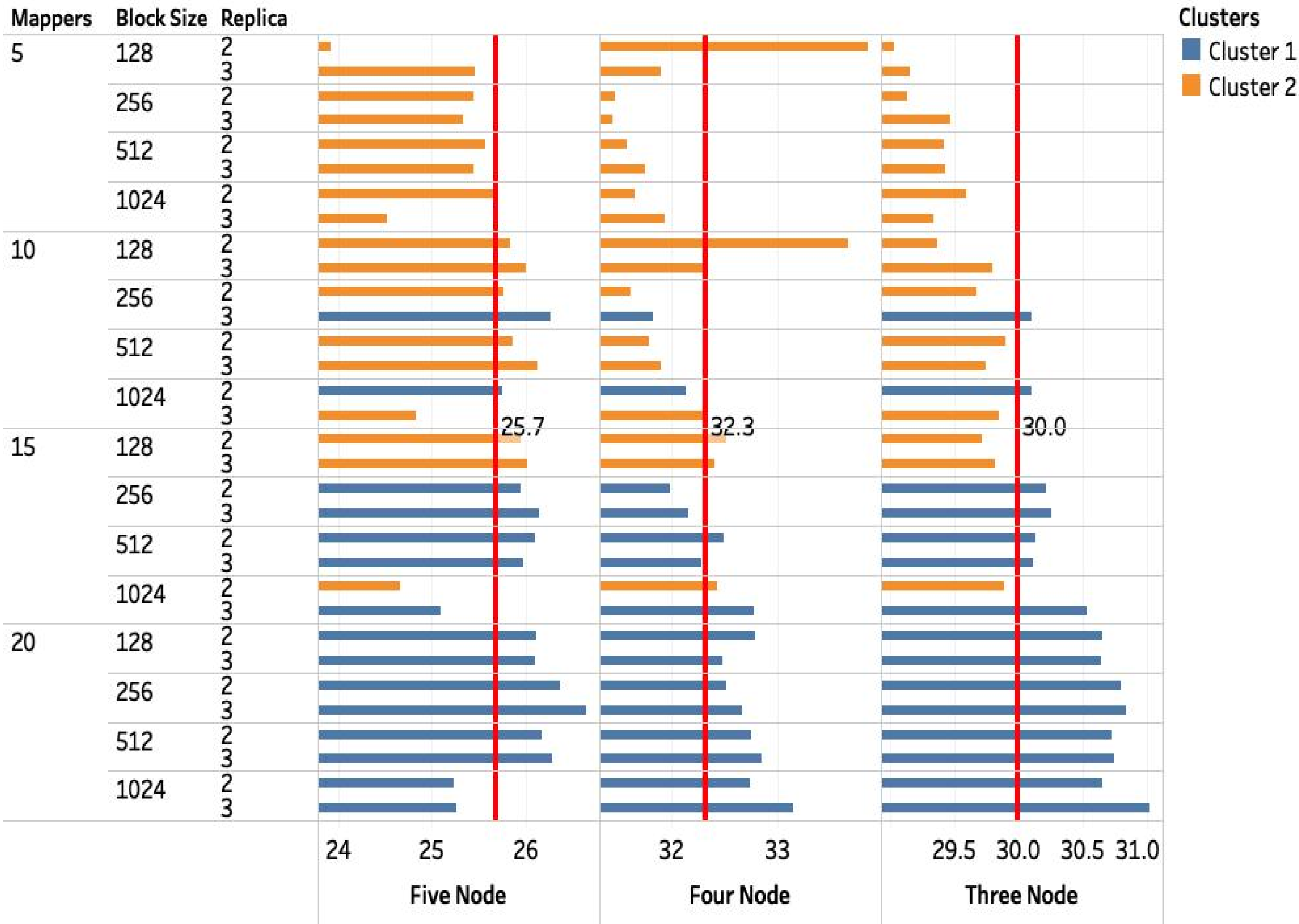

Figure 6.10: Result of k-mean clustering for memory usage for cluster of different sizes.

The *Pearson correlation* coefficient showed that number of nodes and block size are negatively correlated with memory usage while number of mappers and number of replicas are positively correlated with memory usage. The number of nodes were moderately associated with memory usage while number of mappers showed weak association. Both number of replicas and block size showed very weak association.

Network usage

Model predicted 86% of the network utilization correctly for five node cluster. For three node cluster model predicted 100% of the network usage correctly.

Figure D.13 showed the mean network usage for three cluster sizes. When we increased the number of nodes from *four* to five there was a decrease in network traffic by 15.09%. when we decreased the number of nodes from *four* to three, there was an increase in overall mean network traffic by 19.15%. If we compare the change in network usage for increasing number of nodes from three to five, the overall network traffic decreased by 31.36%.

Figure D.14 showed the standard deviation of the network usage for five, *four* nodes and three nodes cluster. Results showed a gradual increase in overall standard deviation of network usage when we reduced the cluster size from five to three. There was a percentage increase of 19.60%.

Figure D.15 showed the boxplot of the results for network usage. Number of replicas seem to have a distinguishing effect over the increase in network traffic when we reduced the cluster size from five to three nodes. Another interesting observation in all three cluster sizes was that when we used 3 number of replicas instead of 2 number of replicas, the network traffic was greater than the overall mean network traffic.

Figure 6.11 showed the k-means clustering of network usage data for the five node, *four* nodes and three nodes Hadoop clusters. The two clusters received equal number of instances. The results showed that less number of replicas always resulted in mean network traffic less than overall mean network usage while the number of replicas with value 3 resulted in mean network usage higher than the overall mean network usage. For both clusters, three node Hadoop cluster showed the highest mean network traffic, followed by *four* nodes and five nodes. The between-group sum of squares was 9 times greater than within-group sum of squares. This shows that the two cluster centers were not close to the overall mean and their boundaries were very distinct. The individual values of each cluster were very close to each other.

The *F-statistics* values showed that *four* nodes Hadoop cluster was the highest distinguishing variable for clustering followed by three node and five node

Hadoop clusters. The *p-values* also showed that the difference between the means of two clusters is highly significant for the three experimental setups.

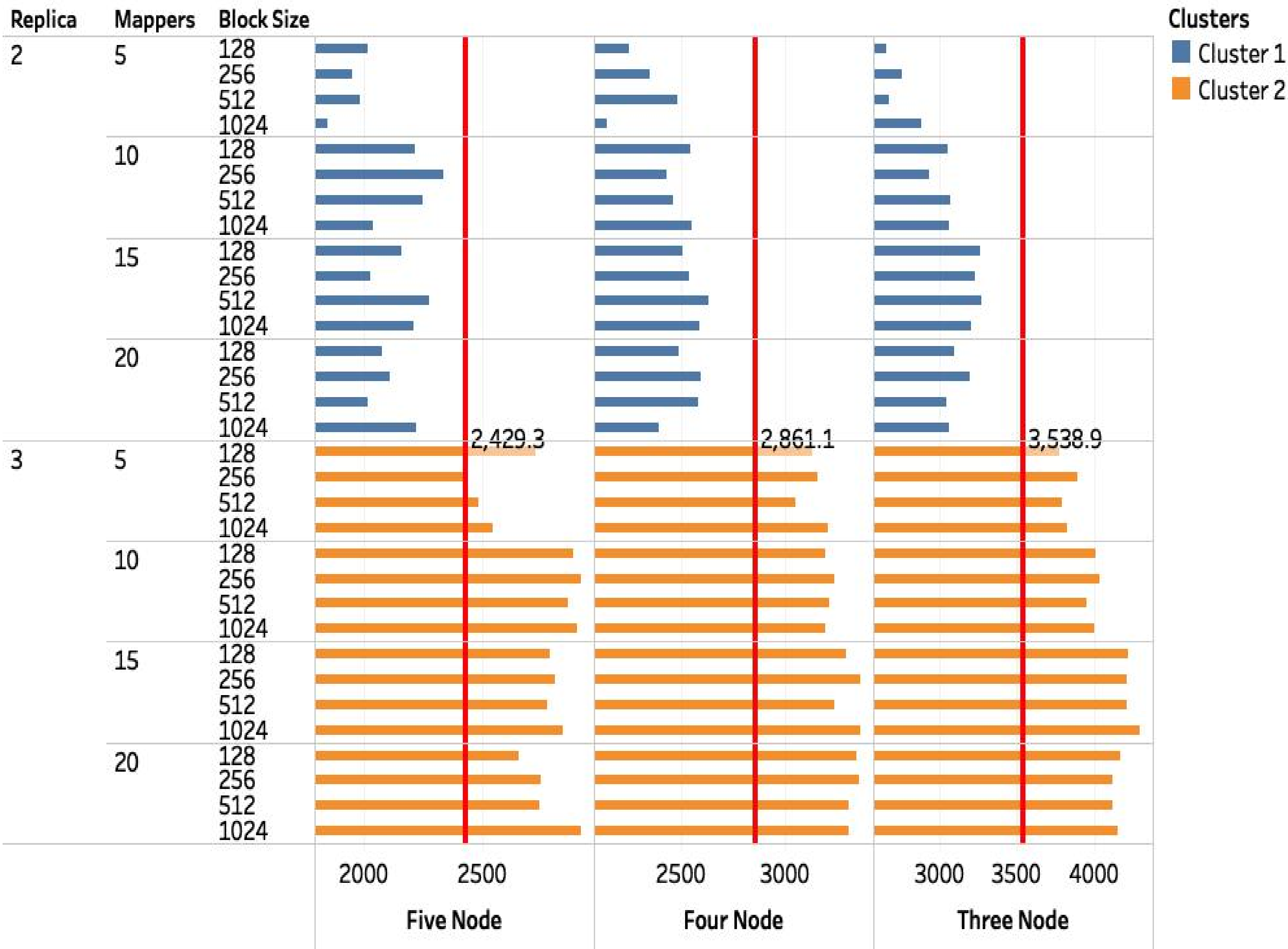

Figure 6.11: Result of k-mean clustering for network usage for cluster of different sizes.

The *correlation analysis,* by using regression M5P tree for network usage data of different cluster sizes is shown in Figure D.16. The outcome showed 18 linear models with the correlation coefficient of 0.88. The results showed that number of replicas and number of nodes were highly correlated with network usage. Linear models 2, 13, 14 and 15 covered higher number of observations and lower RMSE value as compared to the rest of the models. The individual linear models are shown in Section D.4.

The *Pearson correlation* coefficient for network usage showed that number of nodes was negatively correlated with network usage while block size , number of replicas and number of mappers were positively correlated with network usage. The number of nodes in the cluster showed a strong association with network usage while number of replicas showed moderately strong association. Number of mappers showed weak relationship with network usage while block size showed very weak association.

Execution time

For both three node and five node clusters, the model built on the basis of *four* nodes cluster was able to predict 100% of the outcome correctly. All the rules were able to generalize the resource utilization.

Figure D.17 showed that adding or removing nodes in existing *four* nodes cluster can result an increase in the overall execution time of the job. Increasing a node in the cluster resulted in 5.9% increase in the overall mean execution time of Hadoop job. Similarly decreasing a node from *four* nodes to three nodes also resulted in 3.2% increase in the overall mean execution time of job. The results showed that number of replicas and number of mappers had the most distinguishing effect on the mean execution time of the job. For five node cluster, there was a gradual decrease in mean execution time with the increasing number of mappers while keeping the number of replicas constant as shown in Figure D.19. For *four* nodes cluster, the mean execution time increases gradually with the increase in number of mappers. But further downsizing the cluster to three nodes resulted in increasing the mean execution time as the number of mappers increased from 5 to 15. But when the number of mappers further goes above 15 the mean execution time dropped below overall execution time.

Figure D.18 showed the standard deviation of the execution time of cluster of five nodes, *four* nodes and three nodes. There was a gradual increase in the overall standard deviation of execution time. Downsizing the cluster from three nodes to five nodes resulted in 13.51% decrease in the overall standard deviation in execution time.

Figure 6.12 showed the output of the k-means clustering performed for execution times of the three Hadoop cluster setups. The cluster 1 received more than 65% of the instances while cluster 2 received less than 35% of the instances. For cluster 1, five node Hadoop cluster showed the highest mean execution time. For cluster 2 five node cluster also showed the highest mean execution time but it was much closer to three node cluster. The between-group sum of squares is 25% less than withing-group sum of squares. This suggests that the difference between the clusters means and overall mean was less and clusters may not have very distinct boundaries.

The *F-statistics* results showed that five node cluster was the highest distinguishing variable for clustering followed by three node and *four* nodes clusters. The *p-value* results also showed that the difference between the means of the two clusters was statistically significant.

The *correlation analysis* by using regression tree, for execution time of Hadoop jobs on different clusters, is shown in Figure D.20. Four linear regression models have been predicted by the tree with a correlation coefficient of 0.45. Number of nodes was the most important attributed associated with execution times. The linear models were quoted in Section D.5 and LM 1 and LM2 covered 66.67% of the observations with a low RMSE.

The *Pearson correlation* coefficient for the overall execution time of same job over Hadoop cluster with different number of nodes showed that number of nodes had positive correlation with execution time while all other attributes showed negative correlation. The analysis of strength of relationship between attributes and execution time showed that number of nodes was moderately associated with execution time while number of replicas, number of mappers and block size were very weakly associated with execution time.

Summary

From the results above we can summarize the following:

- The results showed that changing the size of cluster could greatly impact

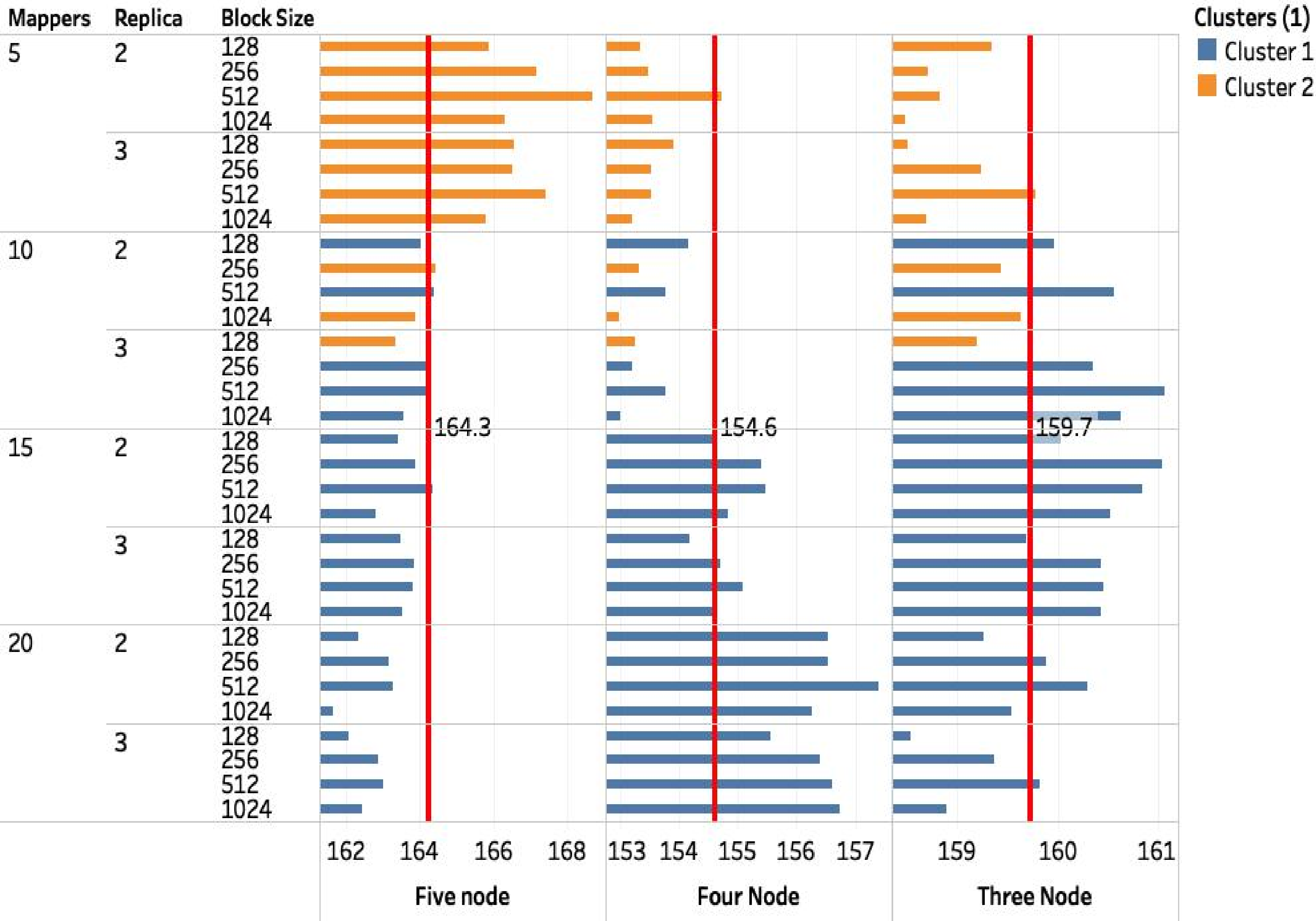

Figure 6.12: Result of k-mean clustering for execution time for cluster of different sizes.

resource utilization and overall execution time of Hadoop jobs. Number of mappers can greatly impact overall CPU usage, memory usage of Hadoop cluster and execution times of the Hadoop job. While different values of replica can affect disk usage and network usage of the cluster. Block size value had the highest impact over disk usage of Hadoop cluster. These results are mostly aligned with our previous findings. The clus-

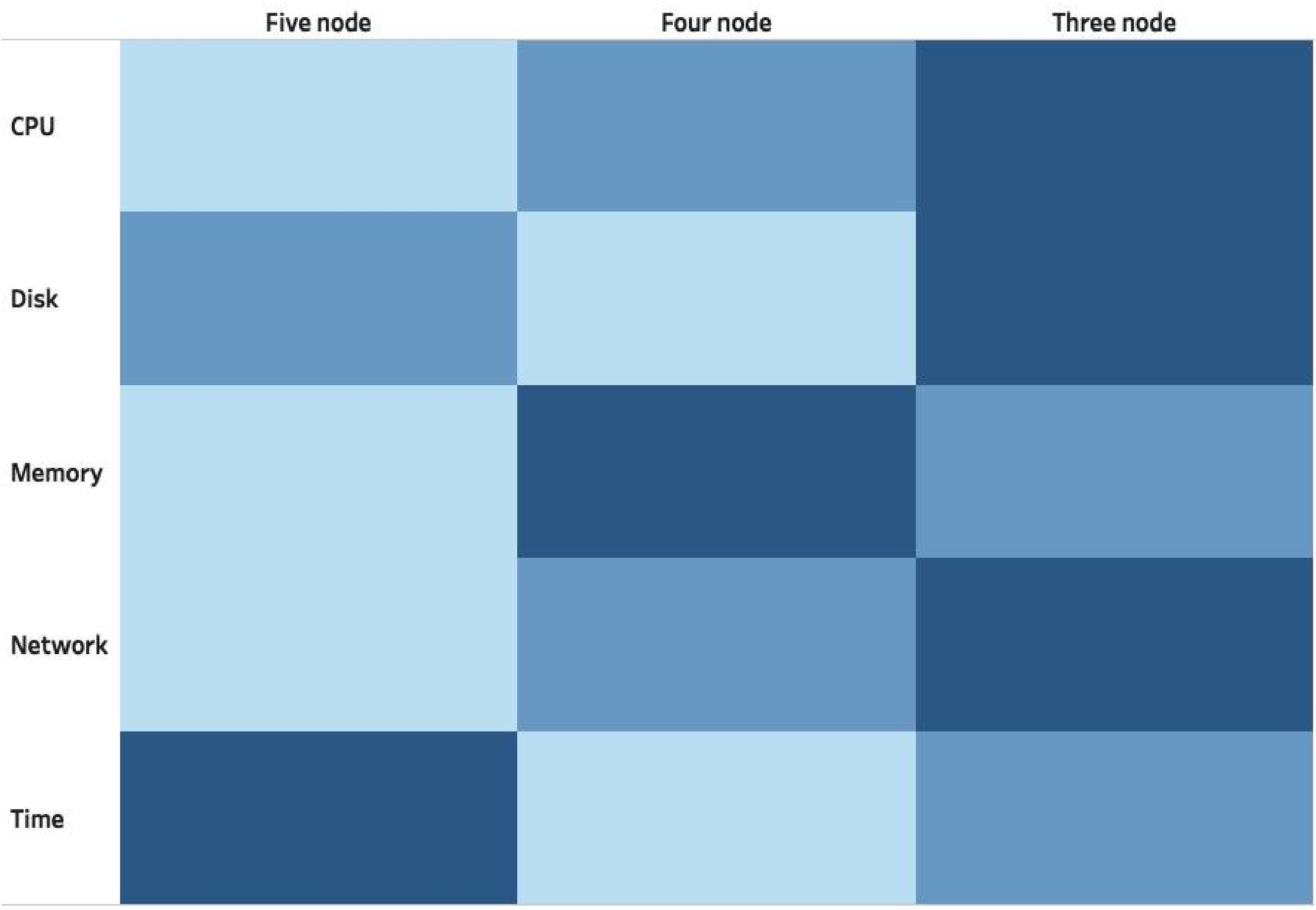

Figure 6.13: Heat map representation of the overall performance of different performance measures

tering results also showed very distinct performance differences between the clusters of different sizes.

- We analyzed the statistical significance of these results by applying *z-test*. We used *z-test* because the number of observations were greater than 30. For all of our performance measures for example CPU usage, disk usage, memory usage, network usage and execution time, the change was significant between cluster of all sizes. This significance test was run to record the change between five nodes to *four* nodes, *four* nodes to three nodes and five nodes to three nodes.

- The correlation analysis, Pearson's correlation coefficient and M5P tree, was carried out for the experiments run on cluster size of five nodes, *four* nodes and three nodes. The results showed that number of nodes in the cluster was the attribute that was strongly correlated with the utilization of different resources and execution time. In case of system resources like CPU usage, disk usage, memory usage, and network usage the correlation was negative which demonstrated that the cluster with less number of nodes showed higher resources usage while cluster with high number of nodes had lower resources utilization.

- Figure 6.13 showed the heat map representation of our performance measure in cluster of five nodes, *four* nodes and three nodes. The lighter color represented the lowest among three and the darkest color represents the highest of all. The results showed that lower cluster sizes always result in higher execution time, CPU usage, disk usage, memory usage and network usage, for example, the three node cluster in current set of experiments. Thus, we concluded that when same job run over Hadoop clusters with different number of nodes, it impacted the overall resource utilization. Among other attributes, number of mappers and replica showed moderate to weak association with our performance measures.

6.4.3 Impact of change in infrastructure

This section presented and discussed the results of impact of change in infrastructure on overall performance of the Hadoop job from dedicated single user to virtual. These experiments were run on AWS EC2 [220] as to evaluate the generalizability of the method. The overall execution time of the jobs had been recorded for different Hadoop configuration parameters using two different queries.

Figure E.1 showed the impact of configuration parameters and number of order by columns on mean execution time. Mean execution time was always higher than the overall mean execution time when the number of columns in order by clause were *three*. While when the number of columns in order by clause was *two*, the mean execution time was always less than the overall mean execution time. This mean execution time gradually decreased as the number of mappers decreased. The change was more prominent when the number of columns in order by clause were *three*.

Figure E.2 showed the impact of number of mappers and number of replicas on mean execution time. Increasing the number of mappers results in gradual decrease in mean execution time. A similar pattern can be observed when we plotted the mean execution times while considering the number of mappers and block size Figure E.3. Figure 6.14 shows that no of order by columns played a vital role, higher number of columns in `Order by` clause always resulted in higher mean execution times. The standard deviations plotted in Figure E.4 also showed a significant difference when the number of columns was *three*. The boxplot shown in Figure E.5, suggests that mean and median execution times of jobs was greater than the overall mean execution time when the number of order by columns was higher and number of mappers ware less.

Figure E.6 and Figure E.7 showed the impact of mean and standard deviation for block size, number of mappers and number of replicas on overall mean execution time. The results showed that when the value of block size was 256MB or less, the mean and standard deviation of the overall execution time was significantly less. Similar pattern was highlighted in Figure E.8 and Figure E.9, where the impact of block size higher than 256Mb was prominent.

Figure E.10 showed the difference in execution times for different Hadoop job attributes between local cluster and AWS cluster. The overall mean execution time for the same job on AWS cluster was 1.5 times higher than running the same job on our *four* nodes local cluster.

Figure E.11 showed the output of the k-means clustering performed on execution times of AWS-based Hadoop cluster. Cluster group 1 received 92.18% of the instances while cluster group 2 only received only 7.8% of the instances. The population standard deviation of cluster group 1 was 6.57 while the population standard deviation of the cluster group 2 was 36.58. So the standard deviation of cluster group 2 was 5.56% higher than cluster group 1. The between-group sum of square was 6.7% higher than than within-group sum of squares. This suggested that both of the clusters groups have distinct boundaries and the p-value also shows that the difference between both of the clusters was statisti-

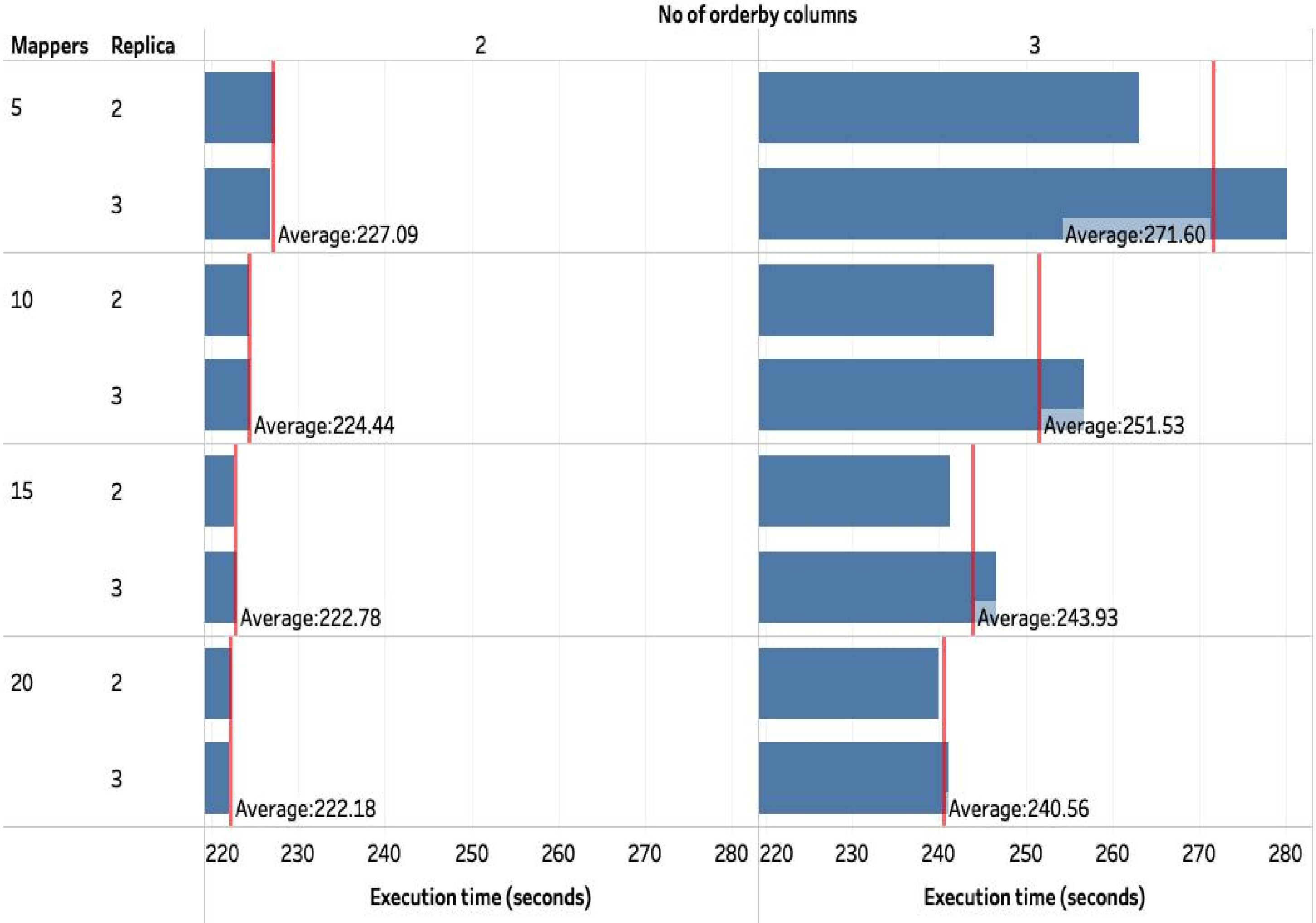

Figure 6.14: Mean execution times showing impact of number of mappers, number of replicas and columns in order by clause on AWS cluster

cally significant. The results suggested that higher columns in order by clause, higher block size and very few number of mappers could result in higher execution times, thus resulting in similar behavior.

The decision tree model in Figure E.12, showed that number of columns in order by clause was the most important feature contributing towards higher execution time. The greater the number of columns, leads to higher execution time. Other important attributes include number of mappers and block size. Another important observation was that if the value of block size is less than

256 and the number of mappers is less than or equal to 5, the execution time remained high. These results were not aligned with our findings in our previous decision tree model as shown in Figure B.3. This showed that the change in infrastructure significantly affects the overall performance of the jobs.

The M5P regression tree for the execution time of Hadoop jobs on AWS cluster was shown in Figure E.13. Five linear regression models were predicted by the tree with a correlation coefficient of 0.75. Number of orderby columns was the most important attribute contributing toward the overall execution time. Other important attributes were number of mappers and Block size. The linear models were quoted in E.1 and both LM2 and LM5 covered 50% of the observations with RMSE lower than other models.

The *Pearson correlation* coefficient showed that the number of order by columns, block size and number of replicas were positively correlated with execution time while number of mappers were negatively correlated with overall execution time of job on AWS cluster. The analysis of strength of relationship showed that number of order by columns was strongly correlated with execution time while number of mappers and block size are moderately correlated with execution time. Number of replicas shown to be weakly related to overall execution time.

6.5 Chapter summary

From the discussion in this chapter, we summarized the following:

- The results from our experiments showed that the change in experiment setup, like dataset, or change in cluster environment, like cluster size, has significantly affected Hadoop cluster resource utilization and overall execution time. Different values of Hadoop parameters and other Hadoop job attributes like size and content of dataset and number of nodes in the cluster are significantly important. These Hadoop job attributes should be chosen carefully while designing and configuring a Hadoop cluster for a particular job as it can greatly impact the overall performance of the cluster resources.

- Our results also highlighted the importance of infrastructure. Same experiments run on different infrastructure like AWS EC2 can impact the overall performance of Hadoop job significantly. So, it will be appropriate to also consider both hardware and software infrastructures while modeling and predicting resources utilization and performance of Hadoop jobs. It also demonstrated the application of our technique to model and predict performance and understand key factors.

- The results also demonstrated that the decision tree-based modeling of the Hadoop jobs was very effective. The results showed that the models have very good predictive ability. These prediction achieved good true positive percentages for completely unseen data, resulting from significant changes in the Hadoop job and cluster environment. Therefore, if we incorporate different attributes of environmental changes like cluster size, infrastructure into the model building process, it could make models more robust.

- The results were analysed for correlations among different attributes of Hadoop jobs and performance measures. The valuable insight into the significantly contributing attributes was demonstrated and verified by using different descriptive, statistical and regression modeling techniques. Most of the time, the findings of each technique were aligned with the findings of our decision tree prediction. So, while optimizing such system it is better to use more than one technique. This is because, firstly, no one technique was efficient enough to understand the complex interaction among different contributing factors towards resource and performance modeling of Hadoop jobs, secondly, using multiple techniques can validate the prediction of decision tree-based models.

Chapter 7

Conclusions and Future Work

This thesis proposed the method to optimize the performance of a Hadoop cluster. This included the development of robust methodology to measure the performance of our system by incorporating configuration parameters, dataset structure and cluster size. This also included the use of machine learning based technique to build model and predict resource utilization of a Hadoop job. This thesis provided insight into commonly discussed configuration parameters of Hadoop such as block size, replication factor and number of mappers. We used different values of each of these parameters to see their impact on overall performance of CPU, disk and memory. We also analyzed the impact of changing the values of these parameters on network usage and overall execution time of job. Setting up the correct value of these parameters require experience as we have to consider the available system resources. We designed experiments not only to analyze the impact of different values of configuration parameters but also the query given. We used Hive queries, which is a data warehouse system built on top of Hadoop to mine structured data. Our results showed that apart from changing the parameters, changing the query, for same dataset, also affects the overall performance of the system resources.

We analyzed results by using different statistical techniques in order to model the behavior of Hadoop job in the context of system resources and execution time. But we find it difficult to draw the underlying patterns, correlations and associations. In order to automate the process of modelling the relationship between variety of parameters and query structures, we used machine learning approach. We used decision trees to model and predict the resource utilization behavior of different Hadoop jobs. The trees showed robustness and good generalizability when we used them to predict the resource utilization and execution time of the jobs running with diverse datasets. Decision trees were also able to predict the resource utilization behavior of clusters of different sizes.

While decision trees were used to predict the resource utilization when using a particular parameter value or query structure. This thesis further investigated to see the relationship or correlation between factors (parameters, query structure etc.) affecting Hadoop job. We used M5P tree algorithm, a classification and regression tree, to see the functional relationship between these factors. These Hadoop job attributes were also correlated by using Pearson's correlation coefficient. We performed k-mean clustering to see job characteristics showing similar resource utilization and execution time behavior. This thesis provided a comprehensive and robust method to model and predict the resource usage, for complex interaction of different characteristics of Hadoop job, which can be very helpful for system designers or administrators.

We investigated our research goal as described in Chapter 1. The overall aim is to investigate the relationship between Hadoop job attributes and resource utilization. We used different machine learning techniques such as decision trees, M5P trees and k-mean clustering. Section 7.1 of this chapter highlight the main conclusions of the thesis and Section 7.2 outline the possible future work in this area. This chapter conclude by providing a summary of the research.

7.1 Conclusions

This section highlights the contributions from different chapters of this thesis.

Robust experimental methodology for Hadoop performance characterisation. Hazard and Operability analysis (HAZOP) was applied to identify potential factors that can hinder or affect the working of hardware, running of experiments and collection of measurements. This process improved the internal validity of our experiments. By applying HAZOP, we not only minimized the risk of biased measurements but also saved time of looking for cause of problem whenever we encountered some deviations from either normal working of cluster or running of jobs on cluster. This practice supported our experimental methodology and made sure that are reproducible and can be easily replicated by anyone in the future. This also highlighted the important hazards and their possible mitigations while designing Hadoop-based experiments.

Characterisation of the impact of Hadoop job attributes for the utilization of different resources and execution time. Performance characterisation was done to see the factor affecting the performance of a system or framework. Hadoop has a complex ecosystem with many of inter-dependencies between its modules. Configuration parameters are defined in *XML* files to control the behavior of different modules, thus contributing to its fault-tolerance and robustness during the execution of job for example, the number of replicas and block size can affect the distribution of data on cluster nodes. There are are more than 200 configuration parameters in Hadoop which can be tuned to optimize the performance of this distributed framework. Tuning these parameters not only requires good understanding of its internal structure but also experience and skill. Also, these parameters can take more than one value, so tuning these parameters is a challenging task.

This thesis provided useful insight into important Hadoop configuration features and their impact on CPU usage, disk usage, memory usage, network usage and execution time. Our focus was not only to highlight the impact of different values of the commonly discussed parameters on system resources but also relationship between impact of query or input executed and dataset used. In order to make sure that our experimental results were repeatable and statistically valid, we run each set of experiment *30 times*.

While exploring the impact of changing one parameter value we kept other parameter values, cluster size, dataset and input constant. But this did not highlight the inter-related impact of all possible combinations of configuration parameters on system resources and execution time. Not only this, but also our experimental setups became more complicated as we used six different query structures for each 32 different combinations of configuration parameters. Therefore, we automated this process of parameter assignment and query assignment to a Hadoop job, and run a single experiment with all possible combination of configuration parameters. As each combination of parameter was run 30 times as mentioned before. There were two advantages of this approach; firstly the results were statistically verified to be significant or not, secondly if a job showed some abnormal behavior due to any hardware-based or software-based failures, the overall effect on our data analysis is being minimized as we easily detected any outlier in the data. This has not only made our methodology repeatable, reliable and robust but also allowed us to collect CPU usage, disk usage, memory usage, network usage and execution time data for all of these experiments.

Modeling and predicting resource utilization of Hadoop jobs by using decision trees. Once we collected resources utilization and execution time data for all possible scenarios including different configuration parameters and queries as part of our experiments to characterize the impact of change in Hadoop job attributes. Therefore, it was impossible to manually draw some conclusions or generic models as the feature space was huge. Therefore, we tried to find patterns by using boxplots and graphs or some descriptive statistics. We used machine learning to automate this process of data mining, data modeling and prediction. We used decision trees (DT) algorithm to achieve this task. DT models are very robust, easy to build and has the advantages of being able to handle variety of data and missing values. For each of the resource such as CPU usage, a tree-based model was built after discretizing the data into low and high resource usage. The model was then tested for its generalizability and predictability. Our results showed that the decision rules, extracted form models by traversing the trees from leaf nods to root nodes, can effectively predict or generalize over such situations like changing the dataset, number of nodes and change in infrastructure.

The decision tree models not only gave us good and generalizable models but they also helped us to identify and predict the important patterns in the data such as the attributes which affected the performance of Hadoop jobs and resource utilization the most. These features included number of mappers, number of columns selected, number of replicas and others.

Correlation of features affecting the Hadoop job performance. Decision trees were used to model and predict the behavior of different configuration parameters, query structure and performance measures. These models were robust in handling such diverse feature space. But can these models show how different job attributes are related? Can different experiments be grouped into similar category on the basis of their performance measure? To answer these questions this thesis demonstrated the use of two methods. Firstly, for different performance measures, to see the association or correlation between configuration parameters, query structure, dataset and number of nodes. We used M5P tree modeling technique which is a classification and regression tree. This predicted linear relationship between different characteristics of Hadoop job in a decision tree-like fashion. Separate linear regression models are predicted for each path from root to leaf node. We further analyzed the strength and direction of relationship for each performance measure by using Pearson's correlation coefficient.

Secondly, we used *k-means* clustering technique to see whether a particular set of experiments showed similar behavior. We used k=2 to categorize the experiments into two main groups. For some performance measures the two groups showed very distinct boundaries. The Hadoop job characteristics grouped into one cluster can be seen as closely related. The overall difference for both between-group and within-group sum-of-squares for clusters demonstrated the strength of correlation for attributes grouped together. The F-statistics further highlighted the experiment with greatest standard deviation to make clusters.

7.2 Future Work

This section lists the potential directions for future work related to the research in this thesis.

7.2.1 Including YARN parameters for performance optimization of Hadoop

YARN is an important part of Hadoop ecosystem which handles the resource scheduling of jobs. It also has a large number of configuration parameters. Also, there are different scheduling strategies which can impact the performance of Hadoop cluster. Also, the impact of these parameters in conjunction with different parameters of HDFS and MapReduce will be interesting. Modeling and prediction of performance of Hadoop jobs for such a huge parameter space along with other characteristics of Hadoop jobs will be a challenging task.

7.2.2 Modeling performance for more datasets and query elements

There are numerous other query structures which need to be explored such as different types of JOINS. Datasets also play important role in different performance measure. It will be interesting to characterize datasets and include them in modeling process.

7.2.3 Expanding discretization for more detailed analysis

Currently, we have discretized the performance of different resources into high or low. We have used median-based discretization in this study. Another possible future direction can be to discretize data into more classes or trying different criteria of discretization.

7.2.4 Extending to multi-class optimization problem

We focused on analyzing individual performance measure with two possible outcomes. So current modeling and prediction only comprises of binary classes. But it would also be interesting to see how we can combine these different measures into one dataset and try to model and predict performance. This will give rise to many different and diverse classes such as predicting a Hadoop job attribute which will give us high CPU usage, low disk usage, low memory usage, high network and high execution time.

7.3 Summary

In summary, this thesis has contributed to the field of optimization of Hadoop performance for different configuration parameters, query structures, different datasets and cluster sizes. We proposed: (a) a robust methodology to design experiments for such systems, (b) characterization of the impact of changing configuration parameters and job query on system resources and execution time, (c) a machine learning approach to model and predict performance of Hadoop, and (d) an in-depth analysis approach to analyze the relationship between different characteristics of Hadoop job optimization problem.

Appendix A

Characterization of Hadoop performance and resources utilization by configuration parameter tuning

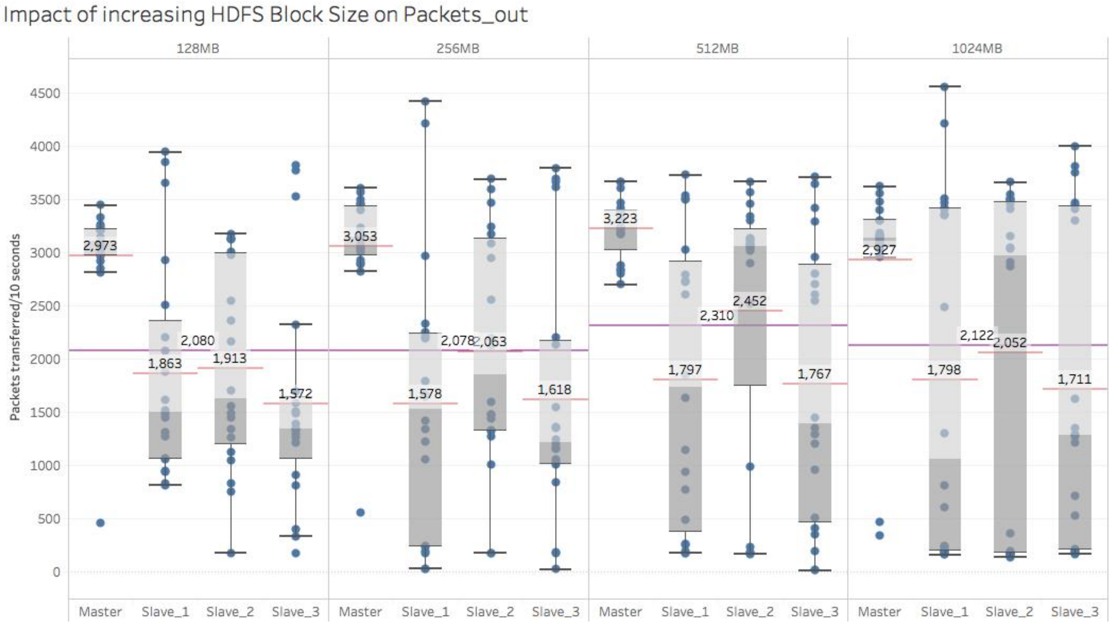

Figure A.1: Box plot and average packets_out data for 30 runs for different block sizes

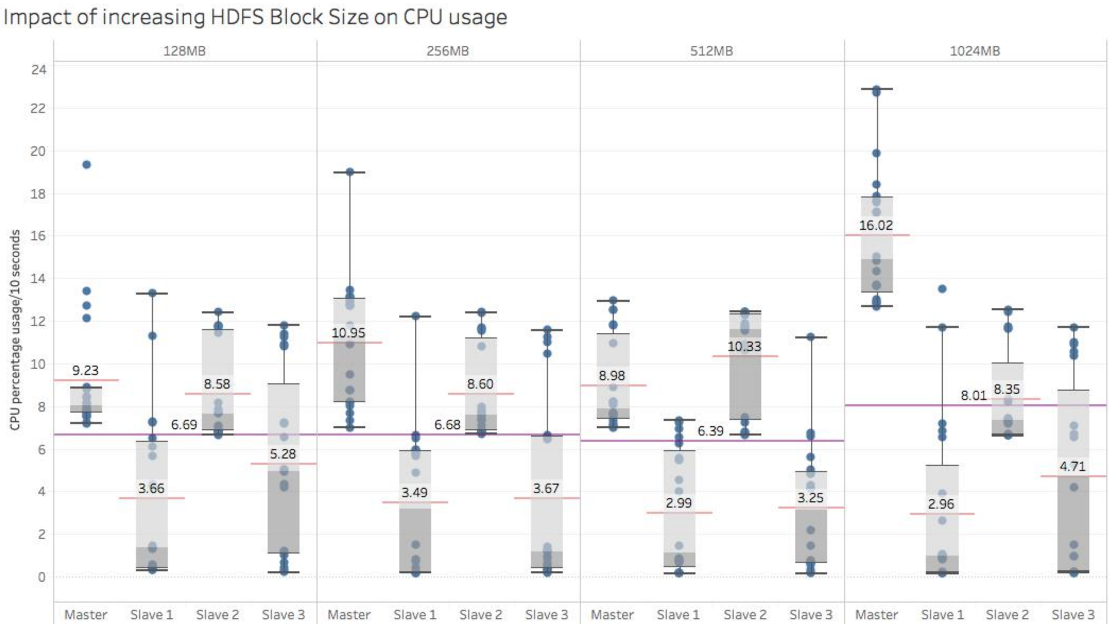

Figure A.2: Box plot and average CPU usage data for 30 runs for different block sizes

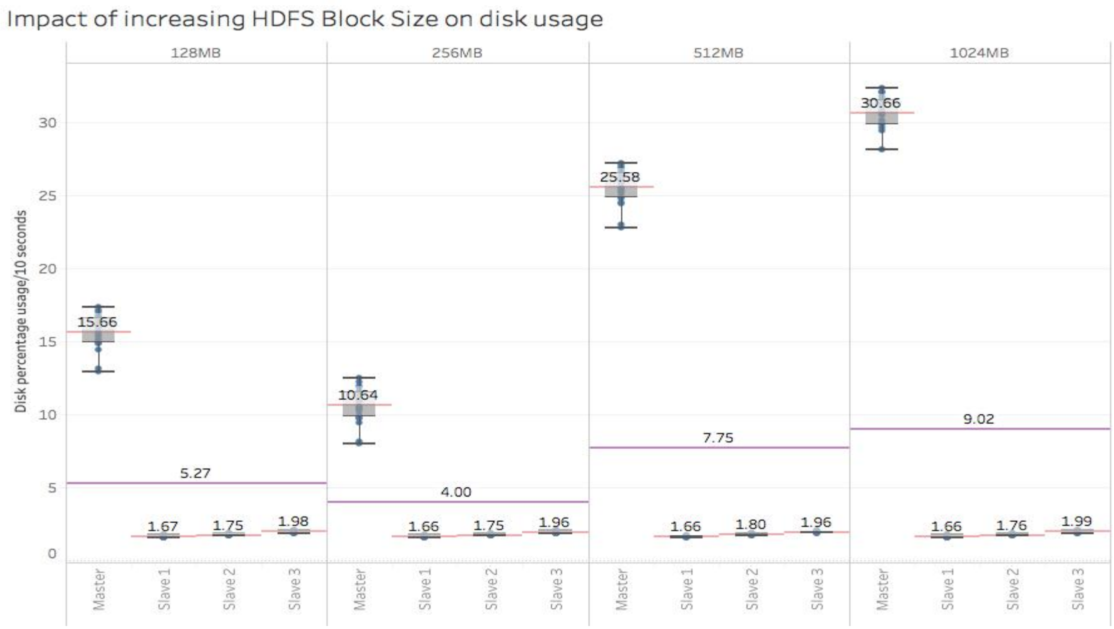

Figure A.3: Box plot and average disk usage data for 30 runs for different block sizes

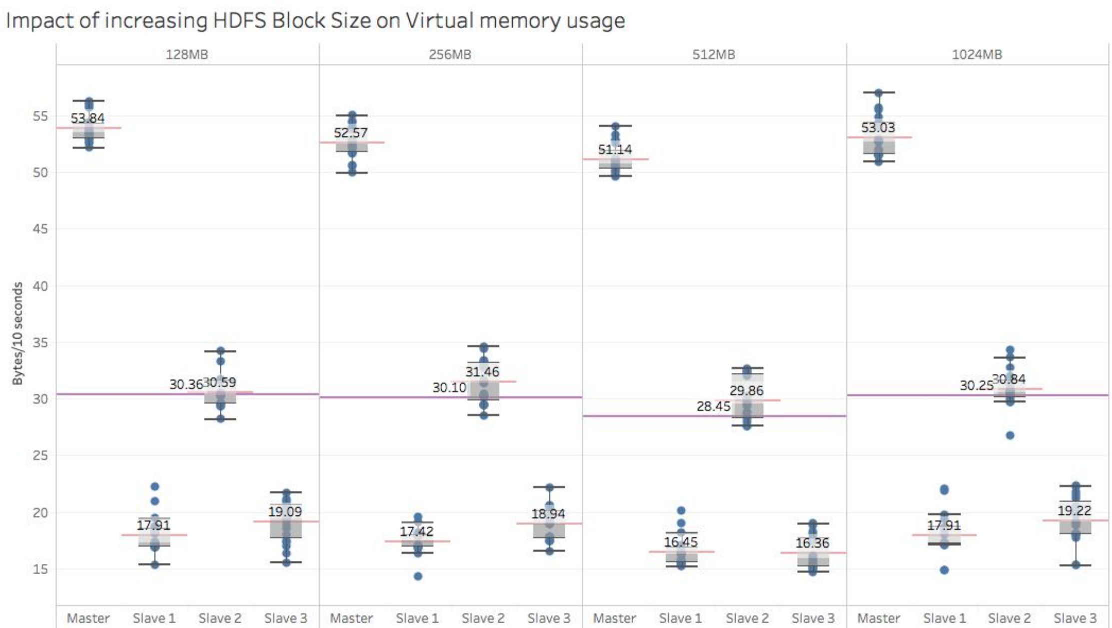

Figure A.4: Box plot and average virtual memory data for 30 runs for different block sizes

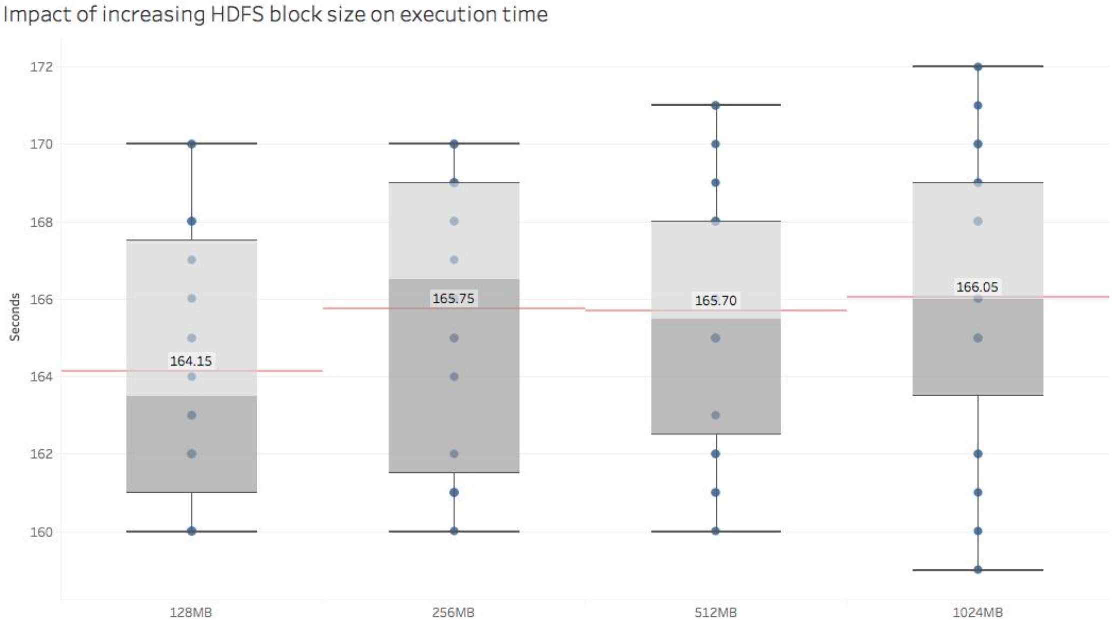

Figure A.5: Box plot and average execution for 30 runs for different block sizes

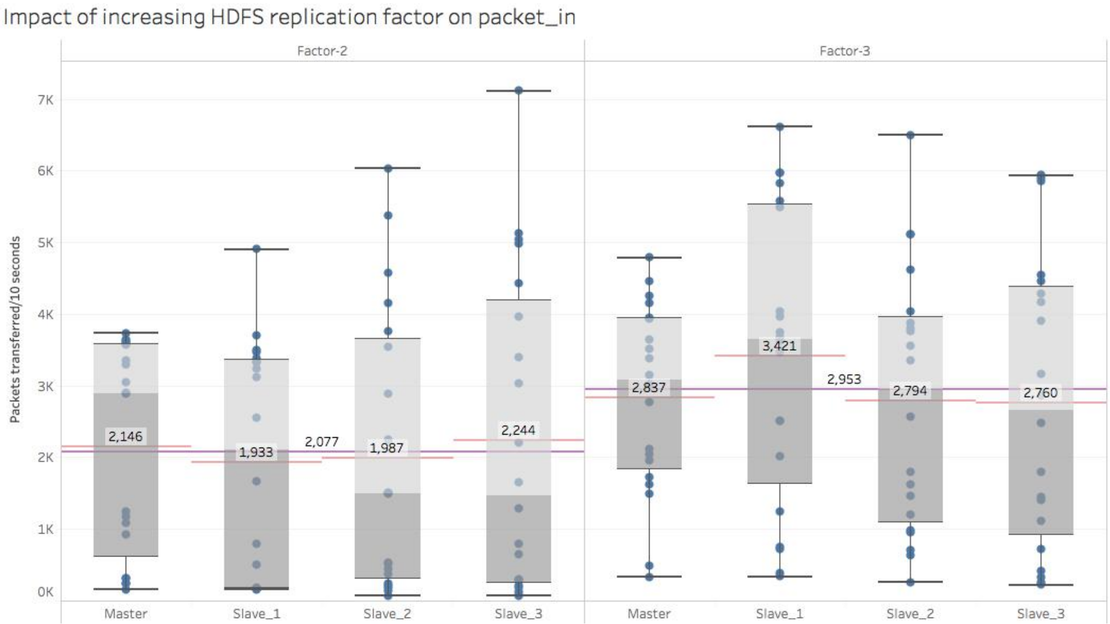

Figure A.6: Box plot and average packets_in data for 30 runs for different replication factors

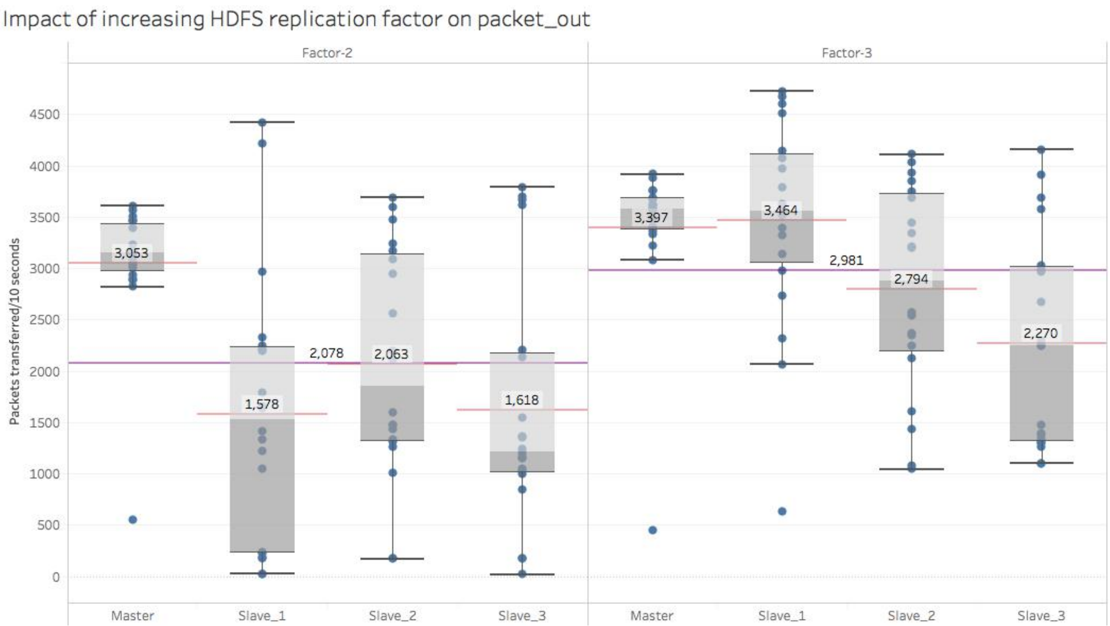

Figure A.7: Box plot and average packets_out data for 30 runs for different replication factors

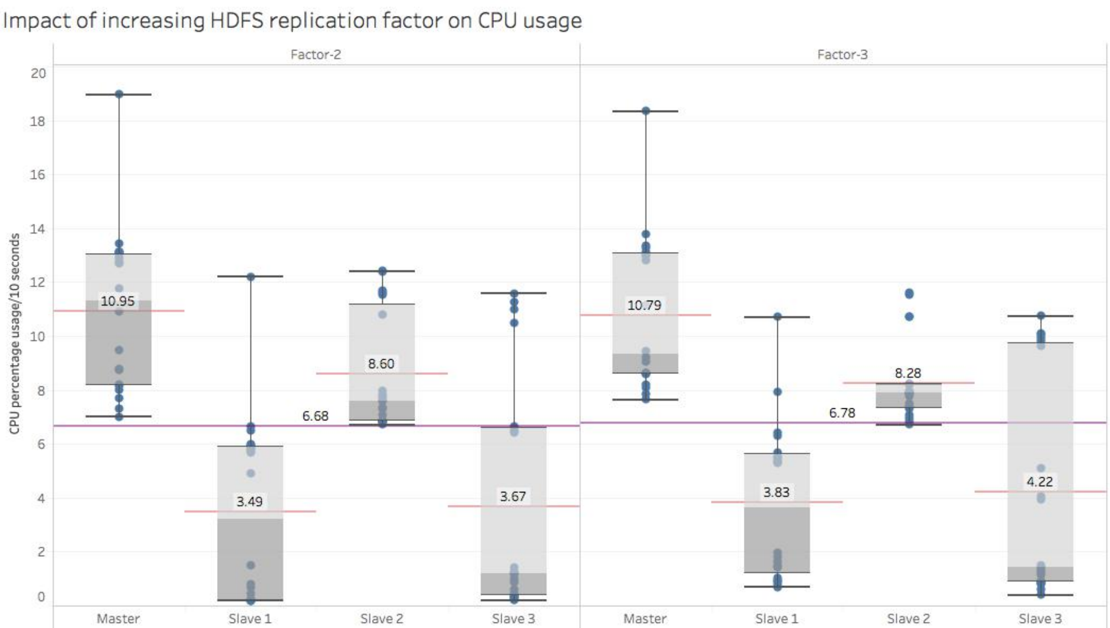

Figure A.8: Box plot and average CPU usage data for 30 runs for different replication factors

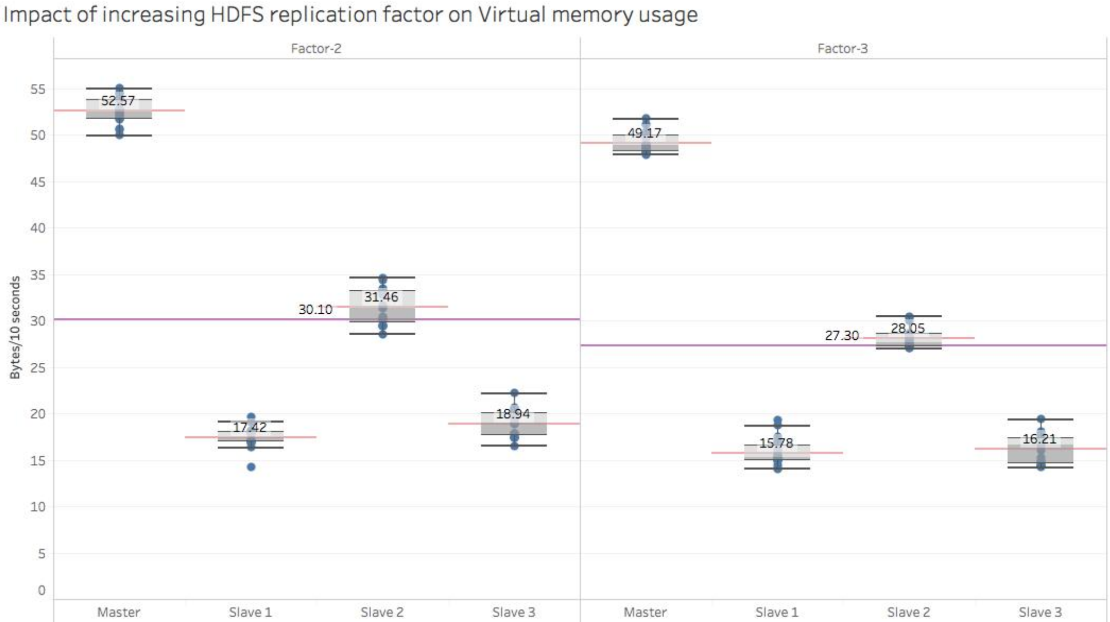

Figure A.9: Box plot and average virtual memory usage data for 30 runs for different replication factors

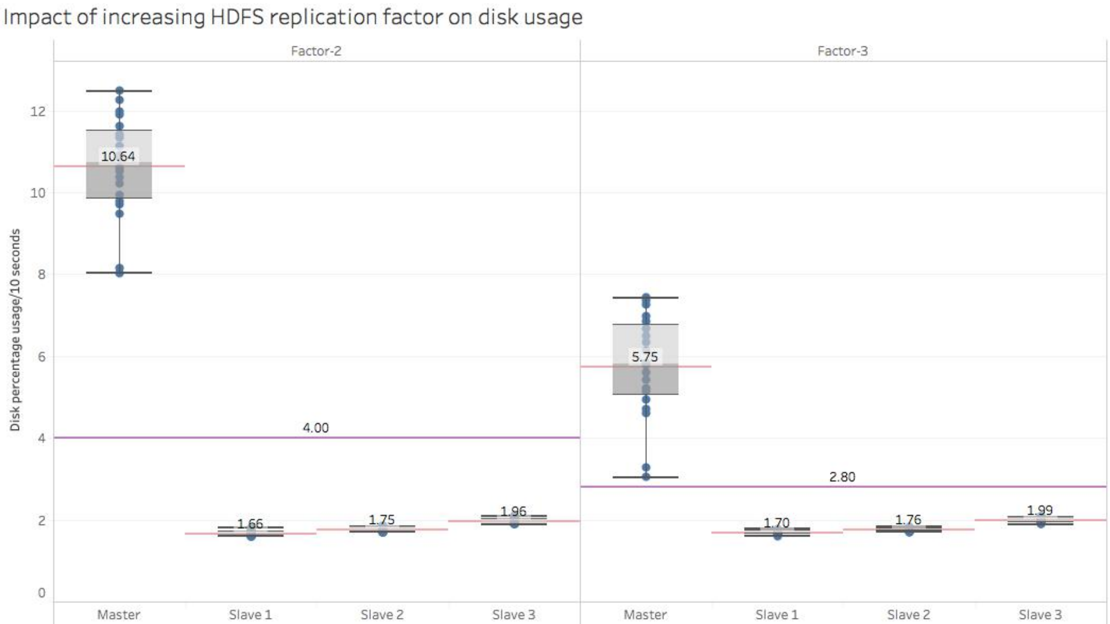

Figure A.10: Box plot and average disk usage data for 30 runs for different replication factors

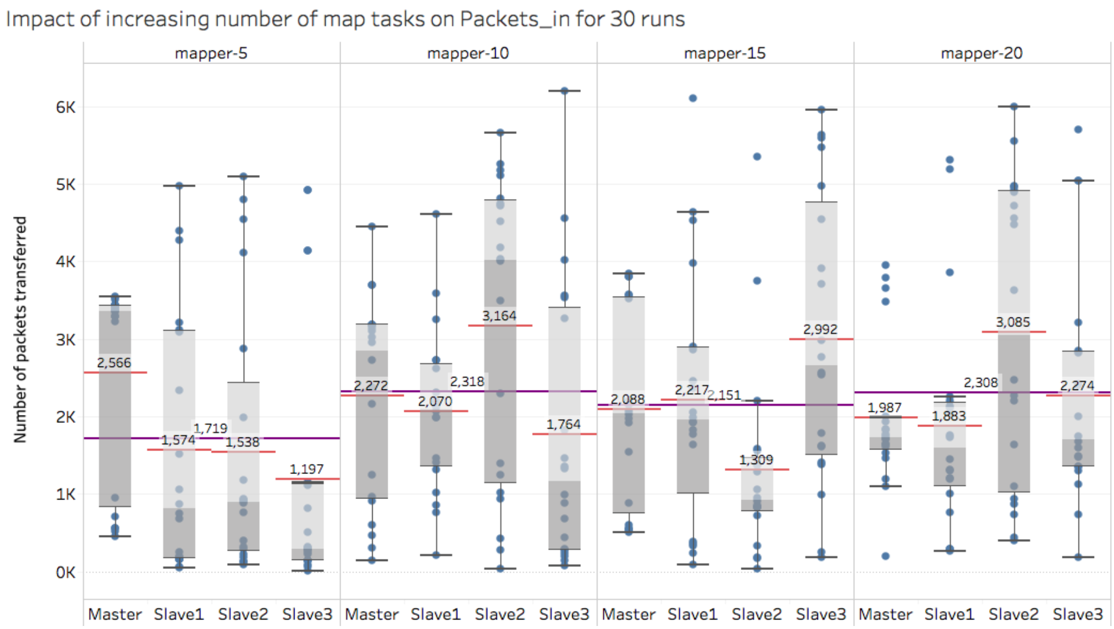

Figure A.11: Box plot and average packets_in data for 30 runs for different number of mappers

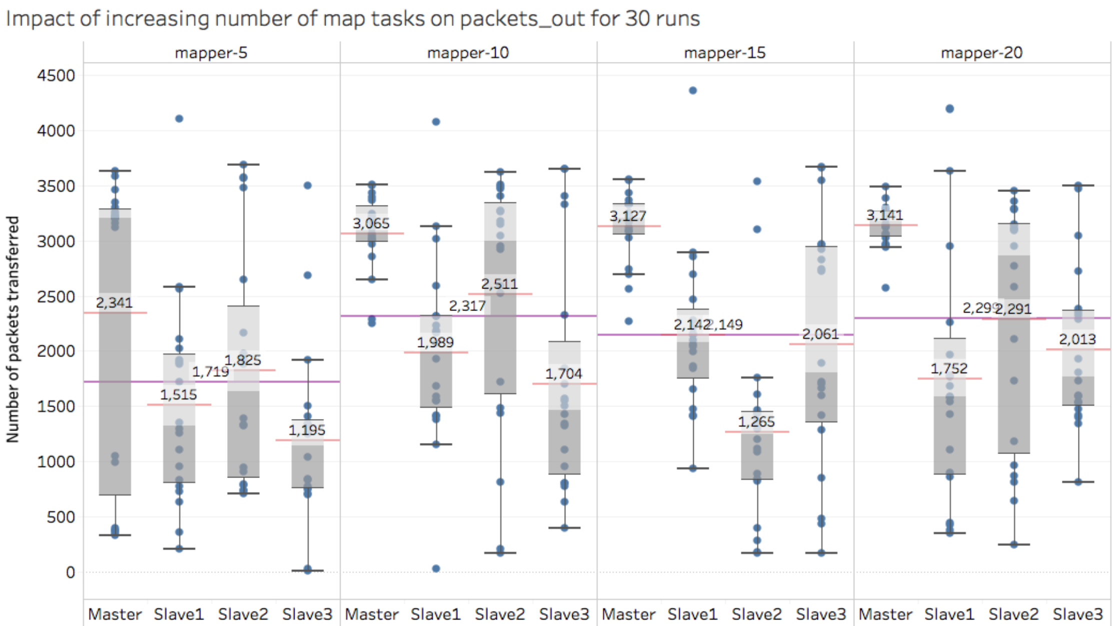

Figure A.12: Box plot and average packets_out data for 30 runs for different number of mappers

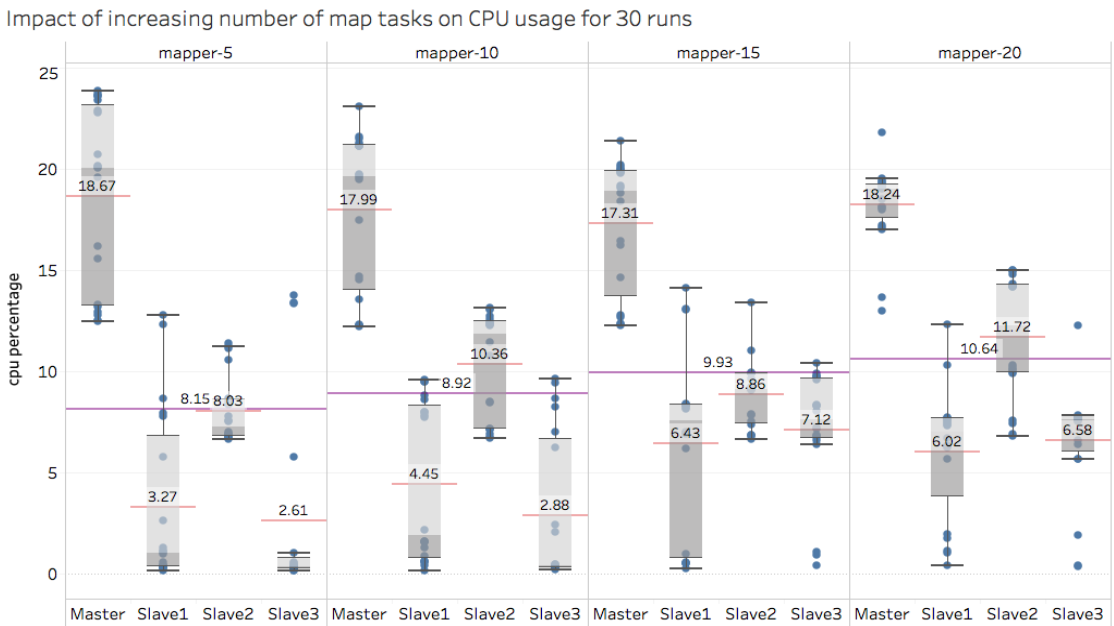

Figure A.13: Box plot and average CPU usage percentage data for 30 runs for different number of mappers

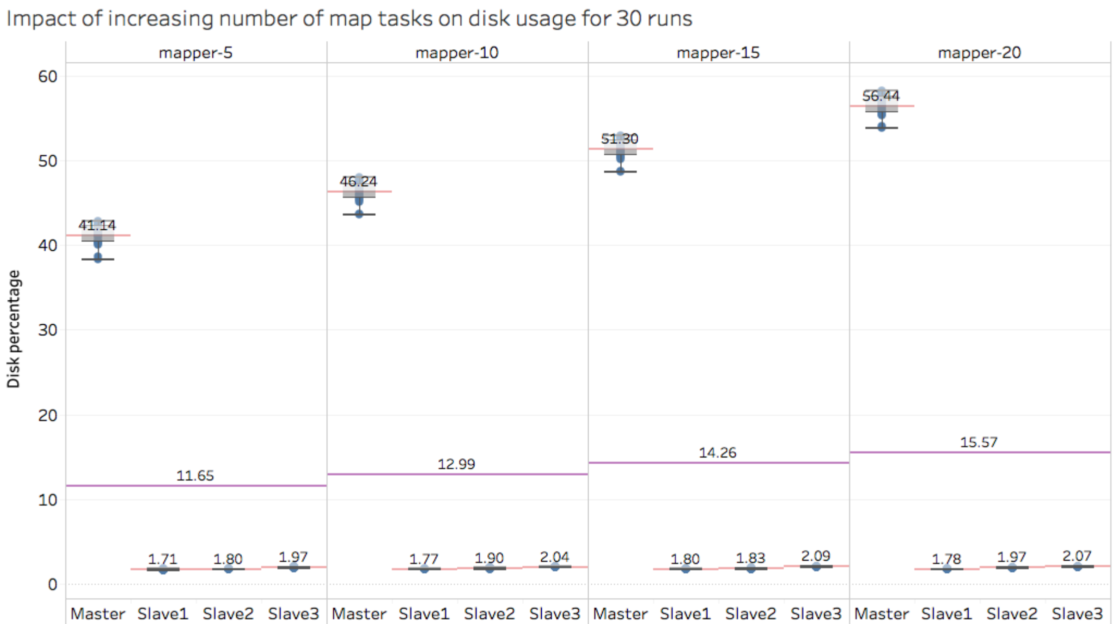

Figure A.14: Box plot and average disk usage percentage data for 30 runs for different number of mappers

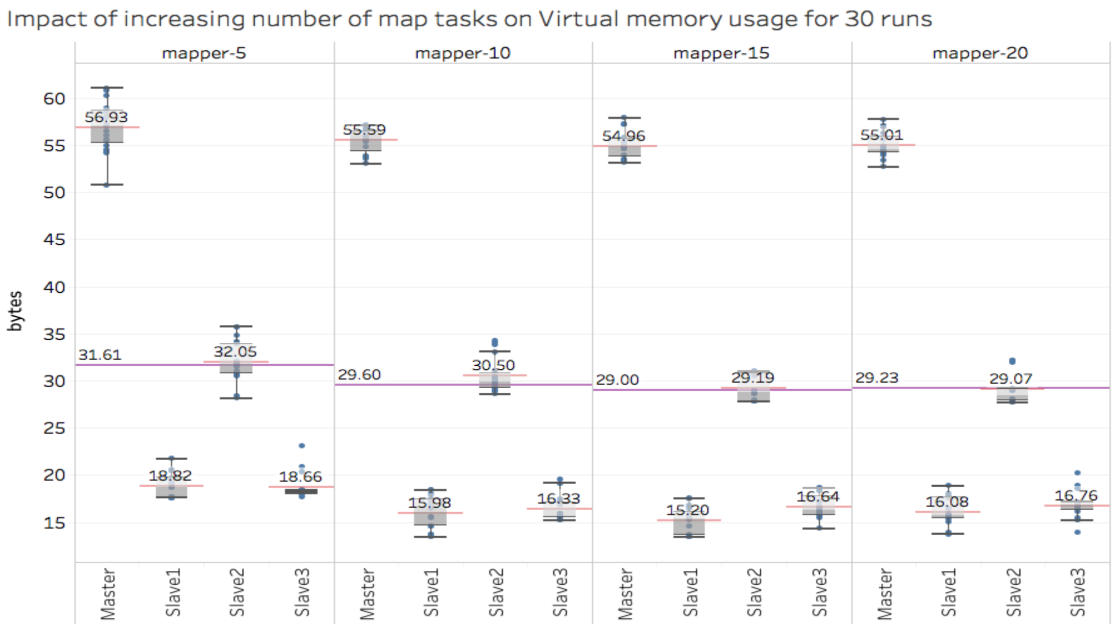

Figure A.15: Box plot and average virtual memory usage data for 30 runs for different number of mappers

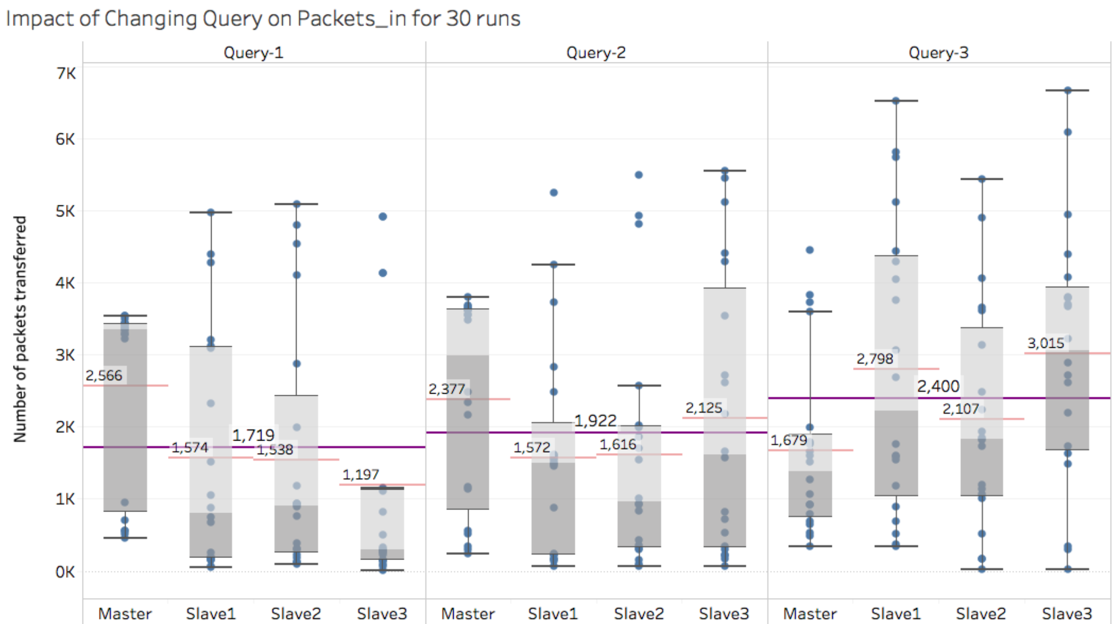

Figure A.16: Box plot and average for packets_in data for 30 runs for different queries

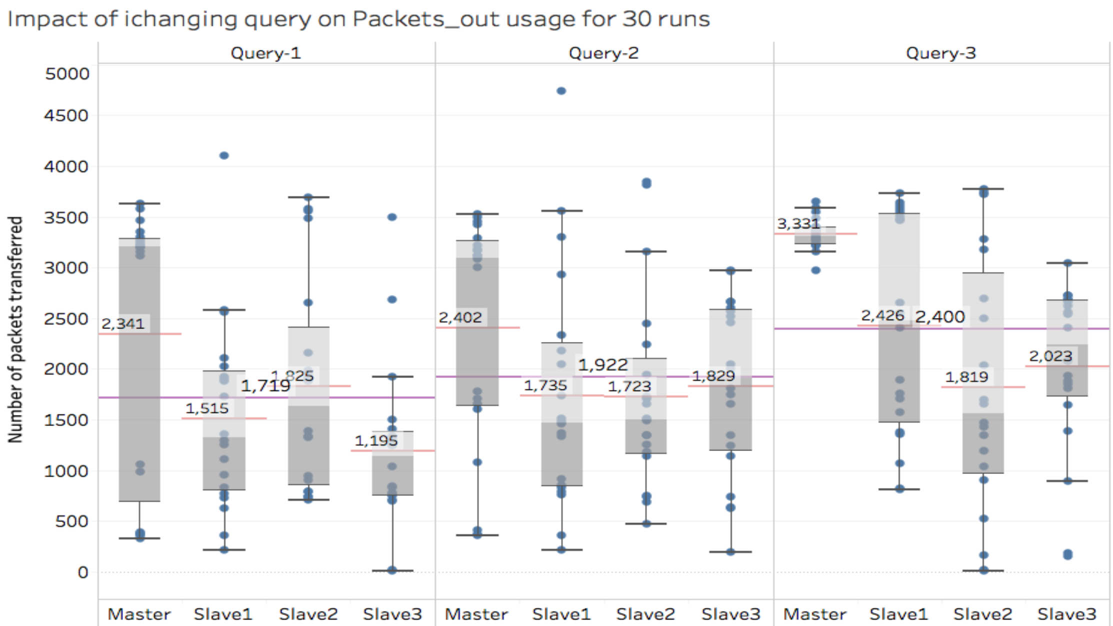

Figure A.17: Box plot and average packets_out data for 30 runs for different queries

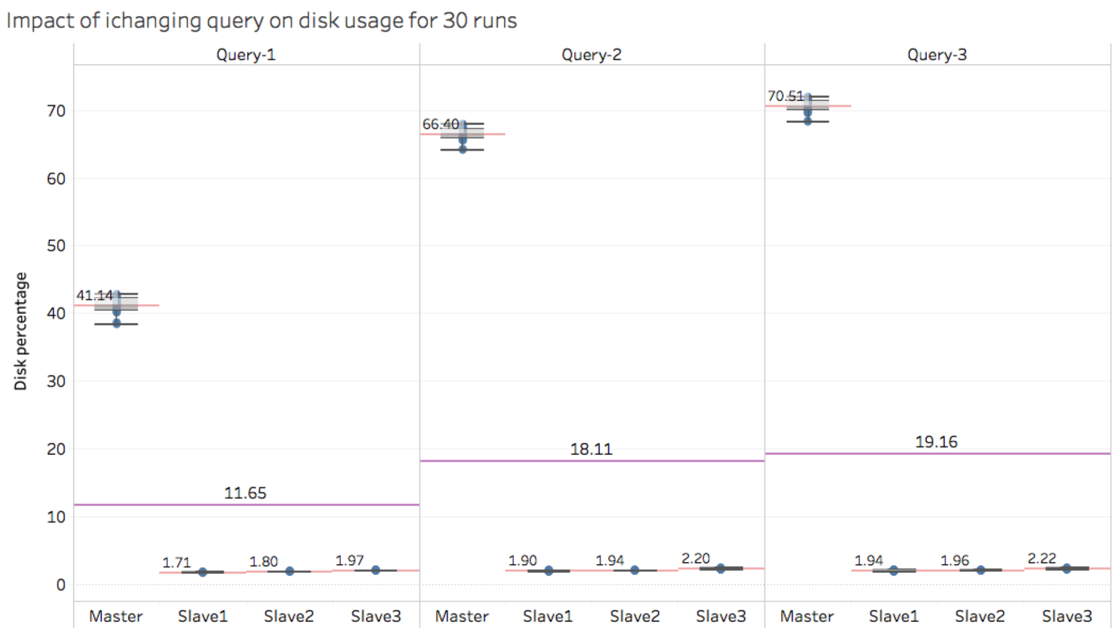

Figure A.18: Box plot and average disk usage percentage data for 30 runs for different queries

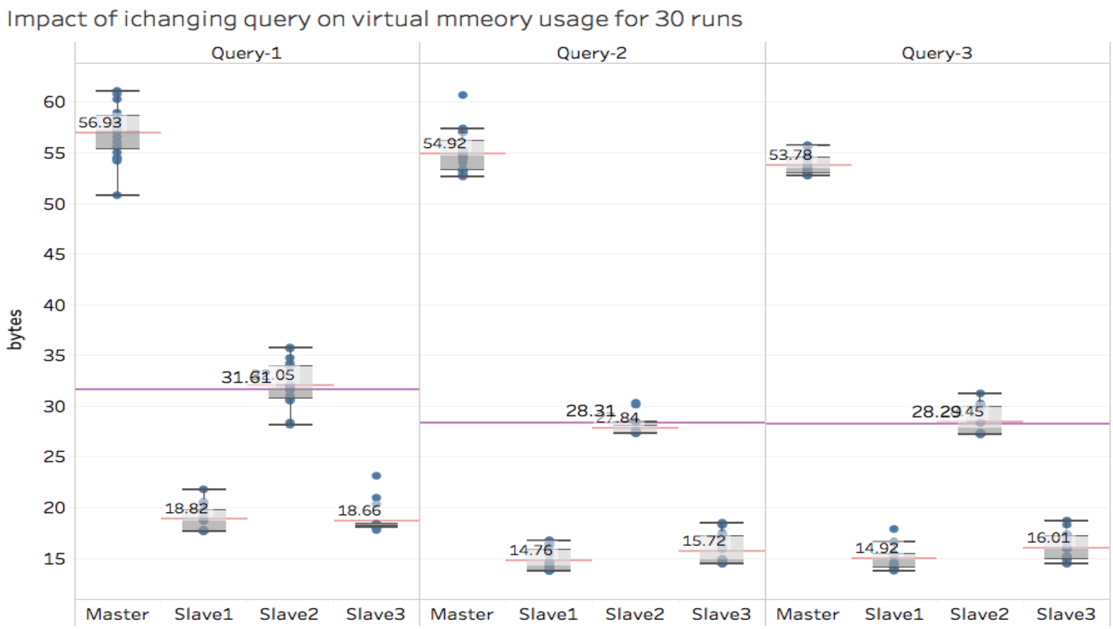

Figure A.19: Box plot and average virtual memory usage data for 30 runs for different queries

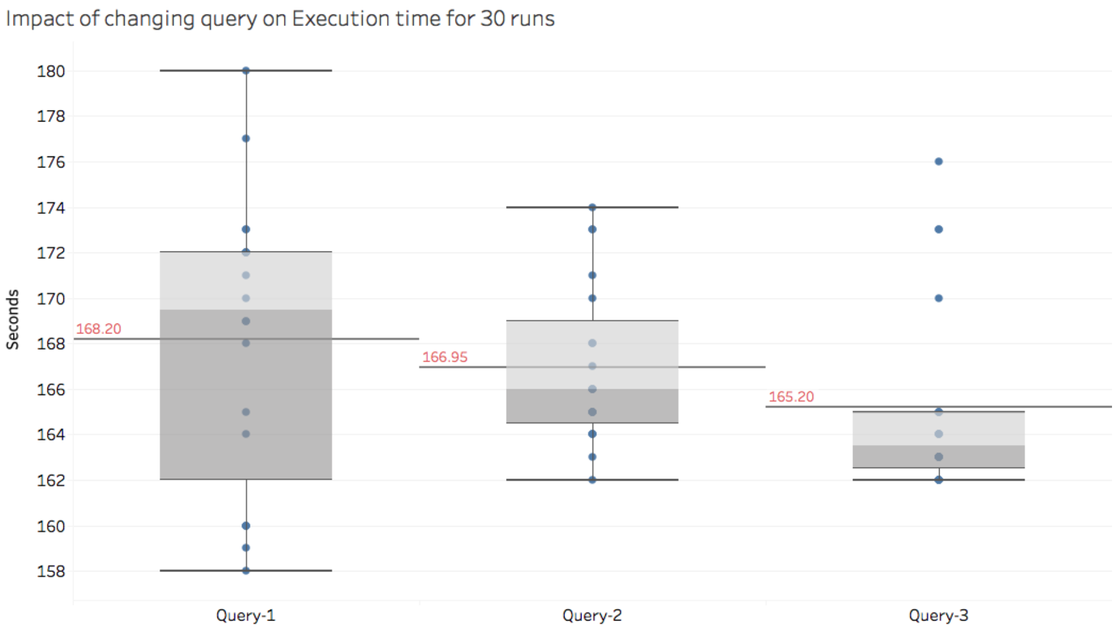

Figure A.20: Box plot and average execution time for 30 runs for different queries

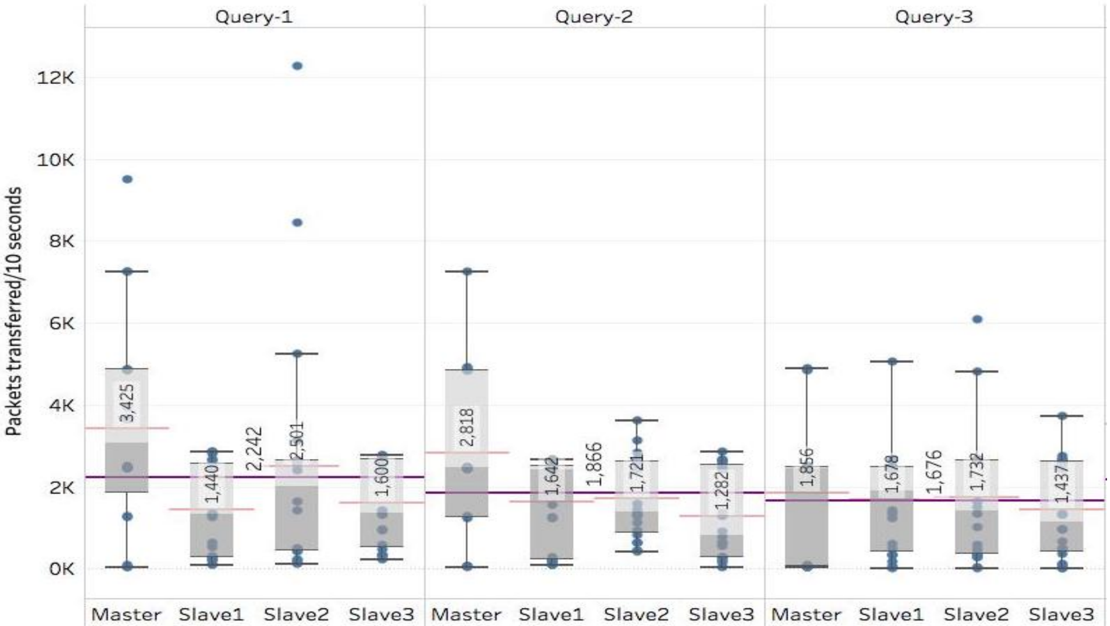

Figure A.21: Box plot and average for packets_in data for 30 runs for different aggregate functions in queries

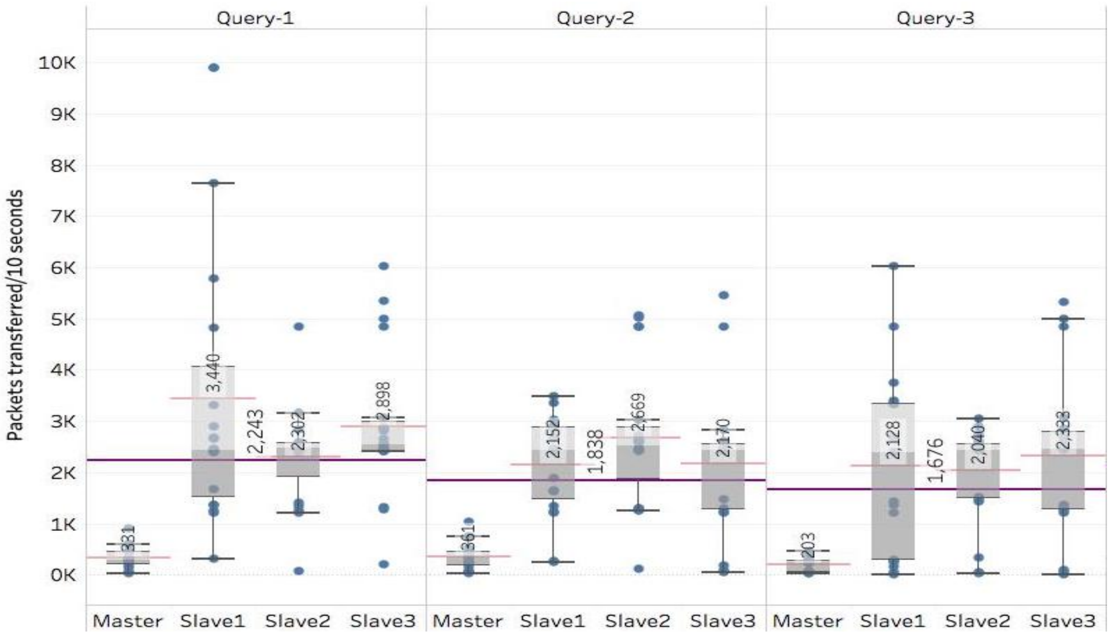

Figure A.22: Box plot and average packets_out data for 30 runs for different aggregate functions in queries

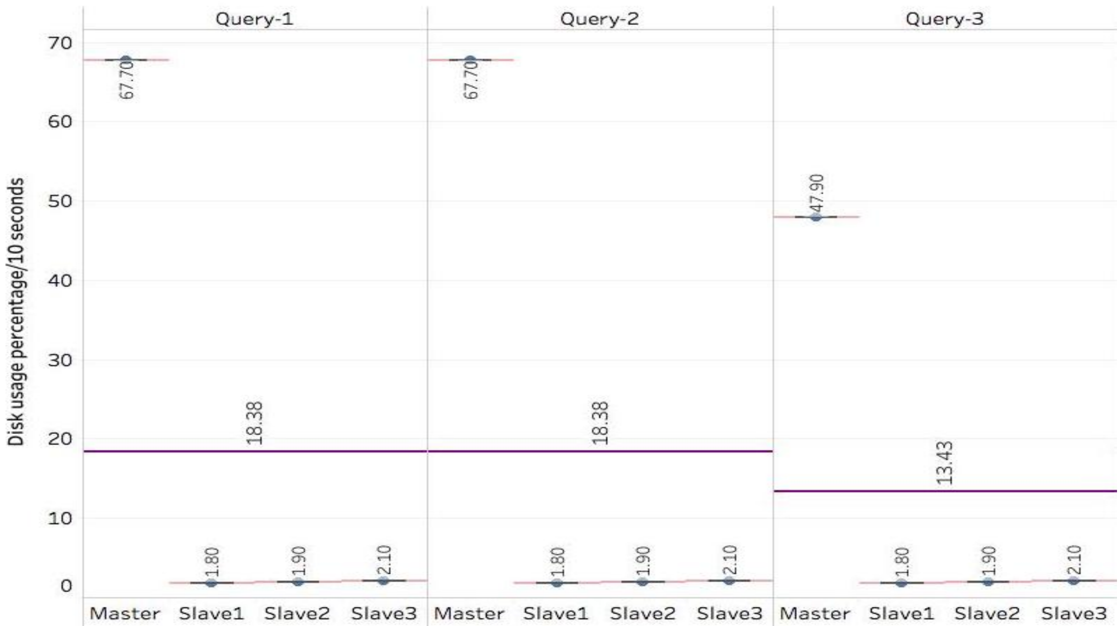

Figure A.23: Box plot and average disk usage percentage data for 30 runs for different aggregate functions in queries

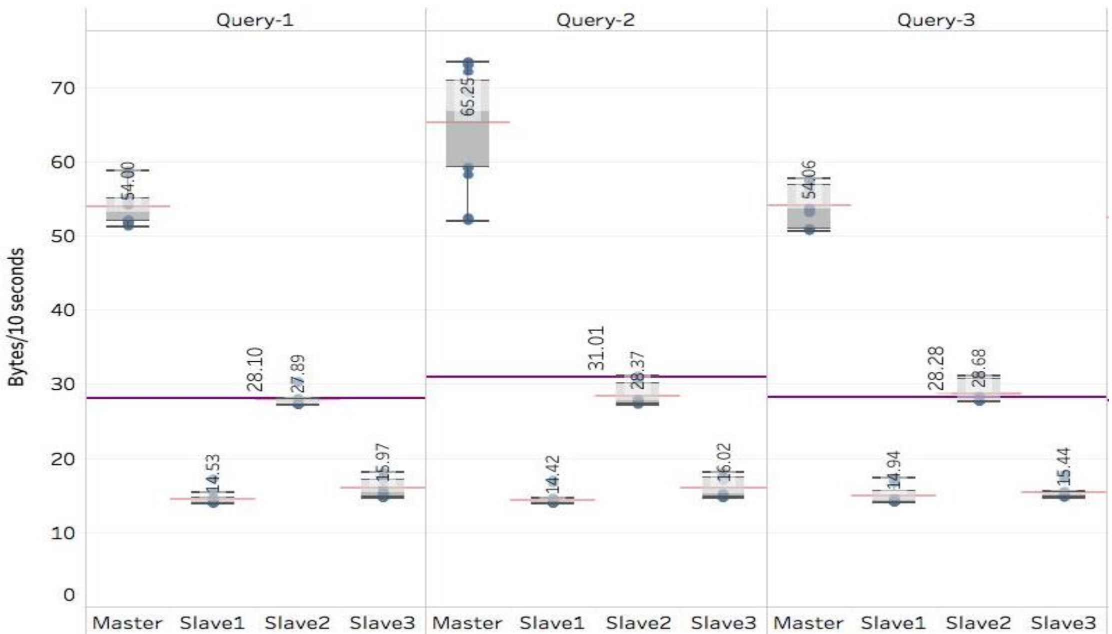

Figure A.24: Box plot and average virtual memory usage data for 30 runs for different aggregate functions in queries

Appendix B

Modelling and prediction of Hadoop cluster using Machine learning

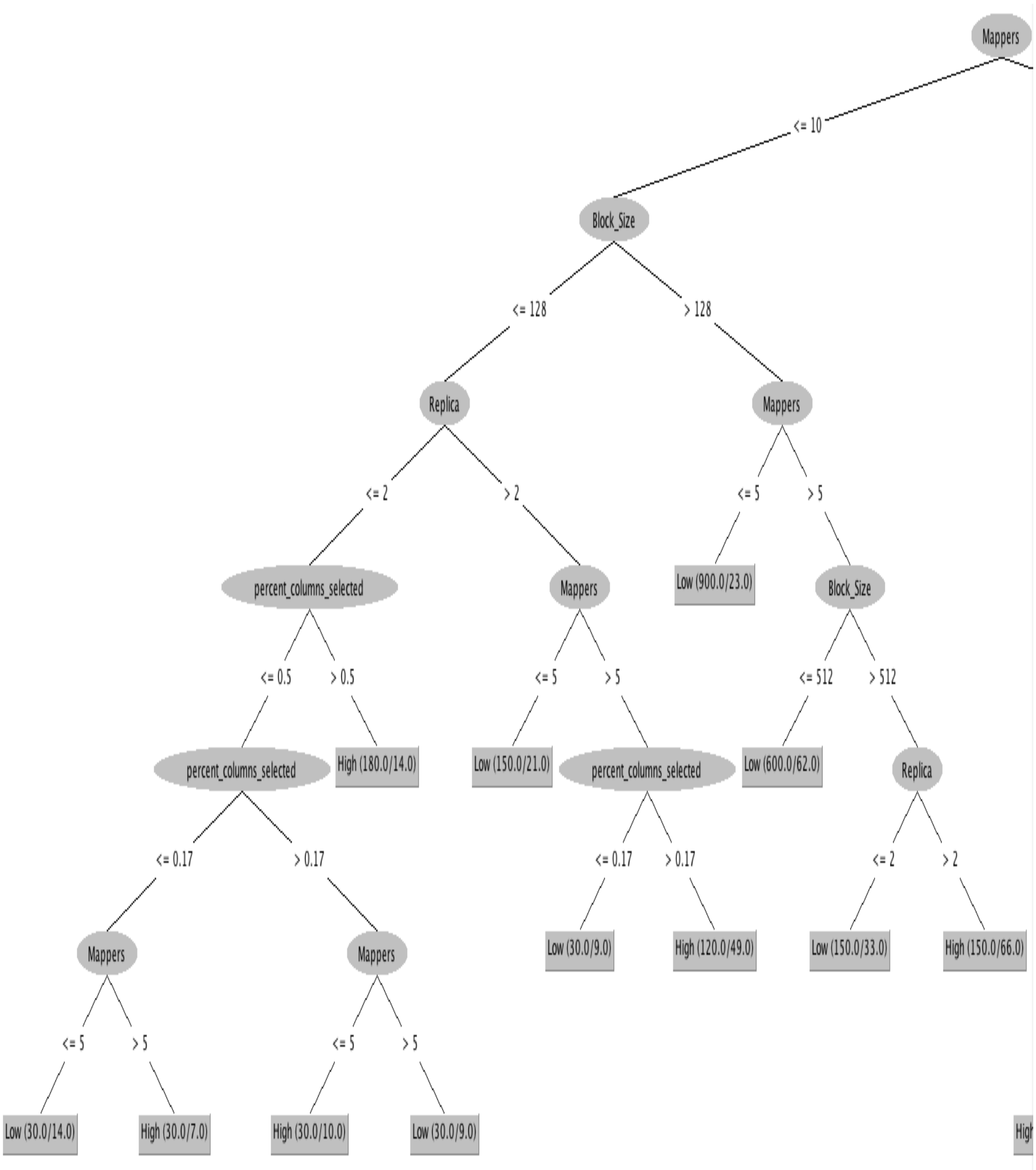

Figure B.1: Decision tree for memory usage

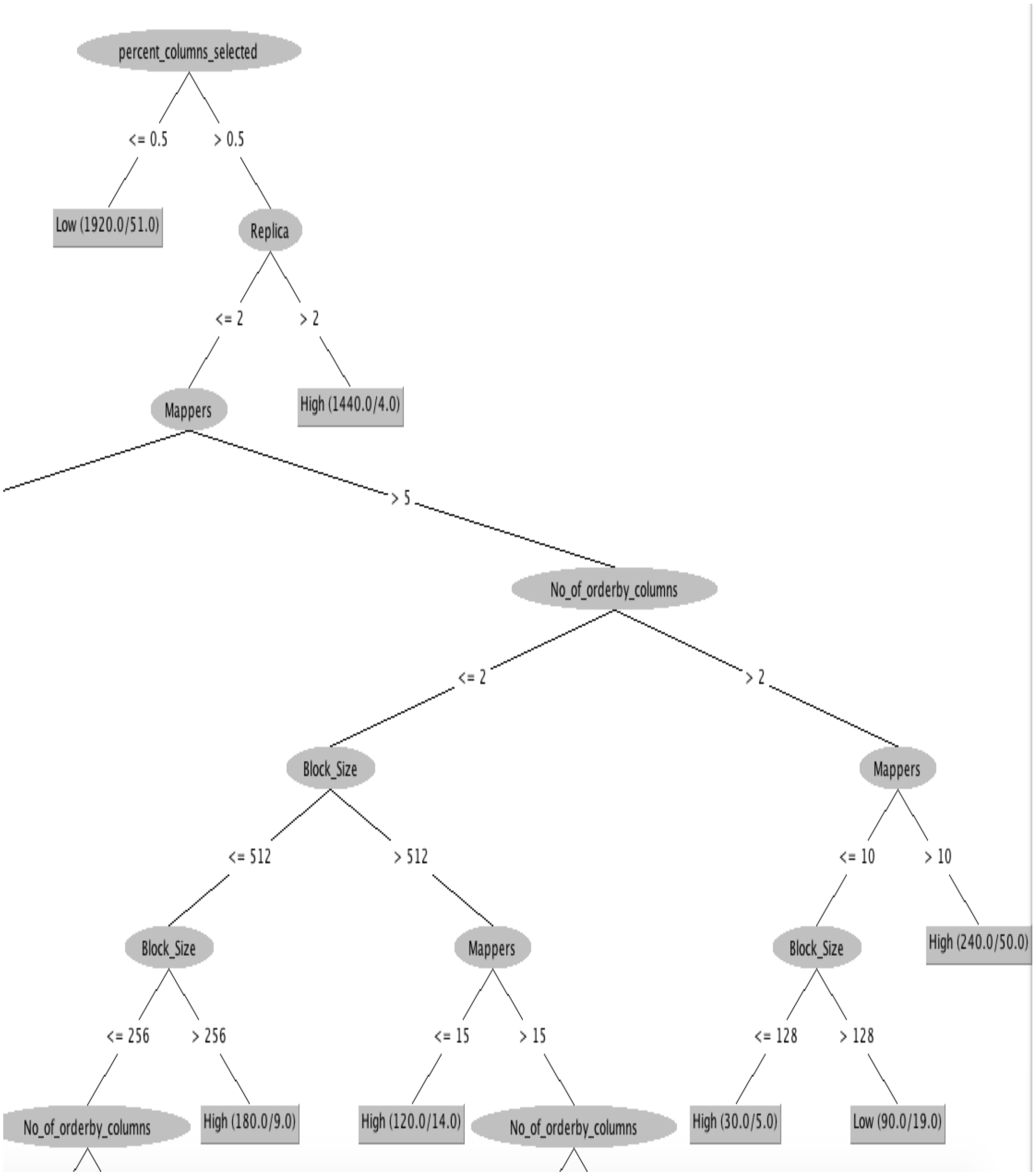

Figure B.2: Decision tree for network usage

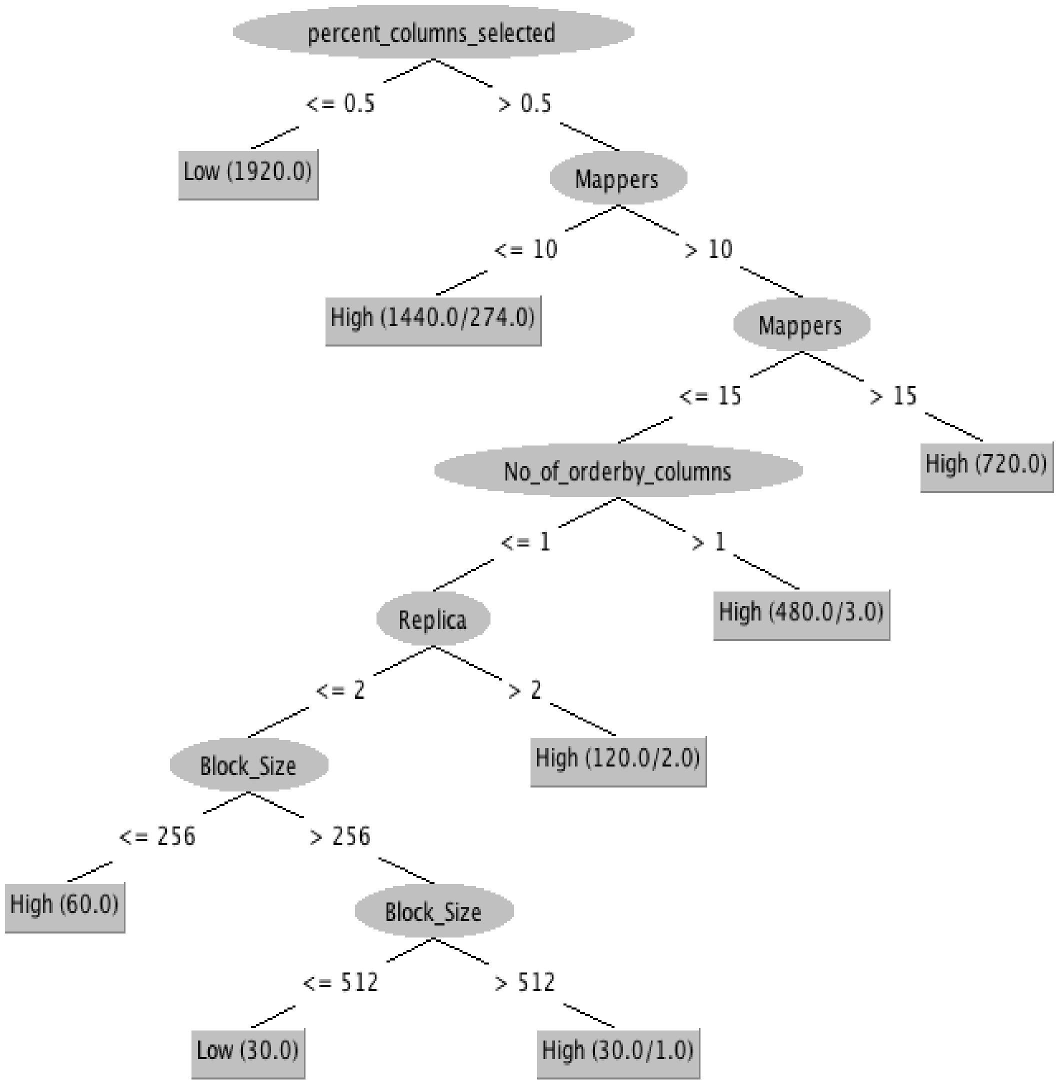

Figure B.3: Decision tree for execution time

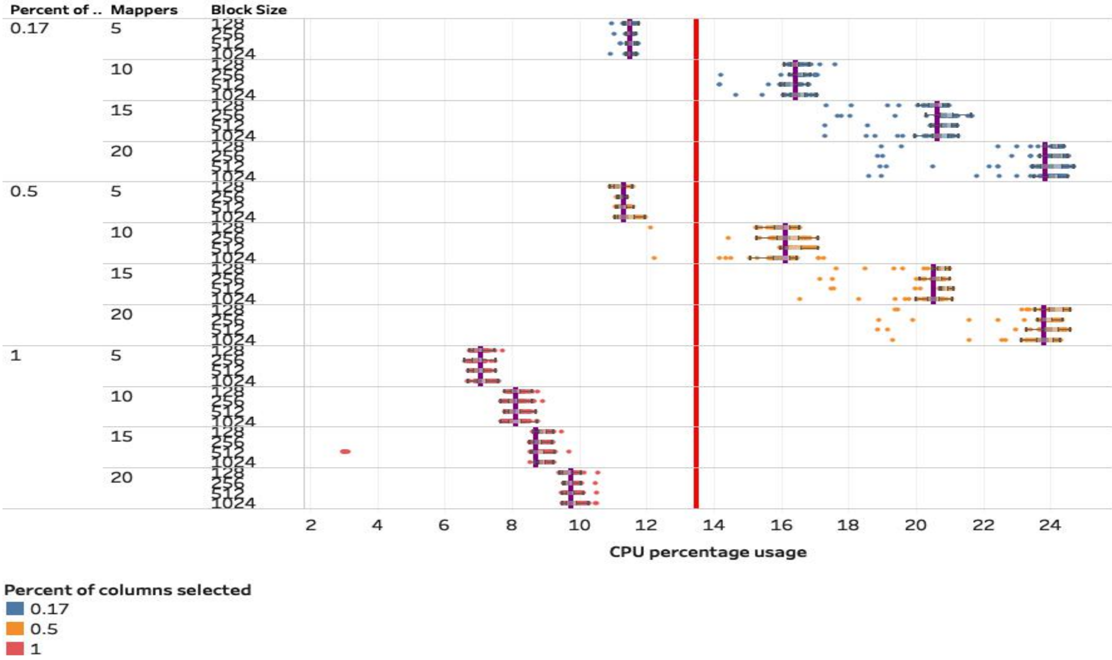

Figure B.4: Impact of percentage of columns selected, number of mappers and block size on mean CPU usage.

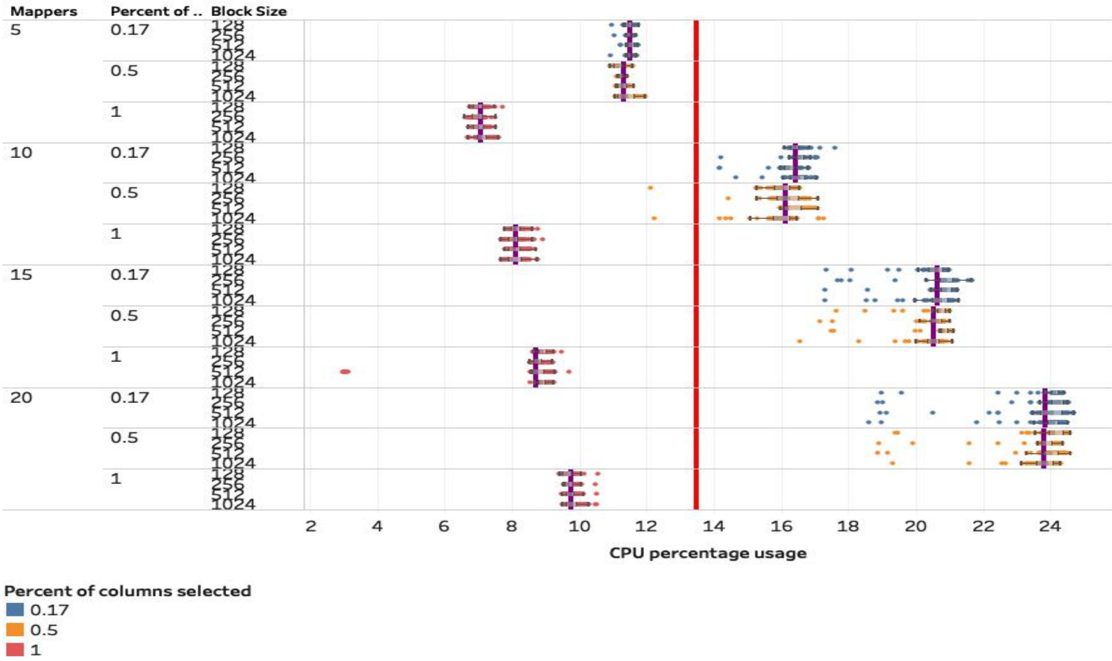

Figure B.5: Impact of percentage of columns selected, number of mappers and block size on mean CPU usage.

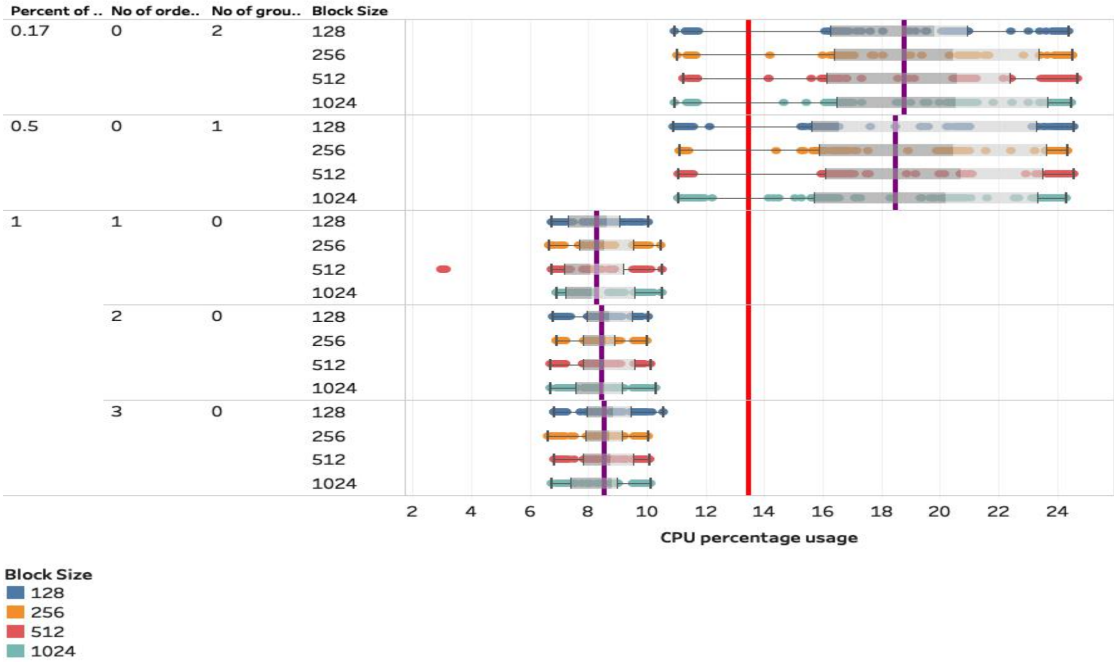

Figure B.6: Impact of query structure elements and block size on mean CPU usage.

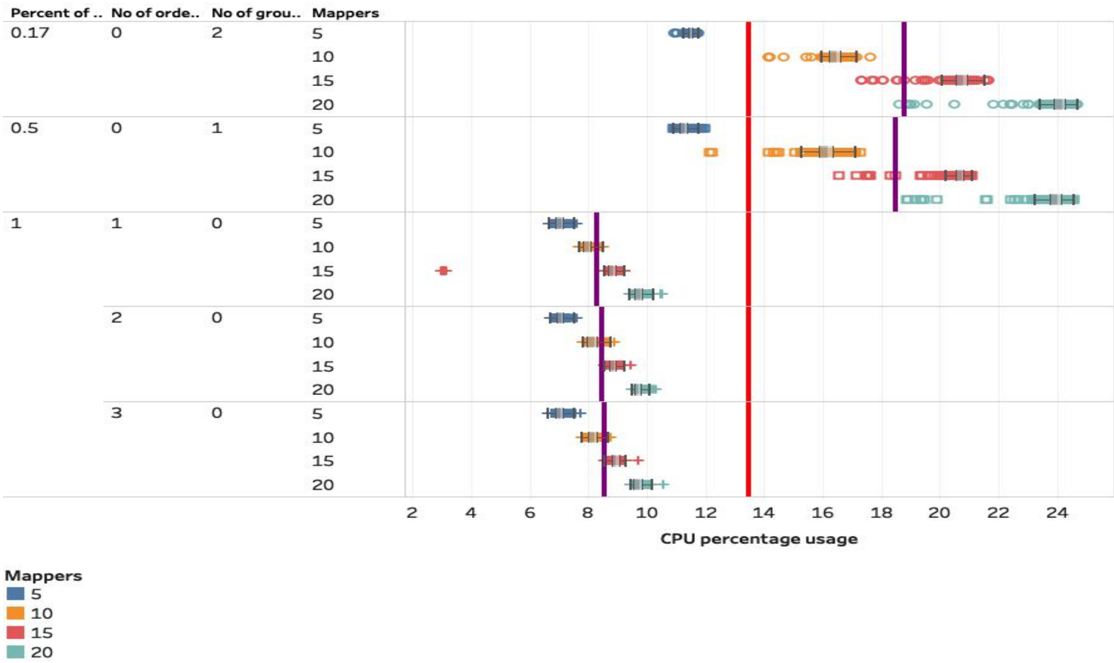

Figure B.7: Impact of query structure elements and number of mappers on mean CPU usage.

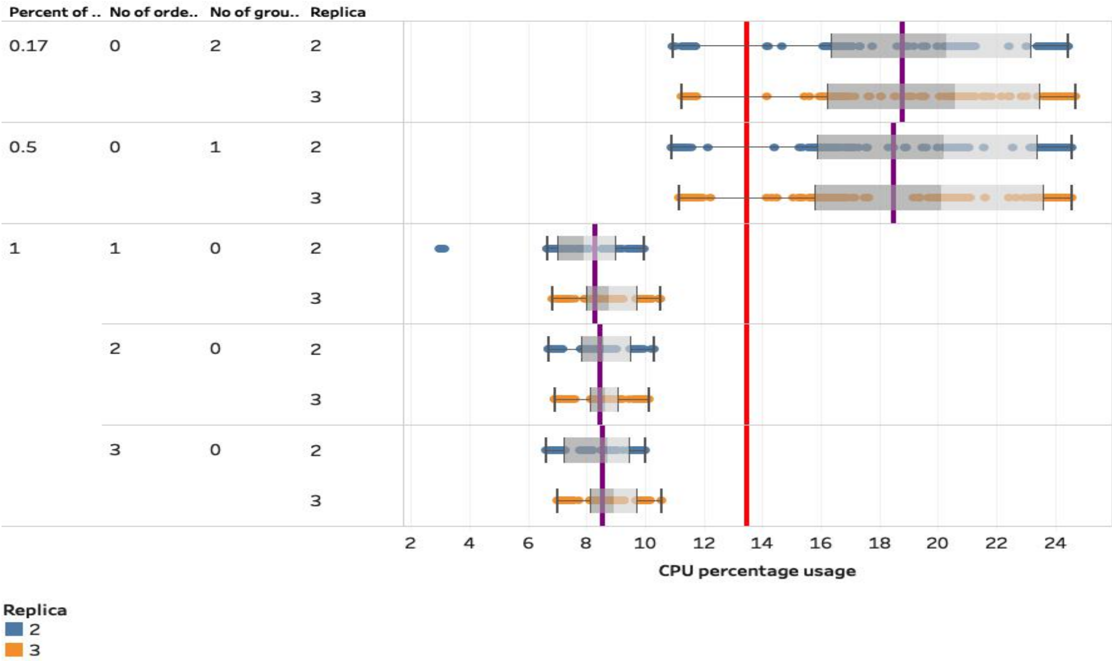

Figure B.8: Impact of query structure elements and number of replicas on mean CPU usage.

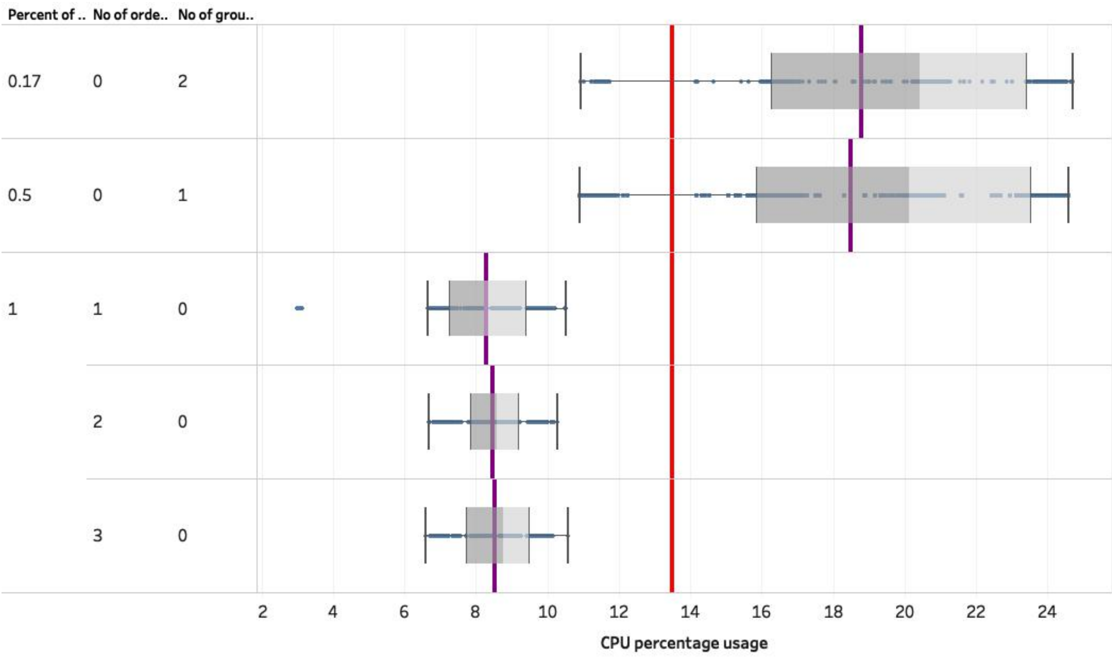

Figure B.9: Impact of different query structure elements on mean CPU usage.

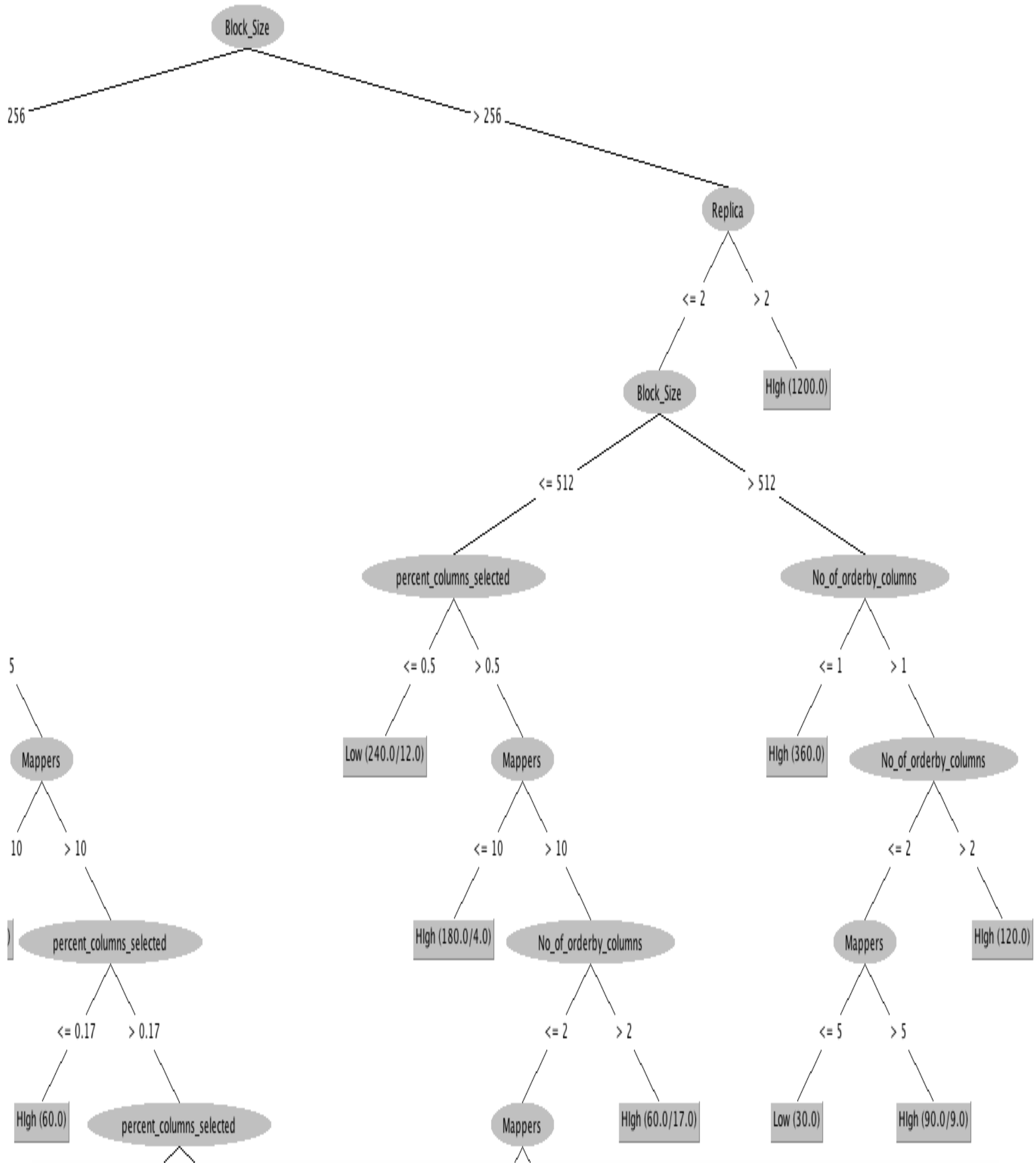

Figure B.10: Decision tree for disk usage

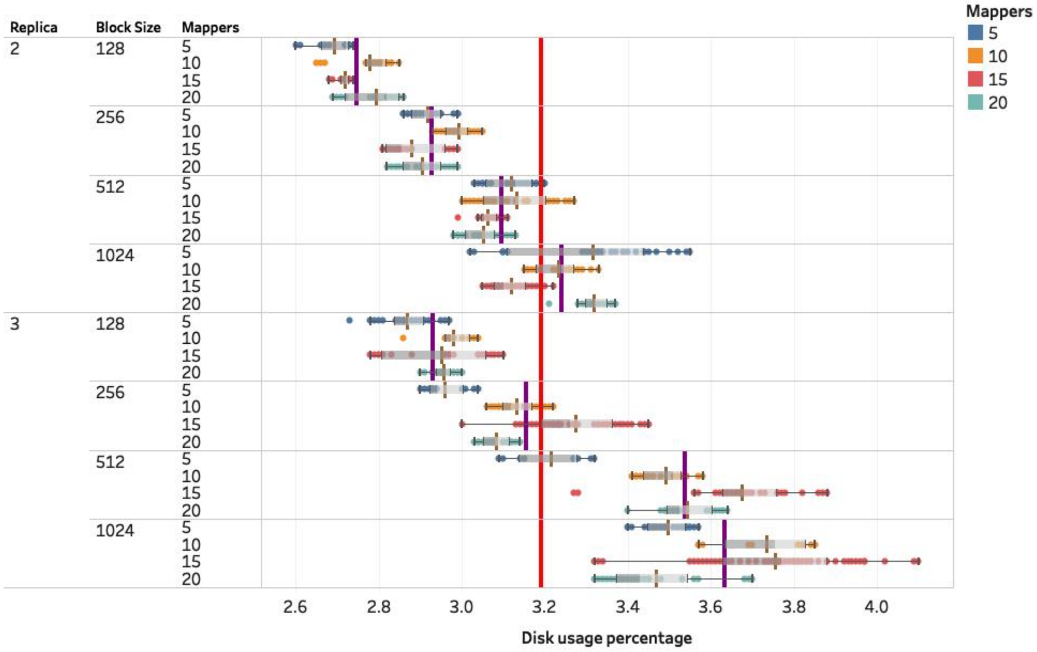

Figure B.11: Impact of different configuration parameters on mean disk usage.

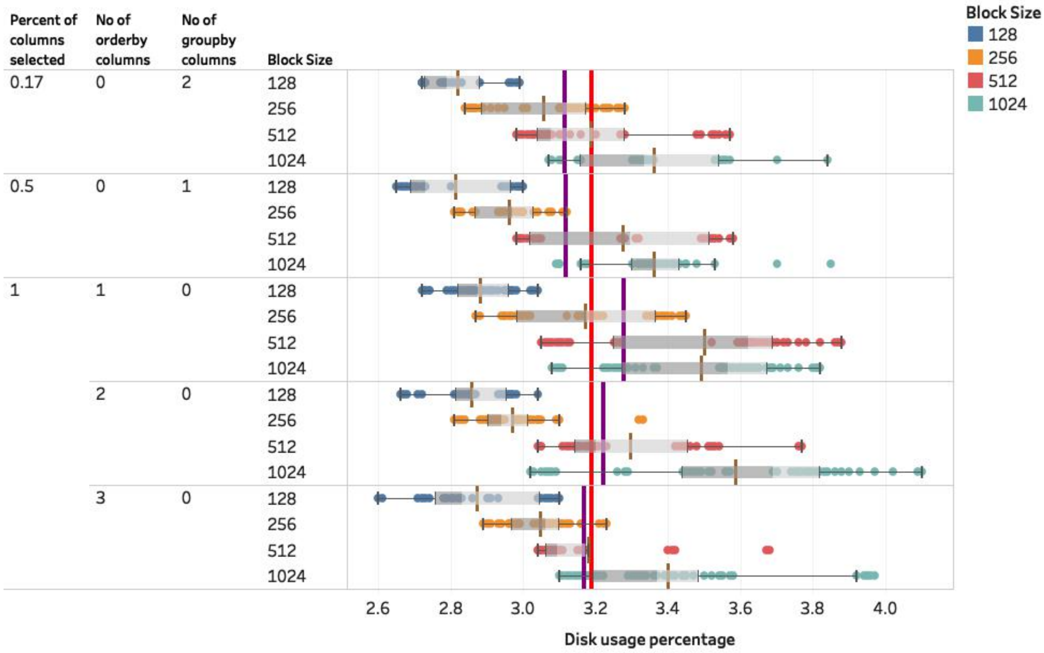

Figure B.12: Impact of different query structure elements and block size on mean disk usage.

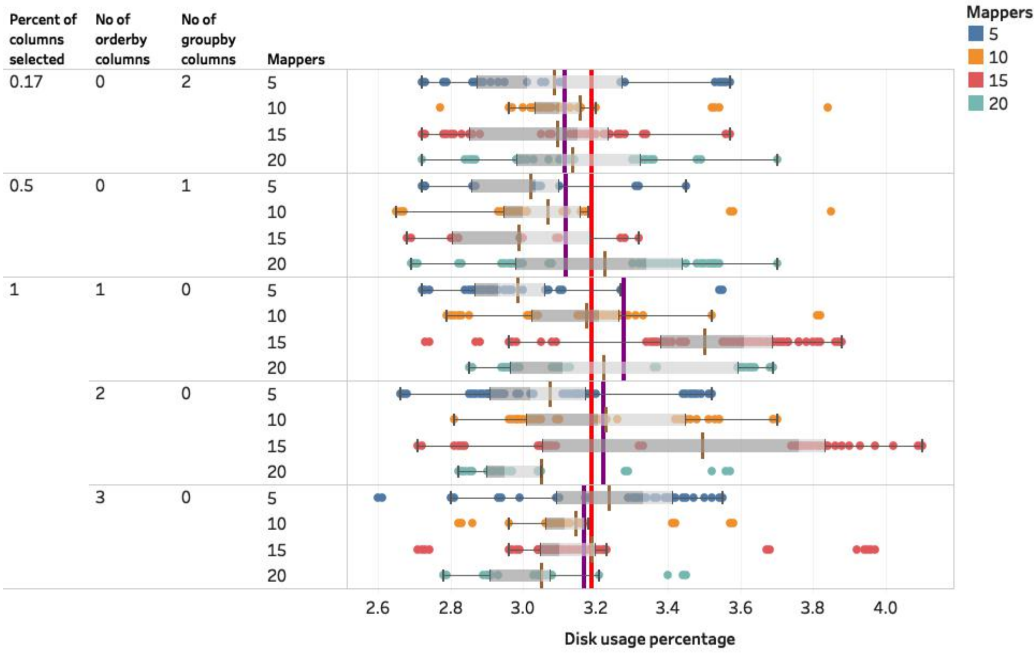

Figure B.13: Impact of different query structure elements and number of mappers on mean disk usage.

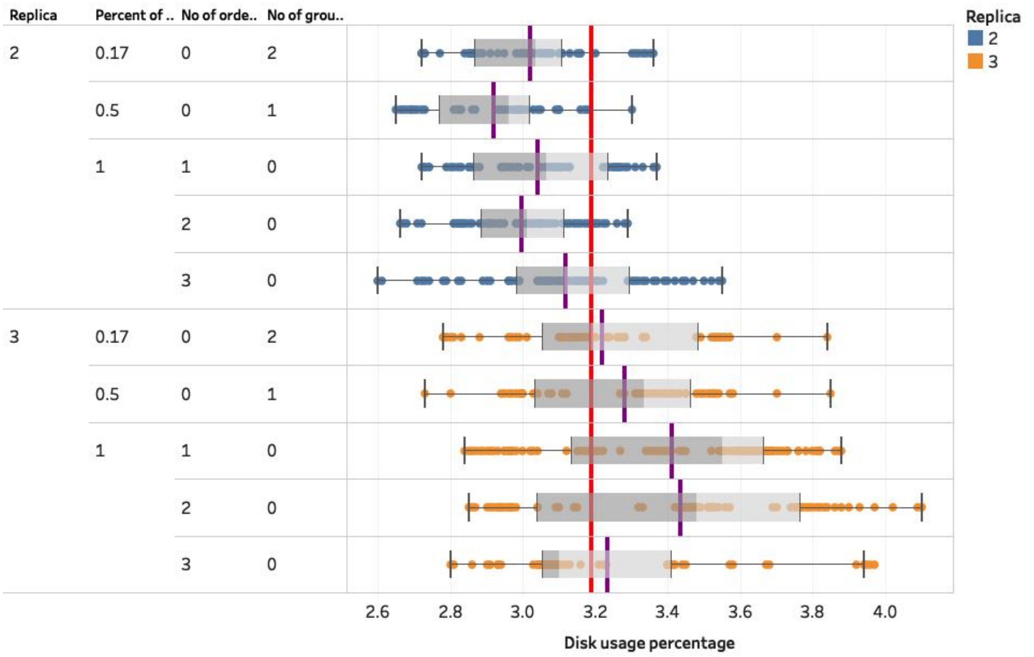

Figure B.14: Impact of different query structure elements and number of replicas on mean disk usage.

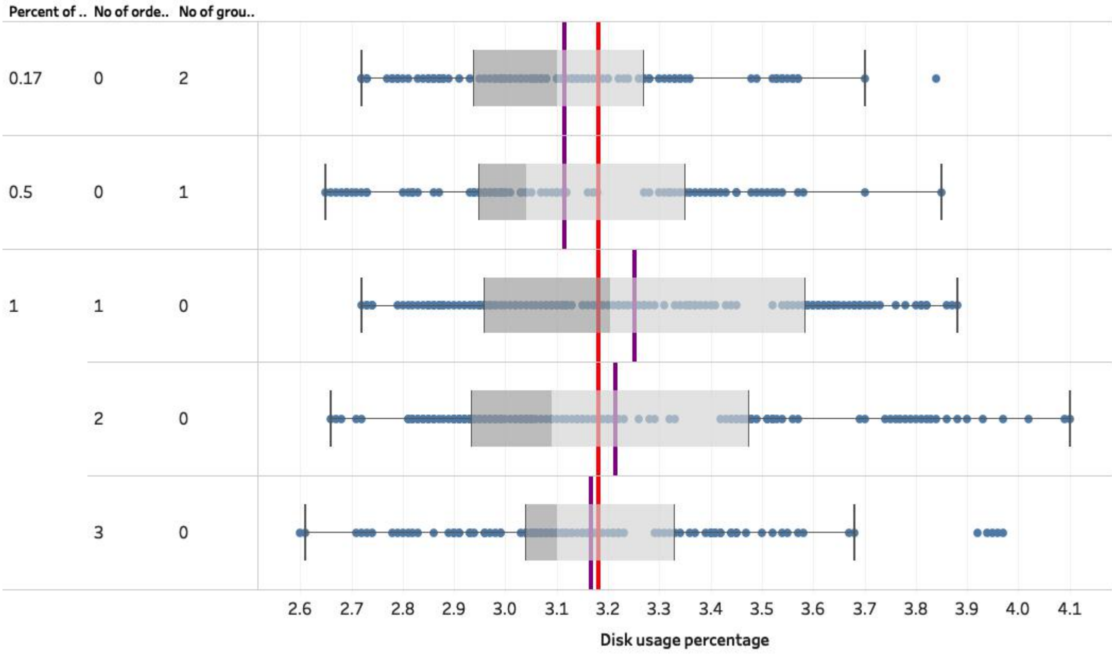

Figure B.15: Impact of different query structure elements on mean disk usage.

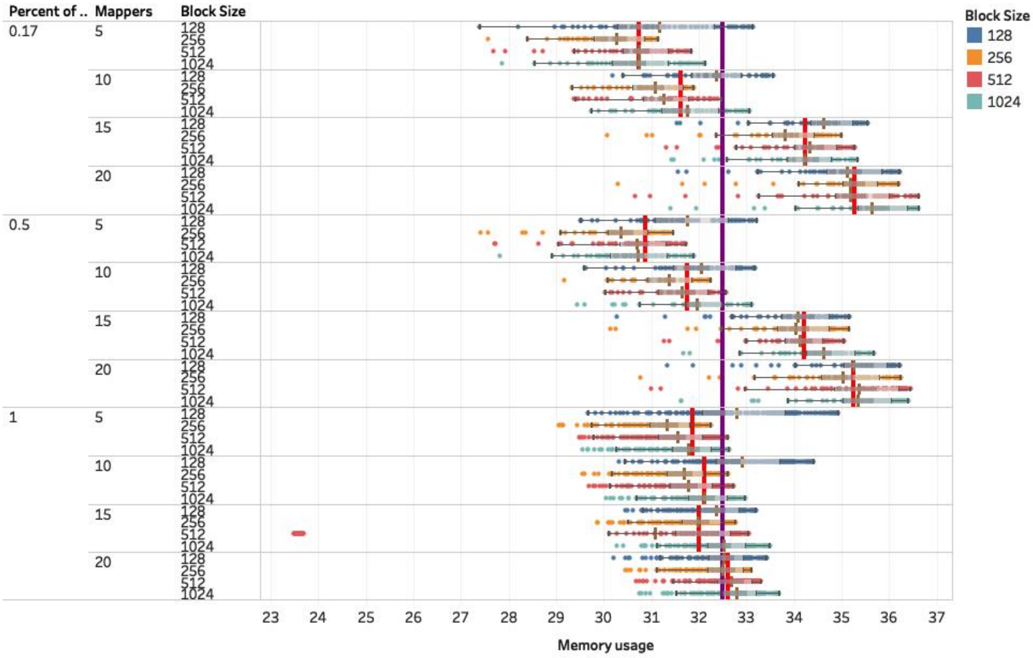

Figure B.16: Impact of percentage of columns selected, number of mappers and block size on mean memory usage.

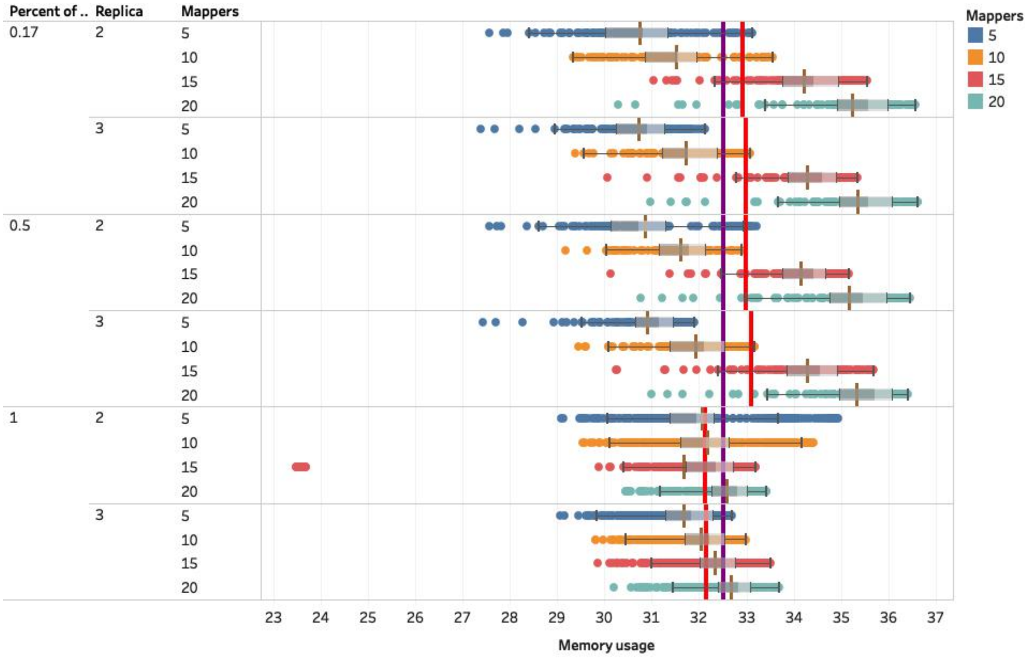

Figure B.17: Impact of percentage of columns selected, number of mappers and number of replicas on mean memory usage.

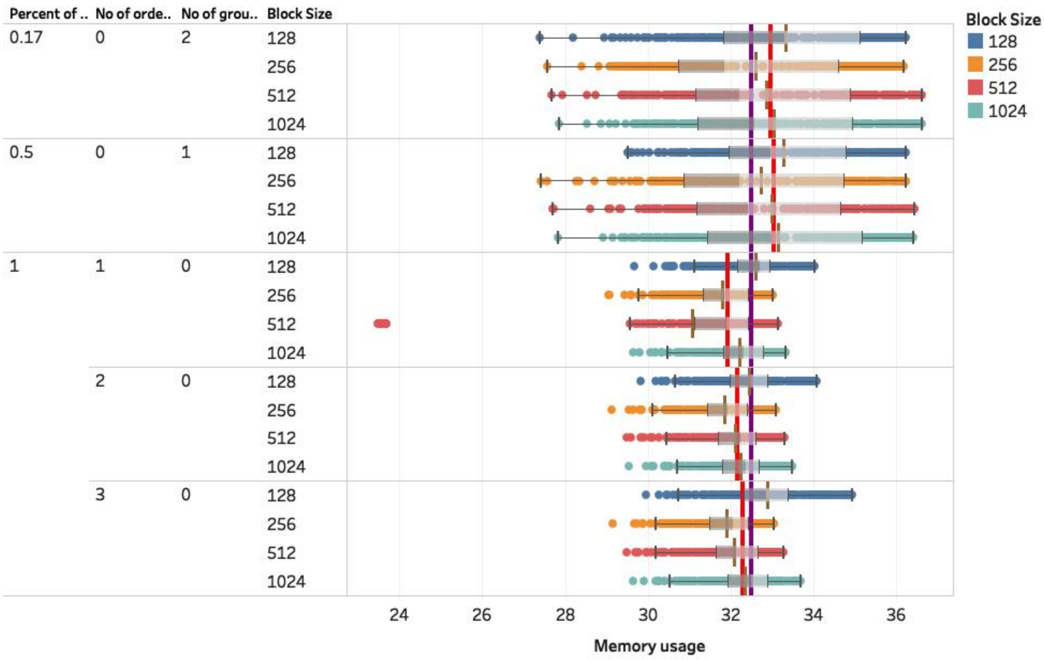

Figure B.18: Impact of query structure elements and block size on mean memory usage.

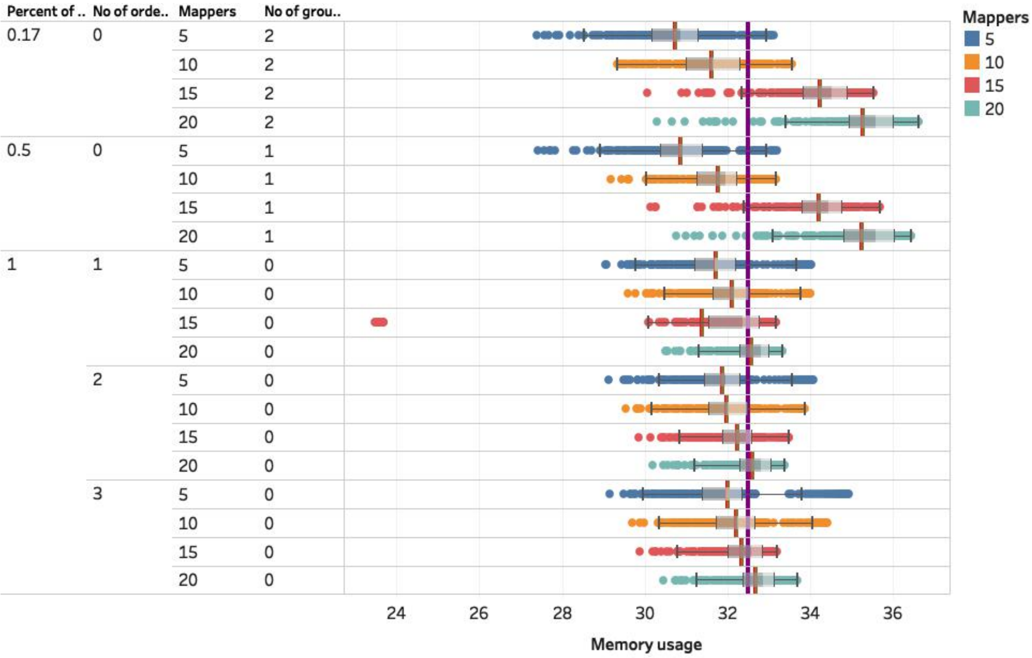

Figure B.19: Impact of query structure elements and number of mappers on mean memory usage.

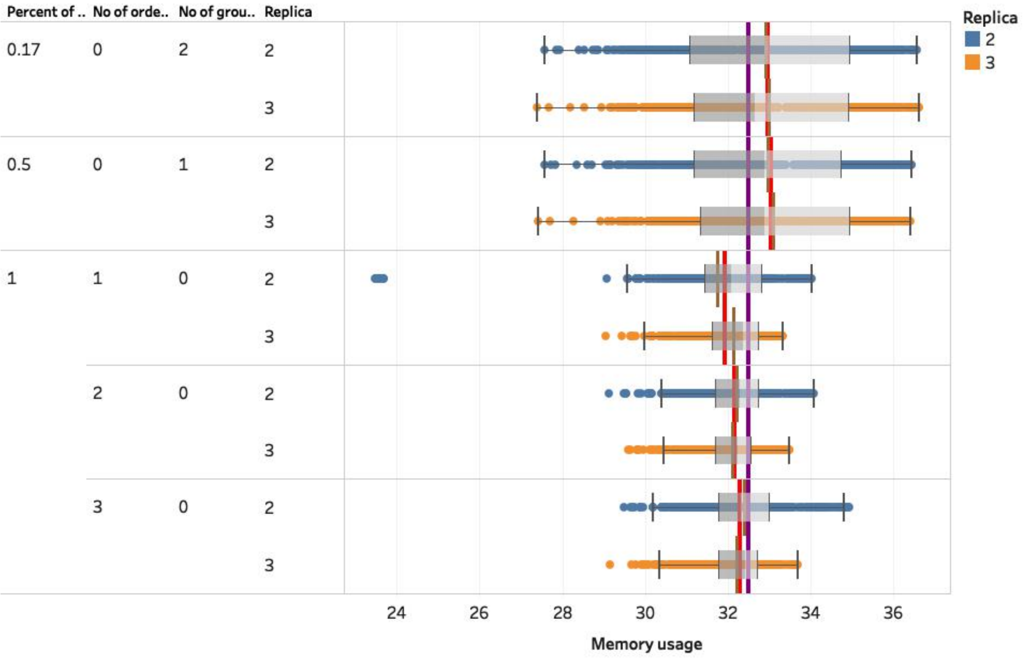

Figure B.20: Impact of query structure elements and number of replicas on mean memory usage.

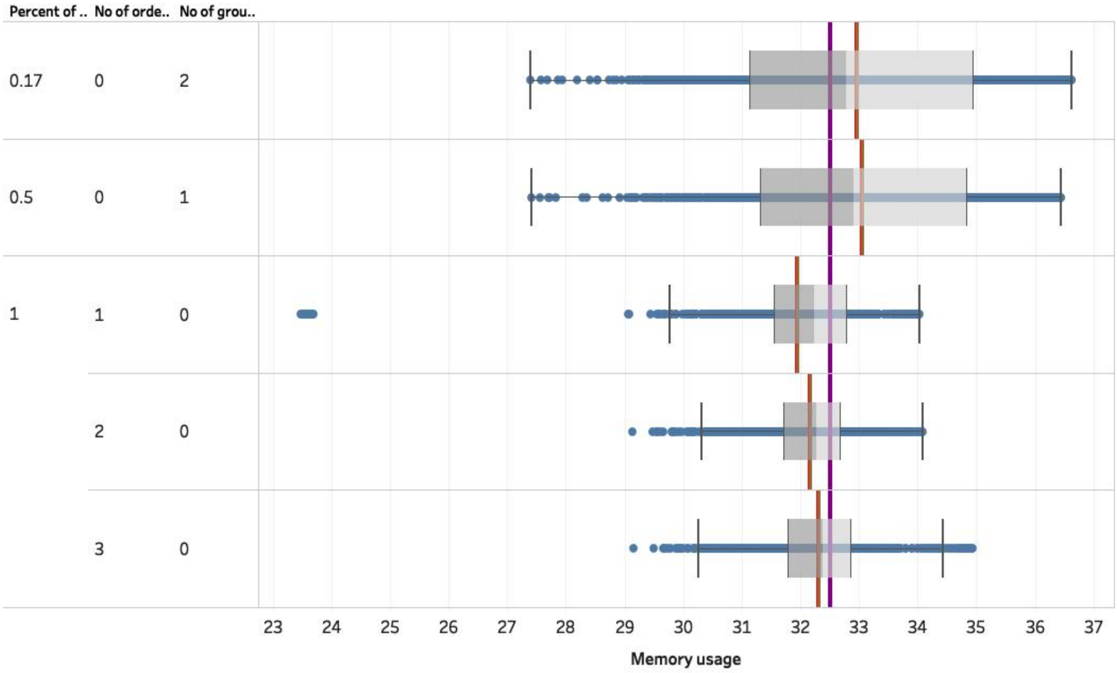

Figure B.21: Impact of different query structure elements on mean memory usage.

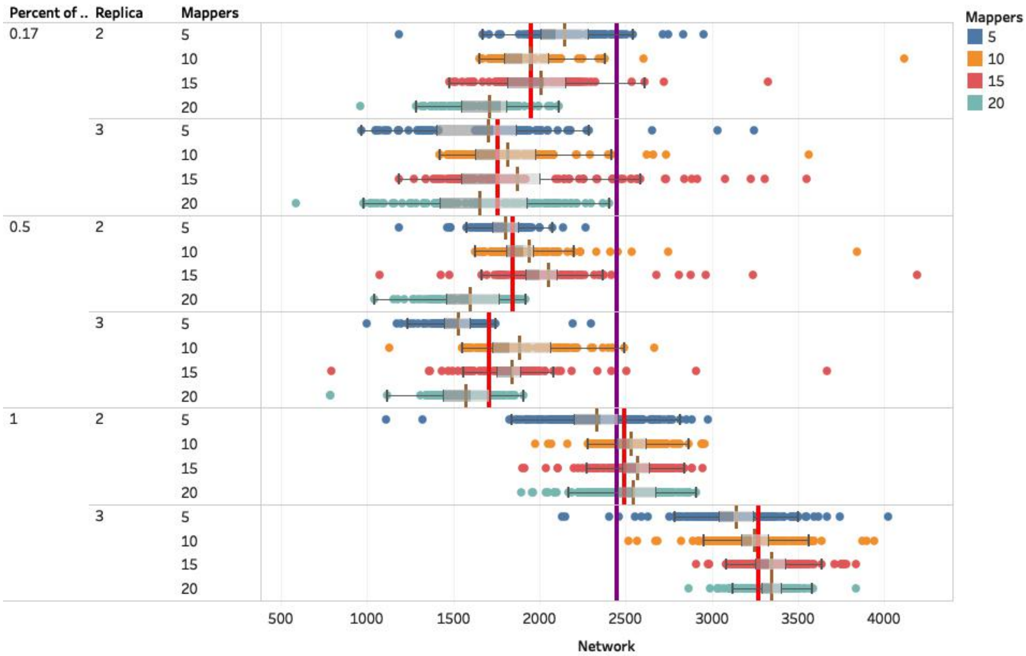

Figure B.22: Impact of percentage of columns selected, number of replicas and number of mappers on mean network usage.

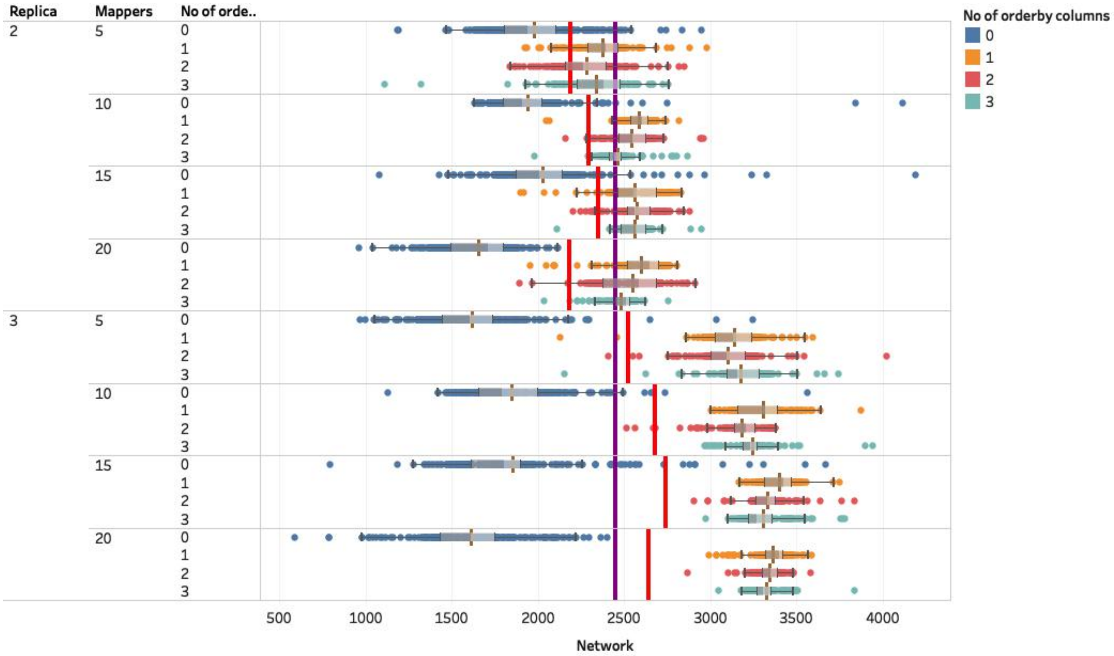

Figure B.23: Impact of Number of columns in order by clause, number of replicas and number of mappers on mean network usage.

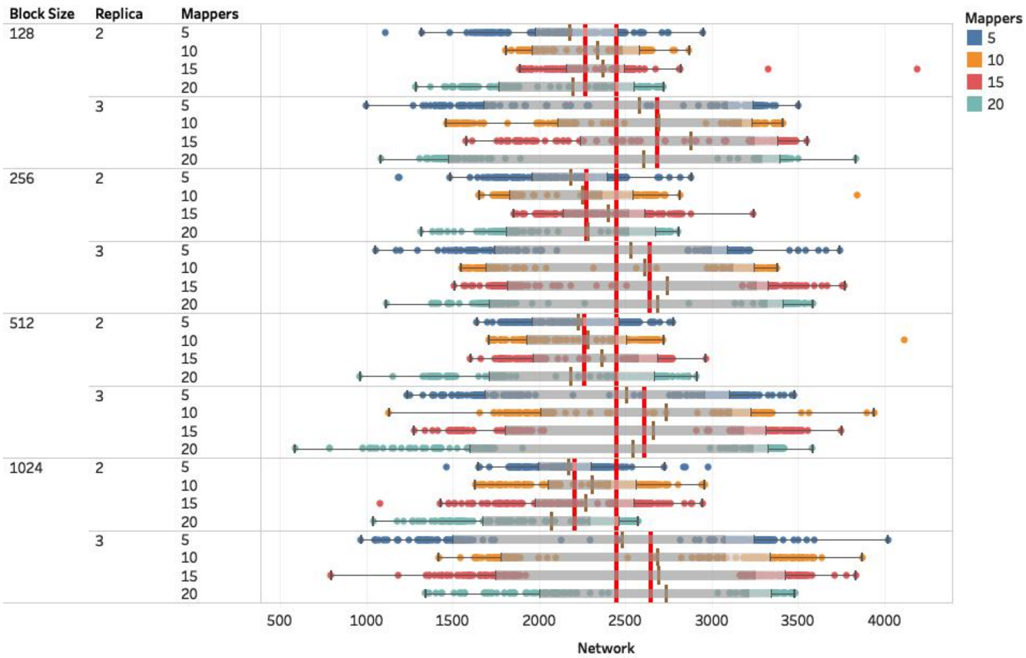

Figure B.24: Impact of Hadoop configuration parameters on mean network usage.

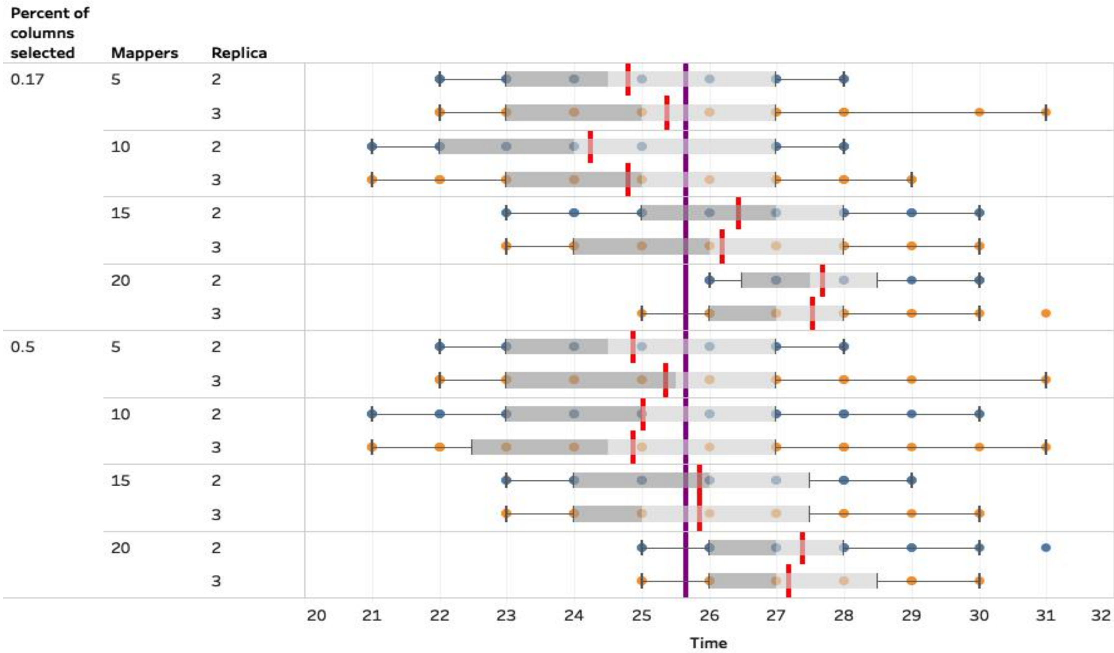

Figure B.25: Impact of percentage of columns selected for group by clause, number of replicas and number of mappers on mean execution time.

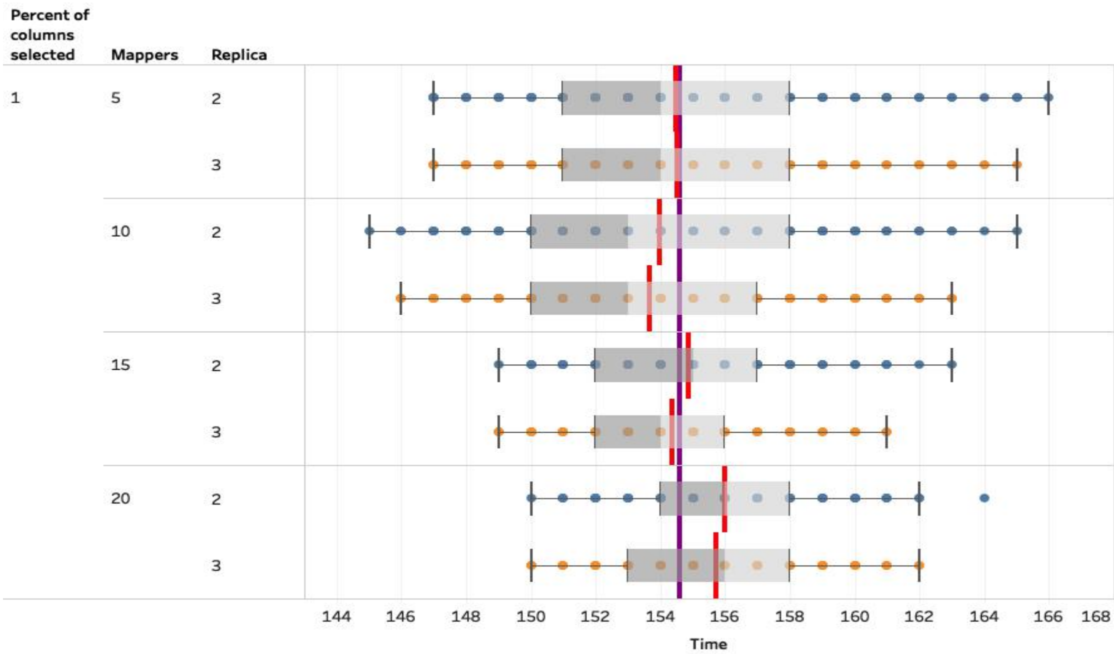

Figure B.26: Impact of percentage of columns selected for order by clause, number of replicas and number of mappers on mean execution time.

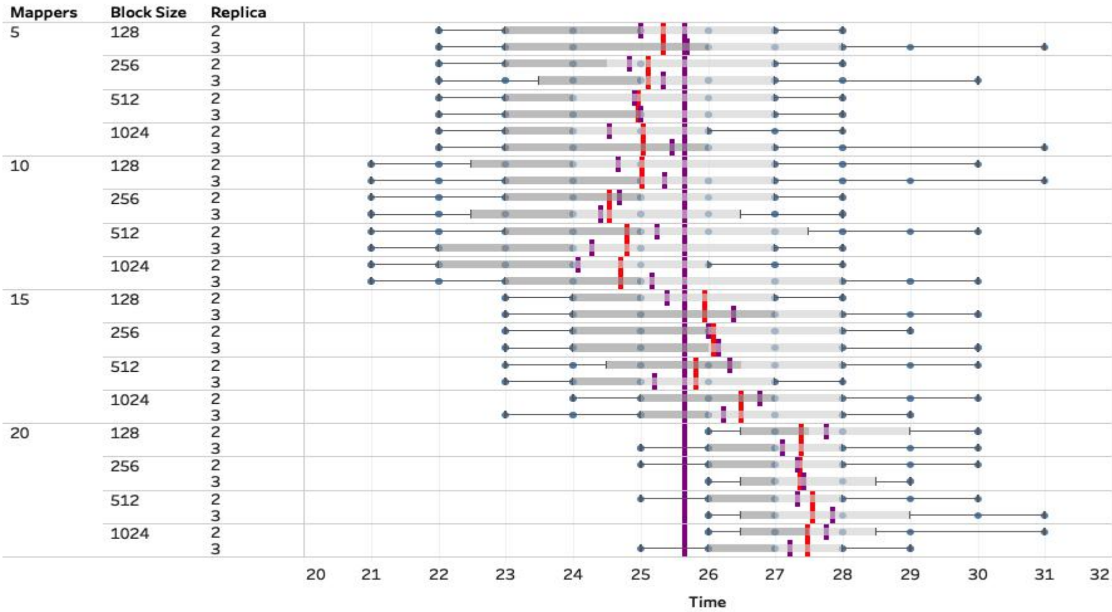

Figure B.27: Impact of Hadoop configuration parameters for group by clause on mean execution time.

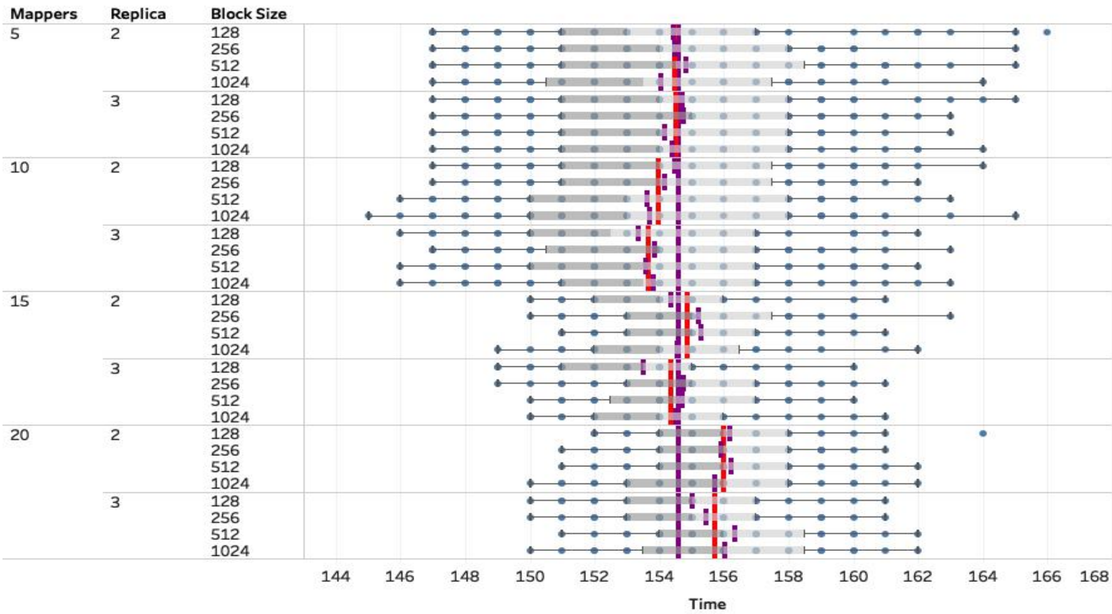

Figure B.28: Impact of Hadoop configuration parameters for order by clause on mean execution time.

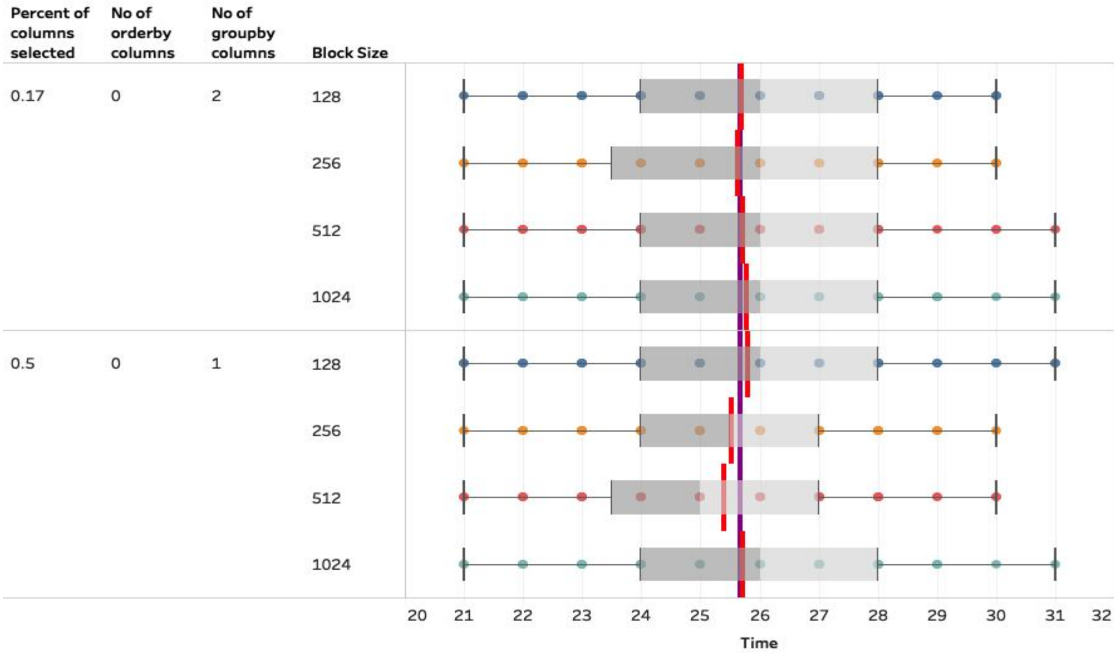

Figure B.29: Impact of query elements and block size for group by clause on mean execution time.

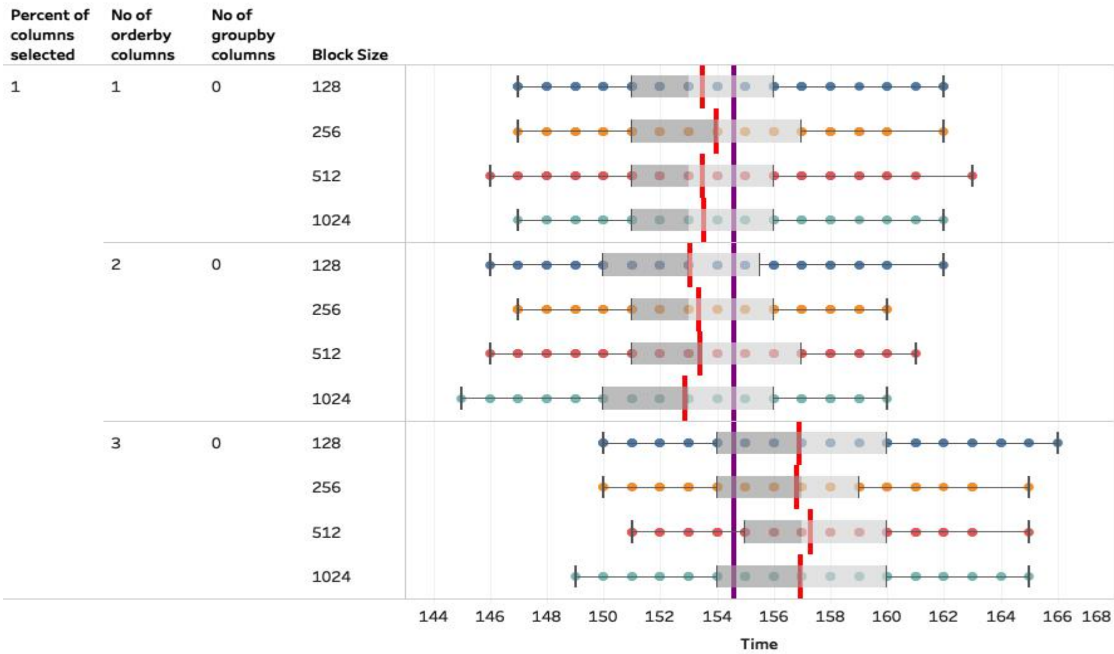

Figure B.30: Impact of query elements and block size for order by clause on mean execution time.

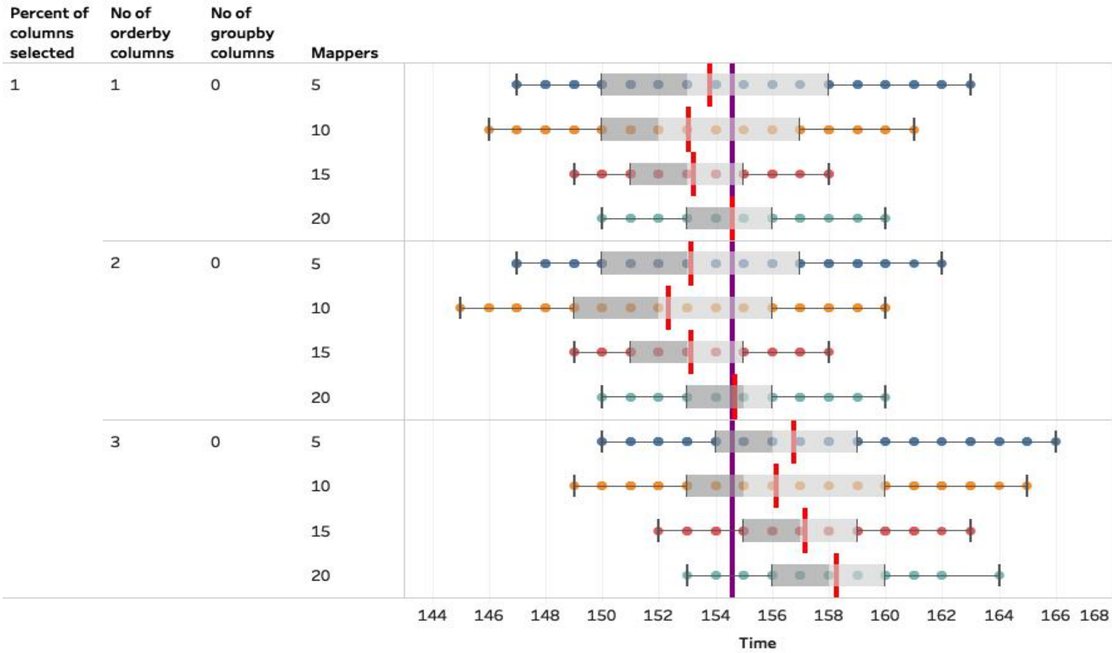

Figure B.31: Impact of query elements and number of mappers for order by clause on mean execution time.

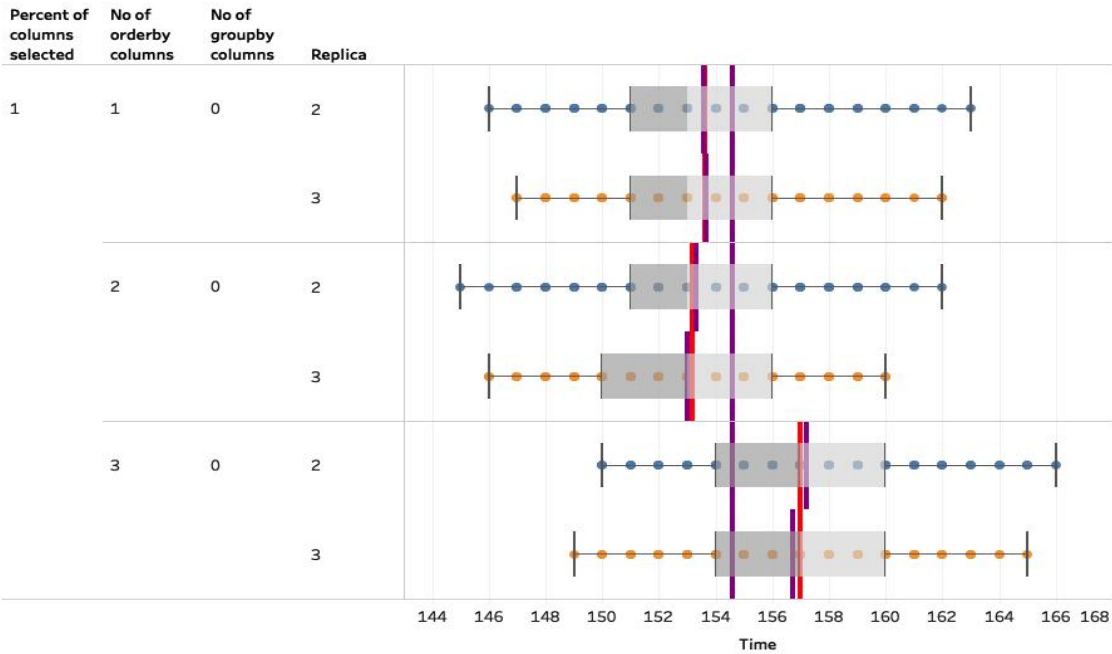

Figure B.32: Impact of query elements and number of replicas for order by clause on mean execution time.

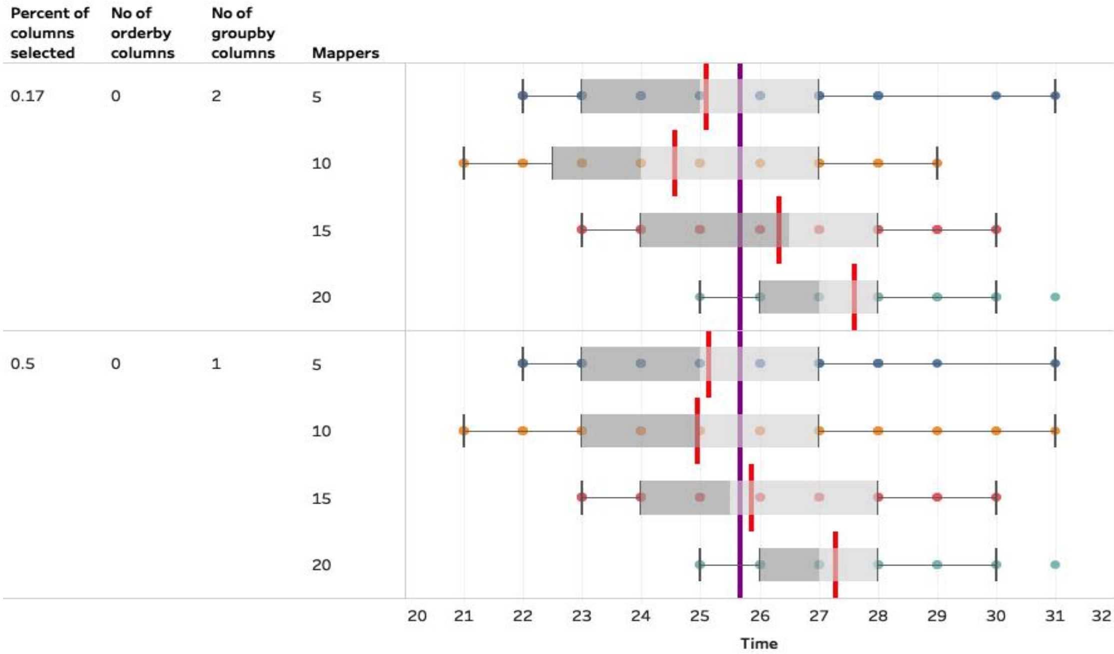

Figure B.33: Impact of query elements and number of mappers for group by clause on mean execution time.

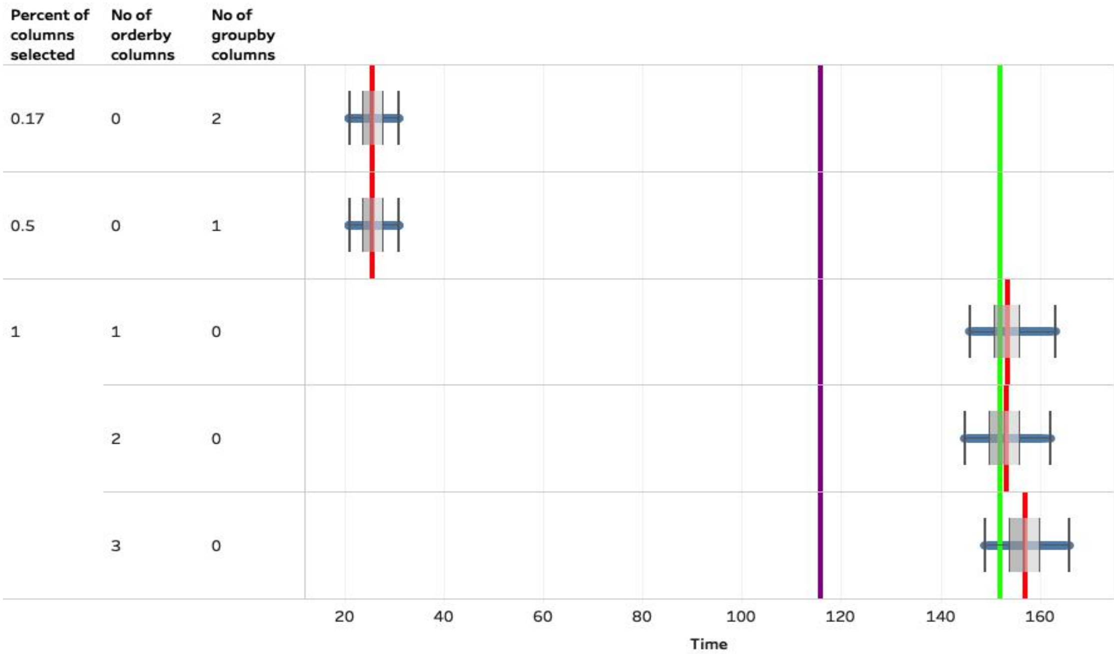

Figure B.34: Impact of different query structure elements on mean execution time.

Appendix C

Modelling and prediction of resource utilization of Hadoop cluster for different datasets

C.1 CPU usage

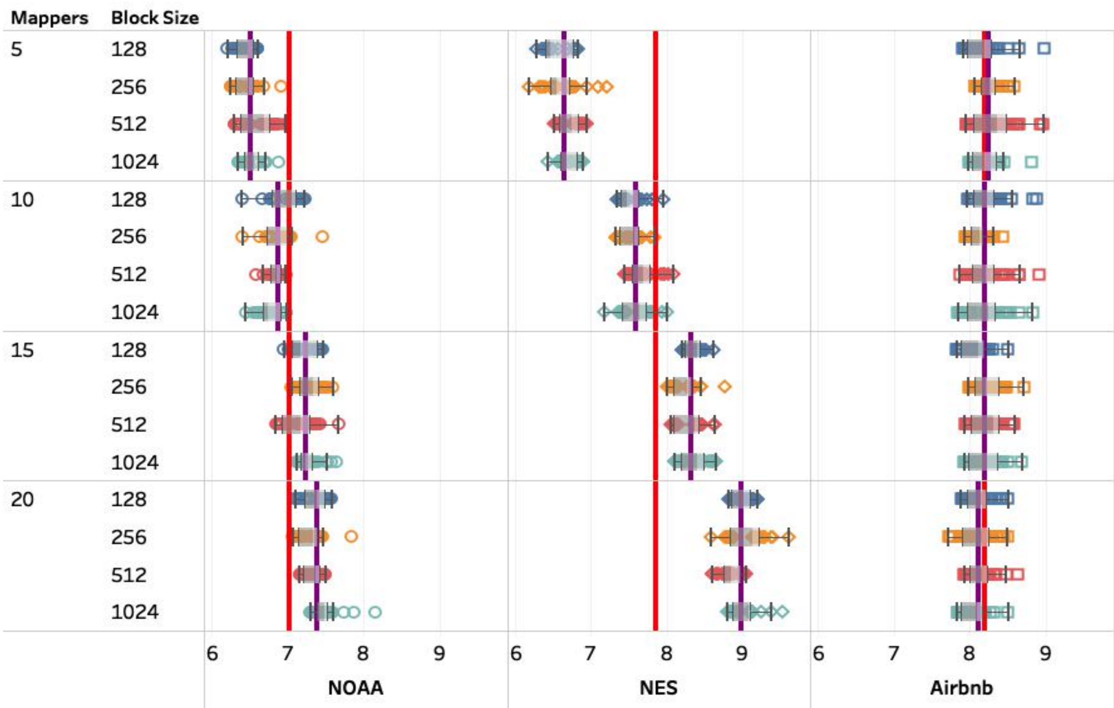

Figure C.1: Impact of number of mappers and block size on CPU usage for different datasets.

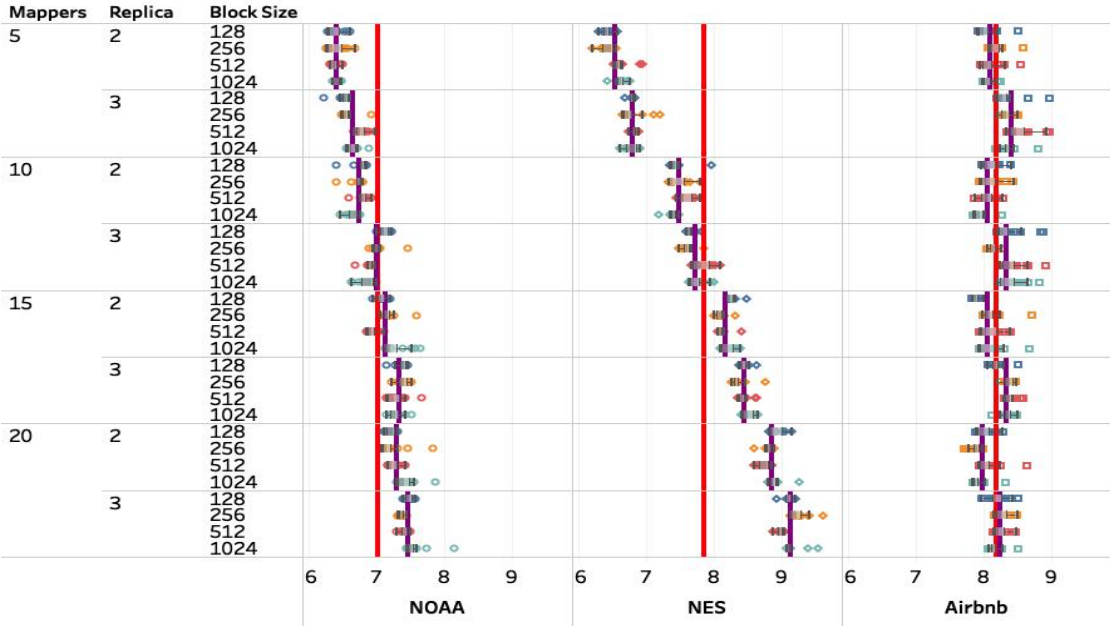

Figure C.2: Impact of number of mappers, number of replicas and block size on CPU usage for different datasets.

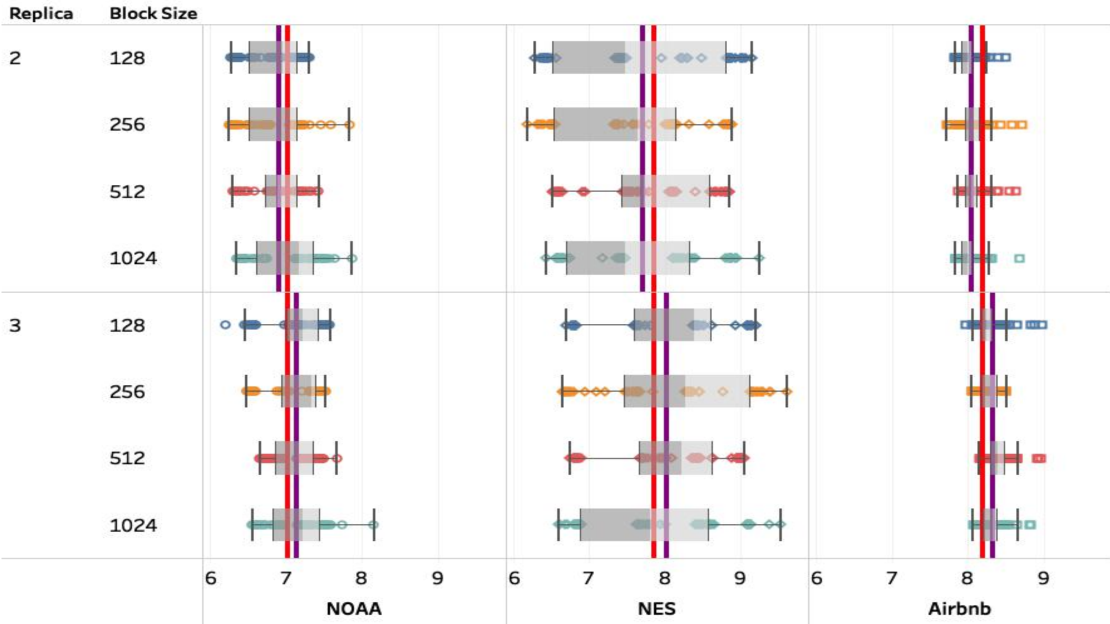

Figure C.3: Impact of number of replicas and block size on CPU usage for different datasets.

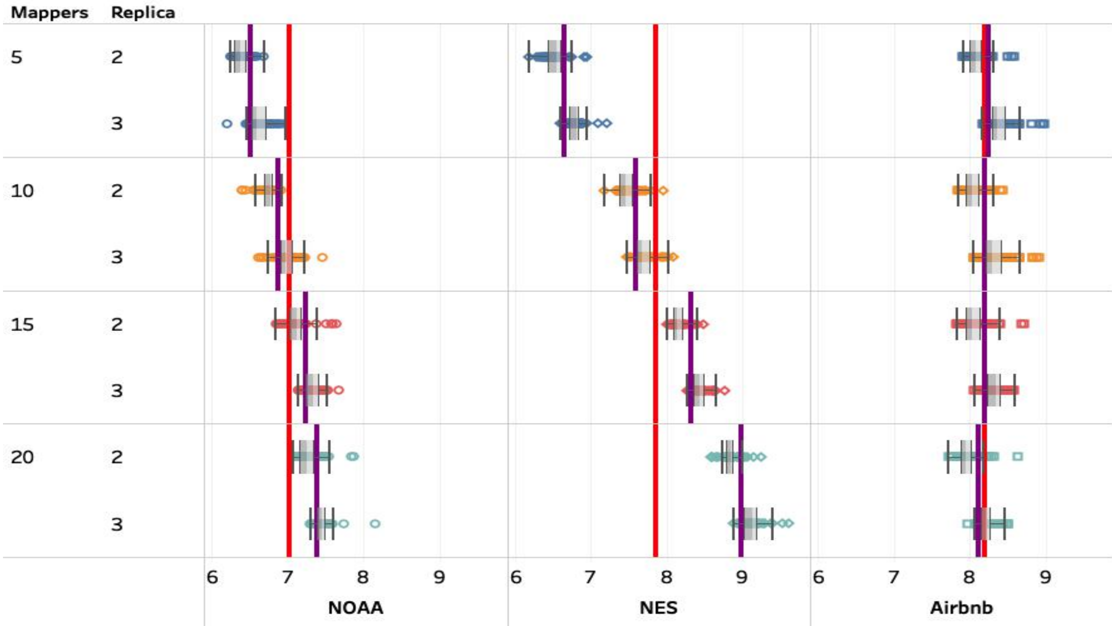

Figure C.4: Impact of number of mappers and number of replicas on CPU usage for different datasets.

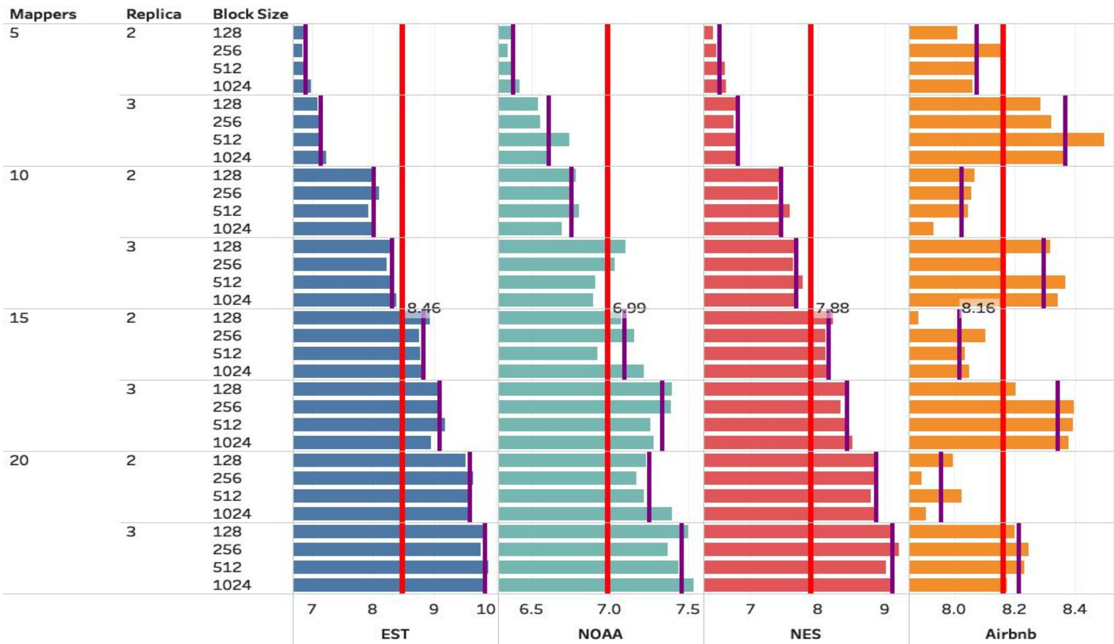

Figure C.5: Mean CPU usage for different dataset.

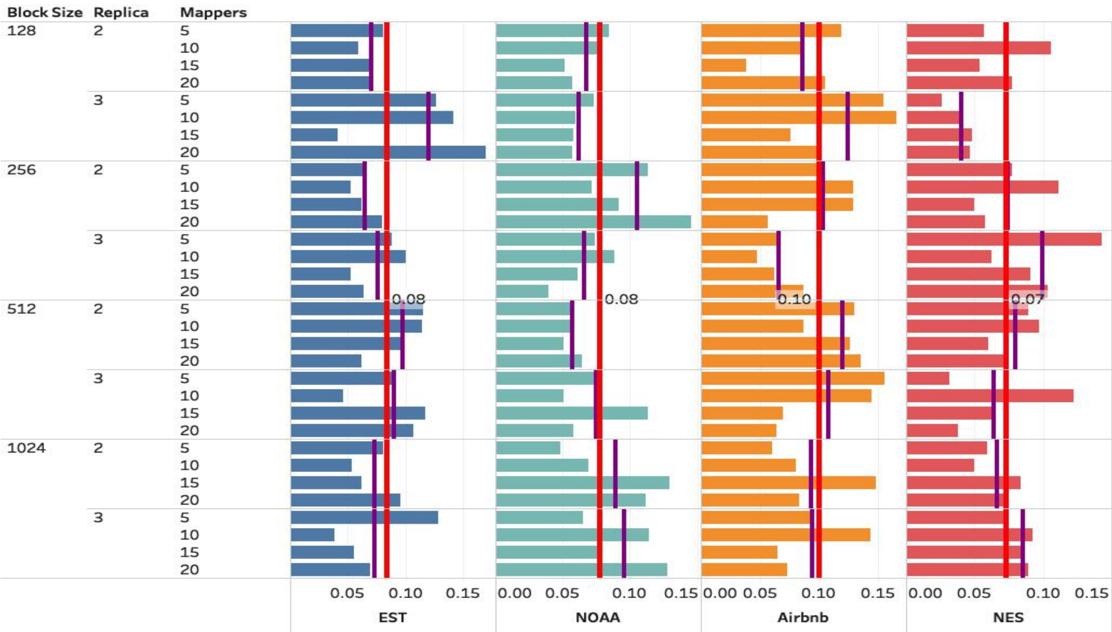

Figure C.6: Standard deviation of CPU usage for different dataset.

C.1.1 Decision tree rules

1. B $<=$ 256MB, R $<=$ 2, M $>$ 15 and N $>$ 0.5, only Airbnb showed 24.04% of the instances were true.

2. B $<=$ 256MB, R $<=$ 2, M $>$ 15, N $>$ 0.3 and O $>$ 2, 34.46% of the instances were true in Airbnb dataset.

C.1.2 CPU usage linear regression models from M5P tree for different datasets

```
LM num: 1
usage =
-0.0181 * No_of_columns
- 0 * Block_size
+ 0.0109 * Replica
+ 0.0004 * Mappers
```

```
+ 0.1795

LM num: 2
usage =
-0.0035 * No_of_columns
- 0 * Block_size
+ 0.0109 * Replica
+ 0.0007 * Mappers
+ 0.4524

LM num: 3
usage =
-0.0057 * No_of_columns
- 0.0004 * Block_size
+ 0.0109 * Replica
+ 0.0257 * Mappers
+ 0.0109

LM num: 4
usage =
-0.0057 * No_of_columns
- 0.0036 * Block_size
+ 0.0109 * Replica
+ 0.0257 * Mappers
+ 0.9981

LM num: 5
usage =
-0.0057 * No_of_columns
- 0 * Block_size
+ 0.0109 * Replica
+ 0.0069 * Mappers
+ 0.0283
```

```
LM num: 6
usage =
-0.0004 * No_of_columns
- 0 * Block_size
+ 0.0093 * Replica
+ 0.0004 * Mappers
- 0.0123

LM num: 7
usage =
0.0007 * No_of_columns
- 0 * Block_size
+ 0.0247 * Replica
+ 0.0173 * Mappers
- 0.2524

LM num: 8
usage =
0.0007 * No_of_columns
- 0 * Block_size
+ 0.0247 * Replica
+ 0.0829 * Mappers
- 0.8406

LM num: 9
usage =
0.0068 * No_of_columns
+ 0.0001 * Block_size
+ 0.0093 * Replica
+ 0.0028 * Mappers
+ 0.7194
```

```
LM num: 10
usage =
0.0068 * No_of_columns
- 0.0007 * Block_size
+ 0.0093 * Replica
+ 0.0028 * Mappers
+ 0.4623

LM num: 11
usage =
0.0068 * No_of_columns
- 0.0001 * Block_size
+ 0.0093 * Replica
+ 0.0028 * Mappers
+ 0.815

LM num: 12
usage =
0.0057 * No_of_columns
- 0 * Block_size
+ 0.0093 * Replica
+ 0.0028 * Mappers
+ 0.8063

LM num: 13
usage =
0.0015 * No_of_columns
- 0 * Block_size
+ 0.0093 * Replica
+ 0.0028 * Mappers
+ 0.8957

LM num: 14
```

```
usage =
0.0008 * No_of_columns
- 0 * Block_size
+ 0.0093 * Replica
+ 0.0012 * Mappers
+ 0.9365
```

C.2 Disk usage

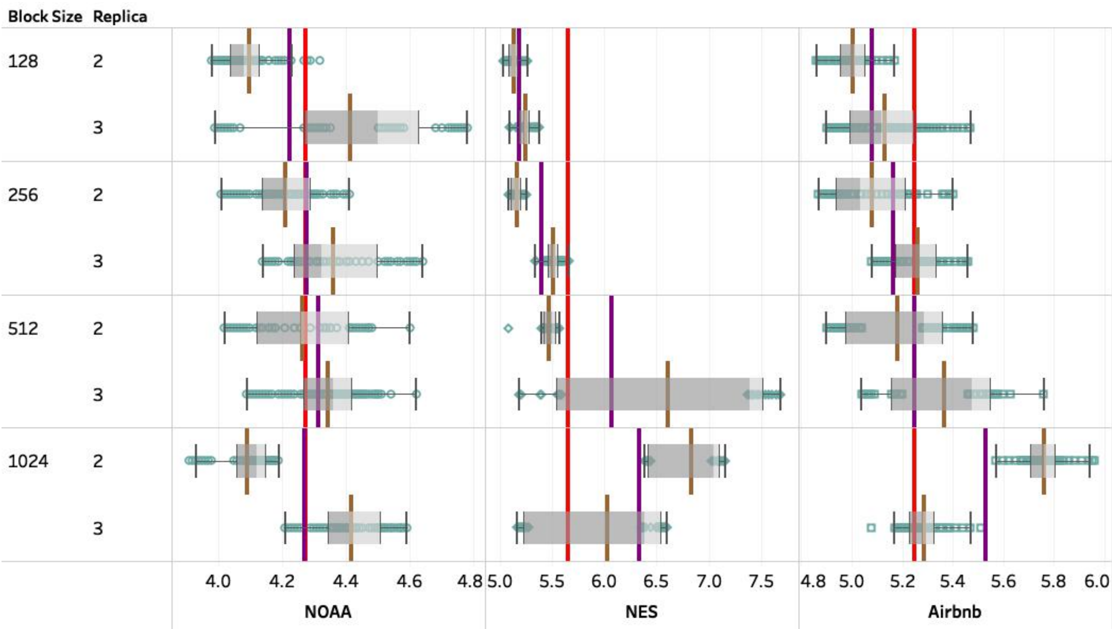

Figure C.8: Impact of block size and number of replicas on disk usage for different datasets.

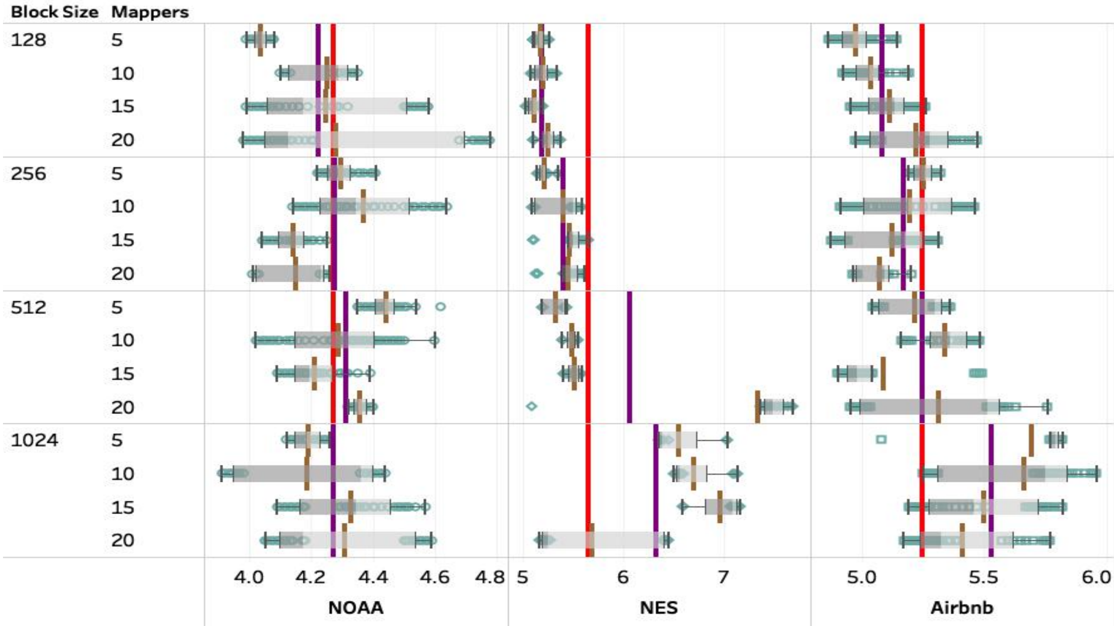

Figure C.9: Impact of block size and number of mappers on disk usage for different datasets.

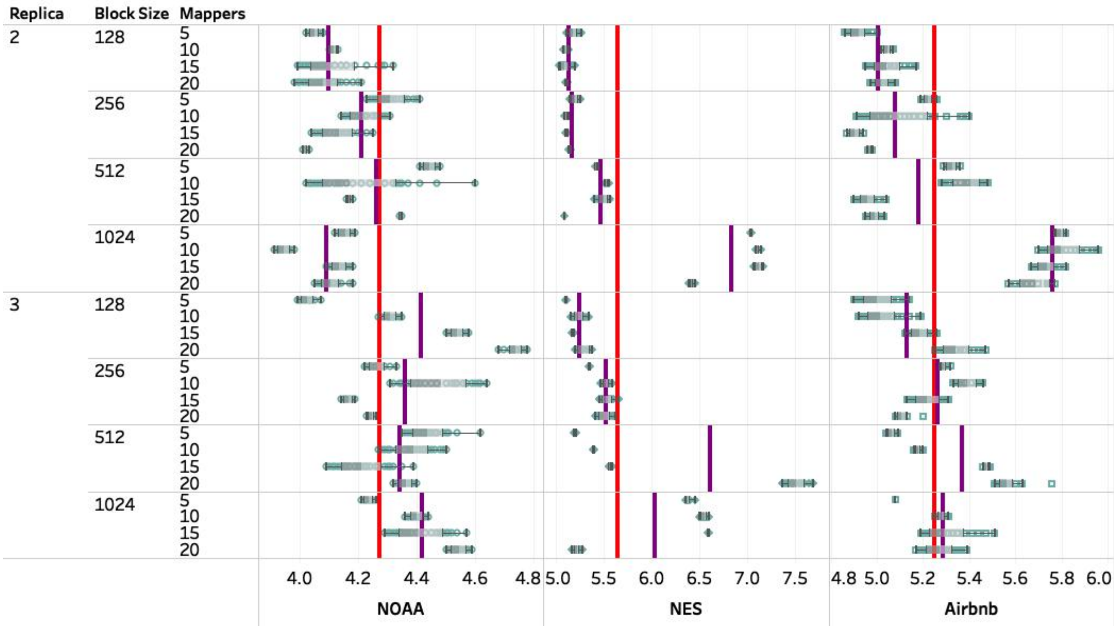

Figure C.10: Impact of Hadoop configuration parameters on disk usage for different datasets.

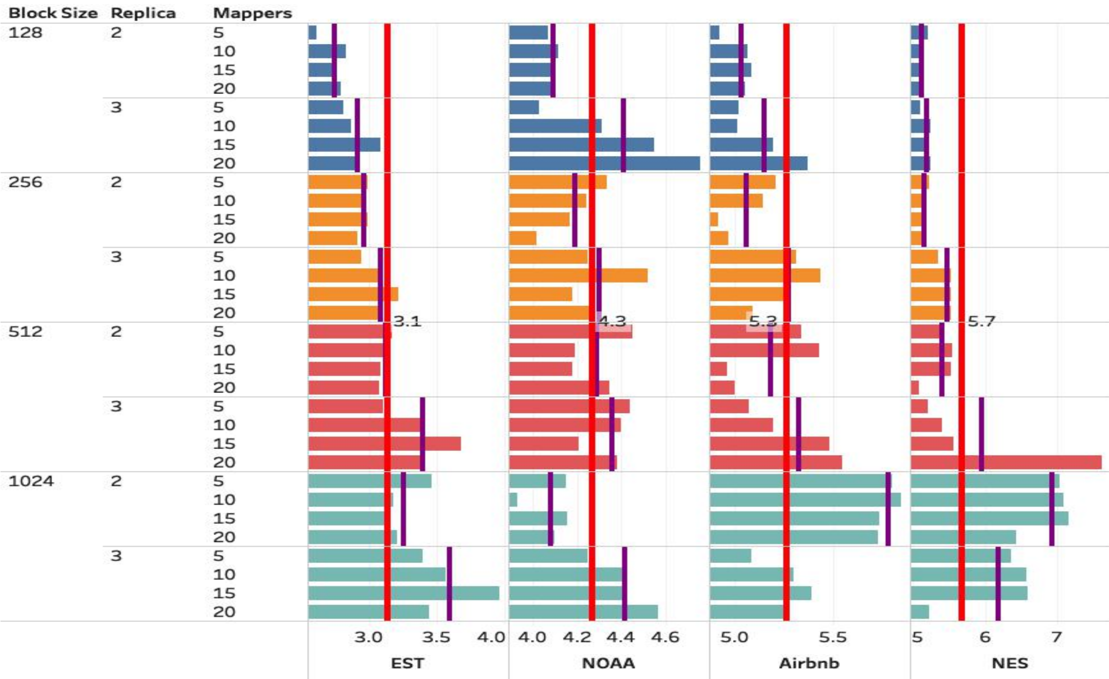

Figure C.11: Mean disk usage for different dataset.

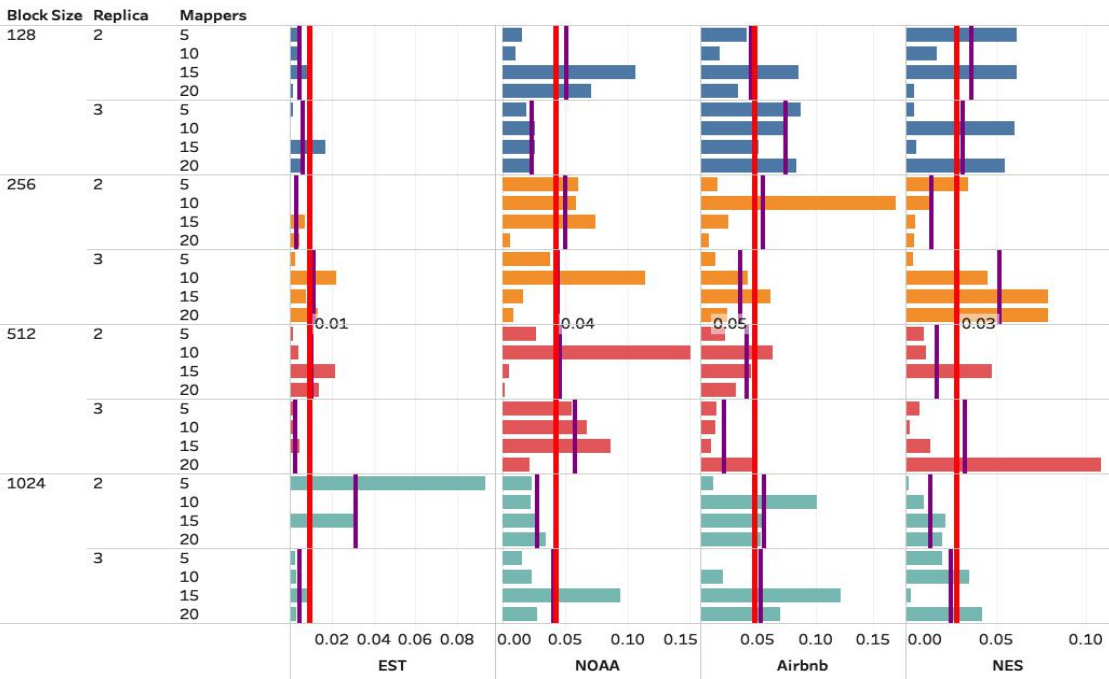

Figure C.12: Standard deviations of disk usage for different dataset.

C.2.1 Decision tree rules

1. B > 512, R $<= 2$ and O > 2, which is 14.05% for NOAA dataset, 35.15% for Airbnb dataset and 43.65% for NES dataset.
2. B $<= 512$, R $<= 2$, M $<= 10$ and N > 0.5, which is 17.06% for NOAA dataset, 19.33% for Airbnb dataset and 12.47% for NES dataset.
3. B $<= 512$, R $<= 2$, M > 10, N > 0.5 and O > 2, which is 14.85% for NOAA dataset and 6.23% for NES dataset.
4. B $<= 128$, R > 2, M $> 10 <= 15$ and O > 2, which is 12.04% for NOAA dataset and 2.92% for Airbnb dataset.
5. B $> 128 <= 256$, R > 2 and M $> 5 <= 10$, which is 14.25% for NOAA dataset, 11.71% for Airbnb dataset and 12.47% for NES dataset.
6. B $> 128 <= 256$, R > 2, M > 15, N > 0.5 and O > 2, which is 17.06% for NOAA dataset, 13.47% for Airbnb dataset and 12.68% for NES dataset.

For low disk usage,

1. B $<= 128$, R > 2, M $<= 10$ and O > 2, which is 6.49% for NOAA dataset, 13.40% for Airbnb dataset and 12.52% for NES dataset.
2. B $<= 128$, R > 2, M > 15 and O > 2, which is 12.31% for NES dataset.
3. B > 128, R > 2 and M $<= 5$, which is 5.84% for NOAA dataset, 13.40% for Airbnb dataset and 18.78% for NES dataset.

C.2.2 Disk usage linear regression models from M5P tree for different datasets

```
LM num: 1
usage =
0.0048 * No_of _columns
+ 0.0002 * Block_Size
```

```
+ 0.008 * Replica
- 0.0013 * Mappers
- 0.0613

LM num: 2
usage =
0.0058 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
+ 0.0142 * Mappers
- 0.1719

LM num: 3
usage =
0.0907 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
- 0.0459 * Mappers
+ 0.0602

LM num: 4
usage =
0.0029 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
- 0.0017 * Mappers
- 0.0393

LM num: 5
usage =
-0.0005 * No_of _columns
+ 0.0001 * Block_Size
+ 0.008 * Replica
```

```
- 0.0008 * Mappers
- 0.0142

LM num: 6
usage =
-0.0023 * No_of _columns
+ 0.0005 * Block_Size
+ 0.008 * Replica
+ 0.0015 * Mappers
- 0.048

LM num: 7
usage =
-0.0296 * No_of _columns
+ 0.0005 * Block_Size
+ 0.008 * Replica
+ 0.0015 * Mappers
+ 0.5242

LM num: 8
usage =
0.0083 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
+ 0.0121 * Mappers
- 0.1255

LM num: 9
usage =
0.0083 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
- 0.0153 * Mappers
```

```
+ 0.7335

LM num: 10
usage =
0.0171 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
+ 0.006 * Mappers
+ 0.6196

LM num: 11
usage =
-0.0043 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
+ 0.004 * Mappers
+ 0.0406

LM num: 12
usage =
-0.0038 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
- 0.0069 * Mappers
+ 0.9792

LM num: 13
usage =
-0.0065 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
- 0.0781 * Mappers
+ 1.9686
```

```
LM num: 14
usage =
-0.0102 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
+ 0.1029 * Mappers
- 1.3092

LM num: 15
usage =
0.0019 * No_of _columns
+ 0.0002 * Block_Size
+ 0.008 * Replica
- 0.004 * Mappers
+ 0.9162

LM num: 16
usage =
-0 * No_of _columns
+ 0 * Block_Size
+ 0.0126 * Replica
- 0.0057 * Mappers
+ 0.972

LM num: 17
usage =
-0 * No_of _columns
+ 0 * Block_Size
+ 0.0126 * Replica
- 0.0057 * Mappers
+ 0.4634
```

```
LM num: 18
usage =
-0 * No_of _columns
+ 0 * Block_Size
+ 0.0126 * Replica
+ 0.0582 * Mappers
+ 0.2091

LM num: 19
usage =
-0 * No_of _columns
+ 0 * Block_Size
+ 0.0126 * Replica
+ 0.0027 * Mappers
+ 0.8828

LM num: 20
usage =
-0 * No_of _columns
+ 0 * Block_Size
- 0.0048 * Replica
+ 0.0006 * Mappers
+ 0.9778

LM num: 21
usage =
-0 * No_of _columns
+ 0 * Block_Size
- 0.0048 * Replica
+ 0.0048 * Mappers
+ 0.5408

LM num: 22
```

```
usage =
-0 * No_of _columns
+ 0 * Block_Size
- 0.0048 * Replica
- 0.006 * Mappers
+ 1.0383

LM num: 23
usage =
0.0035 * No_of _columns
- 0.0001 * Block_Size
+ 0.0106 * Replica
- 0.0012 * Mappers
+ 0.8138

LM num: 24
usage =
0.0035 * No_of _columns
- 0.0001 * Block_Size
+ 0.0106 * Replica
- 0.0284 * Mappers
+ 0.6653

LM num: 25
usage =
0.0035 * No_of _columns
- 0.0001 * Block_Size
+ 0.0106 * Replica
+ 0.0139 * Mappers
+ 0.5244

LM num: 26
usage =
```

```
0.0035 * No_of _columns
- 0.0001 * Block_Size
+ 0.0106 * Replica
- 0.0012 * Mappers
+ 0.1313

LM num: 27
usage =
0.0035 * No_of _columns
+ 0 * Block_Size
+ 0.0106 * Replica
- 0.0033 * Mappers
+ 0.8988

LM num: 28
usage =
0.0035 * No_of _columns
+ 0.0015 * Block_Size
+ 0.0106 * Replica
- 0.0066 * Mappers
- 0.5416

LM num: 29
usage =
-0.0063 * No_of _columns
- 0.0002 * Block_Size
+ 0.0106 * Replica
- 0.0003 * Mappers
+ 1.0626

LM num: 30
usage =
-0.0063 * No_of _columns
```

```
+ 0.0015 * Block_Size
+ 0.0106 * Replica
- 0.0003 * Mappers
- 0.4644

LM num: 31
usage =
-0.0003 * No_of _columns
- 0 * Block_Size
+ 0.0106 * Replica
- 0.0028 * Mappers
+ 0.9859

LM num: 32
usage =
-0.0008 * No_of _columns
+ 0.0001 * Block_Size
+ 0.0106 * Replica
+ 0.0928 * Mappers
- 1.0039

LM num: 33
usage =
-0.0008 * No_of _columns
+ 0.0001 * Block_Size
+ 0.0106 * Replica
+ 0.0076 * Mappers
+ 0.7005

LM num: 34
usage =
-0.0008 * No_of _columns
- 0.0002 * Block_Size
```

```
+ 0.0106 * Replica
- 0.0183 * Mappers
+ 1.3358

LM num: 35
usage =
-0.0008 * No_of _columns
- 0.0002 * Block_Size
+ 0.0106 * Replica
- 0.14 * Mappers
+ 3.1619
```

C.3 Memory usage

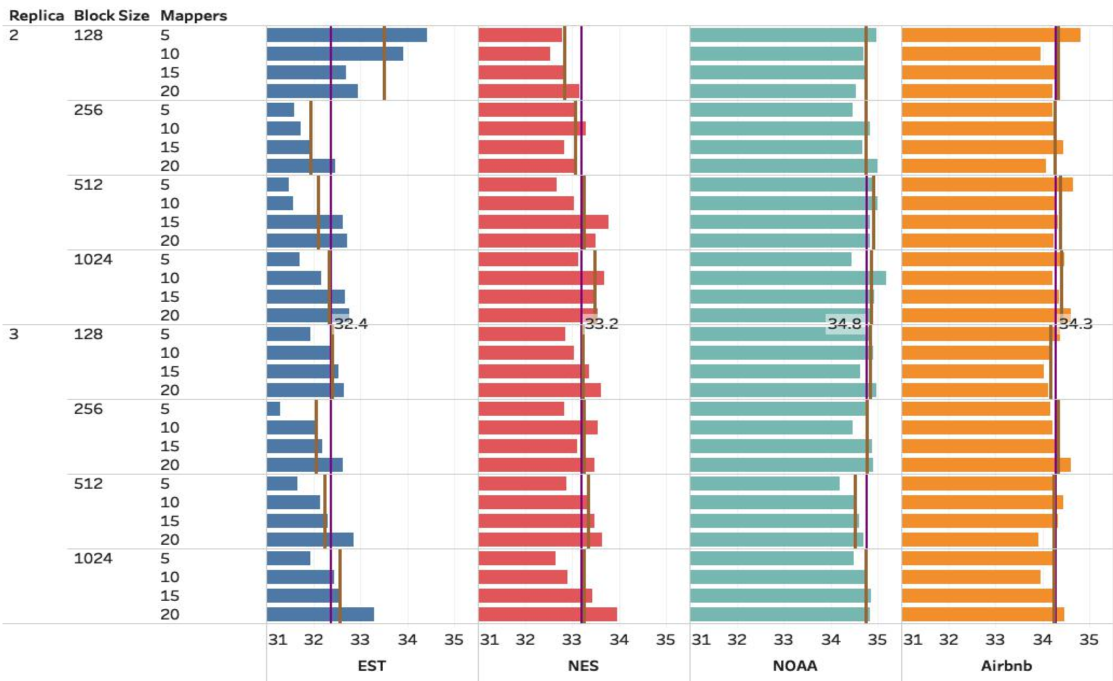

Figure C.14: Mean memory usage for different datasets.

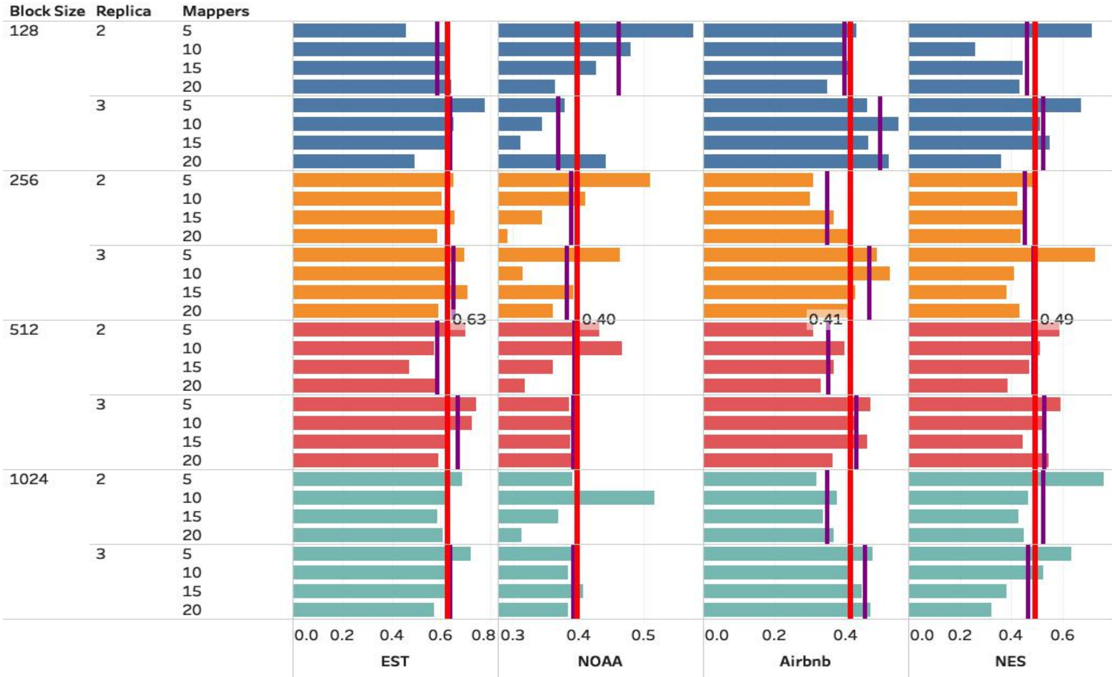

Figure C.15: Standard deviations of memory usage for different datasets.

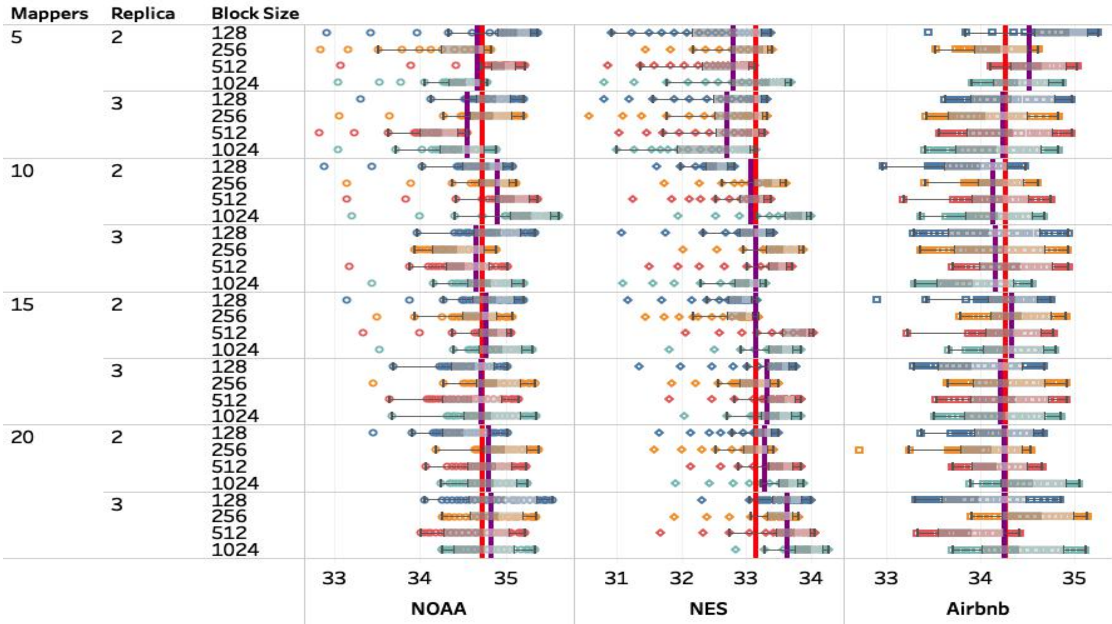

Figure C.16: Impact of Hadoop configuration parameters on memory usage for different datasets.

C.3.1 Decision tree rules

For low memory usage,

1. B $<= 128$ MB, R > 2 and M $<= 5$ which is 2.71% for NOAA, 2.92% for Airbnb dataset and 5.6% for NES dataset.

2. B $> 128 <= 256$ MB, R $<= 2$, M $> 10 <= 15$ N > 3 and O > 1 that makes 11.48% for NOAA, 12.94% for Airbnb and 20.62% for NES dataset.

3. B $> 128 <= 512$MB and M $> 5 <= 10$ that comprises of 38% NOAA, 35.49% Airbnb and 38.33% of NES of the total number of instances.

4. B > 512 MB, R $<= 2$ and M $> 5 <= 10$ which is 7.09% for NOAA, 10.85% for Airbnb and 6.6% for NES dataset.

High memory usage,

1. M $> 10 <= 15$, N > 3 and R > 2 that is 21.62% for NOAA, 23.07% for Airbnb dataset and 29.11% for NES dataset.

2. B $<= 128$ MB, N > 3, O > 1, R $<= 2$ and M$> 10 <= 15$ which is 6.8% for NOAA and 5.19% for Airbnb.

3. B > 512 MB, R > 2 and M $> 5 <= 10$ which is NOAA's 4.78% and Airbnb's 11.64%.

4. B $<= 128$MB R > 2 and M $> 5 <= 10$ and O > 1 that is 7.48% for NOAA and 7.27% for Airbnb dataset.

C.3.2 Memory usage linear regression models from M5P tree for different datasets

```
LM num: 1
usage =
-0.0301 * No_of _columns
- 0 * Block_Size
```

```
- 0.3569 * Replica
+ 0.0003 * Mappers
+ 1.7823

LM num: 2
usage =
0.0151 * No_of _columns
- 0 * Block_Size
- 0.0068 * Replica
+ 0.0003 * Mappers
+ 0.1232

LM num: 3
usage =
0.0656 * No_of _columns
- 0 * Block_Size
- 0.0068 * Replica
+ 0.0003 * Mappers
+ 0.0262

LM num: 4
usage =
-0.0078 * No_of _columns
- 0 * Block_Size
- 0.0068 * Replica
+ 0.0003 * Mappers
+ 0.289

LM num: 5
usage =
-0.0479 * No_of _columns
- 0 * Block_Size
- 0.0068 * Replica
```

```
+ 0.0003 * Mappers
+ 0.615

LM num: 6
usage =
0.0052 * No_of _columns
- 0 * Block_Size
- 0.0068 * Replica
+ 0.0003 * Mappers
+ 0.4364

LM num: 7
usage =
-0.0161 * No_of _columns
- 0 * Block_Size
- 0.0068 * Replica
+ 0.0003 * Mappers
+ 0.4219

LM num: 8
usage =
-0.0001 * No_of _columns
- 0.001 * Block_Size
+ 0.0019 * Replica
+ 0.0004 * Mappers
+ 0.6844

LM num: 9
usage =
-0.0004 * No_of _columns
+ 0.0006 * Block_Size
+ 0.0479 * Replica
- 0.0147 * Mappers
```

```
+ 0.0671

LM num: 10
usage =
-0.0004 * No_of _columns
+ 0.0006 * Block_Size
+ 0.0479 * Replica
- 0.1058 * Mappers
+ 1.4332

LM num: 11
usage =
-0.0004 * No_of _columns
+ 0.0001 * Block_Size
+ 0.0479 * Replica
+ 0.0531 * Mappers
- 0.2473

LM num: 12
usage =
-0.0004 * No_of _columns
+ 0.0001 * Block_Size
+ 0.0479 * Replica
- 0.064 * Mappers
+ 1.2272

LM num: 13
usage =
0.0003 * No_of _columns
+ 0 * Block_Size
+ 0.1408 * Replica
+ 0.0013 * Mappers
+ 0.0704
```

```
LM num: 14
usage =
-0.0331 * No_of _columns
+ 0.0001 * Block_Size
- 0.0391 * Replica
+ 0.0013 * Mappers
+ 1.1237

LM num: 15
usage =
-0.0044 * No_of _columns
+ 0.0001 * Block_Size
- 0.0391 * Replica
+ 0.0013 * Mappers
+ 0.858

LM num: 16
usage =
0.0297 * No_of _columns
- 0.0001 * Block_Size
- 0.0391 * Replica
+ 0.0013 * Mappers
+ 0.197

LM num: 17
usage =
-0.0314 * No_of _columns
- 0.0001 * Block_Size
- 0.0391 * Replica
+ 0.0013 * Mappers
+ 0.9769
```

```
LM num: 18
usage =
0.0128 * No_of _columns
+ 0 * Block_Size
- 0.0969 * Replica
+ 0.0013 * Mappers
+ 0.7303

LM num: 19
usage =
-0.0208 * No_of _columns
+ 0.0001 * Block_Size
+ 0.0014 * Replica
+ 0.0003 * Mappers
+ 0.6947

LM num: 20
usage =
0.0006 * No_of _columns
+ 0.0001 * Block_Size
+ 0.027 * Replica
+ 0.0003 * Mappers
+ 0.3936

LM num: 21
usage =
0.0006 * No_of _columns
+ 0.0001 * Block_Size
+ 0.027 * Replica
+ 0.0003 * Mappers
+ 0.6449

LM num: 22
```

```
usage =
0.0006 * No_of _columns
+ 0 * Block_Size
+ 0.027 * Replica
+ 0.0003 * Mappers
+ 0.7387
```

C.4 Network usage

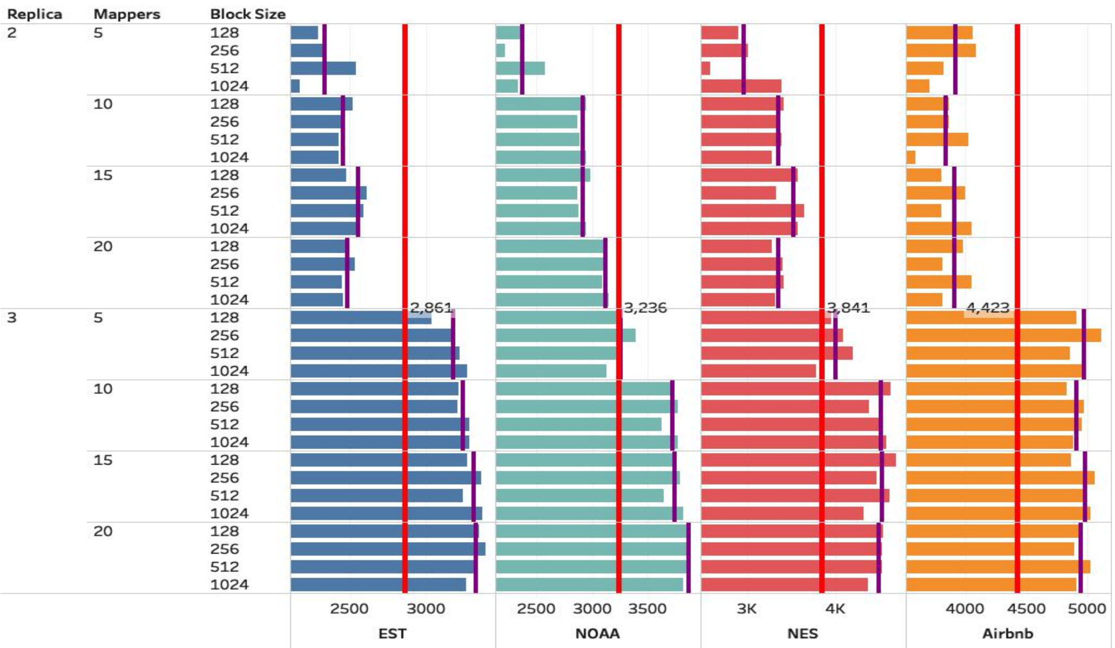

Figure C.18: Mean network usage for different datasets

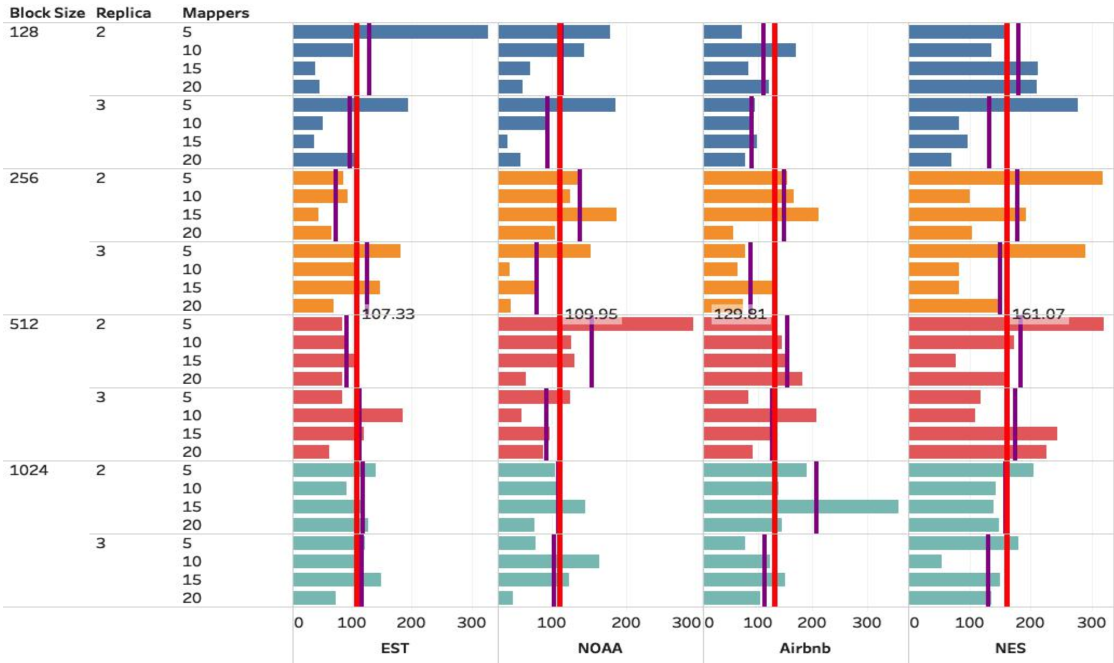

Figure C.19: Standard deviations of network usage for different datasets

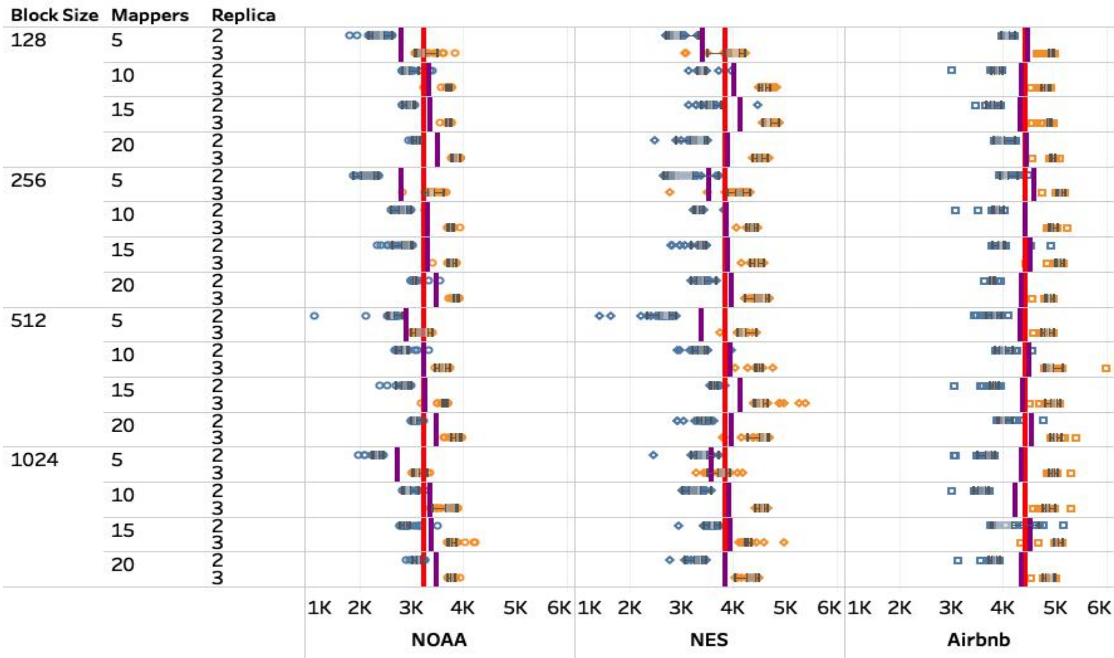

Figure C.20: Impact of Hadoop configuration parameters on network usage for different datasets.

C.4.1 Decision tree rules

For low network usage,

1. B > 512, R $<= 2$, M $<= 5$, N 0.53, which is 12.5% for both NOAA and Airbnb datasets, and 12.29% for NES dataset.

2. B $<= 256$, R $<= 2$, M $<= 5$, N > 0.5 and O > 1, which is also 12.5% for both NOAA and Airbnb datasets, and 12.29% for NES dataset.

3. B > 128, R $<= 2$, M $> 5 <= 10$, N > 0.5 and O > 2, which is 48.95% for NOAA dataset, 50% for Airbnb dataset and 48.95% for NES dataset.

C.4.2 Network usage linear regression models from M5P tree for different datasets

```
LM num: 1
Usage =
-0.0003 * No_of _columns
- 0.0014 * Block_Size
+ 0.0061 * Replica
+ 0.0365 * Mappers
+ 0.1698

LM num: 2
Usage =
-0.0003 * No_of _columns
- 0 * Block_Size
+ 0.0061 * Replica
- 0.0504 * Mappers
+ 0.6466

LM num: 3
Usage =
```

```
-0.0003 * No_of _columns
- 0 * Block_Size
+ 0.0061 * Replica
+ 0.0008 * Mappers
+ 0.0574

LM num: 4
Usage =
-0.0003 * No_of _columns
- 0 * Block_Size
+ 0.0061 * Replica
- 0.0166 * Mappers
+ 0.6898

LM num: 5
Usage =
-0.0004 * No_of _columns
- 0 * Block_Size
+ 0.0061 * Replica
+ 0.004 * Mappers
- 0.0152

LM num: 6
Usage =
-0.0034 * No_of _columns
- 0 * Block_Size
+ 0.0061 * Replica
+ 0.0008 * Mappers
+ 0.3849

LM num: 7
Usage =
-0.0034 * No_of _columns
```

```
+ 0.0004 * Block_Size
+ 0.0061 * Replica
+ 0.0008 * Mappers
+ 0.0385

LM num: 8
Usage =
-0.0034 * No_of _columns
- 0 * Block_Size
+ 0.0061 * Replica
+ 0.0008 * Mappers
+ 0.0511

LM num: 9
Usage =
-0.0004 * No_of _columns
- 0 * Block_Size
+ 0.0061 * Replica
+ 0.0003 * Mappers
+ 0.9767

LM num: 10
Usage =
0.0147 * No_of _columns
- 0 * Block_Size
+ 0.0061 * Replica
+ 0.0003 * Mappers
+ 0.6231

LM num: 11
Usage =
0.0345 * No_of _columns
- 0.0001 * Block_Size
```

```
+ 0.0061 * Replica
+ 0.0003 * Mappers
+ 0.0709

LM num: 12
Usage =
0.0004 * No_of _columns
- 0 * Block_Size
+ 0.0061 * Replica
+ 0.0001 * Mappers
+ 0.9707
```

C.5 Execution time

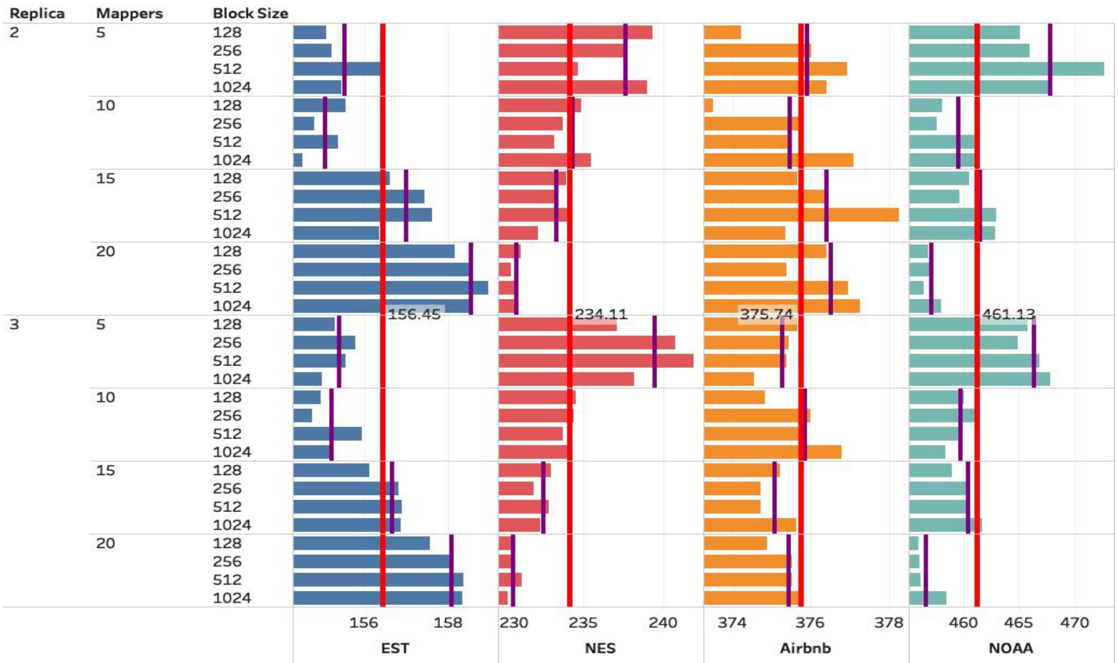

Figure C.22: Mean execution time for different datasets

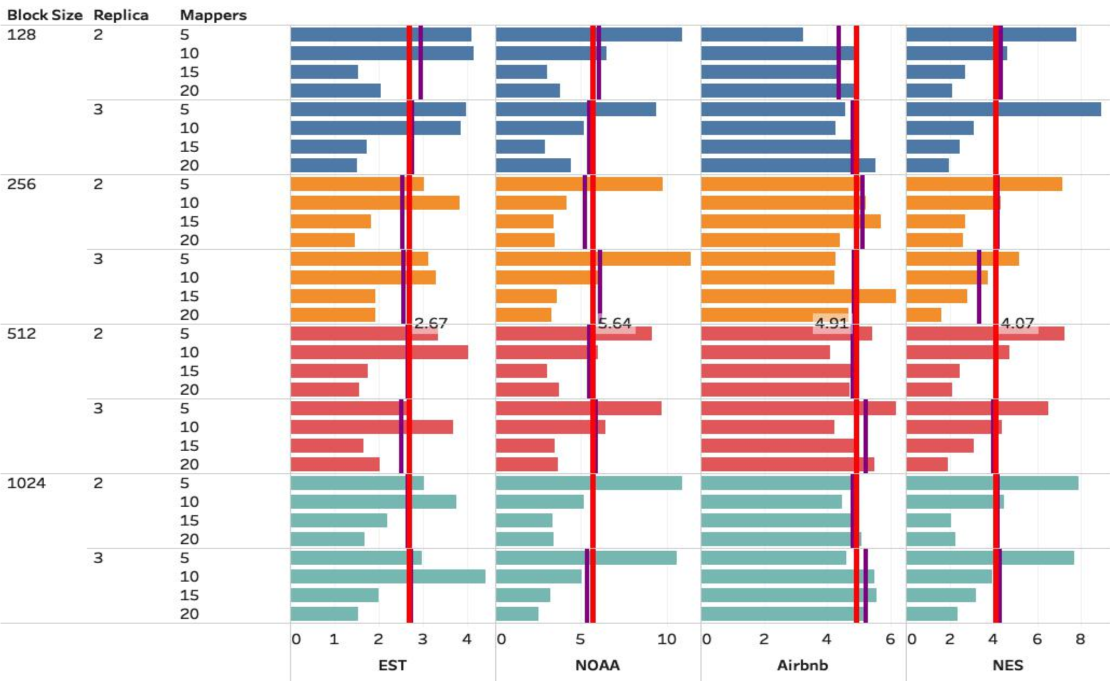

Figure C.23: Standard deviations of execution time for different datasets

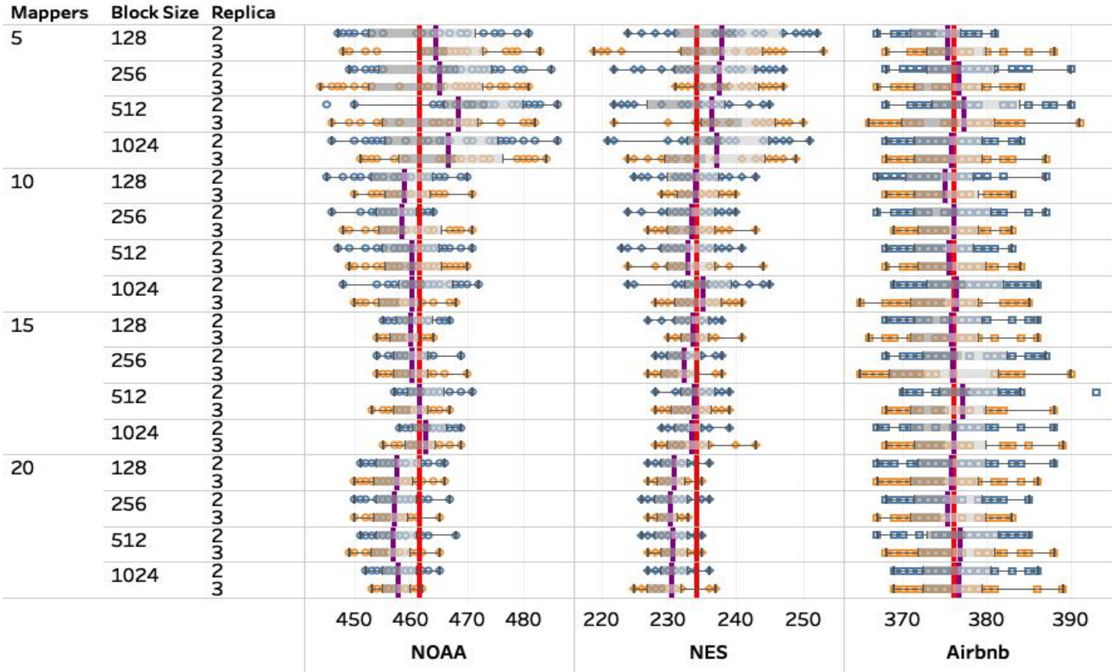

Figure C.24: Impact of Hadoop configuration parameters on overall execution time for different datasets.

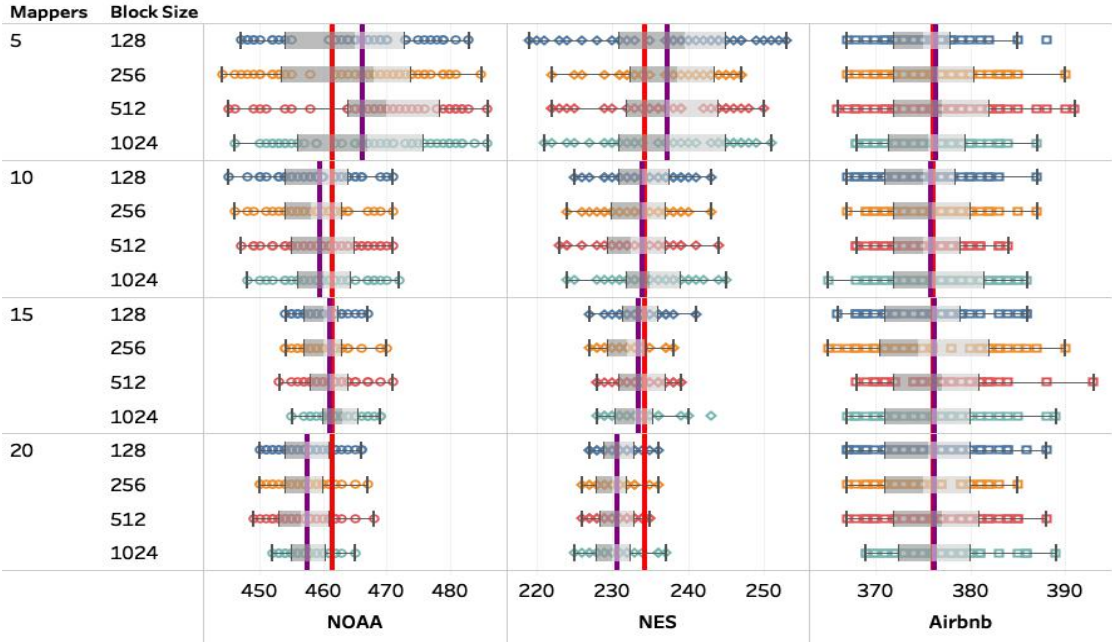

Figure C.25: Impact of number of mappers and block size on overall execution time for different datasets.

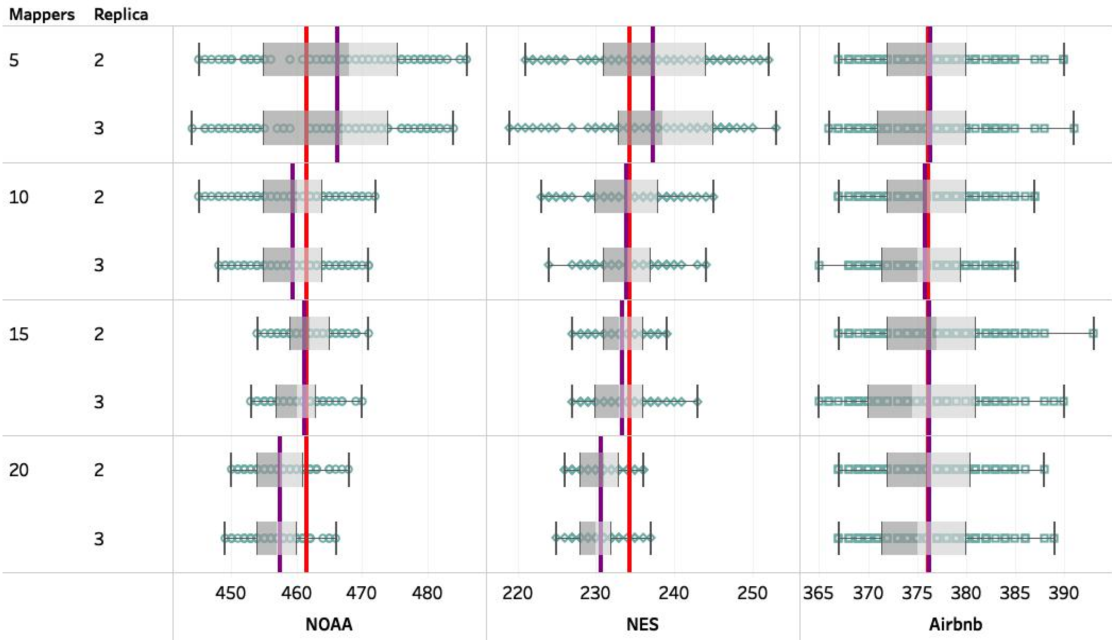

Figure C.26: Impact of number of mappers and number of replicas on overall execution time for different datasets.

C.5.1 Decision tree rules

For high execution time,

1. N > 3 and M > 15, which is 11.70% for NOAA dataset, 21.49% for Airbnb dataset and 32.06% for NES dataset of the total instances.

2. N > 3, M $> 10 <= 15$ and O > 1, which is 53.74% for NOAA, 49.34% for Airbnb dataset and 53.95% for NES dataset.

C.5.2 Execution time linear regression models from M5P tree for different datasets

```
LM num: 1
usage =
-0.0001 * No_of _columns
- 0.0281 * Replica
- 0.0001 * Mappers
+ 0.8446

LM num: 2
usage =
-0.0001 * No_of _columns
- 0.0005 * Replica
- 0.0195 * Mappers
+ 0.8344

LM num: 3
usage =
-0.0001 * No_of _columns
- 0.0011 * Replica
- 0.0011 * Mappers
+ 0.2594
```

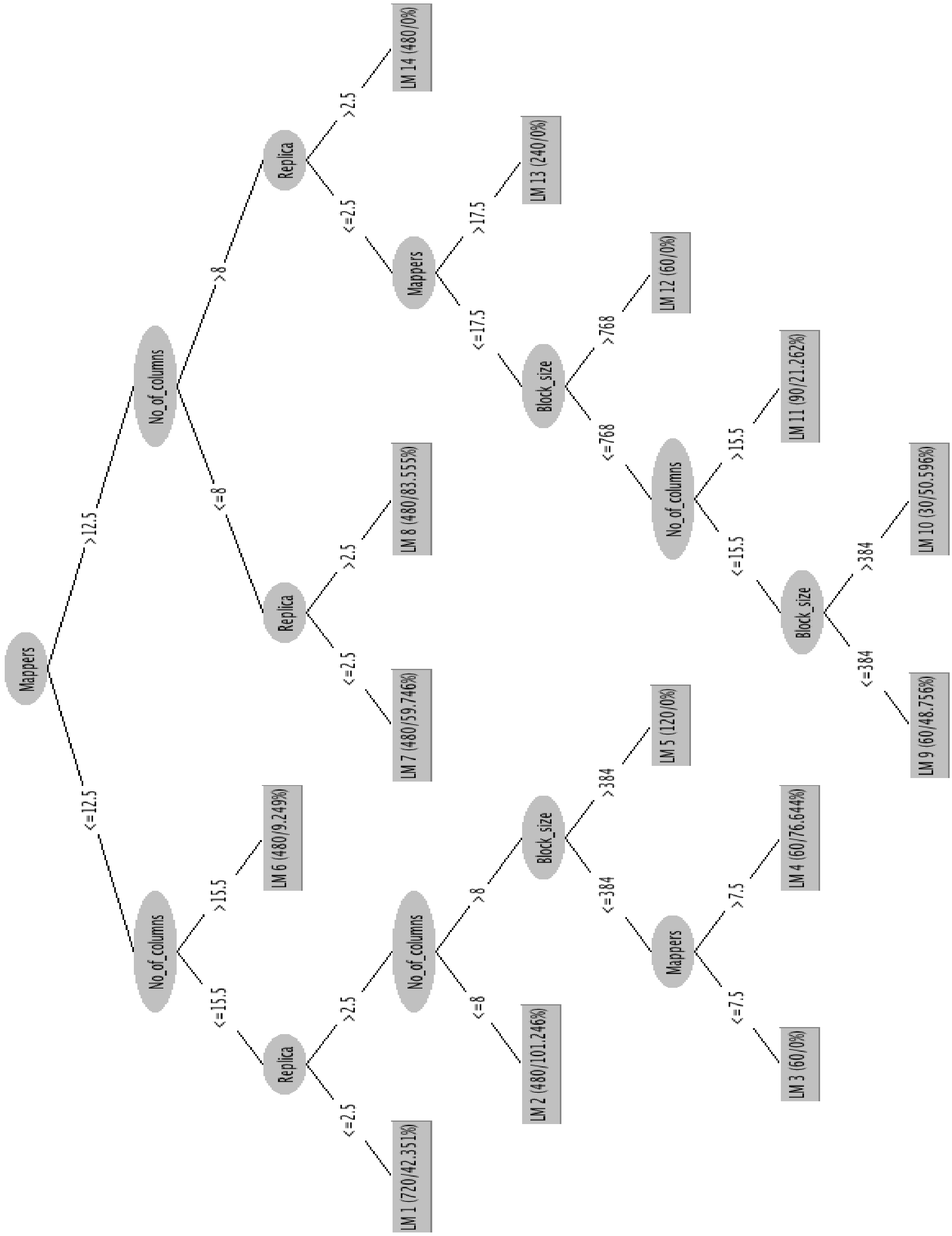

Figure C.7: M5P tree for CPU usage.

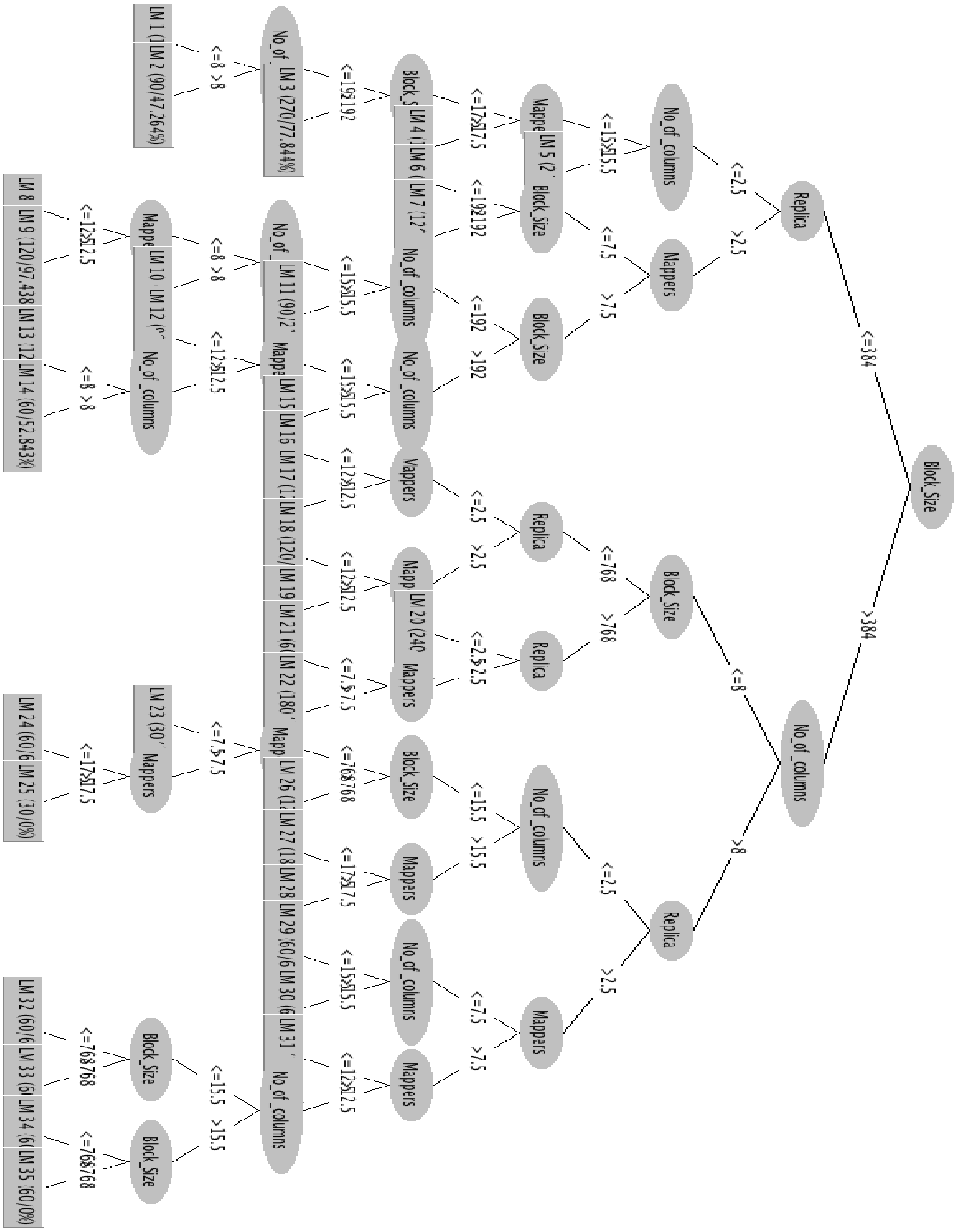

Figure C.13: M5P tree for disk usage.

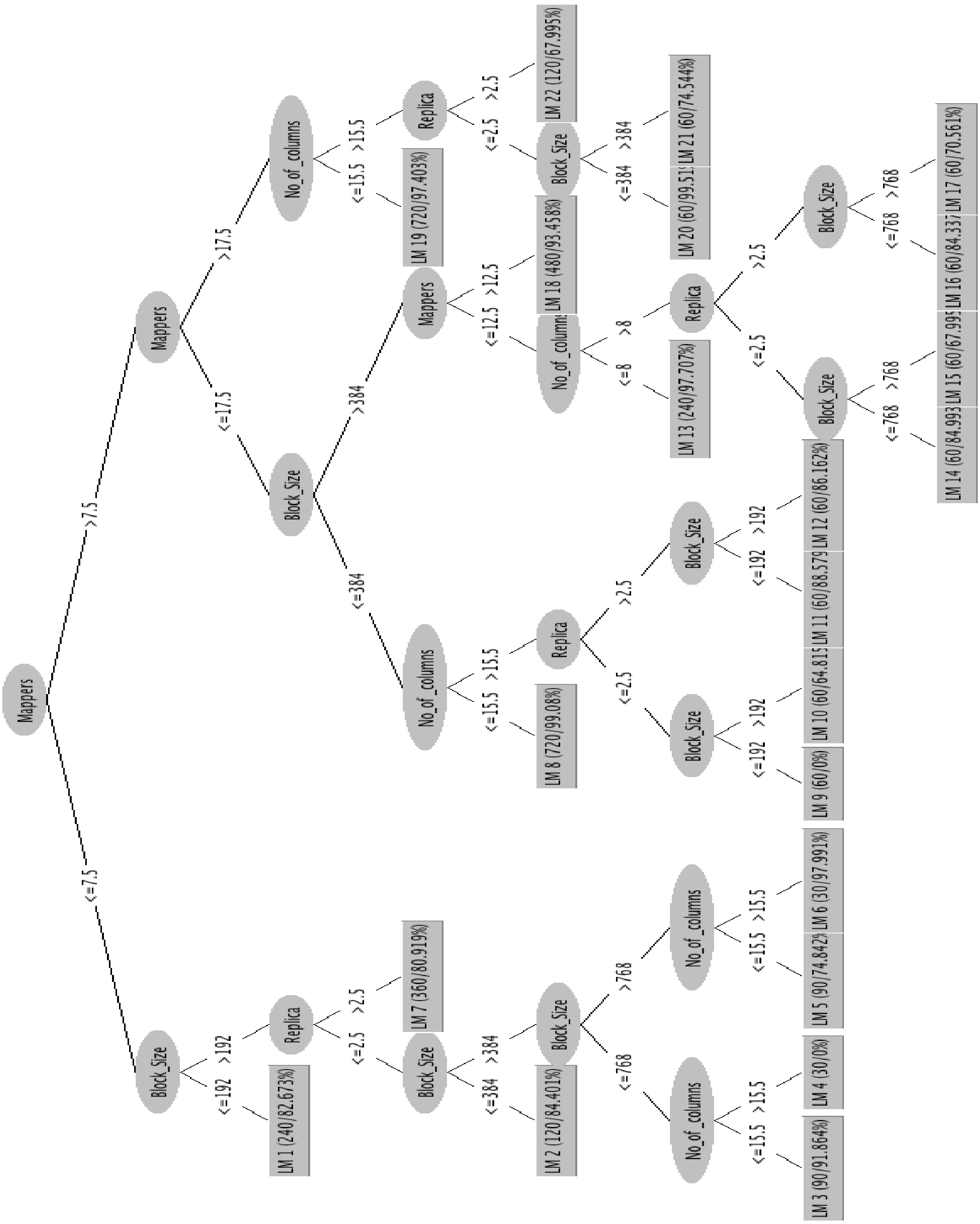

Figure C.17: M5P tree for memory usage.

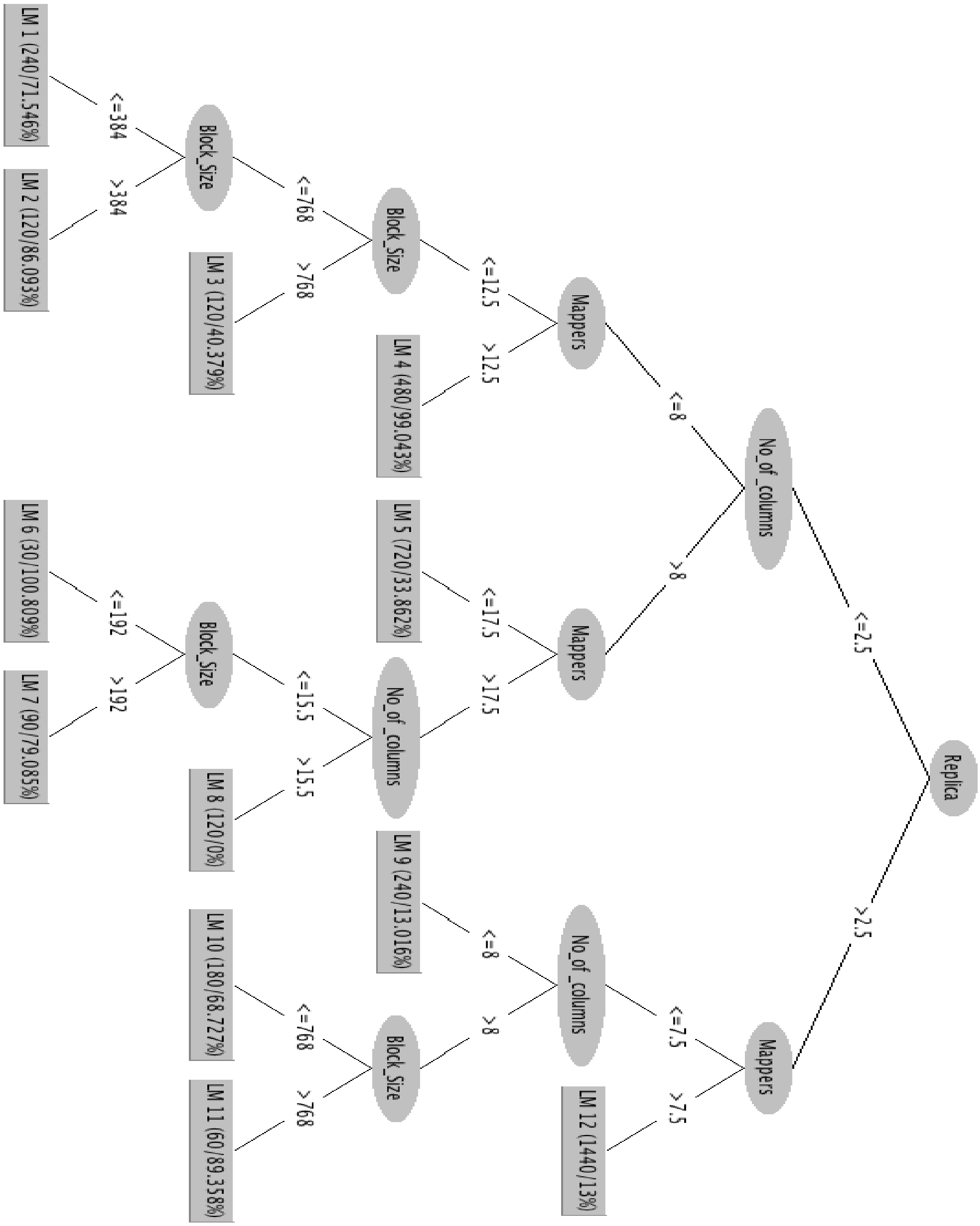

Figure C.21: M5P tree for network usage

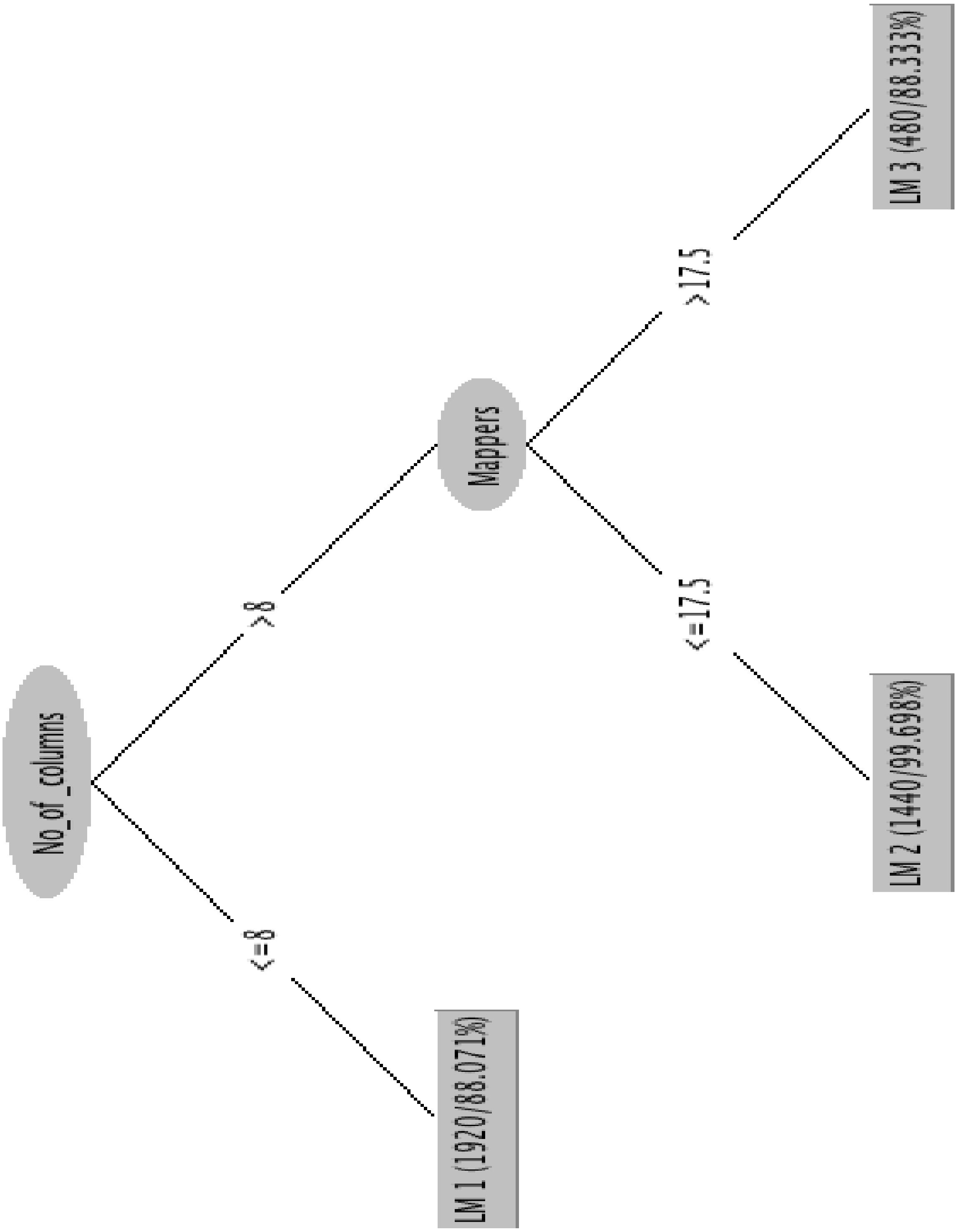

Figure C.27: M5P tree for execution time

Appendix D

Modelling and prediction of resource utilization of Hadoop cluster for different cluster sizes

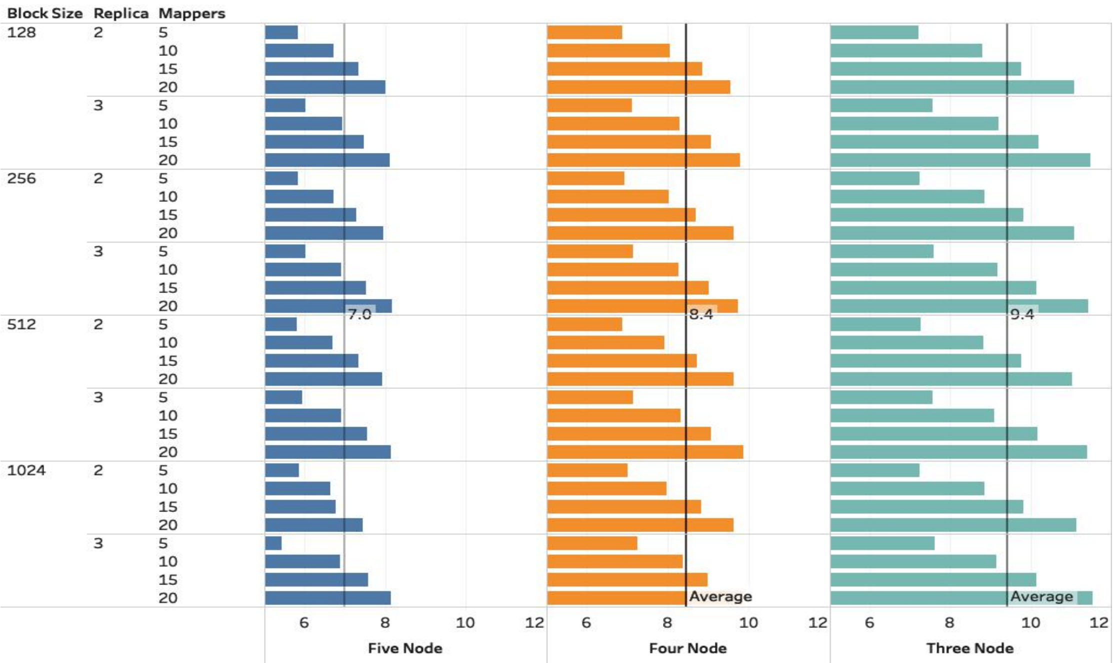

Figure D.1: Mean CPU percentage usage for different cluster sizes

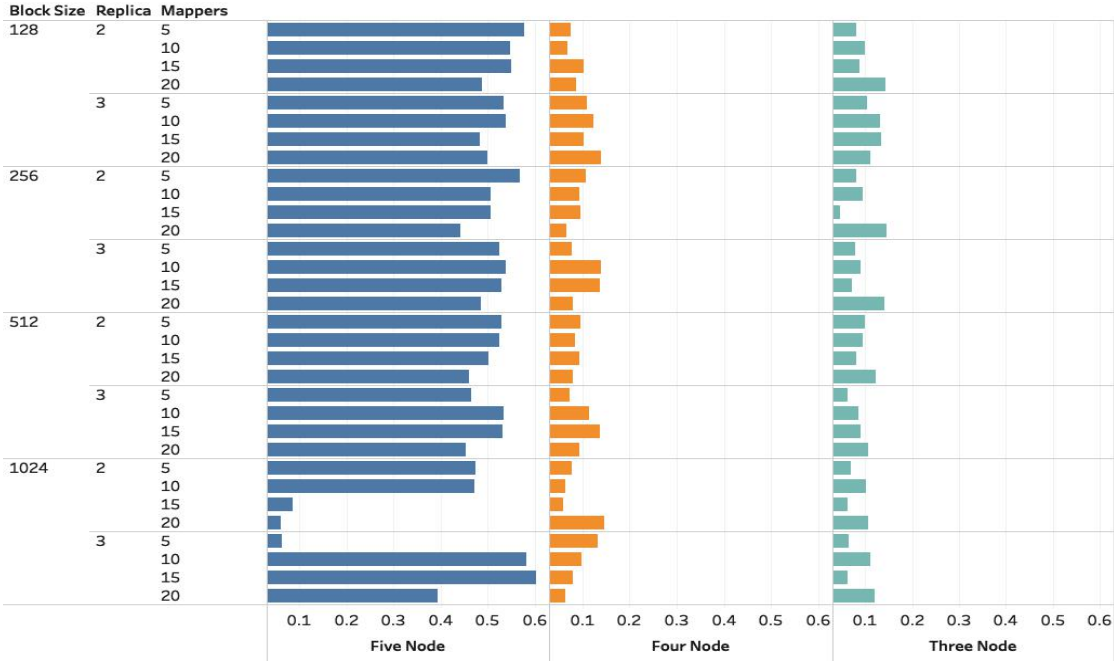

Figure D.2: Standard deviation of CPU usage for different cluster sizes

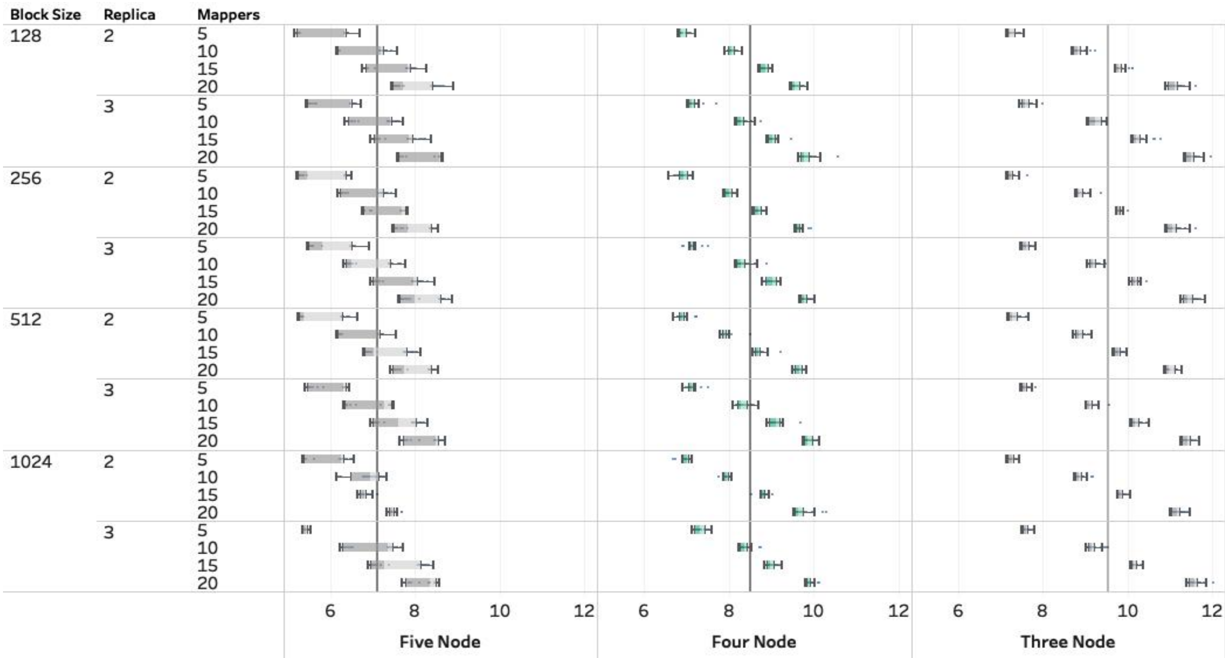

Figure D.3: Box plot for CPU usage for different environmental changes

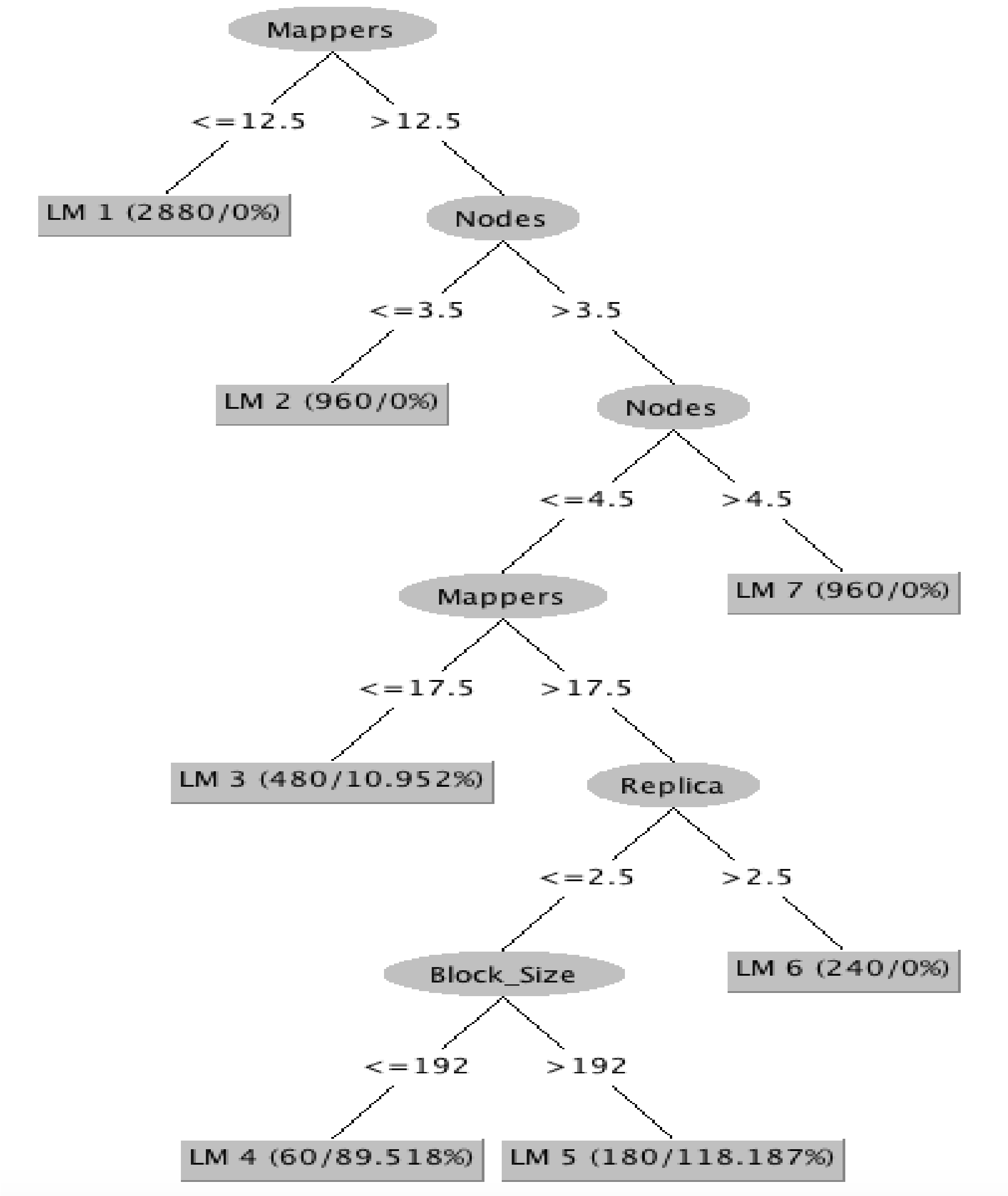

Figure D.4: M5P regression tree of CPU usage for changing cluster sizes

D.1 CPU usage linear regression models from M5P tree for different cluster sizes

```
LM num: 2
Usage =
-0.0089 * Nodes
+ 0.002 * Replica
+ 0.0009 * Mappers
+ 1.0038

LM num: 3
Usage =
-0.0103 * Nodes
+ 0 * Block_Size
+ 0.0132 * Replica
+ 0.0055 * Mappers
- 0.0667

LM num: 4
Usage =
-0.0103 * Nodes
+ 0 * Block_Size
+ 0.0493 * Replica
+ 0.0055 * Mappers
+ 0.0434

LM num: 5
Usage =
-0.0103 * Nodes
+ 0 * Block_Size
+ 0.0493 * Replica
+ 0.0055 * Mappers
```

```
+ 0.2318

LM num: 6
Usage =
-0.0103 * Nodes
+ 0 * Block_Size
+ 0.0493 * Replica
+ 0.0055 * Mappers
+ 0.7636

LM num: 7
Usage =
-0.0103 * Nodes
+ 0.0036 * Replica
+ 0.0016 * Mappers
+ 0.0154
```

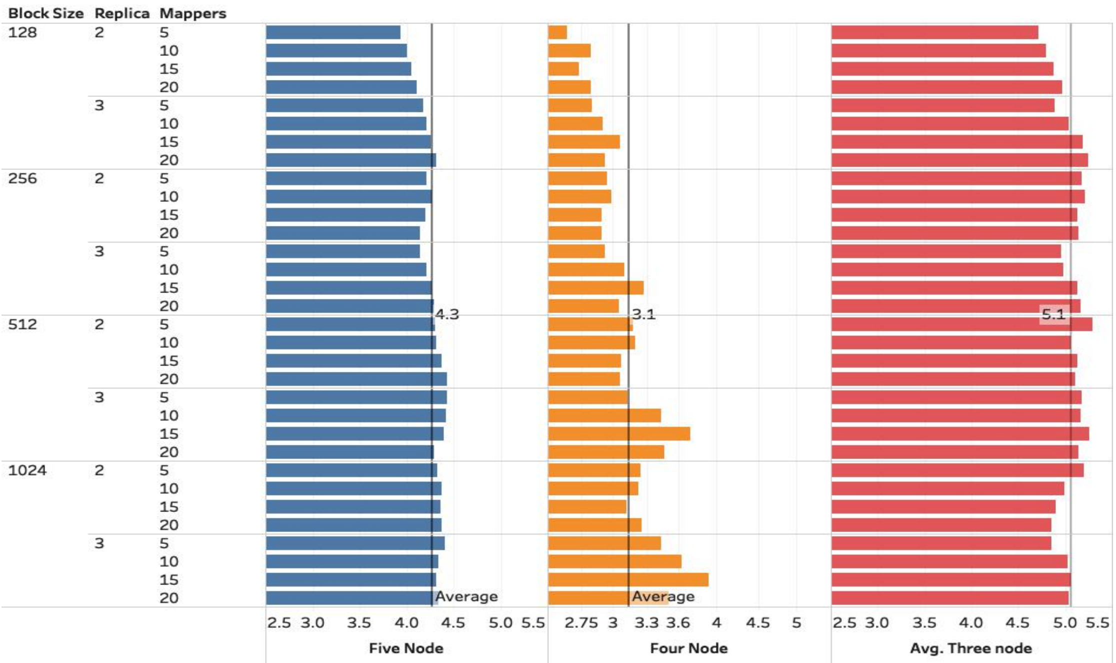

Figure D.5: Mean disk percentage usage for different cluster sizes

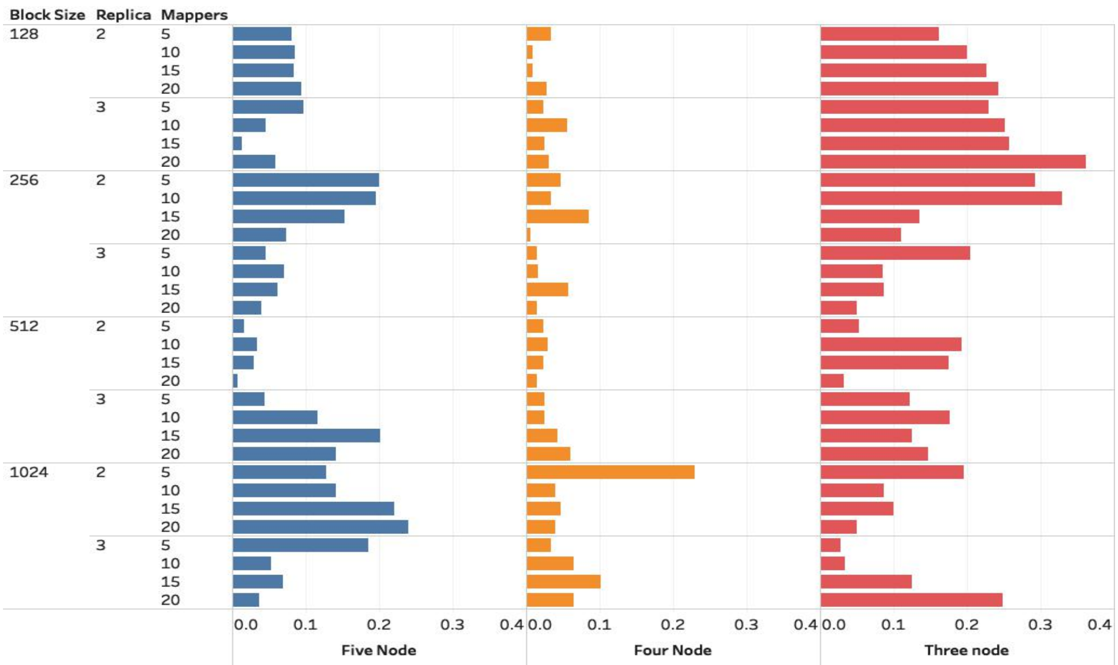

Figure D.6: Standard deviation of disk usage for different cluster sizes

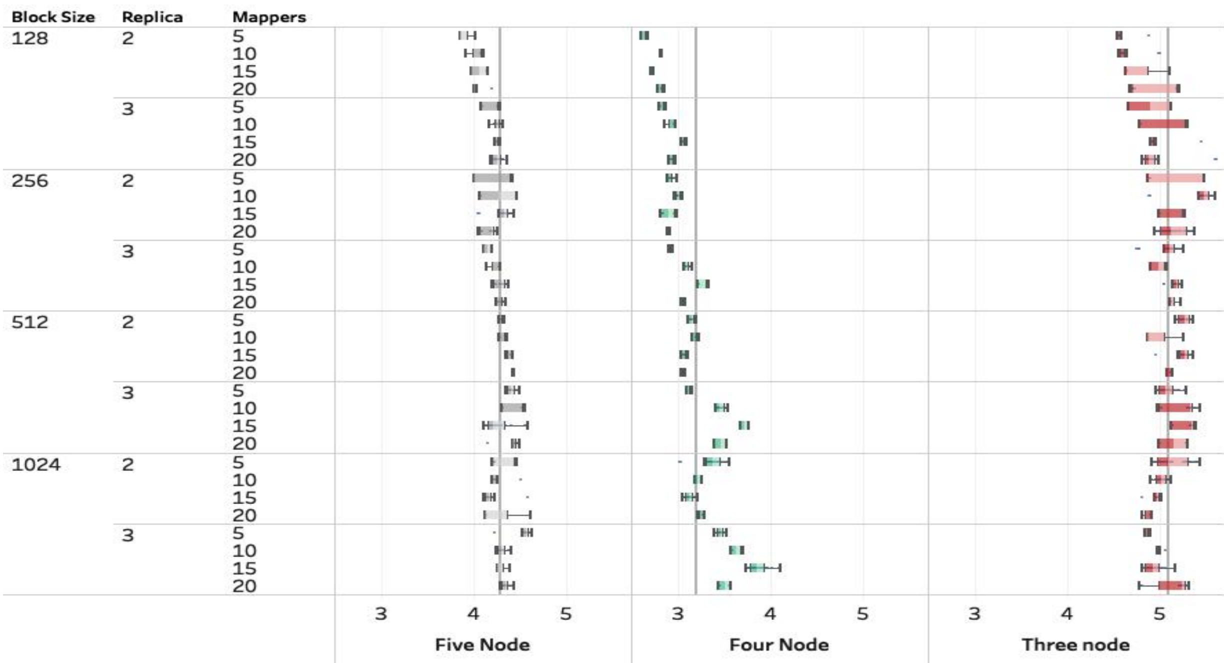

Figure D.7: Box plot for disk usage for different cluster sizes

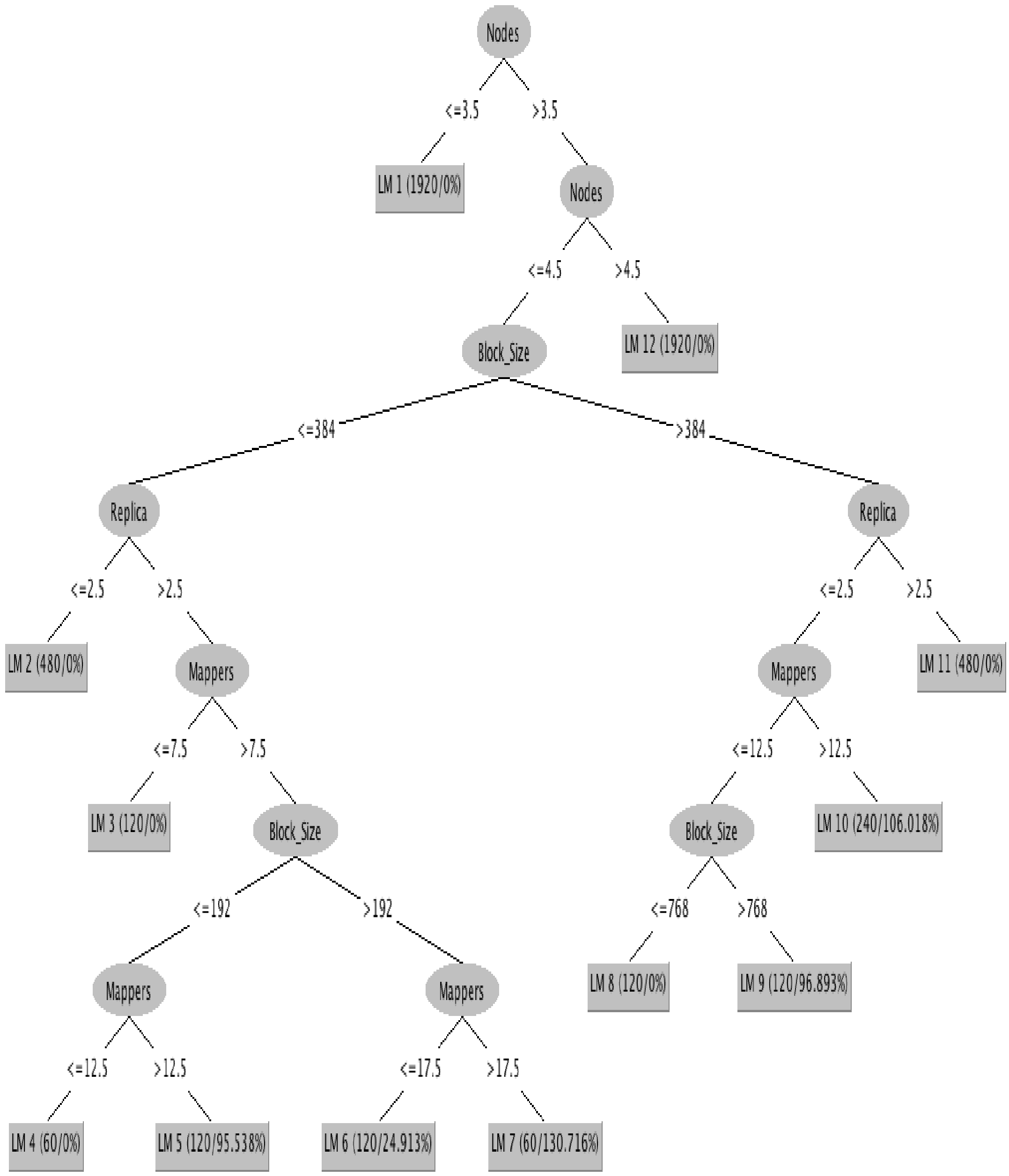

Figure D.8: M5P regression tree of disk usage for changing cluster sizes

D.2 Disk usage linear regression models from M5P tree for different cluster sizes

```
 LM num: 2
usage =
0.0037 * Nodes
+ 0.0001 * Block_Size
+ 0.0161 * Replica
+ 0.0002 * Mappers
- 0.057

LM num: 3
usage =
0.0037 * Nodes
+ 0.0005 * Block_Size
+ 0.0161 * Replica
+ 0.0018 * Mappers
- 0.1158

LM num: 4
usage =
0.0037 * Nodes
+ 0.0006 * Block_Size
+ 0.0161 * Replica
- 0.0014 * Mappers
- 0.0567

LM num: 5
usage =
0.0037 * Nodes
+ 0.0006 * Block_Size
+ 0.0161 * Replica
```

```
- 0.0634 * Mappers
+ 1.1719

LM num: 6
usage =
0.0037 * Nodes
+ 0.0006 * Block_Size
+ 0.0161 * Replica
- 0.0073 * Mappers
+ 0.8178

LM num: 7
usage =
0.0037 * Nodes
+ 0.0006 * Block_Size
+ 0.0161 * Replica
- 0.012 * Mappers
+ 0.4432

LM num: 8
usage =
0.0037 * Nodes
- 0 * Block_Size
+ 0.0131 * Replica
+ 0.0041 * Mappers
+ 0.8968

LM num: 9
usage =
0.0037 * Nodes
- 0 * Block_Size
+ 0.0131 * Replica
+ 0.083 * Mappers
```

```
+ 0.1073

LM num: 10
usage =
0.0037 * Nodes
+ 0.001 * Block_Size
+ 0.0131 * Replica
- 0.0009 * Mappers
- 0.1641

LM num: 11
usage =
0.0037 * Nodes
+ 0 * Block_Size
+ 0.0131 * Replica
- 0.0002 * Mappers
+ 0.9271

LM num: 12
usage =
0.0037 * Nodes
+ 0 * Block_Size
+ 0.0015 * Replica
+ 0.975
```

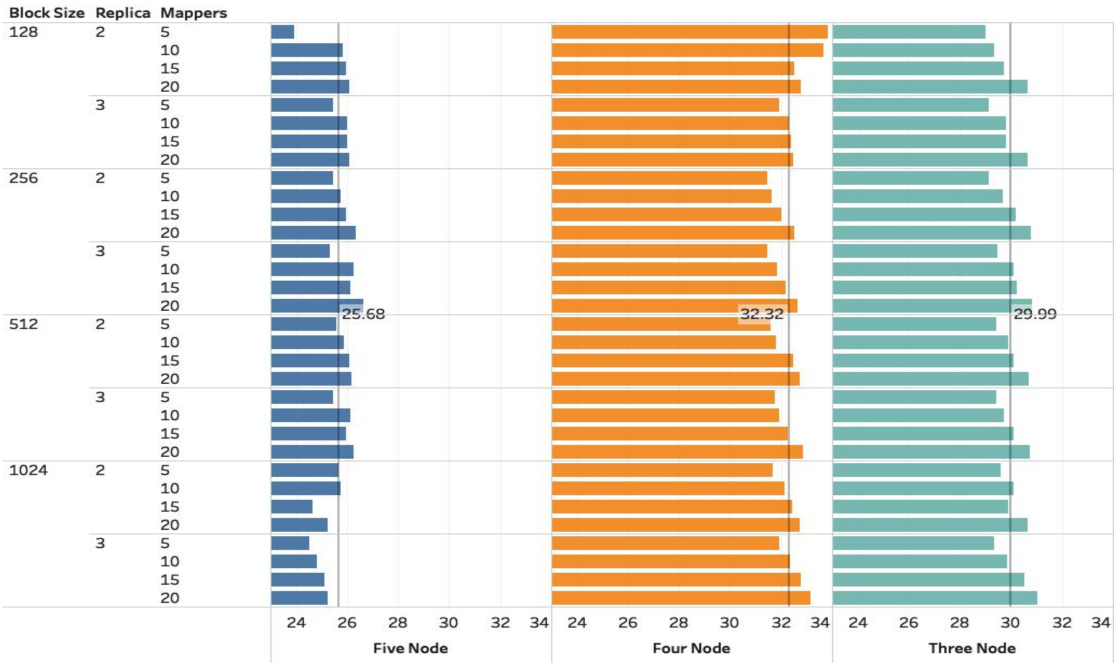

Figure D.9: Mean memory usage for different cluster sizes

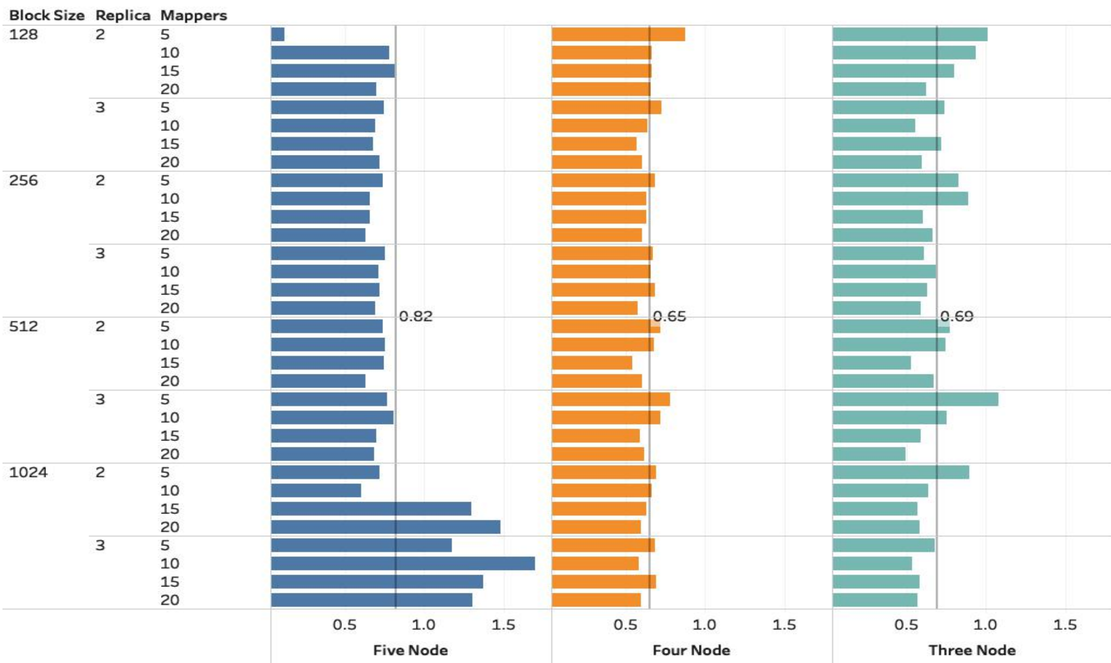

Figure D.10: Standard deviation of memory usage for different cluster sizes

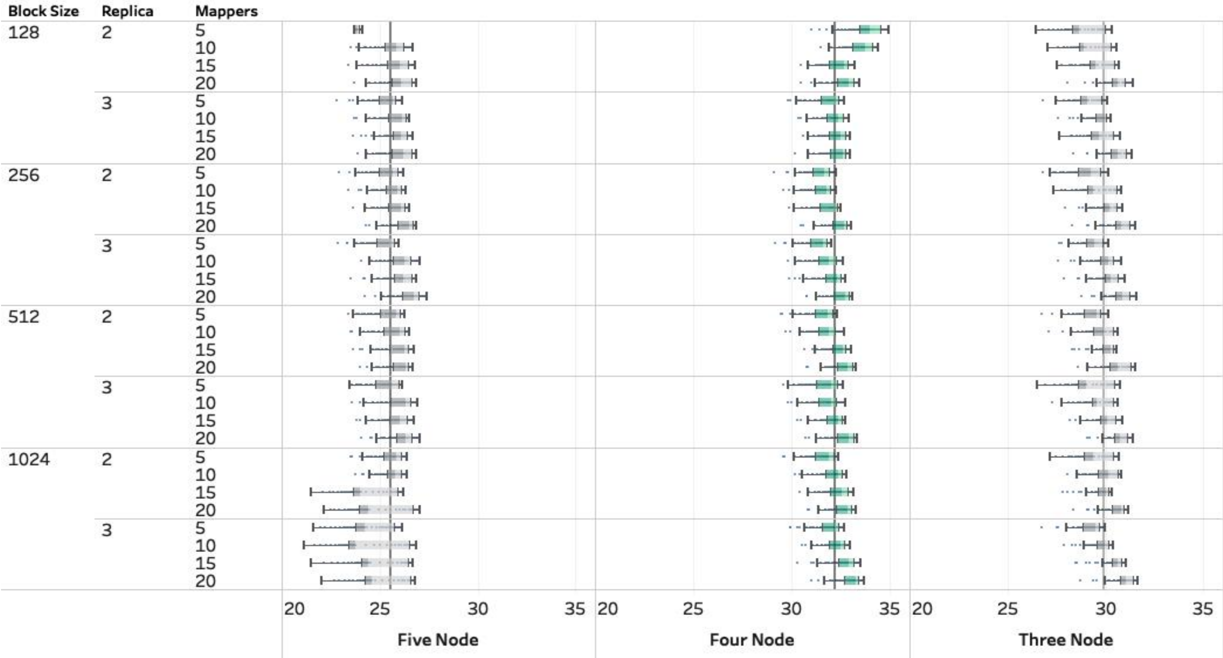

Figure D.11: Box plot for memory usage for different cluster sizes

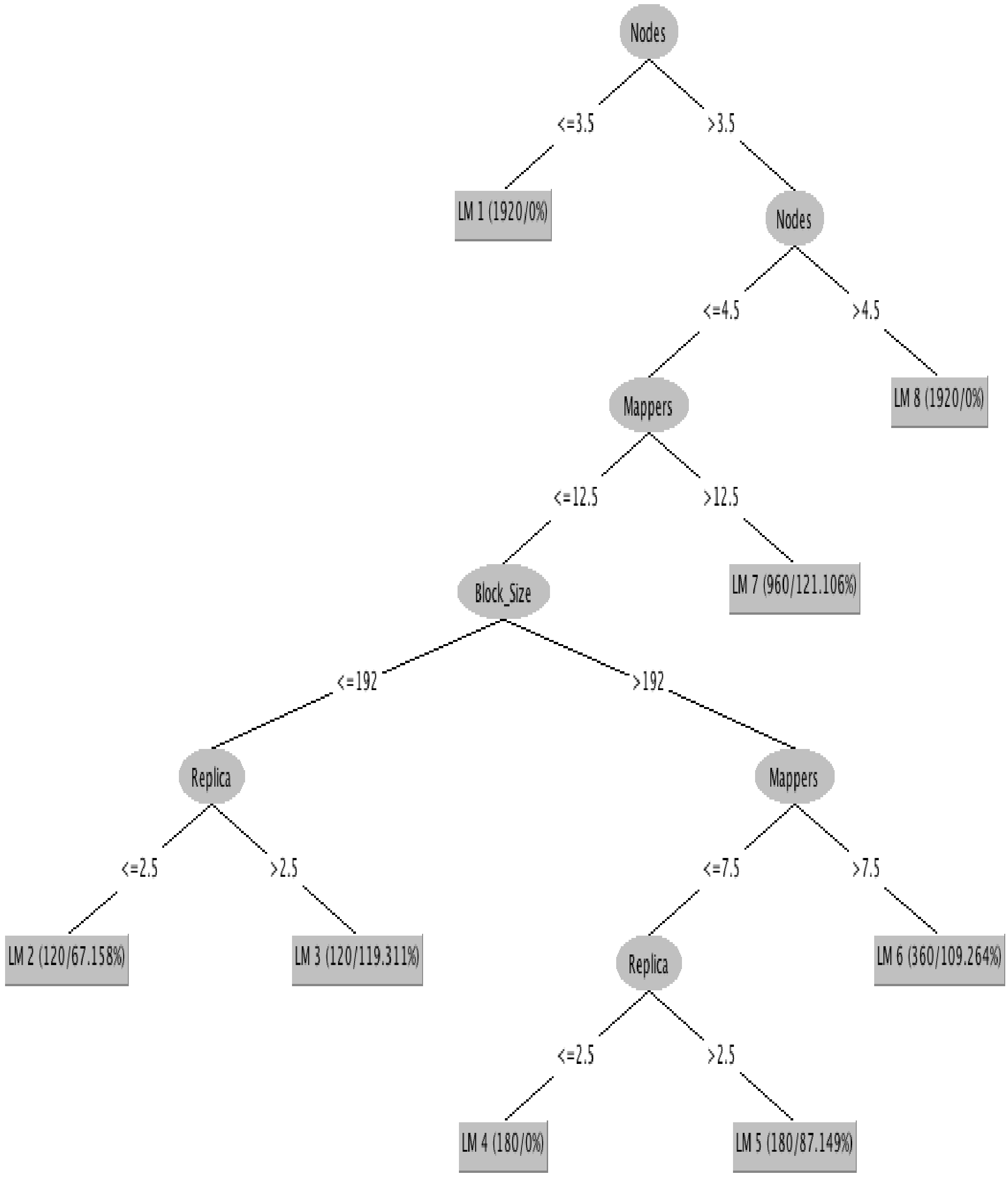

Figure D.12: M5P regression tree of memory usage for changing cluster sizes

D.3 memory usage linear regression models from M5P tree for different cluster sizes

```
LM num: 1
usage =
0.0001 * Mappers
+ 0

LM num: 2
usage =
-0.0038 * Nodes
- 0 * Block_Size
- 0.056 * Replica
+ 0.0077 * Mappers
+ 0.9525

LM num: 3
usage =
-0.0038 * Nodes
- 0 * Block_Size
- 0.056 * Replica
+ 0.0729 * Mappers
+ 0.0163

LM num: 4
usage =
-0.0038 * Nodes
+ 0 * Block_Size
+ 0.0141 * Replica
+ 0.0034 * Mappers
- 0.0324
```

```
LM num: 5
usage =
-0.0038 * Nodes
+ 0.0002 * Block_Size
+ 0.0141 * Replica
+ 0.0034 * Mappers
- 0.0536

LM num: 6
usage =
-0.0038 * Nodes
+ 0.0005 * Block_Size
+ 0.1322 * Replica
+ 0.0034 * Mappers
- 0.384

LM num: 7
usage =
-0.0038 * Nodes
+ 0.0001 * Block_Size
+ 0.0649 * Replica
+ 0.0316 * Mappers
- 0.0668

LM num: 8
usage =
-0.0038 * Nodes
+ 0.0002 * Mappers
+ 0.0172
```

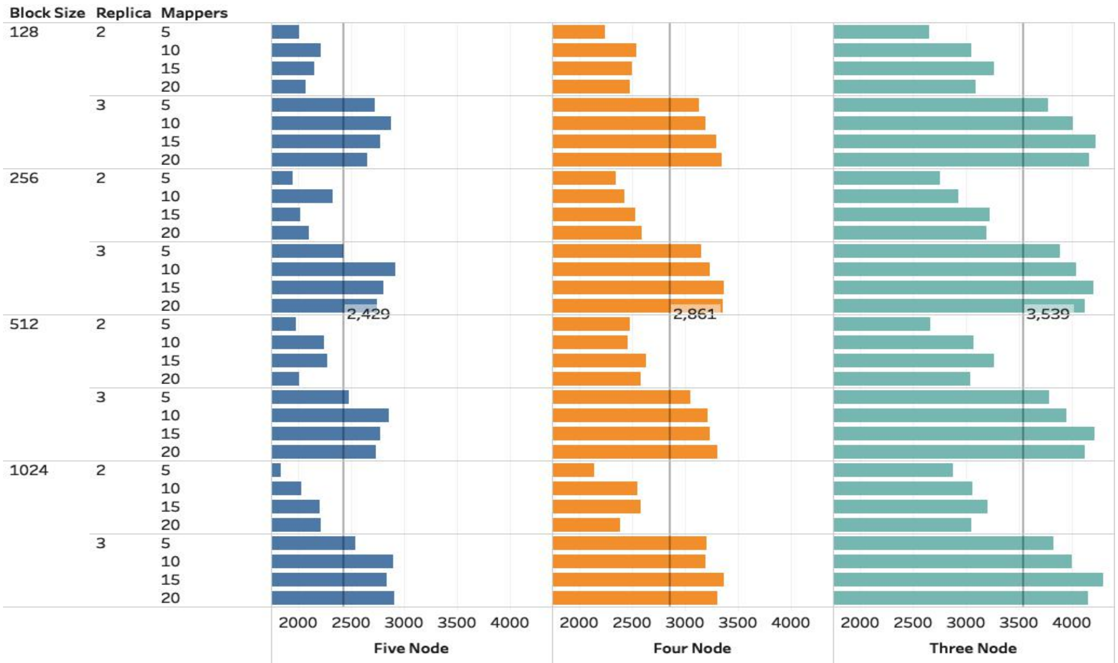

Figure D.13: Mean network usage for different cluster sizes

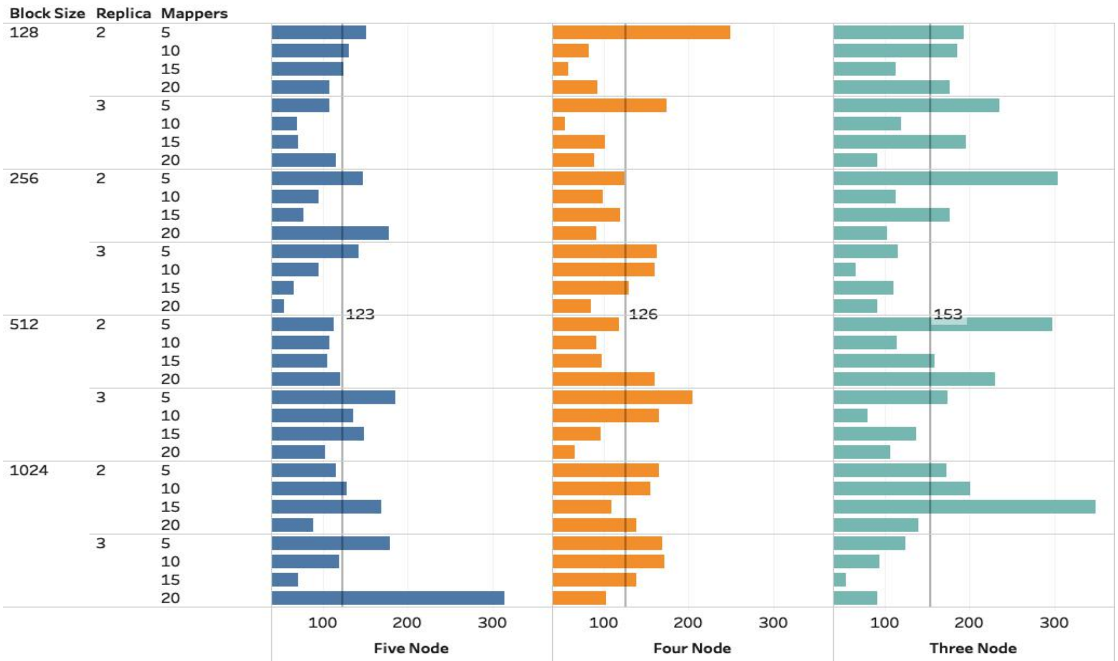

Figure D.14: Standard deviation of network usage for different cluster sizes

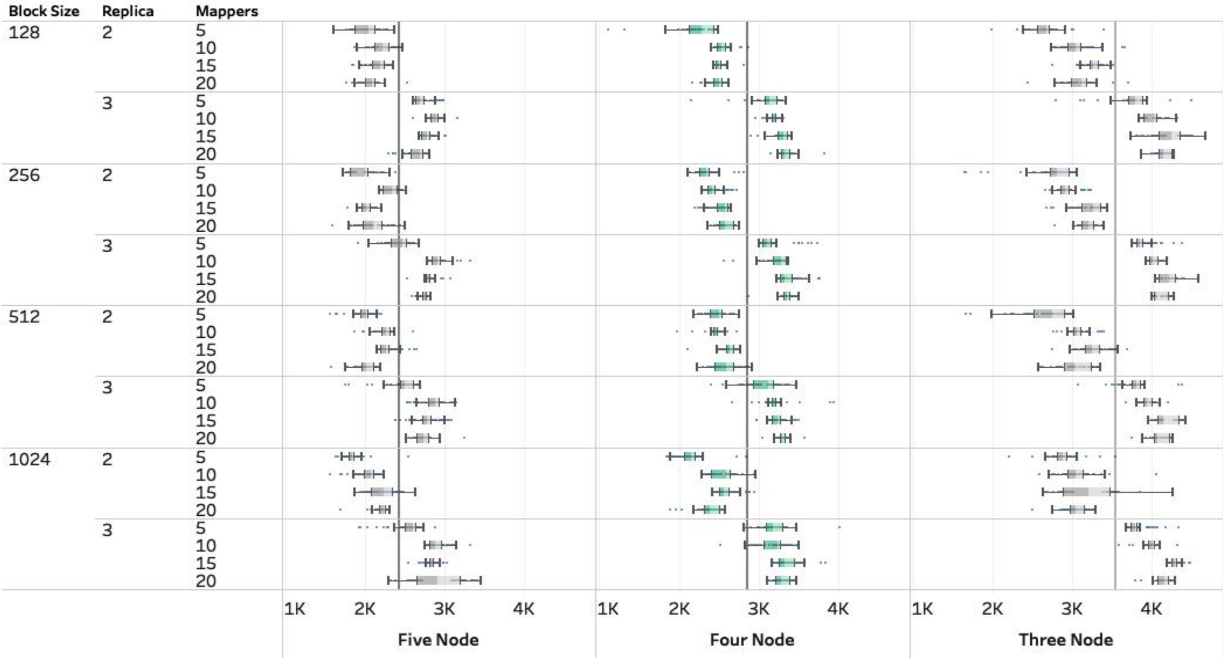

Figure D.15: Box plot for network usage for different cluster sizes

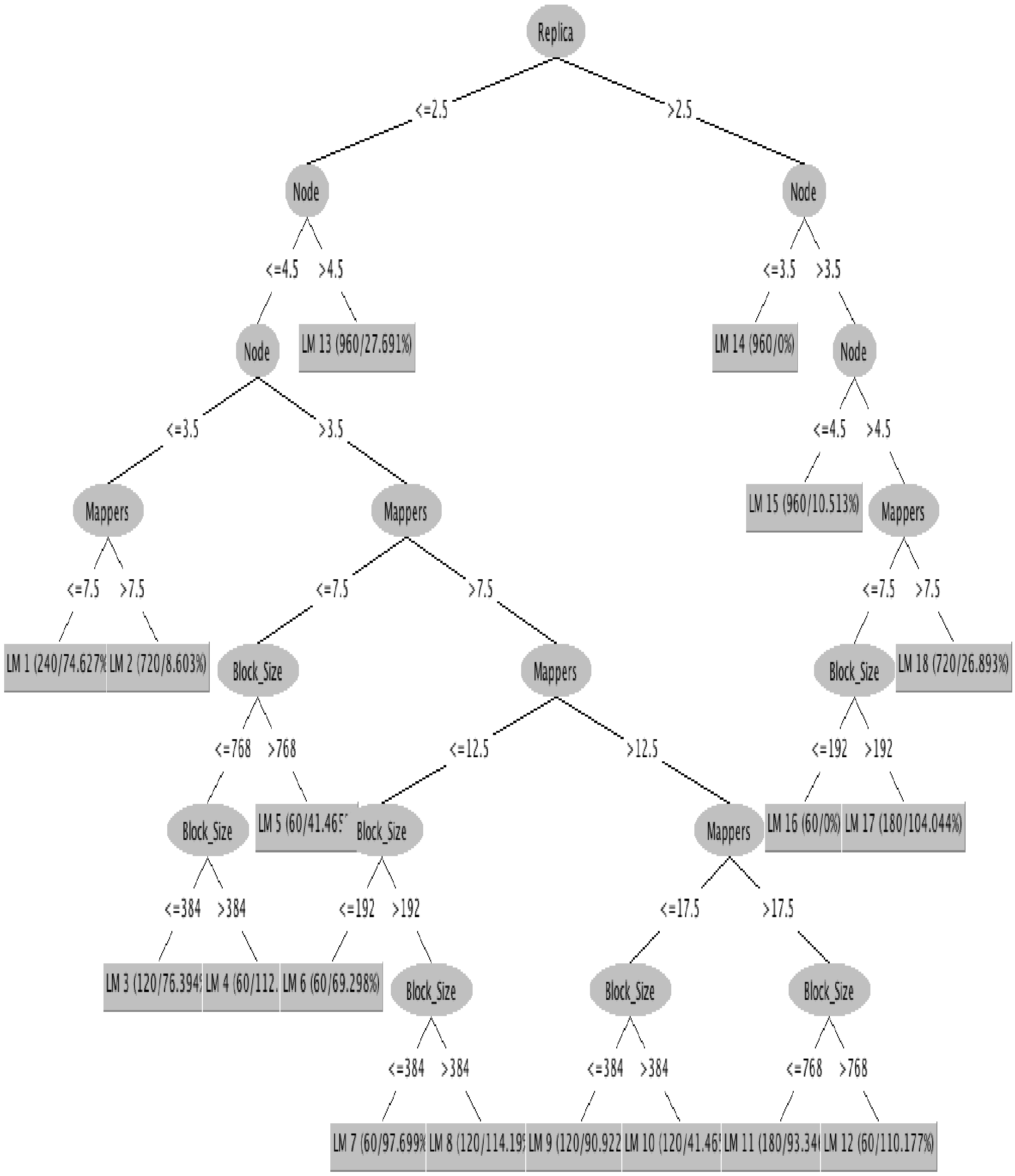

Figure D.16: M5P regression tree of network usage for changing cluster sizes

D.4 Network usage linear regression models from M5P tree for different cluster sizes

```
LM num: 1
usage =
-0.0107 * Node
+ 0.0001 * Block_Size
+ 0.0023 * Replica
+ 0.0009 * Mappers
+ 0.8472

LM num: 2
usage =
-0.0107 * Node
- 0 * Block_Size
+ 0.0023 * Replica
+ 0.0006 * Mappers
+ 1.0166

LM num: 3
usage =
-0.0107 * Node
+ 0.0001 * Block_Size
+ 0.0023 * Replica
+ 0.0024 * Mappers
+ 0.1614

LM num: 4
usage =
-0.0107 * Node
+ 0.0002 * Block_Size
+ 0.0023 * Replica
```

```
+ 0.0024 * Mappers
+ 0.476

LM num: 5
usage =
-0.0107 * Node
- 0 * Block_Size
+ 0.0023 * Replica
+ 0.0024 * Mappers
+ 0.1275

LM num: 6
usage =
-0.0107 * Node
- 0 * Block_Size
+ 0.0023 * Replica
+ 0.0018 * Mappers
+ 0.8353

LM num: 7
usage =
-0.0107 * Node
+ 0 * Block_Size
+ 0.0023 * Replica
+ 0.0018 * Mappers
+ 0.3215

LM num: 8
usage =
-0.0107 * Node
+ 0 * Block_Size
+ 0.0023 * Replica
+ 0.0018 * Mappers
```

```
+ 0.5721

LM num: 9
usage =
-0.0107 * Node
- 0 * Block_Size
+ 0.0023 * Replica
- 0.0008 * Mappers
+ 0.8591

LM num: 10
usage =
-0.0107 * Node
- 0 * Block_Size
+ 0.0023 * Replica
- 0.0008 * Mappers
+ 0.9814

LM num: 11
usage =
-0.0107 * Node
- 0.0001 * Block_Size
+ 0.0023 * Replica
- 0.0008 * Mappers
+ 0.8568

LM num: 12
usage =
-0.0107 * Node
- 0.0001 * Block_Size
+ 0.0023 * Replica
- 0.0008 * Mappers
+ 0.563
```

```
LM num: 13
usage =
-0.0086 * Node
- 0 * Block_Size
+ 0.0023 * Replica
+ 0.0003 * Mappers
+ 0.0519

LM num: 14
usage =
-0.0019 * Node
+ 0.0023 * Replica
+ 0.0001 * Mappers
+ 0.9986

LM num: 15
usage =
-0.0027 * Node
+ 0.0023 * Replica
+ 0.0007 * Mappers
+ 0.9929

LM num: 16
usage =
-0.0027 * Node
+ 0.0023 * Replica
+ 0.0009 * Mappers
+ 0.9428

LM num: 17
usage =
-0.0027 * Node
```

```
+ 0.0003 * Block_Size
+ 0.0023 * Replica
+ 0.0009 * Mappers
+ 0.485

LM num: 18
usage =
-0.0027 * Node
+ 0.0023 * Replica
- 0.0023 * Mappers
+ 1.0249
```

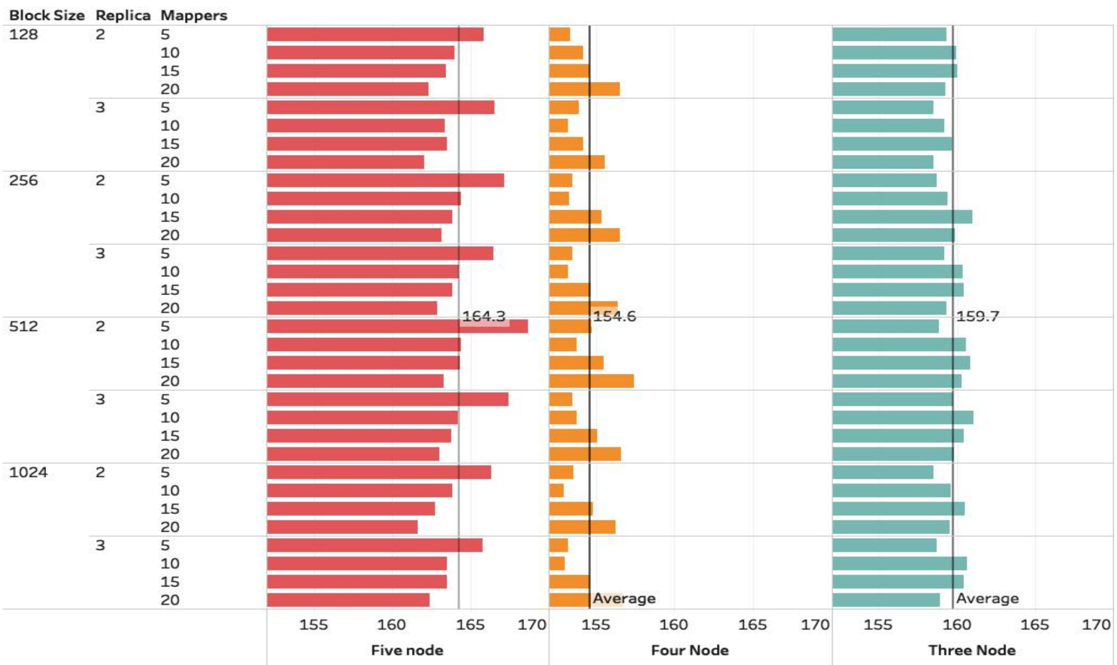

Figure D.17: Mean execution time for different cluster sizes

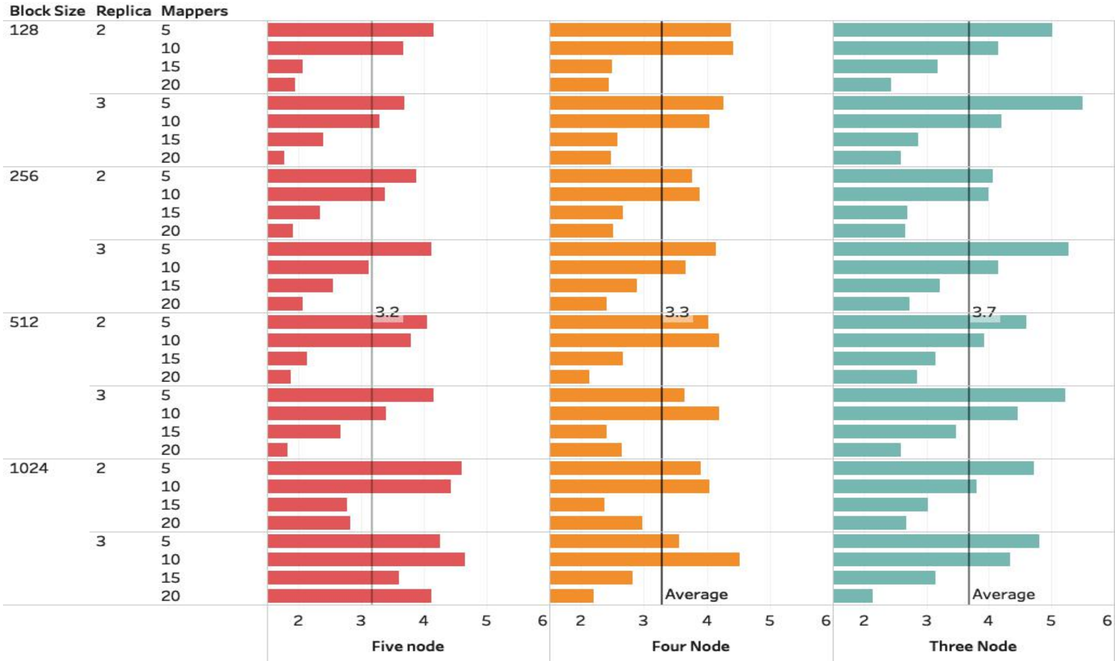

Figure D.18: Standard deviation of execution time for different cluster sizes

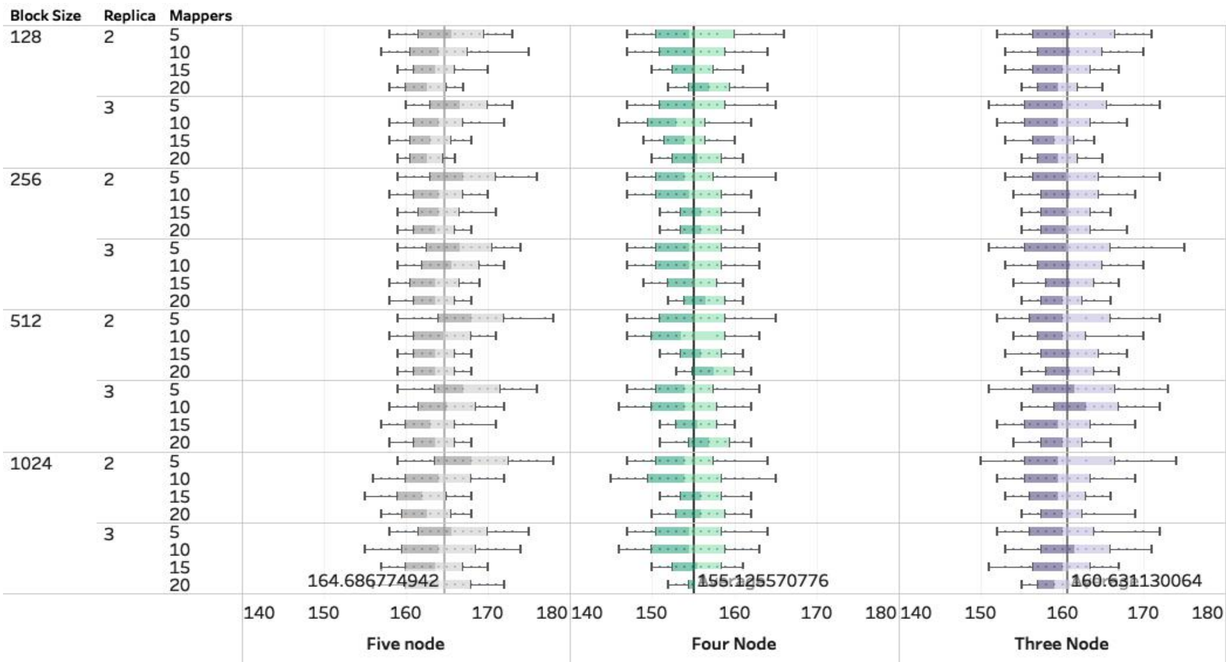

Figure D.19: Box plot for job execution time for different cluster sizes

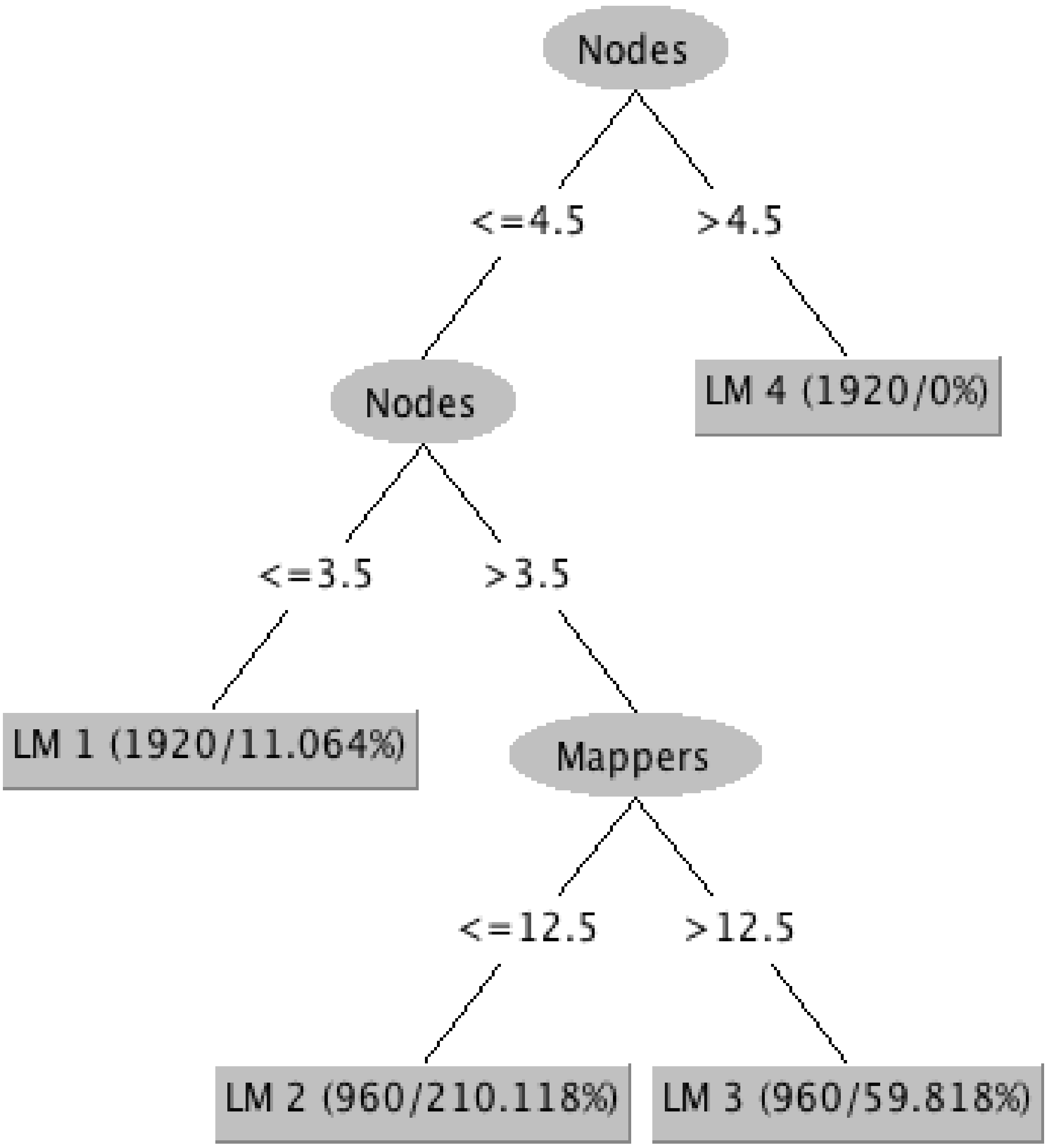

Figure D.20: M5P regression tree of overall time for changing cluster sizes

D.5 Execution time linear regression models from M5P tree for different cluster sizes

```
LM num: 1
usage =
-0.001 * Nodes
- 0 * Block Size
+ 0.0001 * Mappers
+ 1.0023

LM num: 2
usage =
-0.001 * Nodes
- 0.0003 * Replica
+ 0.0004 * Mappers
+ 0.7535

LM num: 3
usage =
-0.001 * Nodes
- 0.0145 * Replica
+ 0.0048 * Mappers
+ 0.9391

LM num: 4
usage =
0 * Mappers
+ 0.999
```

Appendix E

Modelling and prediction of performance of Hadoop cluster for different infrastructure

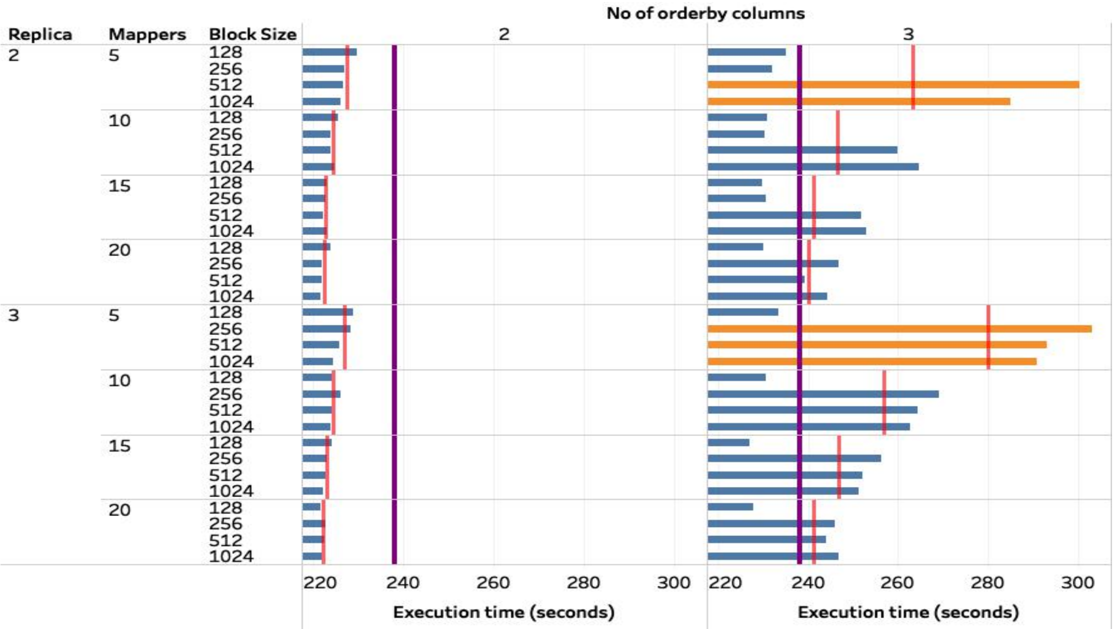

Figure E.1: Mean of execution times showing impact of number of mappers, number of replicas, block size and columns in order by clause on AWS cluster

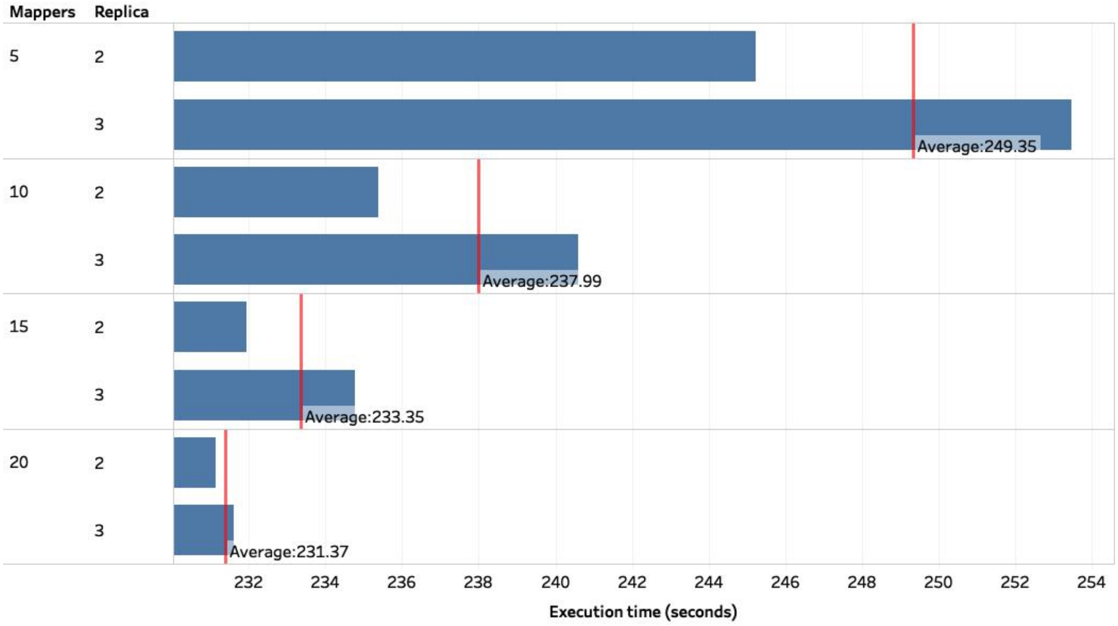

Figure E.2: Mean execution times showing impact of number of mappers and number of replicas on AWS cluster

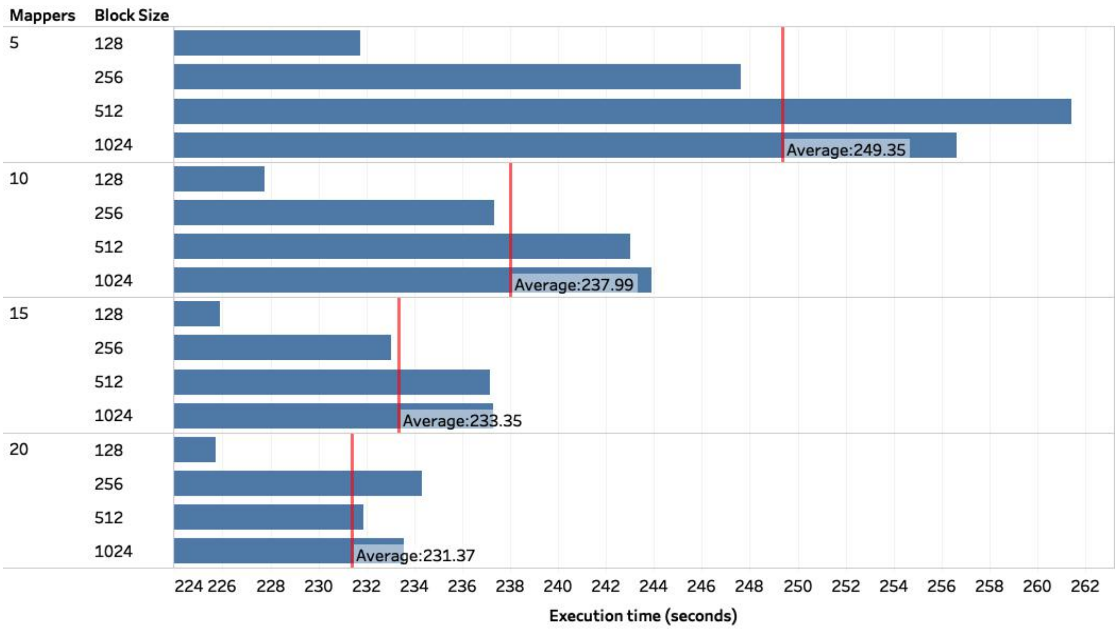

Figure E.3: Mean execution times showing impact of number of mappers and block size on AWS cluster

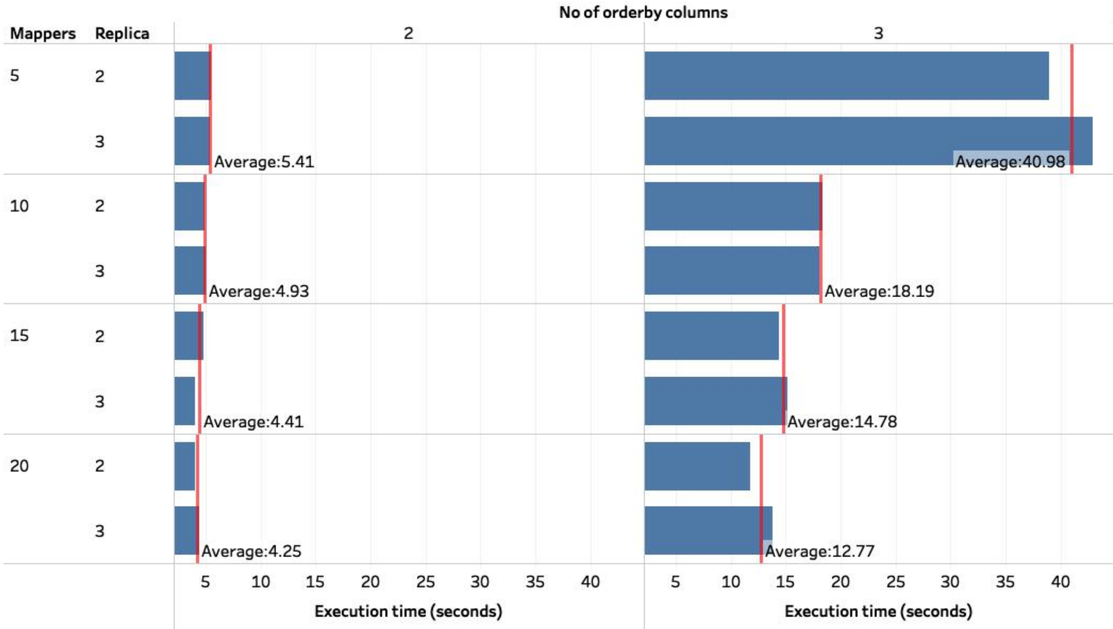

Figure E.4: Standard deviations of execution times showing impact of number of mappers, number of replicas and orderby clause on AWS cluster

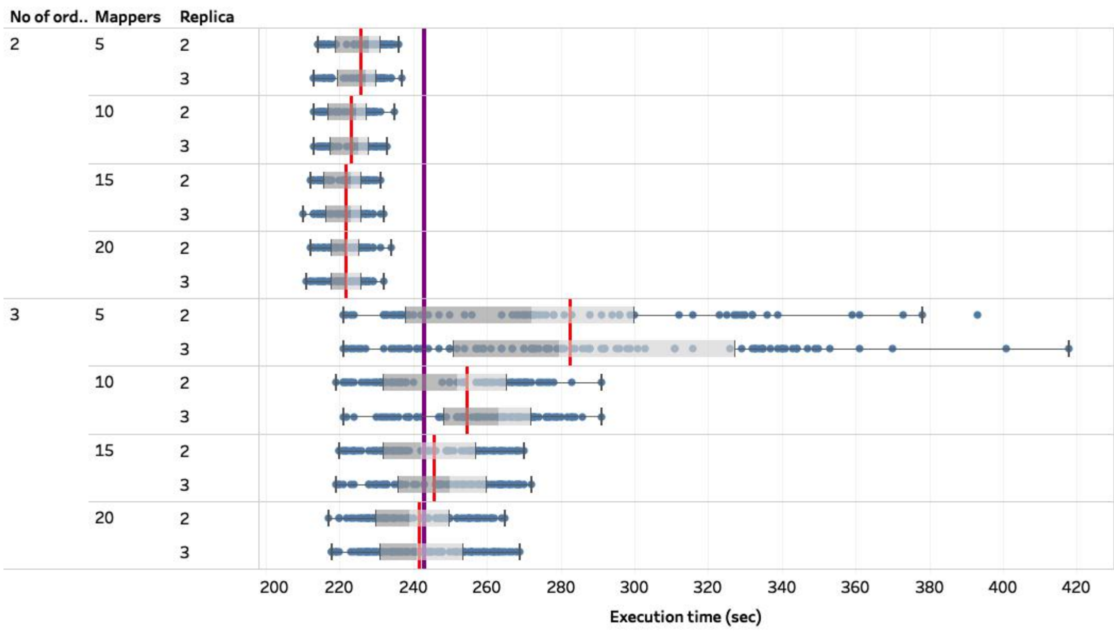

Figure E.5: Boxplot for execution time on AWS cluster

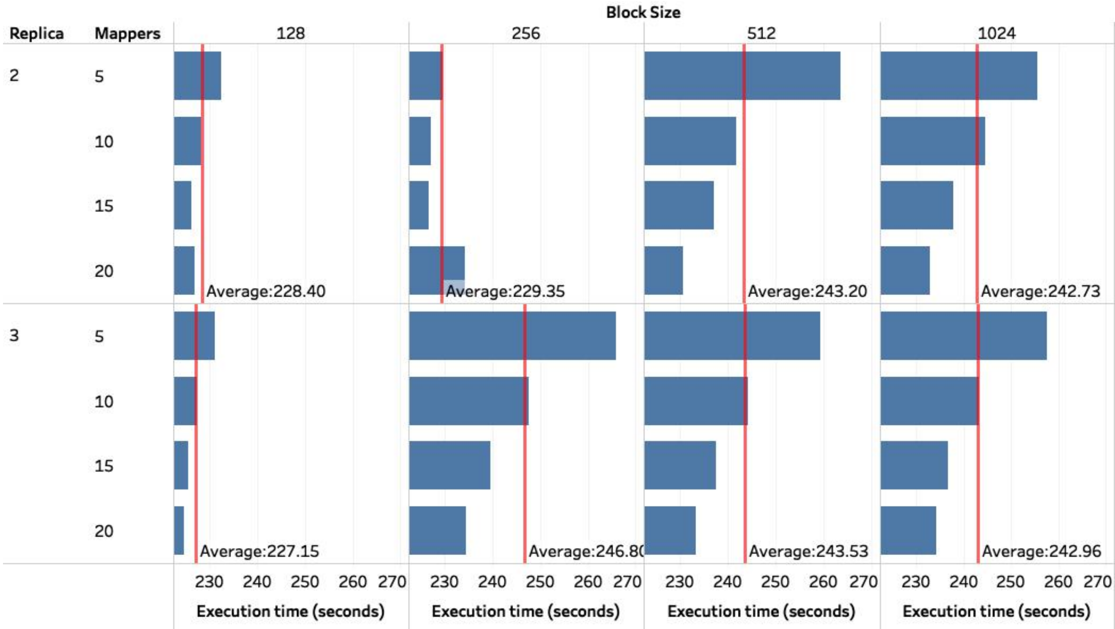

Figure E.6: Mean execution times showing impact of number of mappers, number of replicas and block size on AWS cluster

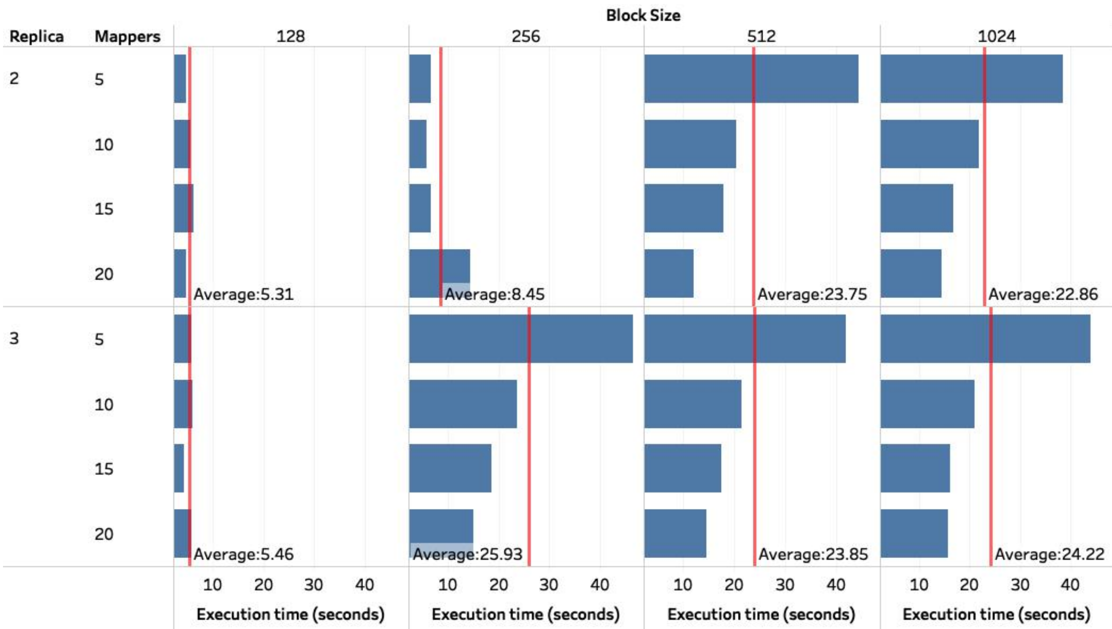

Figure E.7: Standard deviation of execution times showing impact of number of mappers, number of replicas and block size on AWS cluster

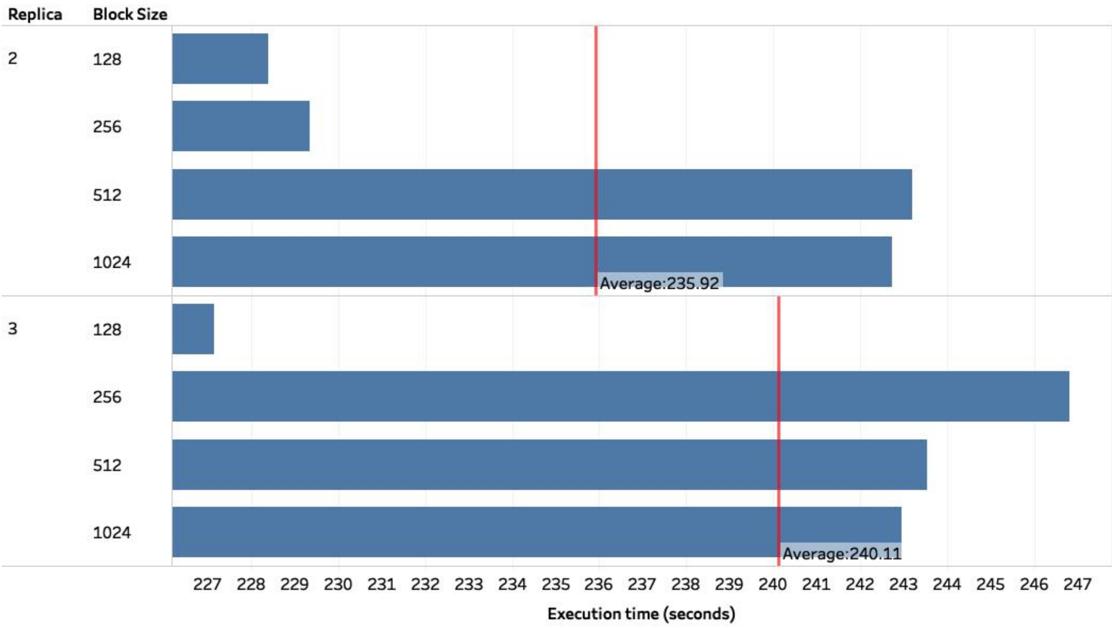

Figure E.8: Mean execution times showing impact of number of replicas and block size on AWS cluster

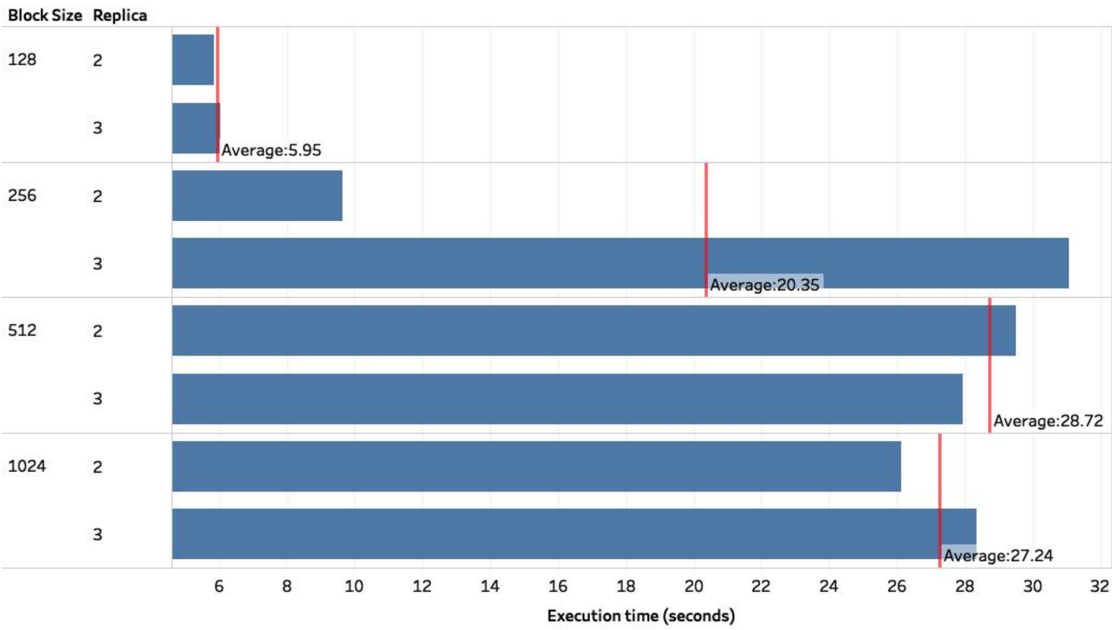

Figure E.9: Standard deviation of execution times showing impact of number of replicas and block size on AWS cluster

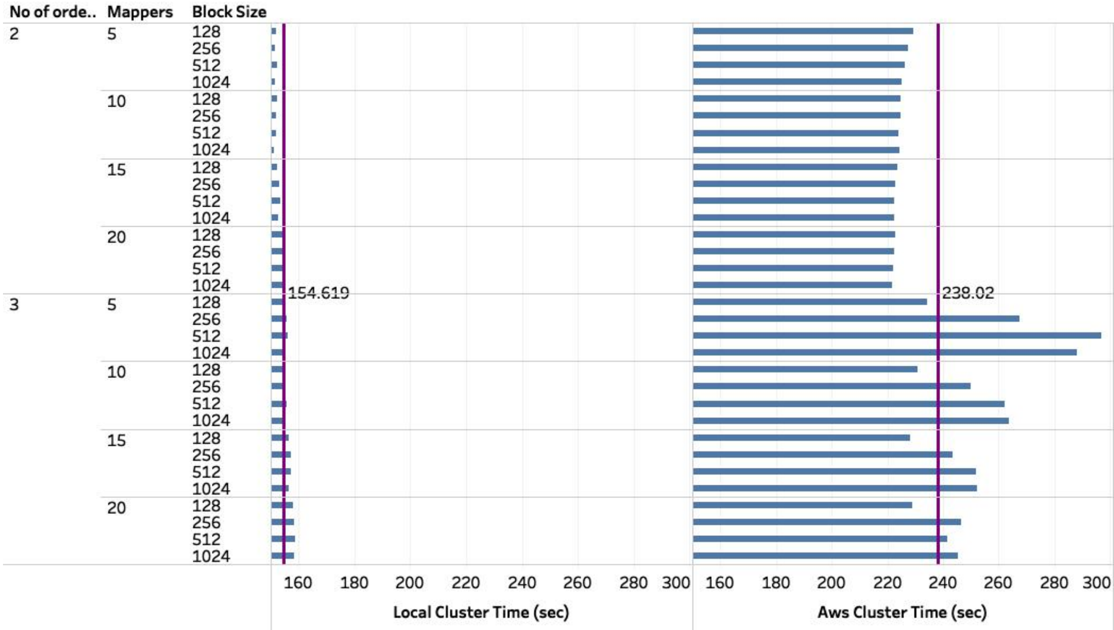

Figure E.10: Impact of infrastructure change on the performance of Hadoop jobs

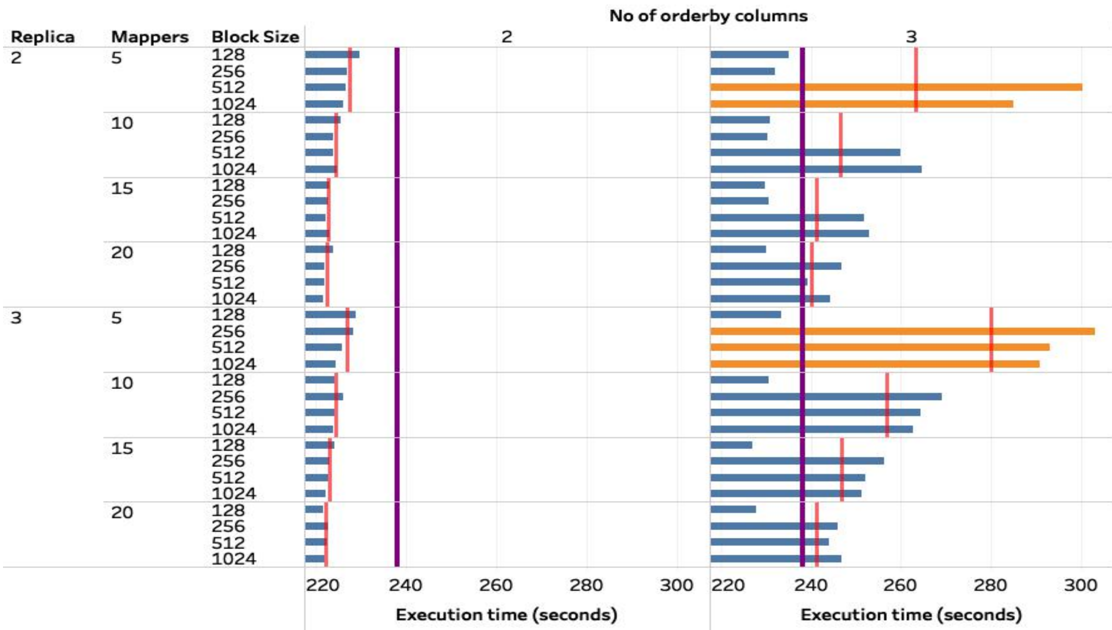

Figure E.11: k-mean clustering for execution time on AWS cluster

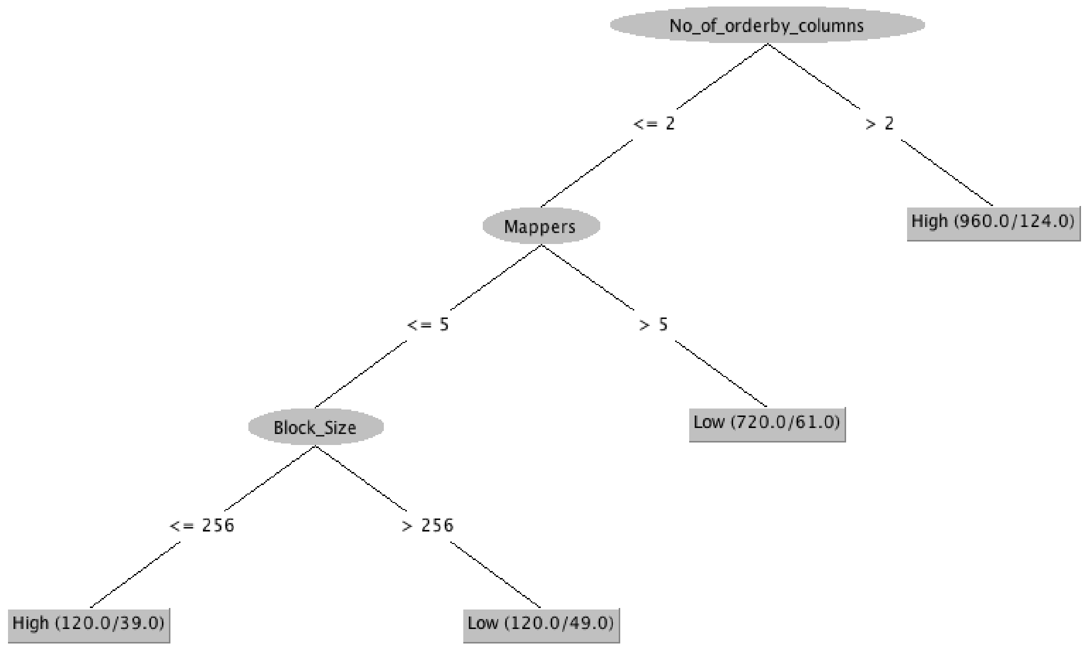

Figure E.12: Decision tree for the performance of Hadoop jobs on AWS cluster

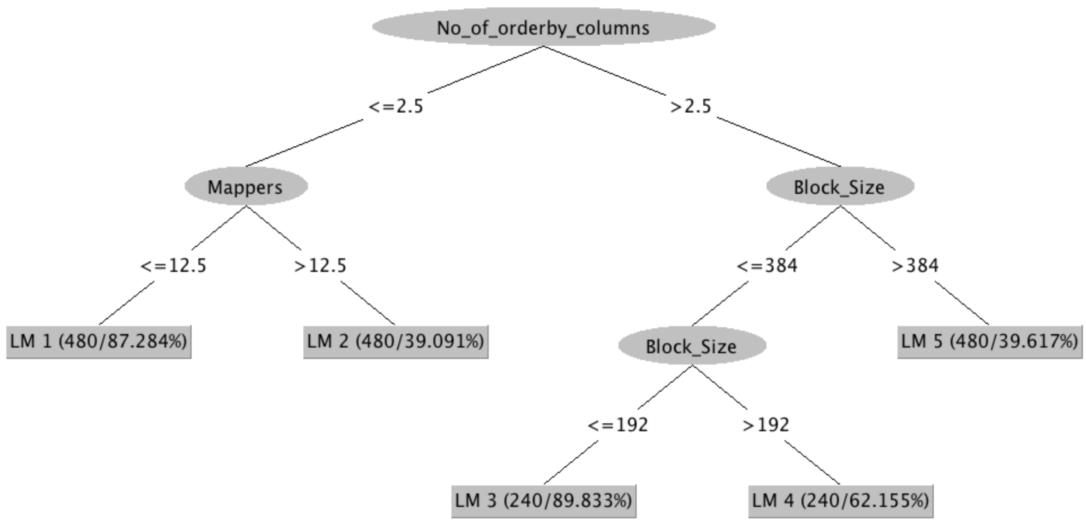

Figure E.13: M5P tree for the performance of Hadoop jobs on AWS cluster

E.1 Execution time Linear regression models from M5P tree for AWS cluster

```
    LM num: 1
Usage =
-0.0002 * Block_Size
- 0.0713 * Mappers
+ 0.0103 * No_of_orderby_columns
+ 0.9833

LM num: 2
Usage =
-0 * Block_Size
- 0.0013 * Mappers
+ 0.0103 * No_of_orderby_columns
+ 0.0438

LM num: 3
Usage =
0.0001 * Block_Size
- 0.1573 * Replica
- 0.0137 * Mappers
+ 0.0103 * No_of_orderby_columns
+ 1.2136

LM num: 4
Usage =
0.0001 * Block_Size
+ 0.1498 * Replica
- 0.0008 * Mappers
+ 0.0103 * No_of_orderby_columns
+ 0.461
```

```
LM num: 5
Usage =
0.0001 * Block_Size
- 0.0089 * Mappers
+ 0.0103 * No_of_orderby_columns
+ 0.9754
```

Bibliography

[1] S. J. Walker, "Big data: A revolution that will transform how we live, work, and think," *International Journal of Advertising*, vol. 33, no. 1, pp. 181–183, 2014.

[2] M. R. Bendre and V. R. Thool, "Analytics, challenges and applications in big data environment: a survey," *Journal of Management Analytics*, vol. 3, no. 3, pp. 206–239, 2016.

[3] A. Vera-Baquero, R. Colomo-Palacios, and O. Molloy, "Towards a process to guide big data based decision support systems for business processes," *Procedia Technology*, vol. 16, pp. 11–21, 2014.

[4] S. Masih and S. Tanwani, "Data mining techniques in parallel and distributed environment-a comprehensive survey," *International Journal of Emerging Technology and Advanced Engineering*, vol. 4, no. 3, pp. 453–461, 2014.

[5] K. Venkatram and M. A. Geetha, "Review on big data & analytics–concepts, philosophy, process and applications," *Cybernetics and Information Technologies*, vol. 17, no. 2, pp. 3–27, 2017.

[6] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Big data technologies: A survey," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 4, pp. 431–448, 2018.

[7] D. Cirillo and A. Valencia, "Big data analytics for personalized medicine," *Current Opinion in Biotechnology*, vol. 58, pp. 161–167, 2019.

[8] S. Kumar and K. K. Mohbey, "A review on big data based parallel and distributed approaches of pattern mining," *Journal of King Saud University - Computer and Information Sciences*, 2019.

[9] P. Zikopoulos, C. Eaton, and IBM, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 1st ed., 2011.

[10] K. Bakshi, "Considerations for big data: Architecture and approach," in *Proceedings of the Aerospace Conference*, pp. 1–7, IEEE, 2012.

[11] A. Ching, R. M. D. Molkov, R. Vadali, and P. Yang, "Under the hood: Scheduling mapreduce jobs more efficiently with corona," 2012.

[12] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, "Hive-a petabyte scale data warehouse using hadoop," in *Proceedings of the 26th International Conference on Data Engineering*, pp. 996–1005, IEEE, 2010.

[13] W. Dai, I. Ibrahim, and M. Bassiouni, "An improved replica placement policy for hadoop distributed file system running on cloud platforms," in *Proceedings of the 4th International Conference on Cyber Security and Cloud Computing*, pp. 270–275, IEEE, 2017.

[14] K. Qu, L. Meng, and Y. Yang, "A dynamic replica strategy based on markov model for hadoop distributed file system (HDFS)," in *Proceedings of the 4th International Conference on Cloud Computing and Intelligence Systems*, pp. 337–342, IEEE, 2016.

[15] W. Dai, I. Ibrahim, and M. Bassiouni, "A new replica placement policy for hadoop distributed file system," in *Proceedings of the 2nd International Conference on Big Data Security on Cloud, IEEE International Conference on High Performance and Smart Computing, and IEEE International Conference on Intelligent Data and Security*, pp. 262–267, IEEE, 2016.

[16] C. Y. Lin and Y. C. Lin, "A load-balancing algorithm for hadoop distributed file system," in *Proceedings of the 18th International Conference on Network-Based Information Systems*, pp. 173–179, IEEE, 2015.

[17] S. Nishanth, B. Radhikaa, T. J. Ragavendar, C. Babu, and B. Prabavathy, "Cohadoop++: A load balanced data co-location in hadoop distributed file system," in *Proceedings of the 5th International Conference on Advanced Computing*, pp. 100–105, IEEE, 2013.

[18] R. Moraveji, J. Taheri, M. Reza, N. B. Rizvandi, and A. Y. Zomaya, "Data-intensive workload consolidation for the hadoop distributed file system," in *Proceedings of the 13th International Conference on Grid Computing*, pp. 95–103, ACM/IEEE, 2012.

[19] M. M. Shetty and D. H. Manjaiah, "Data security in hadoop distributed file system," in *Proceedings of the International Conference on Emerging Technological Trends*, pp. 1–5, IEEE, 2016.

[20] B. Agrawal, R. Hansen, C. Rong, and T. Wiktorski, "Sd-hdfs: Secure deletion in hadoop distributed file system," in *Proceedings of the International Congress on Big Data*, pp. 181–189, IEEE, 2016.

[21] S. Saranya, M. Sarumathi, B. Swathi, P. V. Paul, S. S. Kumar, and T. Vengattaraman, "Dynamic preclusion of encroachment in hadoop distributed file system," *Procedia Computer Science*, vol. 50, pp. 531–536, 2015.

[22] C. Liao, A. Squicciarini, and D. Lin, "LAST-HDFS: Location-aware storage technique for hadoop distributed file system," in *Proceedings of the 9th International Conference on Cloud Computing*, pp. 662–669, IEEE, 2016.

[23] J. Zhang, G. Wu, X. Hu, and X. Wu, "A distributed cache for hadoop distributed file system in real-time cloud services," in *Proceedings of the 13th International Conference on Grid Computing*, pp. 12–21, ACM/IEEE, 2012.

[24] J. Kim, T. K. A. Kumar, K. M. George, and N. Park, "Performance evaluation and tuning for mapreduce computing in hadoop distributed file

system," in *Proceedings of the 13th International Conference on Industrial Informatics*, pp. 62–68, IEEE, 2015.

[25] T. Yeh and Y. Sun, "Enabling prioritized cloud I/O service in hadoop distributed file system," in *Proceedings of the International Conference on High Performance Computing and Communications, 6th International Symposium on Cyberspace Safety and Security, 11th International Conference on Embedded Software and Systems*, pp. 256–259, IEEE, 2014.

[26] T. L. S. R. Krishna, T. Ragunathan, and S. K. Battula, "Performance evaluation of read and write operations in hadoop distributed file system," in *Proceedings of the 6th International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 110–113, IEEE, 2014.

[27] X. Hua, H. Wu, Z. Li, and S. Ren, "Enhancing throughput of the hadoop distributed file system for interaction-intensive tasks," *Journal of Parallel and Distributed Computing*, vol. 74, no. 8, pp. 2770–2779, 2014.

[28] I. Tomašić, J. Ugovšek, A. Rashkovska, and R. Trobec, "Multicluster hadoop distributed file system," in *Proceedings of the 35th International Convention*, pp. 301–305, IEEE, 2012.

[29] V. Chang, "A proposed social network analysis platform for big data analytics," *Technological Forecasting and Social Change*, vol. 130, pp. 57–68, 2018.

[30] S. Jeon, B. Hong, and V. Chang, "Pattern graph tracking-based stock price prediction using big data," *Future Generation Computer Systems*, vol. 80, pp. 171–187, 2018.

[31] M. Bendechache, A.-K. Tari, and M.-T. Kechadi, "Parallel and distributed clustering framework for big spatial data mining," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 34, no. 6, pp. 671–689, 2019.

[32] D. Usha and A. J. Aps, "A survey of big data processing in perspective of hadoop and mapreduce," *International Journal of Current Engineering and Technology*, vol. 4, no. 2, pp. 602–606, 2014.

[33] D. DeRoos, P. Zikopoulos, B. Brown, R. Coss, and R. B. Melnyk, *Hadoop for Dummies*. John Wiley & Sons, Incorporated, 2014.

[34] A. Hadoop, "Hadoop documentation 2.8.0." `https://hadoop.apache.org/docs/r2.8.0/`, 2017. [Online; accessed 2017-10-08].

[35] A. B. Patel, M. Birla, and U. Nair, "Addressing big data problem using hadoop and map reduce," in *Proceedings of the Nirma University International Conference on Engineering*, pp. 1–5, IEEE, 2012.

[36] Z. Bei, Z. Yu, N. Luo, C. Jiang, C. Xu, and S. Feng, "Configuring inmemory cluster computing using random forest," *Future Generation Computer Systems*, vol. 79, pp. 1–15, 2018.

[37] V. Kalavri and V. Vlassov, "Mapreduce: Limitations, optimizations and open issues," in *Proceedings of the 12th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 1031–1038, IEEE, 2013.

[38] P. Saxena and P. Kumar, "Performance evaluation of HDD and SSD on 10GigE, IPoIB amp; RDMA-IB with hadoop cluster performance benchmarking system," in *Proceedings of the 5th International Conference - Confluence The Next Generation Information Technology Summit*, pp. 30–35, IEEE, 2014.

[39] L. Chen, B. Peng, B. Zhang, T. Liu, Y. Zou, L. Jiang, R. Henschel, C. Stewart, Z. Zhang, E. McCallum, Z. Tom, O. Jon, and J. Qiu, "Benchmarking harp-daal: High performance hadoop on knl clusters," in *Proceedings of the 10th International Conference on Cloud Computing*, pp. 82–89, IEEE, 2017.

[40] J. L. Berral, N. Poggi, D. Carrera, A. Call, R. Reinauer, and D. Green, "ALOJA: A framework for benchmarking and predictive analytics in hadoop deployments," *Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 480–493, 2017.

[41] L. J. Mohan, R. L. Harold, P. I. S. Caneleo, U. Parampalli, and A. Harwood, "Benchmarking the performance of hadoop triple replication and

erasure coding on a nation-wide distributed cloud," in *Proceedings of the 2015 International Symposium on Network Coding*, pp. 61–65, 2015.

[42] R. A. Maxion and R. R. M. Roberts, "Methodological foundations: enabling the next generation of security," *IEEE Security Privacy*, vol. 3, no. 2, pp. 54–57, 2005.

[43] M. Mansoori, I. Welch, K. K. R. Choo, and R. A. Maxion, "Application of HAZOP to the design of cyber security experiments," in *Proceedings of the 30th International Conference on Advanced Information Networking and Applications*, pp. 790–799, IEEE, 2016.

[44] A. Eiben and S. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, 2011.

[45] H. Lawley, "Operability studies and hazard analysis," *Chem. Eng. Prog.*, vol. 70, no. 4, pp. 45–56, 1974.

[46] J. Dunjó, V. Fthenakis, J. A. Vílchez, and J. Arnaldos, "Hazard and operability (HAZOP) analysis. a literature review," *Journal of hazardous materials*, vol. 173, no. 1-3, pp. 19–32, 2010.

[47] A. M. Hiremath, S. K. Pandey, and S. R. Asolekar, "Development of ship-specific recycling plan to improve health safety and environment in ship recycling yards," *Journal of cleaner production*, vol. 116, pp. 279–298, 2016.

[48] A. Krzemień, A. S. Sánchez, P. R. Fernández, K. Zimmermann, and F. G. Coto, "Towards sustainability in underground coal mine closure contexts: A methodology proposal for environmental risk management," *Journal of cleaner production*, vol. 139, pp. 1044–1056, 2016.

[49] M. C. Quintella *et al.*, "Adaptação e aplicação da técnica HAZOP na identificação de risco na área de serviço de saúde= estudo de caso hemocentro/unicamp," 2011.

[50] M. V. Fattor and M. G. A. Vieira, "Application of human HAZOP technique adapted to identify risks in brazilian waste pickers' cooperatives," *Journal of Environmental Management*, vol. 246, pp. 247–258, 2019.

[51] J. M. Kościelny, M. Syfert, B. Fajdek, and A. Kozak, "The application of a graph of a process in HAZOP analysis in accident prevention system," *Journal of Loss Prevention in the Process Industries*, vol. 50, pp. 55–66, 2017.

[52] S. Zou, Y. Kuang, D. Tang, Z. Guo, and S. Xu, "Risk analysis of high level radioactive waste storage tank based on HAZOP," *Annals of Nuclear Energy*, vol. 119, pp. 106–116, 2018.

[53] J. Guiochet, "Hazard analysis of human–robot interactions with HAZOP–UML," *Safety Science*, vol. 84, pp. 225–237, 2016.

[54] M. A. O. de la Herrera, A. S. Luna, A. C. A. da Costa, and E. M. B. Lemes, "A structural approach to the HAZOP – hazard and operability technique in the biopharmaceutical industry," *Journal of Loss Prevention in the Process Industries*, vol. 35, p. 1–11, 2015.

[55] H. Jagtman, A. Hale, and T. Heijer, "A support tool for identifying evaluation issues of road safety measures," *Reliability Engineering & System Safety*, vol. 90, no. 2-3, pp. 206–216, 2005.

[56] J. Dunjó, V. Fthenakis, J. A. Vílchez, and J. Arnaldos, "Hazard and operability (HAZOP) analysis. a literature review," *Journal of Hazardous Materials*, vol. 173, no. 1, pp. 19–32, 2010.

[57] M. Khan, Y. Jin, M. Li, Y. Xiang, and C. Jiang, "Hadoop performance modeling for job estimation and resource provisioning," *Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 441–454, 2016.

[58] E. Mjolsness and D. DeCoste, "Machine learning for science: State of the art and future prospects," *Science*, vol. 293, no. 5537, pp. 2051–2055, 2001.

[59] C. Ma, H. H. Zhang, and X. Wang, "Machine learning for big data analytics in plants," *Trends in Plant Science*, vol. 19, no. 12, pp. 798–808, 2014.

[60] Y. W. Foo, C. Goh, and Y. Li, "Machine learning with sensitivity analysis to determine key factors contributing to energy consumption in cloud data centers," in *Proceedings of the 2016 International Conference on Cloud Computing Research and Innovations*, pp. 107–113, IEEE, 2016.

[61] E. Ataie, E. Gianniti, D. Ardagna, and A. Movaghar, "A combined analytical modeling machine learning approach for performance prediction of mapreduce jobs in cloud environment," in *Proceedings of the 2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 431–439, IEEE, 2016.

[62] L. Zheng and Y. Shen, "Improve parallelism of task execution to optimize utilization of mapreduce cluster resources," in *Proceedings of the 17th International Conference on Computational Science and Engineering*, pp. 674–681, IEEE, 2014.

[63] Y. Sheng and S. M. Rovnyak, "Decision tree-based methodology for high impedance fault detection," *IEEE Transactions on Power Delivery*, vol. 19, no. 2, pp. 533–536, 2004.

[64] V. Sugumaran and K. Ramachandran, "Automatic rule learning using decision tree for fuzzy classifier in fault diagnosis of roller bearing," *Mechanical Systems and Signal Processing*, vol. 21, no. 5, pp. 2237–2247, 2007.

[65] W. Sun, J. Chen, and J. Li, "Decision tree and PCA-based fault diagnosis of rotating machinery," *Mechanical Systems and Signal Processing*, vol. 21, no. 3, pp. 1300–1317, 2007.

[66] M. B. Castellanos, A. L. Serpa, J. L. Biazussi, W. M. Verde, and N. S. D. A. do Sassim, "Fault identification using a chain of decision trees in an electrical submersible pump operating in a liquid-gas flow," *Journal of Petroleum Science and Engineering*, vol. 184, p. 106490, 2020.

[67] L. Sun, B. Zou, S. Fu, J. Chen, and F. Wang, "Speech emotion recognition based on DNN-decision tree SVM model," *Speech Communication*, vol. 115, pp. 29–37, 2019.

[68] E. Aliyan, M. Aghamohammadi, M. Kia, A. Heidari, M. Shafie-khah, and J. P. Catalão, "Decision tree analysis to identify harmful contingencies and estimate blackout indices for predicting system vulnerability," *Electric Power Systems Research*, vol. 178, p. 106036, 2020.

[69] M. Czajkowski and M. Kretowski, "Decision tree underfitting in mining of gene expression data. an evolutionary multi-test tree approach," *Expert Systems with Applications*, vol. 137, pp. 392–404, 2019.

[70] H. A. Cuesta, D. L. Coffman, C. Branas, and H. M. Murphy, "Using decision trees to understand the influence of individual- and neighborhood-level factors on urban diabetes and asthma," *Health & Place*, vol. 58, p. 102119, 2019.

[71] H. Wang and M. Hong, "Online ad effectiveness evaluation with a two-stage method using a gaussian filter and decision tree approach," *Electronic Commerce Research and Applications*, vol. 35, p. 100852, 2019.

[72] C. Rodríguez-Pardo, A. Segura, J. J. Zamorano-León, C. Martínez-Santos, D. Martínez, L. Collado-Yurrita, M. Giner, J. M. García-García, J. M. Rodríguez-Pardo, and A. López-Farre, "Decision tree learning to predict overweight/obesity based on body mass index and gene polymporphisms," *Gene*, vol. 699, pp. 88–93, 2019.

[73] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[74] H. Sug, "An effective sampling method for decision trees considering comprehensibility and accuracy," *W. Trans. on Comp*, vol. 8, no. 4, pp. 631–640, 2009.

[75] W.-Y. Loh, "Fifty years of classification and regression trees," *International Statistical Review*, vol. 82, no. 3, pp. 329–348, 2014.

[76] J. Morgan, "Classification and regression tree analysis," *Boston: Boston University*, 2014.

[77] Y.-Y. Song and L. Ying, "Decision tree methods: applications for classification and prediction," *Shanghai Archives of Psychiatry*, vol. 27, no. 2, p. 130, 2015.

[78] J. I. Hoffman, "Chapter 29 - correlation," in *Basic Biostatistics for Medical and Biomedical Practitioners*, pp. 501–524, Academic Press, 2nd ed., 2019.

[79] A. Pal and A. Chakravarty, "Chapter 17 - heritability, repeatability, and genetic correlation of disease-resistance traits," in *Genetics and Breeding for Disease Resistance of Livestock*, pp. 245–258, Academic Press, 2020.

[80] C. Kim, S. Park, and Y. Ha, "Correlation analysis between vehicular traffic and PM using sensor big data," in *Proceedings of the International Conference on Big Data and Smart Computing*, pp. 644–648, IEEE, 2018.

[81] K. K. Lwin, K. Zettsu, and K. Sugiura, "Geovisualization and correlation analysis between geotagged twitter and jma rainfall data: Case of heavy rain disaster in hiroshima," in *Proceedings of the 2nd International Conference on Spatial Data Mining and Geographical Knowledge Services*, pp. 71–76, IEEE, 2015.

[82] K. Ozbay and N. Noyan, "Estimation of incident clearance times using bayesian networks approach," *Accident Analysis & Prevention*, vol. 38, no. 3, pp. 542–555, 2006.

[83] J. R. Quinlan, "Learning with continuous classes," in *Proceedings of the 5th Australian joint conference on artificial intelligence*, vol. 92, pp. 343–348, World Scientific, 1992.

[84] A. Akgündoğdu, I. Öz, and C. Uzunoğlu, "Signal quality based power output prediction of a real distribution transformer station using M5P model tree," *Electric Power Systems Research*, vol. 177, p. 106003, 2019.

[85] S. Singh, A. Yassine, and R. Benlamri, "Towards hybrid energy consumption prediction in smart grids with machine learning," in *Proceedings of the 4th International Conference on Big Data Innovations and Applications*, pp. 44–50, IEEE, 2018.

[86] S.-a. Blaifi, S. Moulahoum, R. Benkercha, B. Taghezouit, and A. Saim, "M5P model tree based fast fuzzy maximum power point tracker," *Solar Energy*, vol. 163, pp. 405–424, 2018.

[87] B. Wang, H. Wang, and S. Zhang, "Research on short-term wind power forecasting based on RMADE-SEN-IRFR," in *Proceedings of the 4th International Conference on Electrical & Electronics Engineering and Computer Science*, Atlantis Press, 2016.

[88] V. B. Krishna, W. S. Wadman, and Y. Kim, "Nowcasting: Accurate and precise short-term wind power prediction using hyperlocal wind forecasts," in *Proceedings of the 9th International Conference on Future Energy Systems*, pp. 63–74, ACM, 2018.

[89] L. Lin, Q. Wang, and A. W. Sadek, "A combined M5P tree and hazard-based duration model for predicting urban freeway traffic accident durations," *Accident Analysis & Prevention*, vol. 91, pp. 114–126, 2016.

[90] A. Behnood, V. Behnood, M. M. Gharehveran, and K. E. Alyamac, "Prediction of the compressive strength of normal and high-performance concretes using M5P model tree algorithm," *Construction and Building Materials*, vol. 142, pp. 199–207, 2017.

[91] C. W. H'ng and W. P. Loh, "A prediction of leaf mechanical properties with data mining," *Computers and Electronics in Agriculture*, vol. 162, pp. 669–676, 2019.

[92] B. Choubin, A. Malekian, and M. Golshan, "Application of several data-driven techniques to predict a standardized precipitation index," *Atmósfera*, vol. 29, no. 2, pp. 121–128, 2016.

[93] K. Bailey, "Numerical taxonomy and cluster analysis," *Typologies and Taxonomies*, vol. 34, p. 24, 1994.

[94] P. Govender and V. Sivakumar, "Application of k-means and hierarchical clustering techniques for analysis of air pollution: A review (1980–2019)," *Atmospheric Pollution Research*, vol. 11, no. 1, pp. 40–56, 2020.

[95] W. Xu and Y. Peng, "Research on classified real-time flood forecasting framework based on k-means cluster and rough set," *Water Science and Technology*, vol. 71, no. 10, pp. 1507–1515, 2015.

[96] J. Sun, W. Chen, W. Fang, X. Wun, and W. Xu, "Gene expression data analysis with the clustering method based on an improved quantum-behaved particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 2, pp. 376–391, 2012.

[97] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society*, vol. 28, no. 1, pp. 100–108, 1979.

[98] N. Akhtar and M. V. Ahmad, "A modified fuzzy C means clustering using neutrosophic logic," in *Proceedings of the 5th International Conference on Communication Systems and Network Technologies*, pp. 1124–1128, IEEE, 2015.

[99] N. Akthar, M. V. Ahamad, and S. Ahmad, "Mapreduce model of improved k-means clustering algorithm using hadoop mapreduce," in *Proceedings of the 2nd International Conference on Computational Intelligence & Communication Technology*, pp. 192–198, IEEE, 2016.

[100] Y. Chen, "Automatic microseismic event picking via unsupervised machine learning," *Geophysical Journal International*, vol. 212, no. 1, pp. 88–102, 2017.

[101] N. Riahi and P. Gerstoft, "Using graph clustering to locate sources within a dense sensor array," *Signal Processing*, vol. 132, pp. 110–120, 2017.

[102] A. Nithya, A. Appathurai, N. Venkatadri, D. Ramji, and C. A. Palagan, "Kidney disease detection and segmentation using artificial neural network and multi-kernel k-means clustering for ultrasound images," *Measurement*, vol. 149, p. 106952, 2020.

[103] Z. kai Feng, W. jing Niu, R. Zhang, S. Wang, and C. tian Cheng, "Operation rule derivation of hydropower reservoir by k-means clustering

method and extreme learning machine based on particle swarm optimization," *Journal of Hydrology*, vol. 576, pp. 229–238, 2019.

[104] R. Ghezelbash, A. Maghsoudi, and E. J. M. Carranza, "Optimization of geochemical anomaly detection using a novel genetic k-means clustering (GKMC) algorithm," *Computers & Geosciences*, vol. 134, p. 104335, 2020.

[105] M. N. Reza, I. S. Na, S. W. Baek, and K.-H. Lee, "Rice yield estimation based on k-means clustering with graph-cut segmentation using low-altitude uav images," *Biosystems Engineering*, vol. 177, pp. 109–121, 2019.

[106] J. Liu, Q. Li, W. Chen, and T. Cao, "A discrete hidden markov model fault diagnosis strategy based on k-means clustering dedicated to pem fuel cell systems of tramways," *International Journal of Hydrogen Energy*, vol. 43, no. 27, pp. 12428–12441, 2018.

[107] M. N. Qureshi and M. V. Ahamad, "An improved method for image segmentation using k-means clustering with neutrosophic logic," *Procedia Computer Science*, vol. 132, pp. 534–540, 2018.

[108] A. Pal and S. Agrawal, "An experimental approach towards big data for analyzing memory utilization on a hadoop cluster using HDFS and MapReduce," in *Proceedings of the 1st International Conference on Networks Soft Computing*, pp. 442–447, 2014.

[109] J. Wang, Y. Yao, Y. Mao, B. Sheng, and N. Mi, "Fresh: Fair and efficient slot configuration and scheduling for hadoop clusters," in *Proceedings of the 7th International Conference on Cloud Computing*, pp. 761–768, IEEE, 2014.

[110] K. Kc and V. W. Freeh, "Tuning hadoop map slot value using cpu metric," in *Proceedings of the Big Data Benchmarks, Performance Optimization, and Emerging Hardware*, pp. 141–153, Springer International Publishing, 2014.

[111] Y. Yao, J. Wang, B. Sheng, and N. Mi, "Using a tunable knob for reducing makespan of mapreduce jobs in a hadoop cluster," in *Proceedings of the 6th International Conference on Cloud Computing*, pp. 1–8, IEEE, 2013.

[112] Xueyuan, Brian, and Yuansong, “Experimental evaluation of memory configurations of hadoop in docker environments,” in *Proceedings of the 27th Irish Signals and Systems Conference*, pp. 1–6, 2016.

[113] W. F. Tichy, “Should computer scientists experiment more?,” *Computer*, vol. 31, pp. 32–40, May 1998.

[114] M. V. Zelkowitz and D. R. Wallace, “Experimental models for validating technology,” *Computer*, vol. 31, pp. 23–31, May 1998.

[115] V. R. Basili and M. V. Zelkowitz, “Empirical studies to build a science of computer science,” *Communications of the ACM*, vol. 50, no. 11, pp. 33–37, 2007.

[116] D. I. K. Sjoberg, T. Dyba, and M. Jorgensen, “The future of empirical methods in software engineering research,” in *Proceedings of the Future of Software Engineering*, pp. 358–378, 2007.

[117] K. TA, “HAZOP and Hazan: identifying and assessing process industry hazards,” *Rugby, UK: Institution of Chemical Engineers*, pp. 1–223, 1999.

[118] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, 1990.

[119] G. Seni and J. F. Elder, “Ensemble methods in data mining: Improving accuracy through combining predictions,” *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 1–126, 2010.

[120] B. J. Mathiya and V. L. Desai, “Apache hadoop Yarn parameter configuration challenges and optimization,” in *Proceedings of the International Conference on Soft-Computing and Networks Security*, pp. 1–6, 2015.

[121] Y. Zhai, Y.-S. Ong, and I. W. Tsang, “The emerging” big dimensionality”,” 2014.

[122] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, 2013.

[123] Y. You, S. L. Song, H. Fu, A. Marquez, M. M. Dehnavi, K. Barker, K. W. Cameron, A. P. Randles, and G. Yang, "Mic-svm: Designing a highly efficient support vector machine for advanced modern multi-core and many-core architectures," in *Proceedings of the 28th International Parallel and Distributed Processing Symposium*, pp. 809–818, IEEE, 2014.

[124] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, pp. 2–2, USENIX Association, 2012.

[125] L. J. Cao, S. S. Keerthi, C. J. Ong, J. Q. Zhang, U. Periyathamby, X. J. Fu, and H. Lee, "Parallel sequential minimal optimization for the training of support vector machines," *IEEE Transactions of Neural Networks*, vol. 17, no. 4, pp. 1039–1049, 2006.

[126] K. Olukotun, "Beyond parallel programming with domain specific languages," *ACM SIGPLAN Notices*, vol. 49, no. 8, pp. 179–180, 2014.

[127] T. White, *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[128] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *proceedings of the SIGOPS Operating Systems Review*, vol. 37, pp. 29–43, ACM, 2003.

[129] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the 26th symposium on Mass Storage Systems and Technologies*, pp. 1–10, IEEE, 2010.

[130] I. Polato, R. Ré, A. Goldman, and F. Kon, "A comprehensive view of hadoop research—a systematic literature review," *Journal of Network and Computer Applications*, vol. 46, pp. 1–25, 2014.

[131] A. V. Hazarika, G. J. S. R. Ram, and E. Jain, "Performance comparision of hadoop and spark engine," in *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud*, pp. 671–674, 2017.

[132] D. P. Acharjya and K. Ahmed, "A survey on big data analytics: Challenges, open research issues and tools," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, pp. 511–518, 2016.

[133] P. Costa, M. Pasin, A. N. Bessani, and M. P. Correia, "On the performance of byzantine fault-tolerant mapreduce," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 5, pp. 301–313, 2013.

[134] S. Mikami, K. Ohta, and O. Tatebe, "Using the Gfarm file system as a POSIX compatible storage platform for hadoop mapreduce applications," in *Proceedings of the 12th International Conference on Grid Computing*, pp. 181–189, IEEE, 2011.

[135] Y. Kang, Y.-s. Kee, E. L. Miller, and C. Park, "Enabling cost-effective data processing with smart SSD," in *Proceedings of the 29th symposium on mass storage systems and technologies*, pp. 1–12, IEEE, 2013.

[136] W. Wei, J. Du, T. Yu, and X. Gu, "Securemr: A service integrity assurance framework for mapreduce," in *Proceedings of the Annual Computer Security Applications Conference*, pp. 73–82, IEEE, 2009.

[137] H.-Y. Lin, S.-T. Shen, W.-G. Tzeng, and B.-S. P. Lin, "Toward data confidentiality via integrating hybrid encryption schemes and hadoop distributed file system," in *Proceedings of the 26th International Conference on Advanced Information Networking and Applications*, pp. 740–747, IEEE, 2012.

[138] S. M. Khan and K. W. Hamlen, "Hatman: Intra-cloud trust management for hadoop," in *Proceedings of the 5th International Conference on Cloud Computing*, pp. 494–501, IEEE, 2012.

[139] C. A. B. S. C. L. Benoy Antony, Konstantin Boudnik and K. Sasaki, *Hadoop Introduction*, ch. 1, pp. 1–14. John Wiley & Sons, Ltd, 2016.

[140] K. V. Shvachko, "HDFS scalability: The limits to growth," *Login*, vol. 35, no. 2, pp. 6–16, 2010.

[141] M. Bhandarkar, "Mapreduce programming with apache hadoop," in *Proceedings of the International Symposium on Parallel & Distributed Processing*, pp. 1–1, IEEE, 2010.

[142] R. Buyya, C. Vecchiola, and S. T. Selvi, "Chapter 1 - introduction," in *Mastering Cloud Computing*, pp. 3–27, Morgan Kaufmann, 2013.

[143] D. Talia, P. Trunfio, and F. Marozzo, "Chapter 2 - introduction to cloud computing," in *Data Analysis in the Cloud*, pp. 27–43, Elsevier, 2016.

[144] D. Sitaram and G. Manjunath, "Chapter 3 - platform as a service," in *Moving To The Cloud*, pp. 73–152, Syngress, 2012.

[145] M. R. Movahedisefat, S. M. R. Farshchi, and D. Mohammadpur, "Chapter 23 - emerging security challenges in cloud computing, from infrastructure-based security to proposed provisioned cloud infrastructure," in *Emerging Trends in ICT Security*, pp. 379–393, Morgan Kaufmann, 2014.

[146] P. Derbeko, S. Dolev, E. Gudes, and S. Sharma, "Security and privacy aspects in mapreduce on clouds: A survey," *Computer Science Review*, vol. 20, pp. 1–28, 2016.

[147] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, p. 5, ACM, 2013.

[148] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: A warehousing solution over a map-reduce framework," *Proc. VLDB Endow.*, vol. 2, pp. 1626–1629, Aug. 2009.

[149] B. Antony, K. Boudnik, C. Adams, B. Shao, C. Lee, and K. Sasaki, *Professional Hadoop*. John Wiley & Sons, 2016.

[150] P. Raj and G. C. Deka, *A Deep Dive into NoSQL Databases: The Use Cases and Applications*, vol. 109. Academic Press, 2018.

[151] G. Hwaiyu and J. McKeeth, *Internet of things and data analytics handbook*. Wiley Online Library, 2016.

[152] A. Dumka, *Innovations in Software-Defined Networking and Network Functions Virtualization*. IGI Global, 2018.

[153] O. Coker and S. Azodolmolky, *Software-defined Networking with OpenFlow: Deliver Innovative Business Solutions*. Packt Publishing Ltd, 2017.

[154] H. Qi and K. Li, *Introduction*, pp. 1–12. Cham: Springer International Publishing, 2016.

[155] P. Heidelberger and S. S. Lavenberg, “Computer performance evaluation methodology,” *IEEE Transactions on Computers*, vol. C-33, pp. 1195–1220, Dec 1984.

[156] F. Ahmad, S. Lee, M. Thottethodi, and T. Vijaykumar, “Puma: Purdue mapreduce benchmarks suite,” 2012.

[157] J. Zhou, T. Wang, P. Cong, P. Lu, T. Wei, and M. Chen, “Cost and makespan-aware workflow scheduling in hybrid clouds,” *Journal of Systems Architecture*, vol. 100, p. 101631, 2019.

[158] A. Baskar and M. A. Xavior, “Optimization of makespan in job and machine priority environment,” *Procedia Engineering*, vol. 97, pp. 22–28, 2014.

[159] G. Bansal, A. Gupta, U. Pyne, M. Singhal, and S. Banerjee, “A framework for performance analysis and tuning in hadoop based clusters,” in *Proceedings of the Smarter Planet and Big Data Analytics Workshop, International Conference on Distributed Computing and Networking*, 2014.

[160] J. P. B. Nascimento, D. O. Capanema, and A. C. M. Pereira, “Assessing and improving the performance and scalability of an iterative algorithm for hadoop,” in *Proceedings of the Computing Conference*, pp. 1069–1076, IEEE, 2017.

[161] Sacerdoti, Katz, Massie, and Culler, "Wide area cluster monitoring with ganglia," in *Proceedings of the International Conference on Cluster Computing*, pp. 289–298, 2003.

[162] I. El-Helw, R. Hofman, and H. E. Bal, "Scaling MapReduce vertically and horizontally," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 525–535, 2014.

[163] D. Beaumont, "How to explain vertical and horizontal scaling in the cloud." `https://www.ibm.com/blogs/cloud-computing/2014/04/09/explain-vertical-horizontal-scaling-cloud/`, 2019. [Online; accessed 11-10-2019].

[164] M. Franzese and A. Iuliano, "Correlation analysis," in *Encyclopedia of Bioinformatics and Computational Biology*, pp. 706–721, Academic Press, 2019.

[165] F. Gagné, "Chapter 12 - descriptive statistics and analysis in biochemical ecotoxicology," in *Biochemical Ecotoxicology*, pp. 209–229, Academic Press, 2014.

[166] P. A. Shaw, L. L. Johnson, and M. A. Proschan, "Chapter 27 - intermediate topics in biostatistics," in *Principles and Practice of Clinical Research*, pp. 383–409, Academic Press, 2018.

[167] M. Glantz and J. Mun, "Chapter 7 - a primer on quantitative risk analysis," in *Credit Engineering for Bankers*, pp. 129–183, Academic Press, 2011.

[168] R. Riffenburgh, "Chapter 21 - regression and correlation," in *Statistics in Medicine*, pp. 443–472, San Diego: Academic Press, 2012.

[169] D. EDMONDS, "Chapter 13 - commodity trading advisors and their role in managed futures," in *Advanced Trading Rules*, pp. 367–387, Butterworth-Heinemann, 2002.

[170] J. Musek, "Chapter 9 - cognitive abilities and personality: Two comprehensive general factors," in *The General Factor of Personality*, p. 257–281, Academic Press, 2017.

[171] F. H. Stephenson, "9 - recombinant dna," in *Calculations for Molecular Biology and Biotechnology*, pp. 186–241, Academic Press, 2003.

[172] R. Nisbet, G. Miner, and K. Yale, "Chapter 10 - numerical prediction," in *Handbook of Statistical Analysis and Data Mining Applications*, pp. 187–213, Academic Press, 2018.

[173] A. F. Siegel, "Chapter 11 - correlation and regression: Measuring and predicting relationships," in *Practical Business Statistics*, pp. 299–354, Academic Press, 2016.

[174] S. M. Nabavinejad and M. Goudarzi, "Faster mapreduce computation on clouds through better performance estimation," *IEEE Transactions on Cloud Computing*, vol. 7, pp. 770–783, July 2019.

[175] J. Sauro and J. R. Lewis, "Chapter 10 - an introduction to correlation, regression, and anova," in *Quantifying the User Experience*, pp. 277–320, Morgan Kaufmann, second edition ed., 2016.

[176] D. Nettleton, "Chapter 6 - selection of variables and factor derivation," in *Commercial Data Mining*, pp. 79–104, Morgan Kaufmann, 2014.

[177] G. Kordelas and P. Daras, "Robust sift-based feature matching using kendall's rank correlation measure," in *Proceedings of the 16th International Conference on Image Processing*, pp. 325–328, IEEE, 2009.

[178] I. Couso, O. Strauss, and H. Saulnier, "Kendall's rank correlation on quantized data: An interval-valued approach," *Fuzzy Sets and Systems*, vol. 343, pp. 50–64, 2018.

[179] C. Li, H. Zhuang, K. Lu, M. Sun, J. Zhou, D. Dai, and X. Zhou, "An adaptive auto-configuration tool for hadoop," in *Proceedings of the 2014 19th International Conference on Engineering of Complex Computer Systems*, pp. 69–72, IEEE, 2014.

[180] Y. Zhu, J. Liu, M. Guo, Y. Bao, W. Ma, Z. Liu, K. Song, and Y. Yang, "Bestconfig: Tapping the performance potential of systems via automatic con-

figuration tuning," in *Proceedings of the Symposium on Cloud Computing*, pp. 338–350, ACM, 2017.

[181] B. S. Kim and T. G. Kim, "Cooperation between data modeling and simulation modeling for performance analysis of hadoop," in *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pp. 1–7, 2017.

[182] G. Wang, J. Xu, and B. He, "A novel method for tuning configuration parameters of spark based on machine learning," in *Proceedings of the 18th International Conference on High Performance Computing and Communications; 14th International Conference on Smart City; 2nd International Conference on Data Science and Systems*, pp. 586–593, IEEE, 2016.

[183] N. Yigitbasi, T. L. Willke, G. Liao, and D. Epema, "Towards machine learning-based auto-tuning of mapreduce," in *Proceedings of the 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 11–20, 2013.

[184] M. Xu, S. Alamro, T. Lan, and S. Subramaniam, "Laser: A deep learning approach for speculative execution and replication of deadline-critical jobs in cloud," in *Proceedings of the 26th International Conference on Computer Communication and Networks*, pp. 1–8, 2017.

[185] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the International Conference on Computer Vision*, pp. 1026–1034, IEEE, 2015.

[186] Y. Ling, F. Liu, Y. Qiu, and J. Zhao, "Prediction of total execution time for mapreduce applications," in *Proceedings of the Sixth International Conference on Information Science and Technology*, pp. 341–345, 2016.

[187] D. T. Campbell and T. D. Cook, *Quasi-experimentation: Design & analysis issues for field settings*. Rand McNally College Publishing Company Chicago, 1979.

[188] P. Lukowicz and S. Intille, "Experimental methodology in pervasive computing," *IEEE Pervasive Computing*, vol. 10, pp. 94–96, April 2011.

[189] J. C. Knight and N. G. Leveson, "An experimental evaluation of the assumption of independence in multiversion programming," *IEEE Transactions on Software Engineering*, vol. SE-12, pp. 96–109, Jan 1986.

[190] K. Killourhy and R. Maxion, "The effect of clock resolution on keystroke dynamics," in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, pp. 331–350, Springer, 2008.

[191] P. A. Fishwick, "Neural network models in simulation: A comparison with traditional modeling approaches," in *Proceedings of the Winter Simulation Conference*, pp. 702–710, 1989.

[192] R. Maxion, "Making experiments dependable," in *Dependable and Historic Computing*, pp. 344–357, Springer, 2011.

[193] R. A. Maxion and K. S. Killourhy, "Should security researchers experiment more and draw more inferences," in *Proceedings of 4th USENIX Workshop on Cyber Security Experimentation and Test*, p. 5, 2011.

[194] C. Seifert, "Cost-effective detection of drive-by-download attacks with hybrid client honeypots," 2010.

[195] T. Srivatanakul, J. A. Clark, and F. Polack, "Effective security requirements analysis: HAZOP and use cases," in *Proceedings of the International Conference on Information Security*, pp. 416–427, Springer, 2004.

[196] K. Lano, D. Clark, and K. Androutsopoulos, "Safety and security analysis of object-oriented models," in *Proceedings of the International Conference on Computer Safety, Reliability, and Security*, pp. 82–93, Springer, 2002.

[197] M. Mansoori, I. Welch, K.-K. R. Choo, R. A. Maxion, and S. E. Hashemi, "Real-world ip and network tracking measurement study of malicious websites with hazop," *International Journal of Computers and Applications*, vol. 39, no. 2, pp. 106–121, 2017.

[198] N. G. Leveson, "Software safety in embedded computer systems," *Commun. ACM*, vol. 34, pp. 34–46, Feb. 1991.

[199] R. Winther, O.-A. Johnsen, and B. A. Gran, "Security assessments of safety critical systems using HAZOPs," in *Proceedings of the International Conference on Computer Safety, Reliability, and Security*, pp. 14–24, Springer, 2001.

[200] F. Paternò and C. Santoro, "Preventing user errors by systematic analysis of deviations from the system task model," *International Journal of Human-Computer Studies*, vol. 56, no. 2, pp. 225–245, 2002.

[201] P. Baybutt, "A critique of the hazard and operability (HAZOP) study," *Journal of Loss Prevention in the Process Industries*, vol. 33, pp. 52–58, 2015.

[202] T. Ylonen and C. Lonvick, "The secure shell (ssh) protocol architecture," 2006.

[203] A. Hadoop, "Hadoop 2.0 installation on windows." `https://cwiki.apache.org/confluence/display/HADOOP2/Hadoop2OnWindows`, 2019. [Online; accessed 2019-10-10].

[204] Faucet, "Faucet version 1.6.8." `https://github.com/faucetsdn/faucet`. [Online; accessed 2017-09-18].

[205] P. Bhavsar, I. Safro, N. Bouaynaya, R. Polikar, and D. Dera, "Chapter 12 - machine learning in transportation data analytics," in *Data Analytics for Intelligent Transportation Systems*, pp. 283–307, Elsevier, 2017.

[206] M. Bramer, *Introduction to Classification: Naïve Bayes and Nearest Neighbour*, pp. 21–37. Springer London, 2016.

[207] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[208] M. Hegland, *Algorithms for Association Rules*, pp. 226–234. Springer Berlin Heidelberg, 2003.

[209] S. Suthaharan, *Support Vector Machine*, pp. 207–235. Springer US, 2016.

[210] A. Najm, A. Zakrani, and A. Marzak, "Decision trees based software development effort estimation: A systematic mapping study," in *Proceedings of the International Conference of Computer Science and Renewable Energies*, pp. 1–6, 2019.

[211] N. Westland, A. S. Dias, and M. Mrak, "Decision trees for complexity reduction in video compression," *arXiv preprint arXiv:1908.04168*, 2019.

[212] S. Idowu, C. Åhlund, O. Schelén, and R. Brännström, *Machine learning in pervasive computing*. Luleå tekniska universitet, 2013.

[213] S. Suthaharan, *Decision Tree Learning*, pp. 237–269. Springer US, 2016.

[214] Apache, "Hive." `https://hive.apache.org/`. [Online; accessed 2019-07-17].

[215] R. Jin, Y. Breitbart, and C. Muoh, "Data discretization unification," in *Proceedings of the 7th International Conference on Data Mining*, pp. 183–92, IEEE, 2007.

[216] R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[217] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[218] InfluxDB, "Influxdb 2.0." `https://www.influxdata.com/`. [Online; accessed 2019-06-23].

[219] Grafana. `https://grafana.com/`. [Online; accessed 2019-06-23].

[220] A. E. C. Cloud. `https://aws.amazon.com/ec2/`. [Online; accessed 2019-12-02].

[221] B. Schmarzo, *Big Data MBA: Driving Business Strategies with Data Science*. John Wiley & Sons, 2015.

[222] L. Chihara and T. Hesterberg, *Mathematical statistics with resampling and R*. Wiley Online Library, 2011.

[223] E. Suárez, C. M. Pérez, R. Rivera, and M. N. Martínez, *Correlation Analysis*, ch. 6, pp. 87–96. John Wiley & Sons, Ltd, 2017.

[224] M. Bramer, "Data for data mining," in *Principles of data mining*, pp. 9–19, Springer, 2016.

[225] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[226] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[227] H. Lee, *Foundations of applied statistical methods*. Springer, 2014.

[228] C. Zhan, A. Gan, and M. Hadi, "Prediction of lane clearance time of freeway incidents using the M5P tree algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1549–1557, 2011.

[229] M. J. Menne, I. Durre, B. Korzeniewski, S. McNeal, K. Thomas, X. Yin, S. Anthony, R. Ray, R. S. Vose, B. E. Gleason, *et al.*, "Global historical climatology network-daily," *NOAA National Climatic Data Center*, vol. 10, 2012.