Summer 2014

Systems For Delivering Electric Vehicle Data Analytics

Vamshi Krishna Bolly
Purdue University

Graduate School ETD Form 9
(Revised 01/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Vamshi Krishna Bolly

Entitled

"SYSTEMS FOR DELIVERING ELECTRIC VEHICLE DATA ANALYTICS"

For the degree of Master of Science

Is approved by the final examining committee:

John A. Springer

J. Eric Dietz

Eric T. Matson

To the best of my knowledge and as understood by the student in the *Thesis/Dissertation Agreement. Publication Delay, and Certification/Disclaimer (Graduate School Form 32)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

John A. Springer

Approved by Major Professor(s): _____

_____

Approved by: Jeffrey L. Whitten 05/06/2014

Head of the Department Graduate Program Date

SYSTEMS FOR DELIVERING ELECTRIC VEHICLE DATA ANALYTICS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Vamshi K. Bolly

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

August 2014

Purdue University

West Lafayette, Indiana

ACKNOWLEDGEMENTS

I take this opportunity to express my sincere gratitude to my Major Professor Dr. John A. Springer who has been a great motivator and guide for my research. His able guidance and unconditional support helped greatly in shaping my research all along. Thanks to him, my journey at Purdue University has been a memorable one.

I would also thank Dr. J. Eric Dietz, for being on my committee and helping my research. He has been active in providing suggestions and inputs that greatly helped the content of this research. I also thank Dr. Eric T. Matson, for serving on my committee and helping me refine my research. It has been a great pleasure working with you.

I thank my family and all my friends who have provided me the support when I needed it the most.

I also thank everyone who directly or indirectly has influenced my graduate studies at Purdue University.

TABLE OF CONTENTS

Page

Page

Page

LIST OF FIGURES

Figure Page

Figure Page

LIST OF TABLES

Appendix Table Page

LIST OF ABBREVIATIONS

NoSQL – Not only Structured Query Language

EV – Electric Vehicle

SoC – State of Charge

CSV – Comma separated file

HDFS – Hadoop distributed file system

MR - Map reduce

VM – Virtual Machine

RHEL – Red Hat Enterprise Linux

ANOVA – Analysis of Variance

GLOSSARY

Big data – A large collection of data that is unstructured, multi-dimensional and complex in nature and needs superior processing systems for performing analytics (Padhy, 2012).

Hadoop – Java based open source software framework for data storage and performing data analytics (Padhy, 2012).

MapReduce – A programming paradigm for batch processing large datasets using programming languages (White, 2012).

ABSTRACT

Bolly, Vamshi Krishna. M.S., Purdue University, August 2014. Systems for delivering electric vehicle data analytics. Major Professor: John A. Springer.

In the recent times, advances in scientific research related to electric vehicles led to generation of large amounts of data. This data is majorly logger data collected from various sensors in the vehicle. It is predominantly unstructured and non-relational in nature, also called Big Data. Analysis of such data needs a high performance information technology infrastructure that provides superior computational efficiency and storage capacity. It should be scalable to accommodate the growing data and ensure its security over a network. This research proposes an architecture built over Hadoop to effectively support distributed data management over a network for real-time data collection and storage, parallel processing, and faster random read access for information retrieval for decision-making.

Once imported into a database, the system can support efficient analysis and visualization of data as per user needs. These analytics can help understand correlations between data parameters under various circumstances. This system provides scalability to support data accumulation in the future and still perform analytics with less overhead. Overall, these open problems in EV data analytics are taken into consideration and a low-cost architecture for data management is researched.

CHAPTER 1. INTRODUCTION

This chapter introduces the essential aspects of the idea being implemented. The research statement is specified in the beginning. The section then discusses the primary reason behind the idea as explained by the statement of purpose, scope and significance. Important definitions focusing on Big Data and NoSQL are specified. The imperative assumptions, limitations and delimitations integral to the idea are also provided.

1.1 Research question

In my research, I plan to research about, “Evaluating the effectiveness of Hadoop in designing systems that satisfy the electric vehicle data analytic requirements?”

1.2 Statement of purpose

In the recent times, advances in scientific research related to electric vehicles led to generation of large amounts of data. This data is primarily logger data collected from instruments or sensors inside the vehicle. The nature of such data is predominantly unstructured and non-relational, also called Big Data. Science has still not been fully capable of inferring meaning from this data and utilizing it for solving real-world issues. Analysis of such data requires a high performance information technology (IT) infrastructure that provides superior computational efficiency and storage capability. This

system has to be scalable to accommodate any future data growth and ensure its security over a network at all times. This research proposes an architecture that is built over Hadoop to effectively support distributed data management over a network. It supports real-time data collection and storage, parallel processing, and faster and random information retrieval for decision-making. This system provides a simplified way of extracting data from data loggers and transforming this raw data into pre-classified buckets or tables. Once imported into the data repository, the system can perform analytics to help identify any correlation between parameters under certain driving conditions. This system is built to scale for any incoming data with negligible change in its performance.

Another important dimension in analytics is visualization of data to better identify any hidden patterns between parameters. Hence, the data management system has to be compatible with various visualization applications. Overall, these open problems in EV data analytics are taken into due consideration and a system for performing data analytics is researched.

1.3 Scope

The scope of this research is primarily focused on designing a data repository for performing electric vehicle (EV) data analytics. This repository is built using components in Hadoop in a high performance distributed computing environment. Essentially, the dataset consists of parameters related to EVs recorded over a period of time under real world commuting conditions. Physically, this data is schema-less and is organized into sets of comma separated value (CSV) files that store information about EV driving

parameters recorded during a driving session. The repository is flexible to store and process large amounts of data in parallel across the multi-node distributed cluster configuration. Due to the schema free nature of the stored data, it allows for seamless inclusion of new data into existing tables with no storage overhead or schema re-design. This negates the need for having a database administrator to manage any volatility of data.

1.4 Significance

A distributed multi-node cluster will have greater processing speeds and storage capacity. NoSQL technologies were the first choice, as they are capable of handling large scale unstructured data. Hadoop Distributed File System (HDFS) works greatly in leveraging the advantages of having distributed storage in a multi-node cluster. Hadoop systems don't have any specific query language, which gives a scope for employing various query methods like MapReduce, HiveQL, etc. This flexibility can help in improving data retrieval speeds and supporting distributed queries.

The analytics on EV datasets can help predict the battery usage patterns under various driving conditions and driver profiles. This data repository can be analyzed and modeled for identifying patterns between parameters and the influence of driving conditions on them. Analyzing the variation of various parameters in the datasets can help relate them to the battery usage during driving conditions.

1.5 Assumptions

The following assumptions are integral to the research being implemented:

- Hadoop systems are suitable for scalable, distributed and data-intensive computing.
- The HDFS is always up and can work directly with any distributed file system of the underlying operating system.
- The mapper and reducer jobs run successful and are capable of parallel processing of large-scale data in a reliable and fault-tolerant way.
- The electric vehicle (EV) data loggers record information and write them consistently.
- The datasets are comma-separated files (CSVs) and have header information.
- All the CSVs at least have EV registration number, recorded event type, value and timestamp information.
- All driving parameters exhibit a correlation with the battery state of charge (SoC).
- The battery state of health (SoH) at the given instant can be calculated from the SoC trend up to that instant.
- The methodology chosen for this research is sufficient to address the research question.

1.6 Limitations

The following limitations are integral to the research being implemented:

- The EVs may not have significant number of recording sessions.
- The data loggers can introduce errors into the recorded data.
- The number of maps depends on the total size of inputs.

- Reduce operations do not take place until all map jobs are complete.

1.7 Delimitations

The following delimitations are integral to the research being implemented:

- This study pertains to use of NoSQL technologies such as Hadoop for designing data repository for EV analytics.
- Users tend to use HDFS more for batch processing needs rather than interactive use.
- The scope of this study is confined to EV data from Project Plug-IN.
- The output of reducer jobs is always smaller than the input to a map.

1.8 Chapter summary

This chapter essentially provides an overview to the area of research being implemented by this project. It introduces the research question being probed by this research in addition to the scope and significance of the idea. The assumptions, limitations and delimitations that govern this research are described in detail. Overall, this chapter provided a concrete foundation in explaining reasons for conducting this research.

CHAPTER 2. REVIEW OF RELEVANT LITERATURE

The major focus of this research has been the concept of Big Data and its role in dealing with unstructured data from scientific research. Big Data, in a way, has greatly assisted data scientists in dealing with unstructured information originating from divergent sources. The technologies of Big Data analytics have come a great way in addressing problems that traditional relational databases failed to solve. The limitations with traditional models in managing unstructured datasets have provided motivation for database research to focus on Big Data related technologies.

This chapter reviews the relevant literature in the area of study accomplished by this research. It essentially covers the significant topics in the area of research and establishes their relationship with the proposed research question.

2.1 The dawn of Big Data

In recent times, scientific research community has been accumulating massive amounts of data from various experiments and analyzing it to extract useful information (Cuzzocrea, Song, & Davis, 2011). As this data is predominantly collected from heterogeneous sources, it tends to be multi-dimensional and unstructured in nature. This kind of large scale unstructured multi-dimensional data is known as Big Data. This kind

of data requires enriched systems that can help integrate it and allow querying on it for information retrieval.

By nature, Big Data is a large complex dataset that is made up of records, variables, intricate structures and dependencies that need complex analysis methods. Hence, any analysis of such data will be cumbersome and in some cases infeasible for certain kinds of analyses, which need parallel processing of different variables (Guha, Kidwell, Hafen, & Cleveland, 2009). In spite of the associated complexities, the main goal of research is to comprehensively study this data without losing any important information. One way for achieving such analysis is to partition it into small subsets and apply numeric methods to each sample. Data visualization is another way that helps in comprehensive understanding of such data (Guha, Kidwell, Hafen, & Cleveland, 2009). Therefore, it is important that this data be extracted from sources, transformed into understandable structures, processed for storage, and managed for analysis so as to easily represent it through media like charts, plots, dashboards, etc., for making informed decisions.

2.2 Challenges in Big Data

Big Data is recorded from various data generating sources like experiments, simulations, surveys, etc. The nature of such data tends to be dynamic, heterogeneous, untidy and sometimes untrustworthy. Even after purging the data, there will be some inherent errors and incompleteness associated with it. These factors should be taken into consideration during its analysis. Nevertheless, with relevant analysis and correlational studies, such data can be used to identify any hidden patterns between parameters across

the dataset. The computing systems need to capture these challenges and help analyze, model and interpret such scientific data.

The proliferation of data everyday poses a significant challenge to accumulation, integration and storage of it. Such a rapid growth demands extensive scalability support from the computational systems. As the size of data increases, so does the query time for retrieving any information from it. Timeliness of information retrieval is another important challenge in Big Data analytics. As a result, it is important that the query method leverages the internal infrastructure and processing capabilities to retrieve the required information from the datasets in minimum time. This inherently creates a need for human collaboration during analytics, as most of the data needs human intervention. The associated analytics environment needs to support shared interpretation of the queried results among multiple data researchers. Hence, these factors have to be holistically considered while designing, building and operating the data processing components.

2.3 Significance of NoSQL

In the world of computing, a database is an essential component that stores and serves large amounts of information as per user demand (Padhy, Patra, & Satapathy, 2011). Distributed storage and computing infrastructures are evolving to contain large amounts of data scaling to petabytes. The main delimitations of the current relational database model are that the data schema is not dynamically scalable, increases in query expense with increasing amounts of data and currently has unstable query plans. A non-relational database system can overcome these limitations as it can store unstructured

data and can differentiate it from another. This type of database is called a NoSQL database, which is schema-free, non-relational, always available through data replication, dynamically scalable and exhibits eventual consistency (Padhy, Patra, & Satapathy, 2011).

Recent trends in computing strongly advocate a non-relational model of data storage (Padhy, Patra, & Satapathy, 2011). The need for providing efficient and scalable databases is the main contributing factor in this transition. Even though, the RDBMS model continues to exist, NoSQL database technologies will keep evolving to better address the storage and performance requirements (Padhy, Patra, & Satapathy, 2011).

2.4 Introduction to Hadoop

The opportunities for new discoveries in research led to the availability of large amounts of data, which created challenges in sharing this data and preserving it for the future (Steinhart, 2010). In order to address these challenges of Big Data, there is a need for enriching computational technologies that support these requirements at their core.

Hadoop, an open source Java-based software framework, is a scalable data management system for distributed computing clusters that provides an ideal solution for performing Big Data analytics (Padhy, 2012). It is part of the Apache Software Foundation and is designed to support data-intensive distributed computing. It enables applications to operate on multiple computing nodes and work with large amounts of data. The two important components that achieve these characteristics are Hadoop Distributed File System (HDFS) and MapReduce processing. The Hadoop architecture

inherently provides support for managing the computing nodes in a cluster and works on reducing the traffic latency between server nodes (Padhy, 2012).

HDFS is a native distributed file system implementation in Hadoop, which makes data readily available to all the nodes in a cluster. It works on a master-subordinate architecture where a NameNode manages multiple DataNodes. The NameNode contains information on the data contained in each DataNode. A large data file is broken into multiple data chunks based on the user-defined block size and these chunks are distributed across all the nodes in the cluster. It provides reliable access to data in the cluster during quick computations. It stores application data in a persistent way and dynamically manages data replication, availability and distribution across the nodes (Attebury, Baranovski, Bloom, Bockelman, Kcira, Letts, Levshina, Lundestedt, Martin, Maier, Pi, Rana, Sfiligoi, Sim, Thomas, & Wuerthwein, 2009). Data coherency is maintained in HDFS as it is implemented to write data once and read it multiple times. Inherently, this implies that the data once written onto HDFS can't be modified or corrupted by external programs.

MapReduce is a programming framework for processing large amounts of data in parallel on huge distributed clusters (Shafer, Rixner, & Cox, 2010). This paradigm is suited to perform batch processing of large datasets containing multiple files. It works on a master-subordinate architecture where a JobTracker breaks a job into multiple tasks, assigns them to individual TaskTrackers, and monitors their status. The JobTracker takes care of reassigning failed tasks to available TaskTrackers and ensures that there is no single point of failure. In the initial map phase, a map function reads each line in the raw

data and converts it into one or more [key, value] pairs. All the values that share the same key are sorted into list of multiple [key, list of value] pairs in an intermediate shuffle phase. A reduce function then takes these sorted pairs and aggregates them based on user defined criteria. The map and reduce functions have the capability of running independently on each key-value pair, extracting large amounts of inter-connected data (Shafer, Rixner, & Cox, 2010). This model can be customized based on the data analysis needs to summarize information from datasets as per user needs.

2.5 Hadoop in EV data mining

In recent times, plug-in electric vehicles (PEVs) have been gaining prominence, attributing to the increasing concerns about global warming and stringent emission regulations in the transportation sector (Amjad, Neelakrishnan, & Rudramoorthy, 2010). Also, a recent study predicts the world oil reserves to deplete by 2038 at current consumption rates (Ehsani, Gao, & Emadi, 2009). All these factors favor the use of PEVs as an alternative for future transportation and strongly advocate research efforts directed in this area. There is a need to study the operational characteristics of PEVs and analyze the usage trends to make them an efficient and trustworthy transportation alternative. This step toward promoting PEVs for commuting needs a thorough analysis of all its internal components.

The battery pack is the most important component and the primary source for powering a PEV (Amjad, Neelakrishnan, & Rudramoorthy, 2010). It gets its charge from plugging into electric outlets and powers the on-board electric systems in a PEV. It essentially provides high-density power capabilities to support the commuting conditions.

Various parameters like driving patterns, charging frequency, ambient temperature, etc. can affect the battery's capacity. It is found that a battery can only deliver about 70% of its rated capacity when working in colder temperatures (Bradley & Frank, 2009). It is essential to derive peak performance from the battery pack by providing facilitative ambient conditions.

Another important aspect in PEVs is the battery's cycle life (Amjad, Neelakrishnan, & Rudramoorthy, 2010). Users expect the PEV battery pack to last the lifetime of the vehicle. Increasing the size of the battery based on its application can help extend battery life. A larger battery discharges less in comparison to a smaller battery, which leads to a longer cycle life. But this will increase the overall cost and weight of battery pack. However, there are concerns related to higher initial costs, subsequent replacement costs, longer charging times, shorter commuting ranges and faster deterioration of battery capacity limiting the usage of PEVs (Chan, 2007). PEVs should be capable of traveling long distances continuously without frequent recharging and minimum external infrastructure requirements. Understanding operational patterns of PEV battery under real commuting conditions can be a good start to address this issue.

In this scenario, the scientific research community needs to analyze the entire PEV sensor data collected from various data loggers under real driving conditions. These data loggers tend to record data from sensors onto a single output stream. This stream contains randomly populated sensor data (i.e., data available at that given instant gets pushed into the logger stream). The data collected by different types of sensors from various components in a PEV is unstructured and non-relational in nature (i.e., Big Data).

This data has to be purged from sources, staged for analysis and stored for future research.

2.7 Chapter summary

This chapter provides a review of literature relevant to the area of research conducted in this project. It introduces the concept of Big Data and the various challenges it introduces in data-intensive scientific computing. It introduces various capabilities of NoSQL technologies and the significance of Hadoop as an alternative for conducting scientific computing in EV data analytics. Toward this end, it details the proposed research design for implementing the idea.

CHAPTER 3. FRAMEWORK AND METHODOLOGY

The main idea of the project was to design an efficient system that facilitates electric vehicle (EV) data analytics. The main requirements of performing analytics were identified and studied thoroughly. A study of NoSQL technologies that met those requirements was done and Hadoop was chosen. Then the repository for data analytics was set up and various types of analyses were performed to verify the system's efficiency.

This chapter delineates the methodology used for the current research and validates the model. It mainly focuses on the research question, the hypotheses and the data set used. Finally, the statistical tools used for analysis and comparisons of the data are discussed.

3.1 Research aim

The battery is considered to be the most essential component of an electric vehicle. It is responsible for providing energy to the mechanical drive train components and other onboard electrical systems. The battery state of charge (SoC) is an estimation of its available capacity and serves as an indicator of vehicle's capable range before next recharge. This SoC discharge pattern is influenced by various parameters like ambient temperature, voltage, current drawn, speed, etc. Accommodating all these parameters for

SoC estimation during real-time driving conditions can help in accurate information to ensure efficient operation (Cox & Perez-Kite, 2000). This research aims to provide an analytics system that is capable of integrating input from multiple sources.

Hence, the primary task in this study was to extract and analyze the SoC correlation with other parameters from a dataset and effectively predict SoC during real commuting conditions in EVs. The observation data had to be first analyzed to identify any patterns or relationships during EV charging cycles over a period of time. These patterns can be a good way of isolating situations where a battery performed better and instances that led to rapid discharge. These patterns have to be validated over a long period of time to eliminate any shortcomings and to strengthen the observed relationships. An accurate SoC prediction can result in optimized performance, extend battery life and mitigate any irreversible damage to the battery (Bhangu, Bentley, Stone, & Bingham, 2005).

Overtime, SoC is influenced by the battery's state of health (SoH), which is the capability of the battery to charge itself to its rated capacity. This analysis has to be extended to identify a way of estimating the battery's SoH at the given instant. One-way to accomplish this is to calculate of the bulk capacitance of the battery from cell capacity (Bhangu, Bentley, Stone, & Bingham, 2005). The SoH of a battery can determine its performance and help estimate its lifetime. EV manufacturers would be interested to predict SoH through precise monitoring techniques and improve their reliability.

3.2 MapReduce framework

Advanced battery-monitoring systems need to perform bulk reads of the sensor logger data collected over a period of time for accurate estimation of parameters. The analytic systems use this information to calculate the parameters that predict current state of battery. It will be helpful if this study showed that it is possible to optimize these reads and provides a mechanism for quick summarization of data.

The MapReduce framework can be a good help for making this job more convenient. It essentially maps all the values from the dataset and first arranges them into predefined clusters of information. Once we have the relevant data mapped into classified groups, it can be used in the reducer jobs to summarize the information using an algorithm. Also, the mapper and reducer jobs in Hadoop can be programmed to be capable of running in parallel over large amounts of data. The outputs of both the jobs can be stored into a separate dataset, which can be used to represent the analyzed information. The processed data can be archived after analysis, which will save the storage space in the repository.

3.3 Proposed research design

This research proposed using Hadoop as a platform for supporting EV data analytics. It found that the various features in Hadoop can greatly support analytics on PEV datasets. This study mainly concentrated on evaluating Hadoop as a data repository tool for storing PEV sensor data. Subsequently, this data can be mined to harvest information that can help in decision-making and support the usage of PEVs for commuting purposes.

This research developed an efficient data warehouse for mining data recorded by PEVs. This data can be modeled and analyzed for identifying patterns between parameters and the influence of driving conditions on them. In general, the data stored was a real time recording of driving and battery parameters of EVs during actual commuting conditions. Hadoop was capable of storing schema-less data and was flexible to accommodate any inflow of data in the future. Due to the schema-free nature of the stored data, it allowed for seamless inclusion of new data with no storage overhead or need for re-designing the schema. This eliminates the need for having a database administrator to manage any volatility in data.

The proposed data repository consisted of various sensor data related to driving and EV usage over a period of time. The logger data stored this information in a set of comma separated value (CSV) files recorded on a session-by-session basis.

The main benefits conceived from the implementation of this repository as stated by Bolly, Springer, & Dietz (ASEE 2014) are:

- Effective integration of data from heterogeneous sources for collaborative analysis and information retrieval.
- Statistical modeling and analysis of sensor data to identify hidden patterns, associations or relationships between driving parameters and their influence on battery pack.
- Using the location data from the data, researchers can identify factors that affect the strategic location of an EV charging infrastructure.

- An opportunity for researchers to test new business models using the available data in the warehouse.
- A unique opportunity to analyze the battery usage requirements and impact of various driving parameters in real world commuting conditions.
- Provide business and technology development opportunities for research partners as they are well positioned to anticipate and solve the technical and practical challenges that emerge with the use of EVs.

This repository was designed to be scalable to fit large amount of research data coming from this area of study. It is capable of supporting multiple writes into the database from various locations in an error-free way. The architecture is configured to provide faster random reads on the data through MapReduce. It was designed to support collaboration of researchers where each can connect to the warehouse and operate on it simultaneously. With appropriate configurations supporting data replication and managing of nodes in the cluster from a master node, it is ensured that the data is always accessible.

3.4 Analysis

The hypothesis for this research can be stated as:

H_0: There is no improvement in the electric vehicle data retrieval speeds from a Hadoop data repository by addition of more computational nodes in the cluster.

H_a: There is improvement in the electric vehicle data retrieval speeds from a Hadoop data repository by addition of more computational nodes in the cluster.

While the null hypothesis states that there is no improvement in EV data retrieval performance in a Hadoop cluster by adding more nodes to it, the alternate hypothesis states that there is an improved data query performance for reading and retrieving EV data in a Hadoop cluster by increasing the number of computational nodes. To perform hypothesis testing, the time taken for information retrieval was used as a measure for calculating the test statistic. A two-factor ANOVA was used to calculate the test statistic for this hypothesis testing as this method is useful for exploratory statistical analysis pertaining to making decisions using observations from experimental data and can explain the statistical significance of data when comparing means of different groups. In this case, each group was a cluster with varying number of nodes. The query time was the average time from running the query five times in each case. For this analysis, observation data from five runs were recorded for each scenario. It was confined to five runs as there was little change in run-time value observed between each run (i.e., low standard deviation for the five observations). Also, the data collected for five runs in each case was adequate enough to reject the null and explain the observed variation.

3.5 Experimental setup

For the experiment, a multi-node Hadoop cluster was configured and used to store the electric vehicle dataset and perform data reads. This setup was configured over a cluster of multiple computing nodes, each with an instance of Hadoop installed on them. The complete setup consisted of 4 physical server machines and 16 VMs distributed among them. The following was the configuration in each scenario:

- 1-node cluster – 1 VM on one machine.

- 2-node cluster – 2 VMs on 2 machines.
- 4-node cluster – 4 VMs on 4 machines.
- 8-node cluster – 8 VMs on 4 machines (2 VMs on each machine).
- 16-node cluster – 16 VMs on 4 machines (4 VMs on each machine).
- 20-node cluster - 20 VMs on 4 machines (5 VMs on each machine).

The repository used HDFS for storing the datasets from the EV sensor loggers. There were mapper jobs that helped in extracting information from the loggers and storing in the HDFS. Sets of mapper and reducer programs to perform reads on the datasets were used. These jobs were programmable based on data summarization needs. These results were then stored as new set of dataset in HDFS.

CHAPTER 4. DATA ANALYSIS

This chapter covers the data analysis done by running MapReduce jobs on the electric vehicle datasets. It essentially encompasses the results of data analysis based on the described implementation and explains the collected statistics. The cluster configured for performing this analysis can be further modified achieving any specific characteristics. The described analytical methods can be extended further to perform other kinds of analysis desired by the researcher.

4.1 Configuring the Hadoop cluster

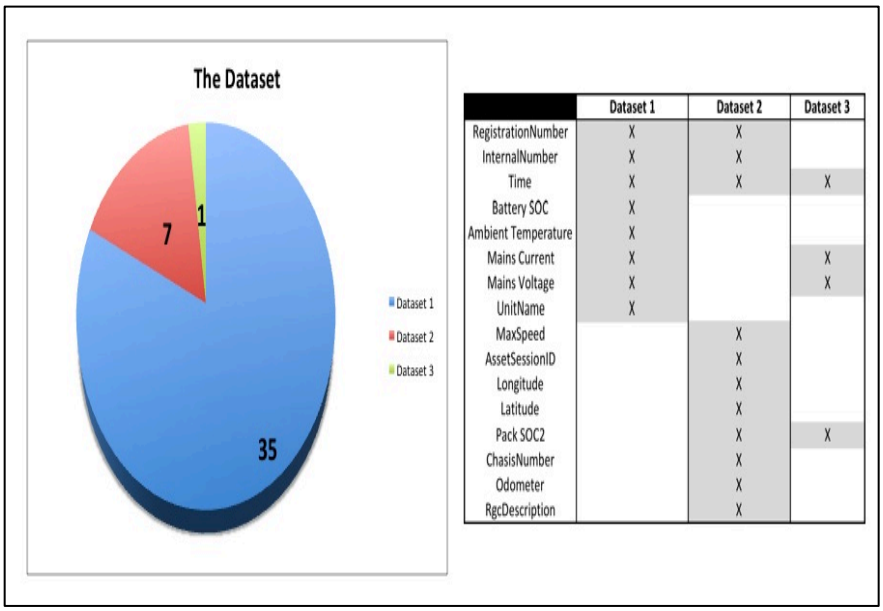
The complete experimental setup was built over a cluster of physical servers configured over a network using VMware ESXi virtualization technology and was accessible through a VPN connection. Every server machine contained 12 CPU cores, each capable of processing at clock speeds of 2.66 GHz. Each server had a storage capacity of holding 3.63 TB of data and the capability of hosting multiple guest VMs.

For this analysis, a Red Hat Enterprise Linux (RHEL) 6.0 server distribution was installed on each virtual machine (VM). An instance of Hadoop was installed on every VM and configured to communicate over keyless SSH protocol. Hadoop-1.2.1 release was used for setting up the data cluster. For the Hadoop cluster nodes to communicate with each other, a keyless SSH capability was provided to all the VMs using the `ssh-keygen` utility.

After installing Hadoop on all VMs, it was configured using the files provided in `conf` folder under the Hadoop installation directory. This configuration is copied to all the subordinate nodes of the cluster. After successfully formatting the NameNode, the cluster is started using the scripts in `bin` folder.

4.2 Analytics on the EV dataset

First, all the files in the dataset were loaded into HDFS for performing data reads. The preliminary analysis on the dataset done by Bolly, Springer, & Dietz (ASEE 2014) has identified the important attributes recorded in the dataset. Figure 4.1 shows the overview of attributes recorded in the dataset. These attributes were analyzed to identify various characteristics of data recorded in this dataset.



	Dataset 1	Dataset 2	Dataset 3
RegistrationNumber	X	X	
InternalNumber	X	X	
Time	X	X	X
Battery SOC	X		
Ambient Temperature	X		
Mains Current	X		X
Mains Voltage	X		X
UnitName	X		
MaxSpeed		X	
AssetSessionID		X	
Longitude		X	
Latitude		X	
Pack SOC2		X	X
ChasisNumber		X	
Odometer		X	
RgcDescription		X	

Figure 4.1. Overview of the Project Plug-IN dataset (Bolly, Springer, & Dietz, ASEE 2014)

Various kinds of analyses were possible with the flexible query mechanisms available in Hadoop. The following were the analytics performed on the data by Bolly, Springer, & Dietz (ASE 2014):

- Grouping the data based on a certain threshold value to perform targeted analysis on relevant subset of data. Figure 4.2 shows the grouping of EVs based on their average battery SoC value recorded over a period of time.

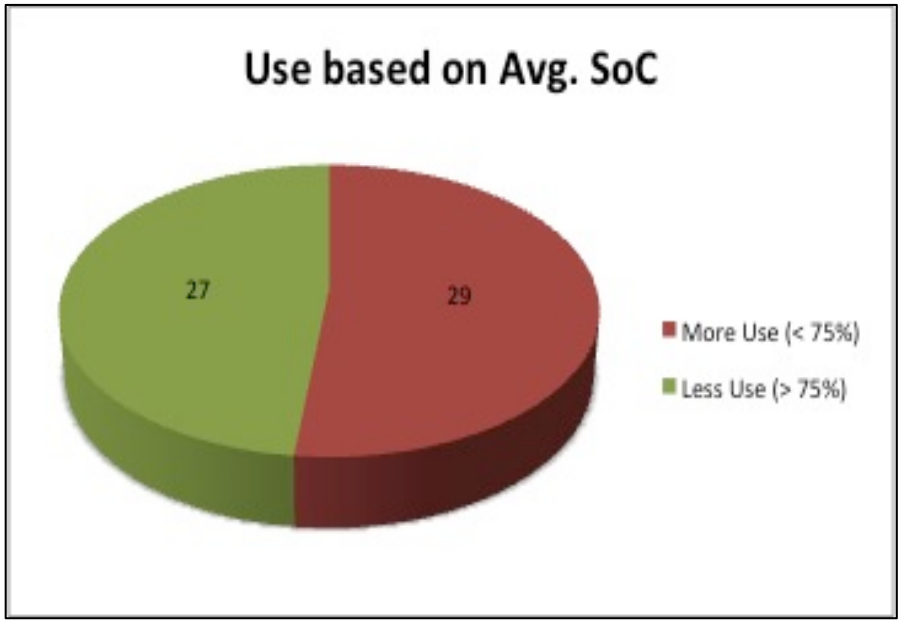


Figure 4.2. Grouping EVs based on their battery usage (Bolly, Springer, & Dietz, ASE 2014)

- Observe the effect of one parameter on the behavior of another and identify a relationship between them. Figure 4.3 shows the variation of battery SoC and Ambient temperature of an EV over a timeline. Here, it can be seen from the

trend line that the SoC values are higher at lower ambient temperatures. An in-depth analysis can help predict the SoC discharge rates better at a given temperature.

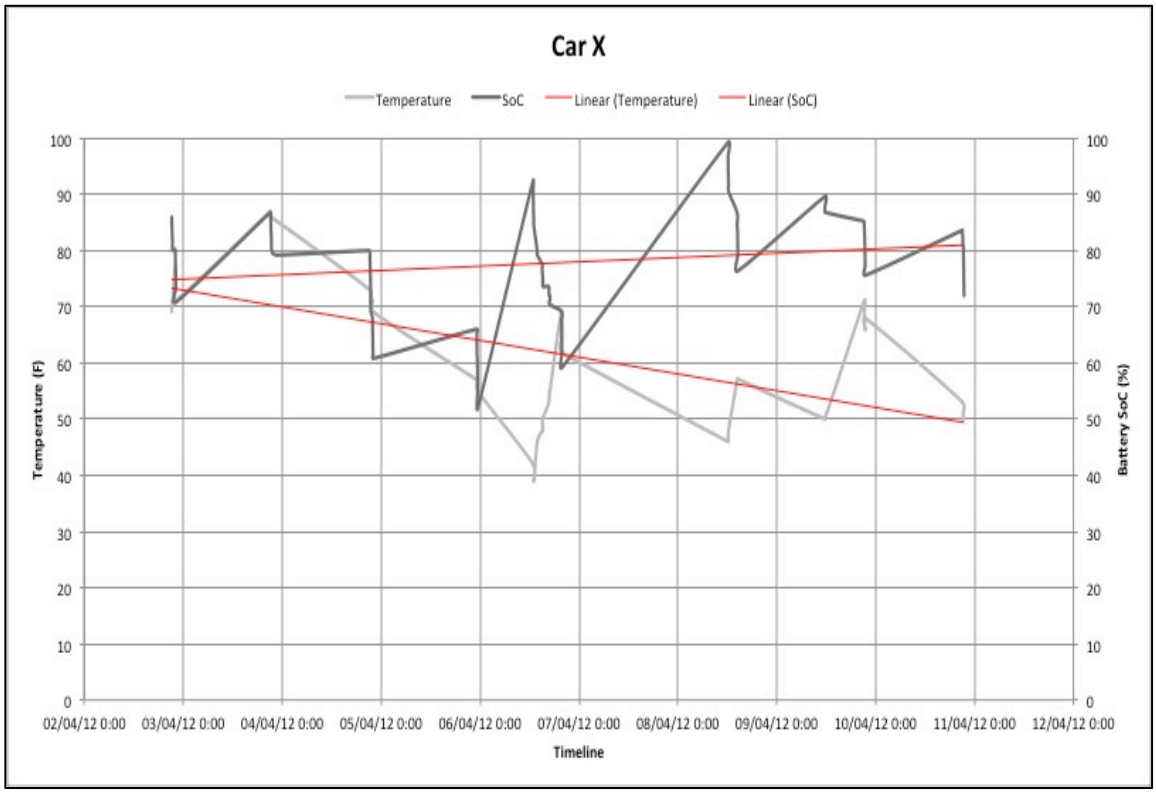


Figure 4.3. Plotting SoC and Ambient Temperature for an EV (Bolly, Springer, & Dietz, ASE 2014)

- Modeling the trip characteristics is another way of understanding the actual usage of EVs by drivers. Figure 4.4 plots the SoC trend for a period of two weeks for three similar EVs from the dataset to observe their typical usage. It can be observed that these EVs mostly were used for weekday commutation

and showed idle activity on weekends. They were charged every evening to almost full charge for next day use. This can be attributed to the drivers using the EVs for commuting to workplace and back home. This modeling of trip for drivers in a particular group can help understand the real world use of EVs and commutation patterns.

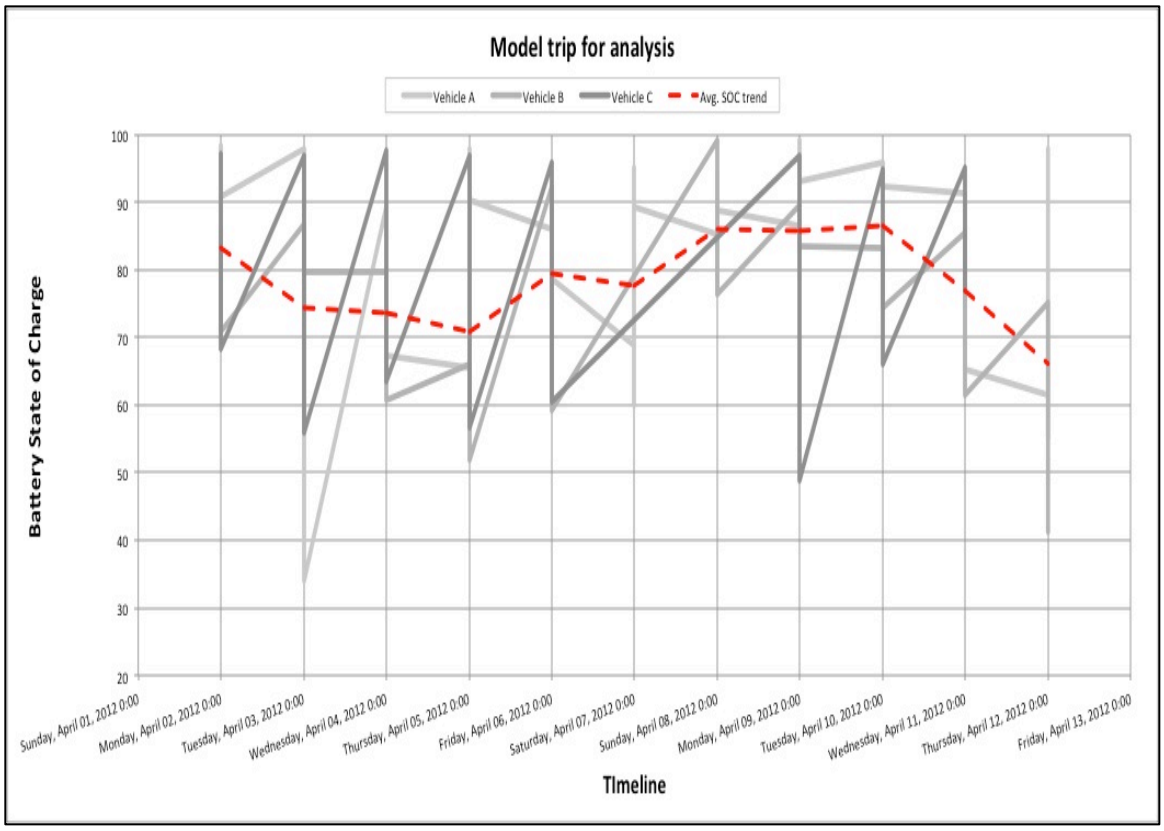


Figure 4.4. Model trip of three EVs in the dataset (Bolly, Springer, & Dietz, ASE 2014)

- The GPS coordinates of EVs in the dataset can provide information on the locations where the data was recorded and the driving patterns in a particular

area. A figure 4.5 show that the EV data recorded in this dataset is majorly from the state of Indiana.

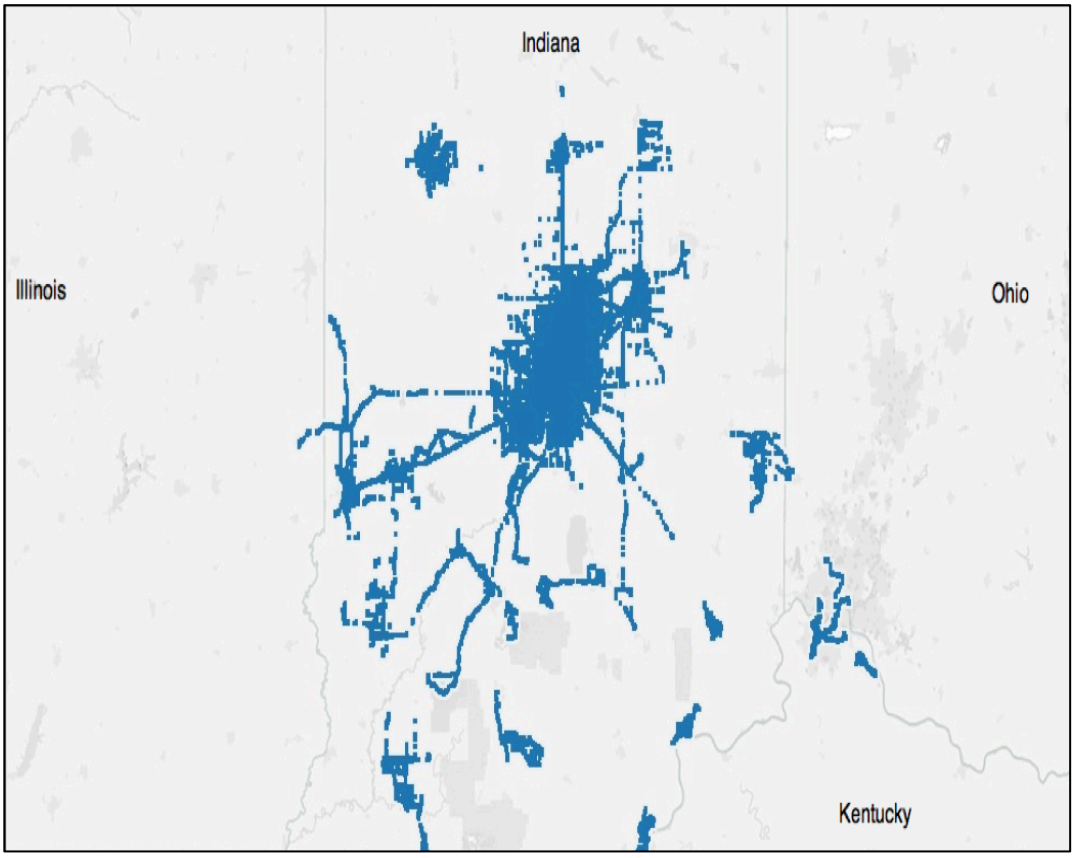


Figure 4.5. The overview of GPS locations in the dataset (Bolly, Springer, & Dietz, ASE 2014)

- This information on GPS values can help provide information of the EV traffic on particular roads in a given county. Figure 4.6 plots the GPS data from Marion County, Indiana. It provides an overview of the roads used by the EVs in this region. This can help in decision-making process of installing a charging infrastructure for EVs at places recording high EV traffic.

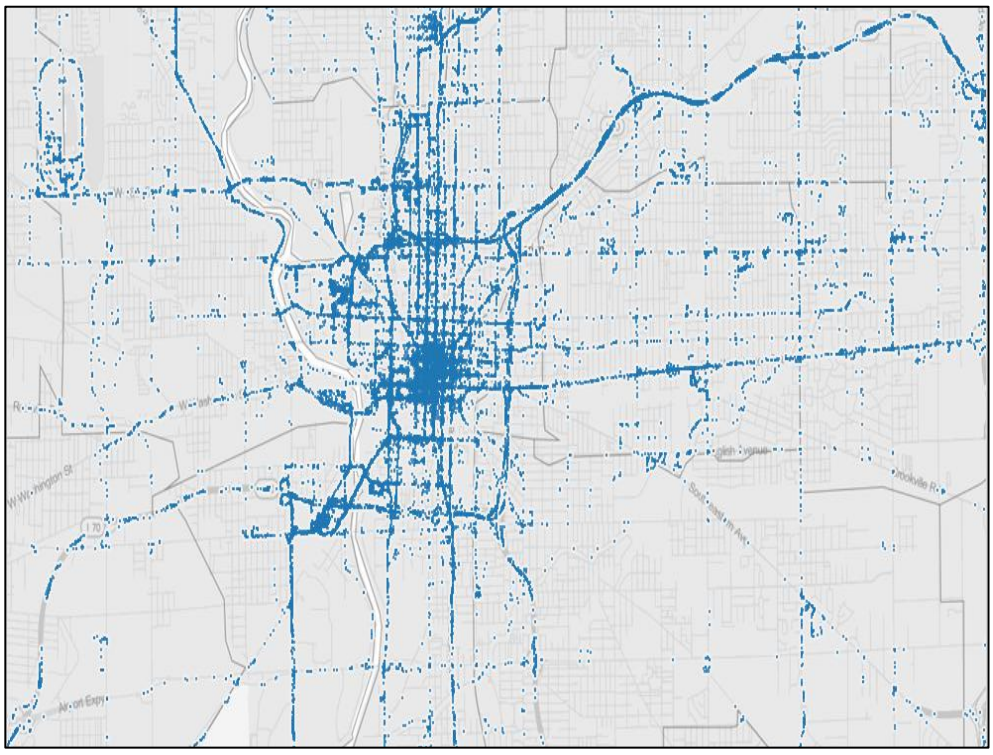


Figure 4.6. EV Traffic from Marion County, Indiana (Bolly, Springer, & Dietz, ASE 2014)

4.3 MapReduce queries for Hypothesis testing

As discussed before, MapReduce is the programming paradigm used to perform data queries in this analysis. For hypothesis testing, three different types of queries were identified to test their data read performance from the cluster. They are as listed below:

- Query 1: Calculate the Average battery SoC for an EV – This query is the most basic type of data read done using MapReduce. It runs over the EV dataset and summarizes the average battery SoC recorded by sensors for each EV over all its drive cycles. This query can be broken into two phases,

mapping the SoC values and reducing these values to summarize to an average as shown in figure 4.7.

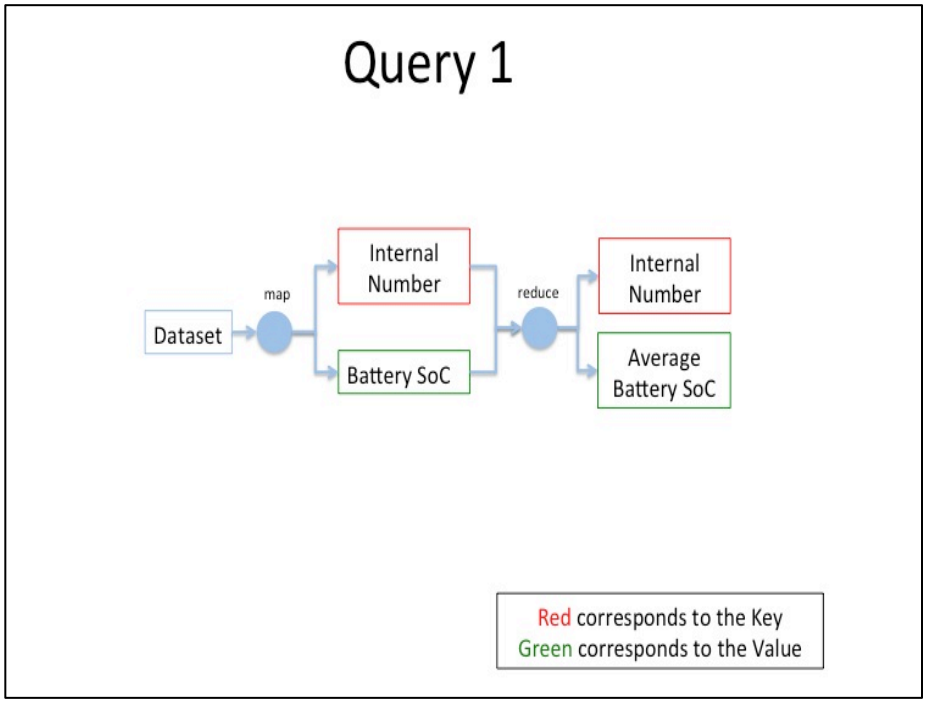


Figure 4.7. MapReduce steps for Query 1

- Query 2: Calculate the Pack Resistance for an EV – Pack resistance is the ratio of the Mains Voltage and Mains Current recorded at the same battery SoC. For this analysis, the voltage, current and SoC values are first mapped from the data for each EV. This mapped data is voltage and current values are combined for the same SoC value. These values are reduced to a ratio corresponding to the Pack resistance of the battery as seen in figure 4.8.

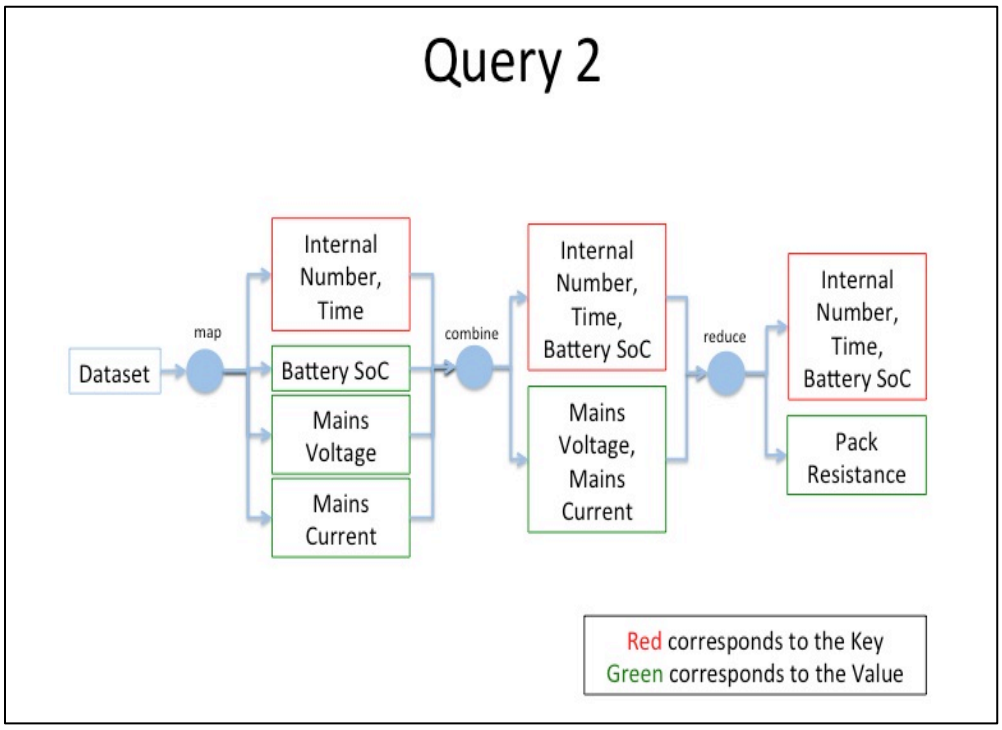


Figure 4.8. MapReduce steps for Query 2

- Query 3: Get the weather station data for an EV – This is a query on heterogeneous data where the weather dataset and EV dataset are processed for analysis. First, all the GPS locations of an EV and weather station data are mapped. Next, the data is reduced to show the weather data for each EV at a particular instant. Figure 4.9 gives an overview on how the data is processed in this query.

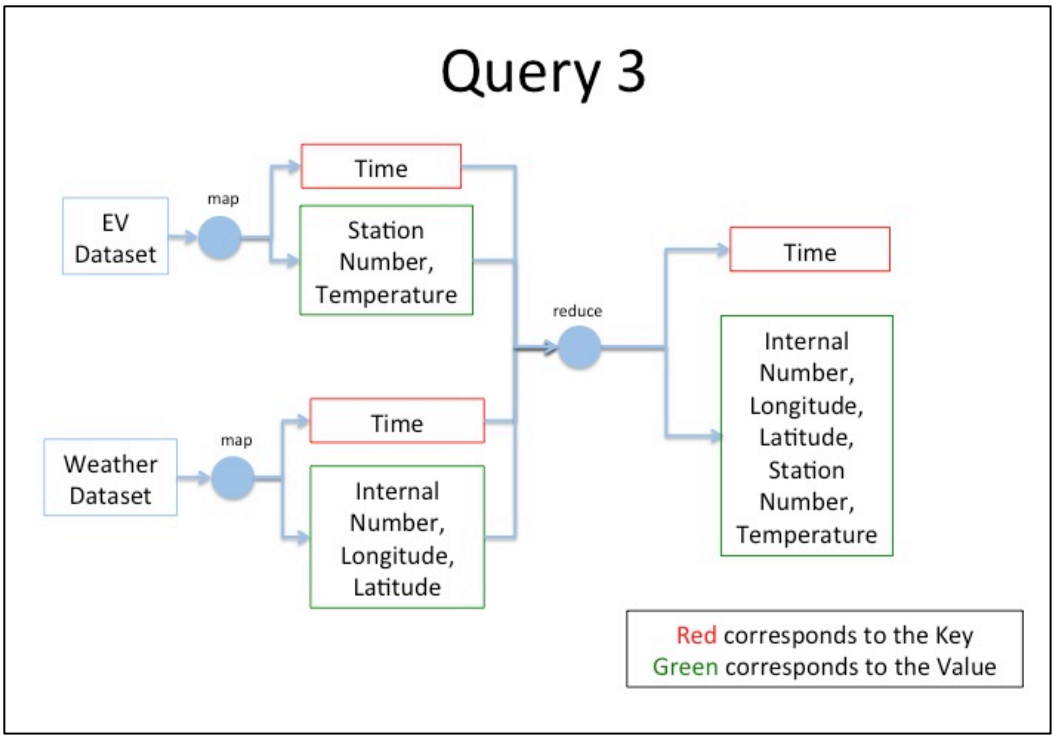


Figure 4.9. MapReduce steps for Query 3

4.4 Data Collection for MR queries

The data collection for the run times of each of these queries was done varying the input data size and the number of nodes in the Hadoop cluster. The data was imported into HDFS and distributed across the cluster before running the MR query. This experiment was done to test the influence of scaling the cluster to multiple nodes on EV data read performance.

The steps followed for the data collection are listed below:

1. First the number of nodes in the cluster was configured.
2. The cluster was started using appropriate scripts in `hadoop/bin` folder.

3. Once the cluster is up and running, each of the MR query was run for five times on datasets of different sizes (folders inside HDFS).
4. The run time in each case was recorded for statistical measure.
5. After recording data for all the three MR queries, more number of nodes were added to the cluster the process was continued from step 2.
6. For this analysis, the cluster was scaled till 16 nodes as in most cases the cluster reached a threshold value after which it didn't show any improvement in run time performance for the queries.

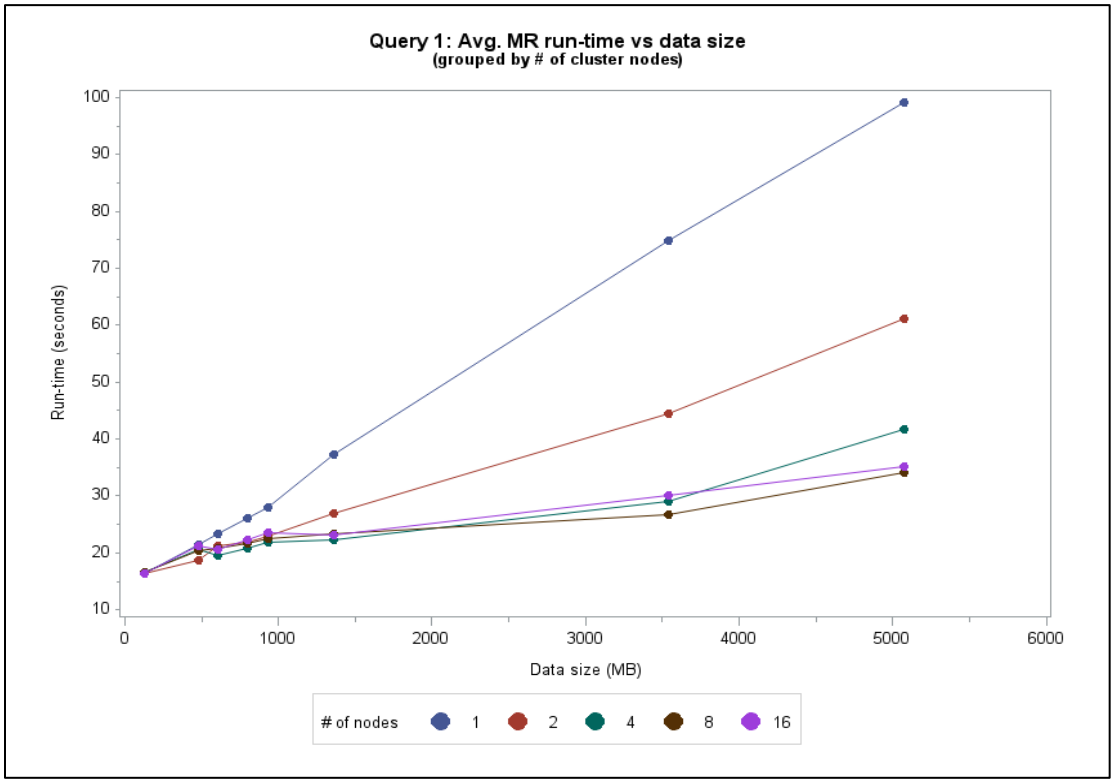


Figure 4.10. Query 1: Run time vs Input data size grouped by number of nodes

Figure 4.10 shows the observed average run times during five iterations of query 1 on the dataset with varying input sizes while increasing the number of nodes in the cluster. The significant observations from the plot are:

- The average MR run-time increases with increasing data size.
- This rate of increase in run-times with data size is less in case of cluster with more nodes.
- For each dataset, there is a threshold value of number of nodes beyond which the run-time performance decreases.
- For larger dataset, the threshold value for number of nodes is higher, which can mean larger datasets need a cluster with more nodes for better data read performance.

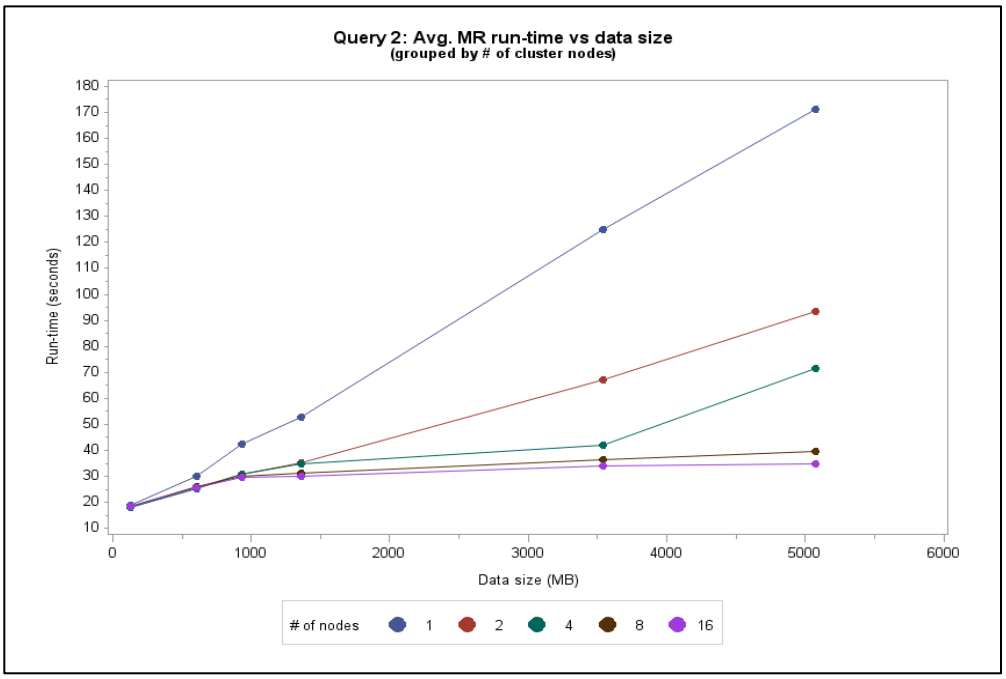


Figure 4.11. Query 2: Run time vs Input data size grouped by number of nodes

Figure 4.11 represents the average MR job run times for five iterations of query 2 on the EV dataset while changing the data input size and number of nodes. This query has a combine phase along with map and reduce phases.

The significant observations from this data are:

- Single node cluster has poor read performance than a multi-node cluster.
- Cluster with more number of nodes has close run-time value range while reading data of different sizes. This observation supports the deployment of a multi-node cluster to help achieve consistent data read performance.
- Using Combiners can help improve the read performances of MR jobs. In action, combiners are pseudo reducers that crunch the amount of data shuffled and fed to reducers across the network (White, 2012). This can result in improvement of the total read time.

Figure 4.12 depicts the average run times during five runs of query 3 over a heterogeneous datasets for different input sizes while increasing the number of nodes. This query processes information from both EV and Weather datasets.

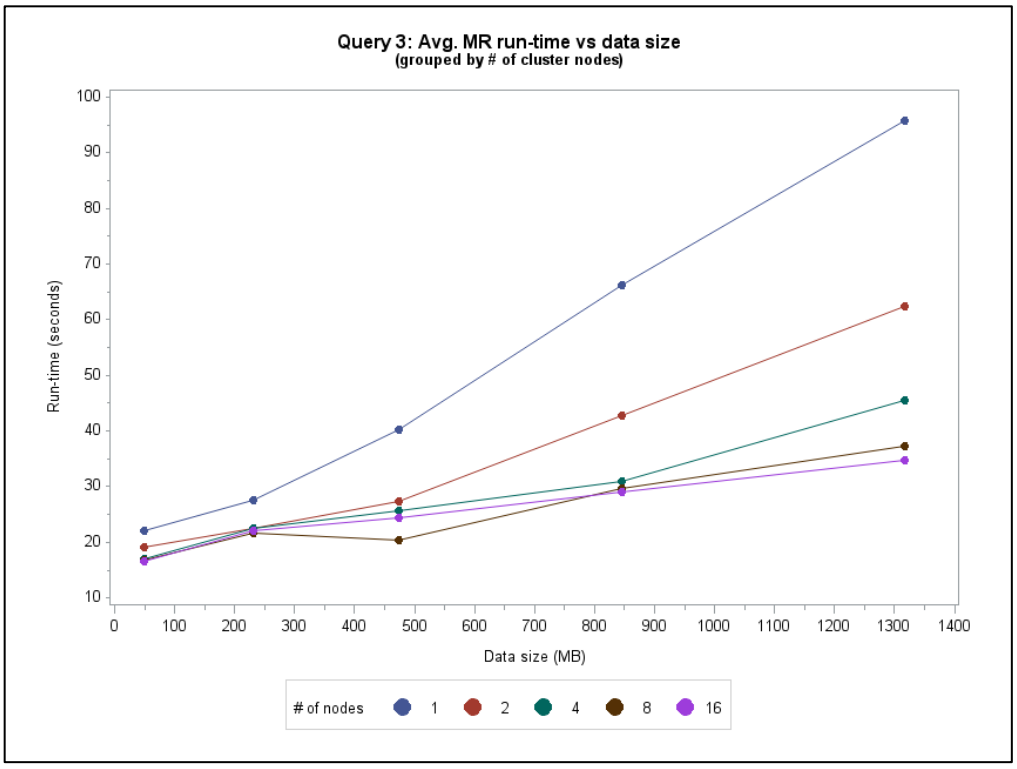


Figure 4.12. Query 3: Run time vs Input data size grouped by number of nodes

The following are the general observations from all the plots:

- The average query run time increases with increase in input size. This can be attributed to the more number of lines to process in case of larger datasets. As there are more lines of data, it takes longer to map the required data.
- For single node cluster, the average run time for a MR query increases almost linearly with input data size.
- As the number of nodes increase, the average run time for the query tends to reduce until a threshold value. This can be attributed to the slicing and distribution of data as blocks in HDFS. These blocks are distributed across all

the nodes in the cluster. Each mapper job can run independently on the nearest data block and reduce the overall mapping time, thus explaining the reduced run times.

- In case of query 1 where the input size is 1361 MB, the best run time was recorded when the cluster had 4 nodes. After 4 nodes, increase in number of nodes increased the run time of the query. Hence, it can be concluded that 4 nodes is the threshold value for this data size in case of query 1. This can be attributed to the shuffle phase in an MR job where all the mapped data moves across the cluster nodes for sorting. As the data size was small, this data movement proved costly in case of more number of nodes.
- For larger datasets, the improvement in run times is more when the cluster is scaled to more number of nodes (until a threshold value). Amdahl's law explains this upper limit on the number of nodes. It states that the sequential part of a code eventually limits the amount of improvement that can be achieved by scaling the system (Amdahl, 2007).
- Overall, scaling to multiple nodes not only helps store more data but also improves its retrieval speeds for EV data when stored in Hadoop clusters.

4.5 Two-way ANOVA and Hypothesis testing

To validate the statistical significance of the data used for analysis, a two-way ANOVA test was carried out on the collected data. Two-way ANOVA (Analysis of Variation) can help examine the influence of two-independent categorical variables (input size and number of cluster nodes) over a dependent variable (MR run time). It can also

help see any significant interaction effect between the independent variables. This statistical analysis was done for a confidence level of 95%. This test can help understand the significance of the data and use it to comment about the null hypothesis (H_0) for this research.

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	39	52737.41811	1352.24149	1845.23	<.0001
Error	160	117.25283	0.73283		
Corrected Total	199	52854.67094			

R-Square	Coeff Var	Root MSE	time2 Mean
0.997782	3.027647	0.856055	28.27460

Source	DF	Type I SS	Mean Square	F Value	Pr > F
size	7	28658.70475	4094.10068	5586.70	<.0001
nodes	4	8790.47292	2197.61823	2998.81	<.0001
size*nodes	28	15288.24044	546.00859	745.07	<.0001

Figure 4.13. Two-way ANOVA for Query 1

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	29	164540.1008	5673.7966	4922.65	<.0001
Error	120	138.3108	1.1526		
Corrected Total	149	164678.4116			

R-Square	Coeff Var	Root MSE	time2 Mean
0.999160	2.501603	1.073587	42.91598

Source	DF	Type I SS	Mean Square	F Value	Pr > F
size	5	71797.13550	14359.42710	12458.4	<.0001
nodes	4	39875.33399	9968.83350	8649.07	<.0001
size*nodes	20	52867.63134	2643.38157	2293.43	<.0001

Figure 4.14. Two-way ANOVA for Query 2

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	24	40992.28364	1708.01182	4135.67	<.0001
Error	100	41.29949	0.41299		
Corrected Total	124	41033.58313			

R-Square	Coeff Var	Root MSE	time2 Mean
0.998994	1.959333	0.642647	32.79926

Source	DF	Type I SS	Mean Square	F Value	Pr > F
size	4	21914.28146	5478.57037	13265.5	<.0001
nodes	4	11186.98657	2796.74664	6771.87	<.0001
size*nodes	16	7891.01562	493.18848	1194.18	<.0001

Figure 4.15. Two-way ANOVA for Query 3

The above ANOVA results were generated from the collected data using Statistical Analysis System (SAS) application. The tables indicate the various parameters measured in ANOVA. The important is the P-value, which can be used to comment on the null hypothesis. For a 95% confidence level, the P-value has to be less than 0.05 to reject the H_0. For all the above queries, the P-value in each case is far less than 0.05. Hence, at 95% confidence level, this data provides enough evidence to reject H_0 i.e., reject the null hypothesis, which states that there is no improvement in query run time by increasing the number of computational nodes. So, this experiment proves that for EV datasets, improvements in read performance can be achieved by scaling the cluster to multiple computational nodes.

The R-square value (coefficient of determination) was very high for each case. It is the ratio of the explained variance to total variance. It also is a measure of the closeness of the data to the fitted regression line. This implies that the model explains most of the response variable variation.

It is also observed that F-value was high in each case. F-value is the ratio of variance between groups to variance within group. High F-value implies that the variation between group means is significant and can't occur by chance. That gives sufficient reason to reject null hypothesis. Also, this supports the argument that five runs (in each case) are sufficient for this hypothesis testing as the variance within group is less which is evident from the low standard deviation values.

4.6 Impact of cluster balancing on performance

When scaling the cluster to multiple nodes, balancing the data across all the nodes is important for performance. In Hadoop, there are three essential types of data locality for the blocks namely data-local, rack-local and remote (White, 2012). Data-local is when the blocks processed are in the same DataNode in the cluster. Mapping data-local blocks is usually faster. Rack-local is when the blocks read are from different datanodes, which are on the same rack. Rack-local maps are usually slower than data-local maps. Remote data is when the blocks read are not in the same node or the same rack.

When the data blocks in a cluster are not fully balanced, there will be more number of rack-local map reads than data-local maps (White, 2012). This can lead to increase in run times for an MR query. To validate this, Query 1 was chosen and was run on the cluster with different balancing threshold values varying the number of nodes.

Figure 4.15 illustrates the improvements in query run times when the cluster is more balanced.

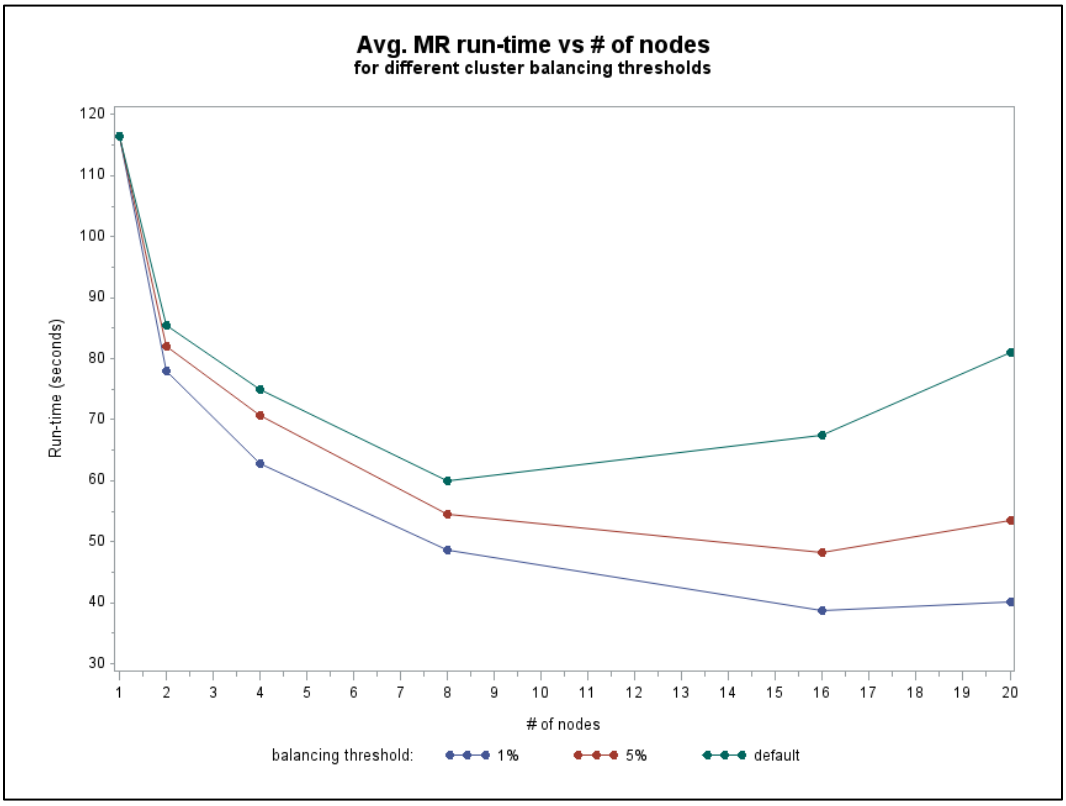


Figure 4.16. Run time vs number of nodes for different cluster balancing thresholds

Figure 4.16 is plotted query 1's average run-times during five iterations while changing the number of cluster nodes for three balancing thresholds namely, unbalanced, 5% and 1%. The lesser the threshold the more balanced is data in the cluster. In each case, the cluster was balanced using Hadoop Balancer utility for the three threshold values.

The following are the observations from analyzing the balancing data in figure 4.16:

- Irrespective of the number of cluster nodes, more balanced cluster performed better data reads.
- By balancing the cluster more, scaling of the cluster nodes showed performance improvement.
- The improvement in read times from balancing are more significant in case of larger cluster.
- This provides enough evidence that a less balanced cluster has slower read times than a more balanced one.
- Overall, after achieving significant improvement through scaling, more enhancements in performance can be achieved by properly balancing the data blocks in the cluster, which maximizes the data-local map reads.

4.7 Setting the right Input split size for data

MapReduce is designed to perform better with large files than small files (White, 2012). Tuning the number of mapper and reducer tasks is another way of improving your query performance. Here, setting the map input split size is a way of fine-tuning the MR query. The properties `mapred.max.split.size` and `mapred.min.split.size` control the data input split size for maps. If the data block size lies between the max and min split sizes, a single block is sent to each mapper (White, 2012). When an input file is large, this input splits specifies the number of maps that run on the data. For each split, a map job is run to process that information. Reducing the number of maps can avoid unnecessary maps that are running and help improve the query performance. It is

recommended that an optimum input split size be set in MR code when reading from a large file based on the HDFS block size for that file.

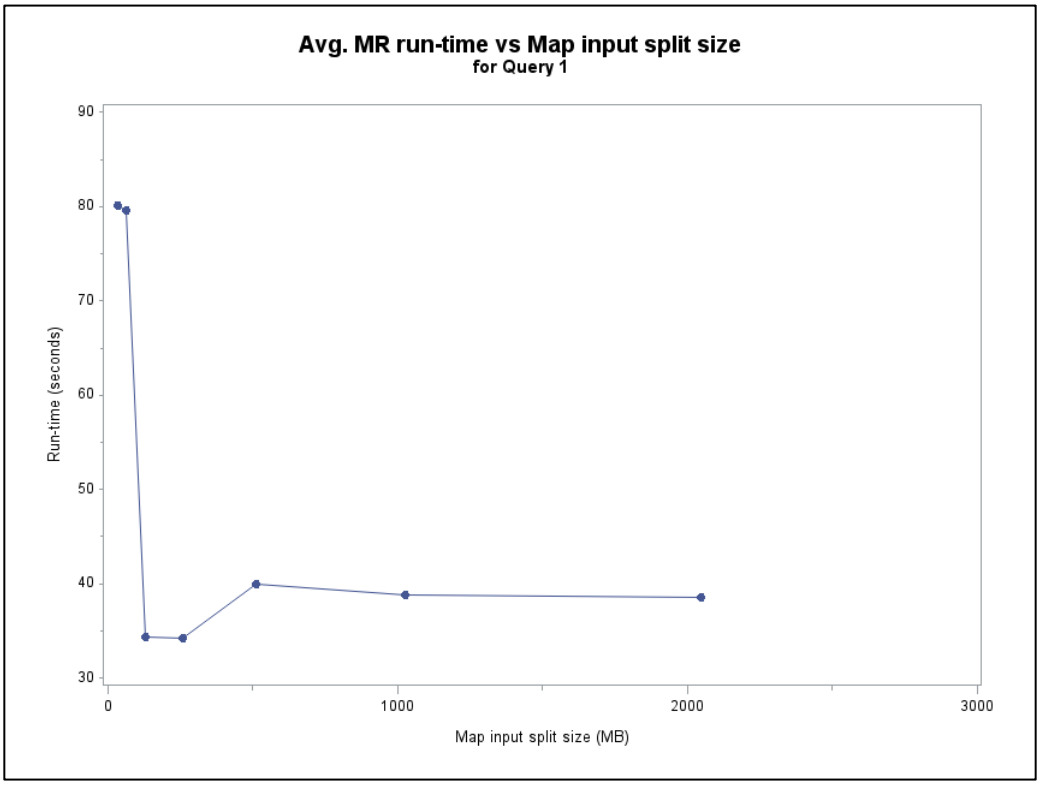


Figure 4.17. Query 1: Run time vs Map input split size

Figure 4.17 plots the average run time of query 1 while changing the input map split size in a 20-node cluster balanced at a threshold value of 1% for about 5 GB input data. In this case, the MR job for query 1 took the least time at map input split size of 256 MB. This is another way of achieving better performance from the MR job after balancing the cluster to an optimum.

CHAPTER 5. RESEARCH CONCLUSIONS AND FUTURE WORK

This chapter summarizes key conclusions of this research study based on the obtained results from experimentation. These results provide an effective argument for supporting the work and comment upon the effectiveness of this system design for delivering EV data analytics. Toward this end, it also discusses the scope of extending this research and help implement a better performing system for EV data analytics.

5.1 Effect of using multiple machines

During the data collection for hypothesis testing using MR queries on the data, 4 physical machines hosted the 4-node, 8-node and 16-node clusters. It was observed (as shown in figures 4.10, 4.11, and 4.12) that the differences in average MR run time between the 4-node, 8-node and 16-node clusters were minimal compared to the improvement observed as the configuration shifted from a 2-node to a 4-node cluster. Hence, to understand the effect of using VMs against using physical machines, another analysis was done with 16 VMs with the following configurations:

- 8 VMs each on 2 machines.
- 4 VMs each on 4 machines.
- 2 VMs each on 8 machines.
- 1 VM each on 16 machines.

Query 1 was chosen for this analysis to run on a 5 GB dataset. The average run time for ten trial runs was recorded in each scenario. This data was plotted on a graph to observe the effect of using more machines for multiple VMs.

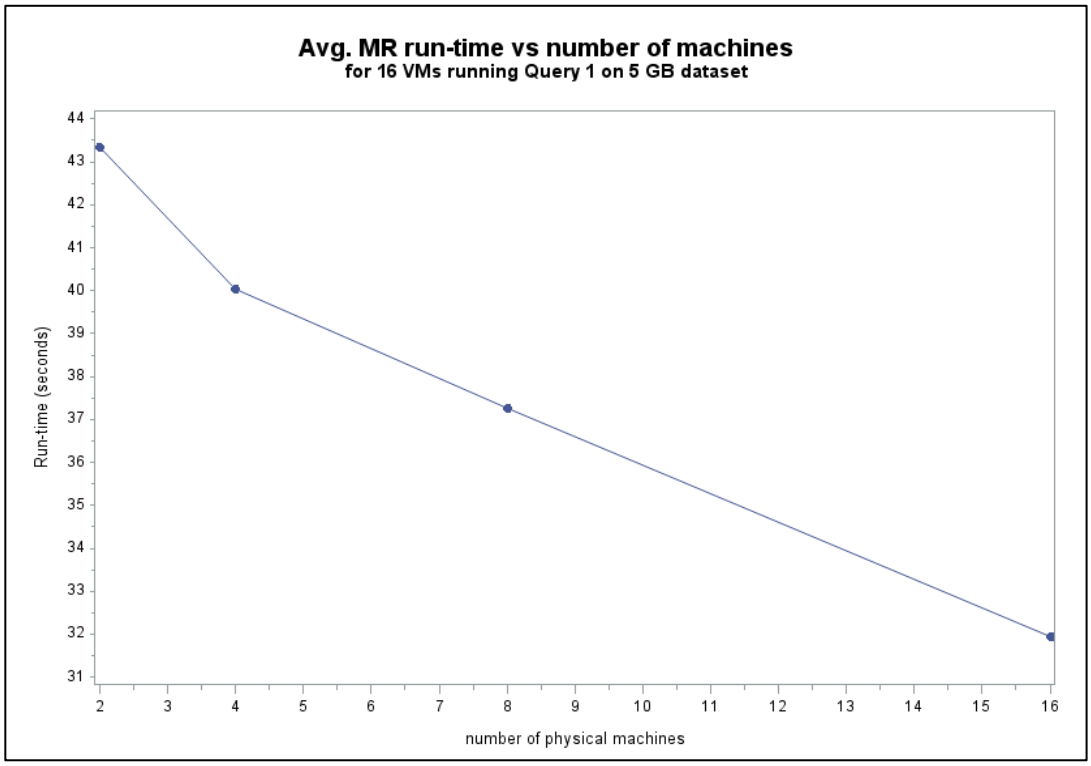


Figure 5.1. Query 1: Run times vs number of physical machines

It was observed from figure 5.1 that when there were more physical machines hosting 16 VMs, the average run time for the MR query showed improvement. This improvement can help us understand the benefits of using more physical machines to host a cluster of VMs.

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	630.6830481	210.2276827	204.57	<.0001
Error	32	32.8857482	1.0276796		
Corrected Total	35	663.5687963			

R-Square	Coeff Var	Root MSE	time2 Mean
0.950441	2.657202	1.013745	38.15086

Source	DF	Type I SS	Mean Square	F Value	Pr > F
machines	3	630.6830481	210.2276827	204.57	<.0001

Figure 5.2. ANOVA of average run times using multiple physical machines

The statistical analysis was done for the above data using ANOVA to determine the significance of the observed improvement. Figure 5.2 contains the results of an ANOVA test for the collected MR run time data for query 1. For a confidence level of 95%, the P-value for this data was very small, and this suggests that the change in average run times is not occurring by chance. Also, the high R-square value suggests that this data explains most of the variation From figure 5.2, it was observed that the data provided enough evidence for suggesting improvement in average run times when more physical machines were used to host the 16 VMs. Hence, the data can be used to conclude that a cluster spread over more physical machines is more efficient in performing data reads than the cluster hosted by fewer physical machines.

5.2 Significant conclusions

The important results achieved from this system implementation are listed below:

- Hadoop is a useful tool for building data analytic systems for EV datasets.
- Scaling a cluster to multiple nodes is handled well in Hadoop and can accommodate any incoming data in the future.
- For large datasets, the effect of scaling the cluster is significantly helpful in improving data reads.
- There can be significant improvements in performance of data reads by scaling the cluster to more nodes.
- There is a threshold value for number of nodes in the cluster beyond which it is not possible to achieve performance improvement. This is because the serial part of the MR code limits the enhancement achieved through multiple nodes working in parallel. In this case, for the available data sizes, a 16-node cluster demonstrates superior read times.
- Balancing the cluster can help achieve superior MR read performance.
- Based on the size of dataset, setting a right map input split size can help reduce the average query run time after optimum cluster balancing.
- A cluster distributed over more physical machines is more efficient than the cluster spread over fewer physical machines.
- HDFS and MR programming paradigm can provide an effective data repository system that meets the requirements for storing and reading large engineering datasets.
- Tuning the mapper and reducer tasks appropriately can help achieve an optimum data query performance.

Hadoop systems provide multiple ways of configuring parameters in HDFS data storage and tuning options in MapReduce framework. These values can be modified to tune the cluster performance based on the user requirements and can be optimized for the available resources. It provides various configuration settings that are programmable in your MR code and are applied dynamically to that MR job.

5.3 Future work

This research pertains to the field of Electric vehicle data analytics using Hadoop and related technologies. In this study, experimental data was collected to provide enough evidence to support the use of multi-node cluster for EV data informatics. It proposes scaling the cluster as a way of handling large datasets and enhancing data read performance. It studied the effect of cluster balancing on MR data read performance. It was observed that the more balanced the cluster, the better is the average run time of the MR job. Also, this study suggests the fine-tuning of map input split size for an MR job to achieve better read times. This research can be further extended to evaluate the effect of other tuning parameters on the cluster performance.

The source code for performing MR queries is written in Java using the basic Hadoop APIs. Other programming languages supported by MapReduce can also be evaluated for efficient performance. Some parts of mapping and reducing code can be parallelized to achieve better improvements from scaling the cluster. This reduces the bottleneck in MR reads due to sequential nature of the code.

The following are some of the configuration parameters in Hadoop that can help tune the system for optimum performance:

- MapReduce works better when there are less number of files to be processed. Hence, by effectively merging similar files into a single file can improve the average runtime of MR query on the dataset.
- Increase the in-memory buffer size for sorting mapped data using `io.sort.mb` property. This can reduce the amount of spilled data from the buffer.
- Compression the map output before writing it onto the disk can save data space and write time. This can be enabled by setting the property `mapred.compress.map.output` to true.
- There are various phases in a MapReduce job and proper tuning of their controlling properties can help improve read performance from the cluster.
- The HDFS block size can be tuned based on the dataset file size using the property `dfs.block.size`. This controls the number of data blocks of the dataset in the cluster. Setting the right block size for a file can help achieve a more balanced cluster and reduce number of unnecessary maps running on the data.

The field of EV research can greatly benefit from the use of this system. Going ahead, this system can be used for engineering study related to EV development and deployment by researchers to address or identify any potential issues in this area. Patterns can be identified between parameters that govern EV performance.

There can be significant benefits from evaluating and implementing the suggested recommendations for this system. Any in-depth analysis of the various configuration

parameters and tuning them to suit the underlying infrastructure and nature of data can help achieve better performing systems for EV data analytics. In overall, there is immense scope for this study to be extended to multiple directions based on the researcher's area of interest.

LIST OF REFERENCES

LIST OF REFERENCES

Amdahl, G. M. (2007). Computer architecture and Amdahl's law. *Solid-State Circuits Society Newsletter, IEEE, 12(3)*, 4-9.

Amjad, S., Neelakrishnan, S., & Rudramoorthy, R. (2010). Review of design considerations and technological challenges for successful development and deployment of plug-in hybrid electric vehicles. In *Renewable and Sustainable Energy Reviews*, *14*(3), 1104-1110.

Attebury, G., Baranovski, A., Bloom, K., Bockelman, B., Kcira, D., Letts, J., Levshina, T., Lundestedt, C., Martin, T., Maier, W., Pi, H., Rana, A., Sfiligoi, I., Sim, A., Thomas, M., & Wuerthwein, F. (2009, October). Hadoop Distributed File System for the grid. In *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE* (pp. 1056-1061). IEEE.

Bhangu, B. S., Bentley, P., Stone, D. A., & Bingham, C. M. (2005). Nonlinear observers for predicting state-of-charge and state-of-health of lead-acid batteries for hybrid-electric vehicles. *Vehicular Technology, IEEE Transactions on*, *54*(3), 783-794.

Bolly, V.K, Springer, J. A., & Dietz, J. E (In Press). *A study of Electric vehicle data analytics*, Poster paper to be presented at 2nd ASE International Conference on Big Data Science and Computing, Stanford, CA (5 MSP).

Bolly, V.K, Springer, J. A., & Dietz, J. E (In Press). *Using Open Source NoSQL technologies in Designing Systems for Delivering Electric Vehicle Data Analytics*, Paper to be presented at 121st ASEE Annual Conference & Exposition, Indianapolis, IN (9 MSP).

Bradley, T. H., & Frank, A. A. (2009). Design, demonstrations and sustainability impact assessments for plug-in hybrid electric vehicles. *Renewable and Sustainable Energy Reviews*, *13*(1), 115-128. Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record, 39(4),* 12-27.

Chan, C. C. (2007). The state of the art of electric, hybrid, and fuel cell vehicles. In *Proceedings of the IEEE*, *95*(4), 704-718. Cox, D. C., & Perez-Kite, R. (2000). Battery state of health monitoring, combining conductance technology with other measurement parameters for real-time battery performance analysis. In *Telecommunications Energy Conference, 2000. INTELEC. Twenty-second International* (pp. 342-347). IEEE.

Cuzzocrea, A., Song, I. Y., & Davis, K. C. (2011, October). Analytics over large-scale multidimensional data: the Big Data revolution. In *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP* (pp. 101-104). ACM.

Edwards, M., Rambani, A., Zhu, Y., & Musavi, M. (2012). Design of Hadoop-based Framework for Analytics of Large Synchrophasor Datasets. *Procedia Computer Science*, *12*, 254-258.

Ehsani, M., Gao, Y., & Emadi, A. (2009). Modern electric, hybrid electric, and fuel cell vehicles: fundamentals, theory, and design. CRC press.

Guha, S., Kidwell, P., Hafen, R., & Cleveland, W. S. (2009). Visualization databases for the analysis of large complex datasets. In *International Conference on Artificial Intelligence and Statistics* (pp. 193-200).

Heer, J., Mackinlay, J., Stolte, C., & Agrawala, M. (2008). Graphical histories for visualization: Supporting analysis, communication, and evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, *14*(6), 1189-1196.

Mackinlay, J. D., Hanrahan, P., & Stolte, C. (2007). Show me: Automatic presentation for visual analysis. *Visualization and Computer Graphics, IEEE Transactions on*, *13*(6), 1137-1144.

Oppel, A. (2011). *Databases DeMYSTiFieD-Hard Stuff made easy*. New York, NY: McGraw-Hill, Inc..

Liu, R., Dow, L., & Liu, E. (2011, January). A survey of PEV impacts on electric utilities. In *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES* (pp. 1-8). IEEE.

Padhy, R. P., Patra, M. R., & Satapathy, S. C. (2011). RDBMS to NoSQL: Reviewing some next-generation non-relational databases. In *International Journal of Advanced Engineering Science and Technologies*, *11*(1), 15-30.

Padhy, R. P. (2012). Big Data Processing with Hadoop-MapReduce in Cloud Systems. *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, *2*(1), 16-27.

Salvini, S., Lopatka, P., & Wallom, D. (2011, November). A hardware and software computational platform for the HiPerDNO (high performance distribution network operation) project. In *Proceedings of the first international workshop on High performance computing, networking and analytics for the power grid* (pp. 75-82). ACM.

Shafer, J., Rixner, S., & Cox, A. L. (2010, March). The Hadoop distributed filesystem: Balancing portability and performance. In *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on* (pp. 122-133). IEEE.

Steinhart, G. (2010). DataStaR: A data staging repository to support the sharing and publication of research data. *International Association of Scientific and Technological University Libraries, 31st Annual Conference*.

Ullman, J., & Widom, J. (1997). *A First course in database systems,* New Jersey: Prentice Hall.

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

APPENDICES

Appendix A

A.1 System configuration parameters

The details of the Hadoop release used in this research is shown below:

```
Hadoop 1.2.1 subversion:
https://svn.apache.org/repos/asf/hadoop/common/branches/bra
nch-1.2 -r 1503152
```

The important files and properties configured for this research are listed below:

- `hadoop-env.sh` – This file is used to configure environment for the Hadoop daemon processes. We have set the `JAVA_HOME` with the path to Java installation.
- `core-site.xml` – In this file, the URI to the Hadoop NameNode and its listening port is set. Usually it's the host name of the master node of the cluster.

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://evmaster:9000</value>
  </property>
<property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/local/cluster/hadoop-1.2.1/tmp</value>
```

```
  </property>
</configuration>
```

- `hdfs-site.xml` – Here, you provide the paths to local directories to be used for physically storing NameNode information and HDFS data blocks. This file is used to define and configure other storage parameters for HDFS like block size, data replication factor, etc.

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.name.dir</name>
    <value>${hadoop.tmp.dir}/dfs/name</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>${hadoop.tmp.dir}/dfs/data</value>
  </property>
  <property>
    <name>fs.checkpoint.dir</name>
    <value>${hadoop.tmp.dir}/dfs/namesecondary</value>
  </property>
  <property>
    <name>dfs.block.size</name>
    <value>10485760</value>
```

```
  </property>
</configuration>
```

- `mapred-site.xml` – This file is used to configure the parameters of MapReduce system. It is used to set the JobTracker information and parameters that control the TaskTrackers performance like number of launched mapper and reducer tasks, heap size for child JVMs of mappers and reducers, etc.

```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>evmaster:9001</value>
  </property>
  <property>
    <name>mapred.local.dir</name>
    <value>${hadoop.tmp.dir}/mapred/local</value>
  </property>
  <property>
    <name>mapred.system.dir</name>
    <value>${hadoop.tmp.dir}/mapred/system</value>
  </property>
  <property>
    <name>mapred.child.java.opts</name>
    <value>-Xmx512m</value>
  </property>
  <property>
```

```
    <name>mapred.reduce.slowstart.completed.maps</name>
    <value>1</value>
  </property>
</configuration>
```

- `masters` – This file defines the master node IP or hostname for the Hadoop cluster.
- `slaves` – This file contains the hostnames/IP addresses list of subordinate nodes in the cluster.

After configuring the files in hadoop `conf` directory of the master node, they are copied to all the subordinate nodes in the cluster. Before starting the cluster for the first time, the NameNode has to be formatted using the command:

```
hadoop namenode –format
```

After the NameNode was successfully formatted, the cluster was started using `bin\start-all.sh` script inside Hadoop directory.

Appendix B

B.1 Data collected for analysis

Table B.1. *The summary of run-time values during 5 runs in Figure 4.10*

Data size (MB)	Number of nodes	Average run-time (milliseconds)	Standard deviation	Standard error
129.63	1	16460.4	8.96	4.01
474.99	1	21499.2	20.96	9.37
604.62	1	23321.8	851.31	380.72
797.54	1	26121.4	550.42	246.15
927.17	1	28089	914.40	408.93
1361.04	1	37173.2	891.56	398.72
3537.93	1	74940	843.64	377.29
5069.72	1	99054.2	846.60	378.61
129.63	2	16467.4	10.45	4.68
474.99	2	18686.4	433.98	194.08
604.62	2	21319	838.53	375.00
797.54	2	21765.4	840.26	375.77
927.17	2	22929.6	894.11	399.86
1361.04	2	26940	555.97	248.64
3537.93	2	44496.6	557.80	249.45
5069.72	2	61103	762.56	341.03
129.63	4	16506.6	693.50	310.14

Table B.1 continued.

129.63	4	16506.6	693.50	310.14
474.99	4	20577	30.36	13.58
604.62	4	19550	994.52	444.76
797.54	4	20771.2	842.10	376.60
927.17	4	21791.2	835.77	373.77
1361.04	4	22228.6	579.61	259.21
3537.93	4	29134.6	865.33	386.99
5069.72	4	41728.8	981.13	438.77
129.63	8	16517.4	17.49	7.82
474.99	8	20364.6	443.69	198.42
604.62	8	20764	445.35	199.17
797.54	8	21625	689.36	308.29
927.17	8	22418.4	842.36	376.71
1361.04	8	22637.4	711.82	318.33
3537.93	8	26779.4	887.56	396.93
5069.72	8	34060.8	708.95	317.05
129.63	16	16371.2	828.70	370.61
474.99	16	20859.8	842.77	376.89
604.62	16	20637.2	1221.70	546.36
797.54	16	22275.4	912.31	408.00
927.17	16	22886.4	1631.39	729.58

Table B.1 continued.

1361.04	16	23365.4	884.06	395.36
3537.93	16	30772.8	837.99	374.76
5069.72	16	35192.8	715.03	319.77

Table B.2. *The summary of run-time values during 5 runs in Figure 4.11*

Data size (MB)	Number of nodes	Average run-time (milliseconds)	Standard deviation	Standard error
129.63	1	18852.2	551.22	246.51
604.62	1	29906.8	895.13	400.32
927.17	1	42535.2	14.82	6.63
1361.04	1	52763.4	453.37	202.75
3537.93	1	123828.8	682.86	305.38
5069.72	1	171327.2	890.42	398.21
129.63	2	18231.8	440.49	196.99
604.62	2	25724	448.54	200.59
927.17	2	30978.2	895.44	400.45
1361.04	2	35189.8	546.80	244.54
3537.93	2	66075	446.19	199.54
5069.72	2	93643.6	557.9	249.5
129.63	4	18139	636.75	284.76
604.62	4	25267.8	908.98	406.51

Table B.2 continued.

927.17	4	30430.8	424.64	189.9
1361.04	4	34726.6	997.55	446.12
3537.93	4	42166.2	453.17	202.66
5069.72	4	70061	1475.51	659.86
129.63	8	18552.4	704.06	314.87
604.62	8	25960.4	885.93	396.20
927.17	8	29860.6	841.63	376.39
1361.04	8	31269.6	869.95	389.05
3537.93	8	36270.2	896.83	401.07
5069.72	8	39655.4	538.47	240.81
129.63	16	18350.6	443.12	198.17
604.62	16	25816.8	845.48	378.11
927.17	16	30066.8	900.18	402.57
1361.04	16	29884.2	834.57	373.23
3537.93	16	33958.2	36.62	16.37
5069.72	16	34671.2	542.81	242.75

Table B.3. *The summary of run-time values during 5 runs in Figure 4.12*

Data size (MB)	Number of nodes	Average run-time (milliseconds)	Standard deviation	Standard error
48.29	1	22058.8	549.92	245.93

Table B.3 continued.

230.97	1	27472.8	4.55	2.03
472.77	1	40260.6	1072.27	479.53
844.58	1	66265.6	627.06	280.43
1317.34	1	95847.8	437.84	195.81
48.29	2	19081.6	549.20	245.61
230.97	2	22486	8.51	3.81
472.77	2	27327.8	455.53	203.72
844.58	2	42773.6	439.15	196.39
1317.34	2	62378	847.87	379.18
48.29	4	17103.6	547.53	244.86
230.97	4	22547.2	6.69	2.99
472.77	4	25612.6	702.23	314.05
844.58	4	30904	433.40	193.82
1317.34	4	45452.8	551.45	246.62
48.29	8	16731.2	444.95	198.99
230.97	8	21574.4	706.41	315.91
472.77	8	20289.6	566.15	253.19
844.58	8	29772.4	1043.90	466.85
1317.34	8	37218.4	894.88	400.20
48.29	16	16548.4	7.44	3.33
230.97	16	22003.2	875.88	391.70

Table B.3 continued.

472.77	16	24477	456.90	204.33
844.58	16	28949.8	465.93	208.37
1317.34	16	34844.2	1216.90	544.21

Table B.4. *The summary of run-time values during 5 runs in Figure 4.16*

Number of nodes	Balancing threshold	Average run-time (milliseconds)	Standard deviation	Standard error
1	1%	112807.8	5968.71	2669.28
2	1%	78009.6	8460.58	3783.69
4	1%	62822.4	1229.29	549.75
8	1%	48635.6	1780.53	796.28
16	1%	38808.4	1858.18	831.01
20	1%	40160.6	883.18	394.97
1	5%	112807.8	5968.71	2669.28
2	5%	82027.8	1618.75	723.92
4	5%	66833.6	1876.45	839.17
8	5%	54433.2	4341.81	1941.72
16	5%	48267.8	2622.34	1172.75
20	5%	53552.2	2735.98	1223.57
1	default	112807.8	5968.71	2669.28
2	default	85555.2	8099.93	3622.40
4	default	74890.4	6791.17	3037.10

Table B.4 continued.

8	default	59974.2	8328.36	3724.55
16	default	67507	8579.50	3836.87
20	default	85343.8	4530.41	2026.06

Table B.5. *The summary of run-time values during 5 runs in Figure 4.17*

Map input split size (MB)	Average run-time (milliseconds)	Standard deviation	Standard error
32	80049.8	265.94	118.93
64	79581.4	482.69	215.86
128	34393.4	541.09	241.98
256	34222.8	559.86	250.38
512	40007	2793.77	1249.41
1024	38840	2723.83	1218.13
2048	38601	3338.77	1493.14

Table B.6. *The summary of run-time values during 10 runs in Figure 5.1*

Number of physical machines	Average run-time (milliseconds)	Standard deviation	Standard error
2	43657.6	1726.38	545.93
4	40535.1	1846.63	583.96
8	37985.1	2325.57	735.41

Table B.6 continued.

16	32185.9	965.33	305.27