

METOCEAN BIG DATA PROCESSING USING HADOOP

By

Nadiatul Akmal binti Md Yusof

A project dissertation submitted to the
Information and Communication Technology Programme
Universiti Teknologi PETRONAS
In partial fulfilment of the requirement for
The BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION AND COMMUNICATION TECHNOLOGY)
MAY 2015

University Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak

CERTIFICATION OF APPROVAL

Metocean Big Data Processing Using Hadoop

By

Nadiatul Akmal binti Md Yusof

15933

A project dissertation submitted to the

Information and Communication Technology Programme

Universiti Teknologi PETRONAS

In partial fulfilment of the requirements for the

BACHELOR OF TECHNOLOGY (Hons)

(INFORMATION AND COMMUNICATION TECHNOLOGY)

Approved by,



(Dr Mohamed Nordin bin Zakaria)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

May 2015

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Nadiatul Akmal binti Md Yusof

ABSTRACT

This report will discuss about MapReduce and how it handles big data. In this report, Metocean (Meteorology and Oceanography) Data will be used as it consist of large data. As the number and type of data acquisition devices grows annually, the sheer size and rate of data being collected is rapidly expanding. These big data sets can contain gigabytes or terabytes of data, and can grow on the order of megabytes or gigabytes per day. While the collection of this information presents opportunities for insight, it also presents many challenges. Most algorithms are not designed to process big data sets in a reasonable amount of time or with a reasonable amount of memory. MapReduce allows us to meet many of these challenges to gain important insights from large data sets. The objective of this project is to use MapReduce to handle big data. MapReduce is a programming technique for analysing data sets that do not fit in memory. The problem statement chapter in this project will discuss on how MapReduce comes as an advantage to deal with large data. The literature review part will explain the definition of NoSQL and RDBMS, Hadoop Mapreduce and big data, things to do when selecting database, NoSQL database deployments, scenarios for using Hadoop and Hadoop real world example. The methodology part will explain the waterfall method used in this project development. The result and discussion will explain in details the result and discussion from my project. The last chapter in this project report is conclusion and recommendation

ACKNOWLEDGEMENTS

First, I would like to thank Allah s.w.t for everything. Without Him, I would be loss and might not be able to do my final year project. To my family, especially my parents, for the continuous support and encouragement.

I would like to express my sincere gratitude to my final year project supervisor, Dr Mohamed Nordin bin Zakaria for the support, his patience and immense knowledge. With his guidance, it helps me a lot in all the time of research and writing the report. I would also like to thank Mr Ade Wahyu and Mr Fadzli from HPCC for their guidance and advice.

To all my friends, thank you.

TABLE OF CONTENTS

CONTENT	PAGE
Title Page	i
Certification	ii
Abstract	iv
Acknowledgements	v
Table of Contents	vi
List of Figures	ix
Abbreviations and Nomenclatures	x
Chapter 1: Introduction	1
1.1 Background	1
1.2 Problem Statement	2
1.3 Objectives	2
1.4 Scope of Study	3
Chapter 2: Literature Review and/or Theory	4
2.1 Definition of NoSQL and Relational Database Model	4
2.2 Hadoop, MapReduce and big data	5
2.3 Things to do when selecting database	7
2.4 Why choose NoSQL over RDBMS	9
2.5 NoSQL database deployments	10

2.6 Hadoop and NoSQL	11
2.7 Scenarios for using Hadoop	12
2.7.1 Three distinct scenarios for Hadoop	12
2.8 Hadoop real world examples	14
Chapter 3: Methodology/Project Work	17
3.1 Waterfall model design	17
3.2 Phases in waterfall model	18
3.2.1 Requirement analysis	18
3.2.2 System design	18
3.2.3 Implementation	19
3.2.4 Testing	19
3.2.5 Deployment	19
3.2.6 Maintenance	20
Chapter 4: Results and Discussion	21
4.1 Hortonworks sandbox	21
4.1.1 Process data with Apache Pig	21
4.1.2 Explore Data with Apache Pig from the Grunt shell	23
4.2 MATLAB and MapReduce	24

4.2.1 MapReduce algorithm phase	26
4.3 Mapreduce in MATLAB	27
4.4 Code sample	28
4.4.1 Interface code	28
4.4.2 Mapper code sample	30
4.4.3 Reducer code sample	31
4.5 Interface	32
Chapter 5: Conclusion and Recommendation	34
5.1 Hortonworks Sandbox and MATLAB in processing big data	34
5.2 Hadoop challenges	35
5.2.1 Talent gap	35
5.2.2 Data security issue	35
5.2.3 Inefficiency	36
References	37

LIST OF FIGURES

Figure	Page
Figure 3.1 Waterfall model	17
Figure 3.2 Metocean data sample	18
Figure 4.1 Lines of codes for processing data using Apache Pig	21
Figure 4.2 Count, Maximum and Sum operation	22
Figure 4.3 Time taken to complete task	22
Figure 4.4 Dump movies	23
Figure 4.5 Describe movies	24
Figure 4.6 Display movies greater than 3.5	24
Figure 4.7 Time taken to run command	24
Figure 4.8: MapReduce algorithm	26
Figure 4.9: Metocean analysis interface	32
Figure 4.10: Result when maximum analysis is being selected	33
Figure 4.11: Result in pdf when documentation is selected	33
Figure 5.1: Hortonworks Sandbox interface	35

ABBREVIATIONS AND NOMENCLATURES

Metocean	Meteorology and Oceanography
ARIMA	Autoregressive integrated moving average
IBM	International Business Machines
RDBMS	Relational Database Management System
SQL	Structured Query Language
NoSQL	Non Structured Query Language
HDFS	Hadoop Distributed File System
YARN	Yet Another Resource Negotiator
JSON	JavaScript Object Notation
API	Application Programming Interface
SAS	Statistical Analysis System
SAN	Storage Area Network
ETL	Extraction, Transformation and Loading
CSV	Comma Separated Values
SDLC	Software Development Life Cycle

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

Metocean according to an article in Definition-of is the abbreviation of “Meteorology” and “Oceanography” that is commonly used in the offshore industry. The term describes the physical environment usually vicinal an offshore platform. Metocean data is the observation measurement of wind, atmospheric pressure, air, temperature, waves, current, water level, salinity, water temperature, etc. (Metocean database description,2014) These data are regularly collected in situ by major oil and gas (O&G) companies. Metocean analyses serve crucial information needed for operations or design work that has health and safety and economic corollary.

Typically, Metocean data is collected by specialist companies and distributed to paying parties who will then set up scientists and engineers to analyze and forecast information based on the information. The data, confidential in nature, will be passed around in large storage devices, or stored in a networked device to be downloaded and studied when needed. For large data set, processing and analysis typically requires long, expensive hours on high-performance workstations and servers, especially when methods such as ARIMA and fuzzy logic is deployed involving large time periods and number of geographical locations.

MapReduce definition according to an article in Search Cloud Computing is a software framework that allows developers to write their own programs that will process massive amounts of data that is unstructured in parallel across a stand-alone computers or distributed cluster of processors. MapReduce is known as the heart of Hadoop. MapReduce concept is quite simple to understand especially for those who are familiar with the clustered scale-out data processing solutions.

According to an article about MapReduce in IBM, the term MapReduce itself actually refers to two separate and discrete tasks that Hadoop programs perform. The first task is the map job, which will takes a set of data and converts it into another set

of data, where individual elements are broken down into tuples (key/value pairs). The reduce job on the other hand, takes the output from a map as input and then combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job will always be performed after the map job.

1.2 PROBLEM STATEMENT

When dealing with Metocean data, big data were involved. More complex data of meteorological and oceanographic data may include multiple levels of nesting. In the past, these data usually modeled into relational tables but it has not fit into the two dimensional row column structure naturally. Other alternatives to store the big and complex data need to be considered. When handling complex data, other than RDBMS, NoSQL can be a lot of help. In NoSQL databases, the multi-level nesting and hierarchies can be easily represented. NoSQL database are primarily non-relational database.

SQL database are vertically scalable. Problem rise when the database need to be scaled, hardware boost is needed on which the DBMS System is installed. On the other hand, NoSQL are horizontally scalable. More nodes need to be added and distribution network need to be created based on our need and power is required. Load on the database will be reduced thus increase the scalability.

1.3 OBJECTIVES

The objective of this project is to use Hadoop to process Metocean Data. Descriptive statistics will be done to display the processed data. MATLAB will be used as a platform to do the descriptive statistics. Descriptive statistics in this project will consists of mean, mode, median, maximum and minimum value.

1.4 SCOPE OF STUDY

Descriptive statistics. Descriptive statistics according to an article in Investopedia titled “Descriptive Statistics” is a set of brief descriptive coefficients that summarizes a given data set, which can either be a representation of the entire population or a sample. The measures used to describe the data set are measures of central tendency and measures of variability or dispersion. The measures of central tendency include mode, mean, median while measures of variability include the standard deviation (or variance), maximum and minimum variables, kurtosis and skewness. Descriptive statistics provides a useful summary of security returns.

This happens when performing empirical and analytical analysis, as they provide a historical account of return behavior. The expectations of future events need to be considered although past information is useful in any analysis. In other words, descriptive statistics helps describe, show or summarize data in a meaningful way. However, descriptive statistics does not allow us to make conclusions beyond the data we have analyzed or reach conclusions regarding any hypotheses we might have made. They are simply a way to describe our data.

CHAPTER 2: LITERATURE REVIEW AND/OR THEORY

2.1 DEFINITION OF NOSQL AND RELATIONAL DATABASE MODEL (RDBMS)

Database is the assemblage of organized data (Christopher Heng,2014). When someone has a lot of files, it is not convenient to save the files with names like “doc1”, “doc2”, and “doc3”. It is inefficient and it will consume a lot of time to retrieve the files back. Database programs were designed to help store tons of data in an organized way so it will be easier to retrieve them back. According to O.S Tezer (2014) there are two types of database model; relational model and the model-less (NoSQL) approach.

Each and every database system will use a different database model that is suitable for them. The selection of database model is very important as it is the first step to determine how the database model will work and handle the data. The Relational Model is the most popular and widely used database model. Although it is the most popular database model, it has several issues that were not being solved. Then comes the new model NoSQL that were promised to solve the problems faced by Relational Model users. NoSQL also offers interesting functionality as a bonus.

NoSQL according to an article in Techopedia is a class of database management system (DBMS). It does not follow all the rules of a relational database management system and it cannot use the traditional SQL to query its data. NoSQL normally misunderstood as the replacement of SQL and RDBMS. NoSQL actually the complementary of RDBMS and SQL. The implementation of NoSQL-based system usually involves large databases where SQL and relational model of databases couldn't handle due to performance problems. Some examples implementations of NoSQL are Google's BigTable, Facebook's Cassandra database and Amazon's SimpleDB and Dynamo. Meanwhile Relational Database management system (RDBMS) is a database engine or system based on the relational model (Techopedia,

2014). Data in RDBMS must be presented and stored as relations. For an example, tables will have relationships with each other; primary key and foreign key.

2.2 HADOOP, MAPREDUCE AND BIG DATA

Hadoop by definition according to Search Cloud Computing site, is a free, Java-based programming framework. Hadoop supports the processing of large data sets in distributed computing environment. Sponsored by the Apache software Foundation, Hadoop is part of the Apache project. To understand Hadoop, one has to understand the two fundamental things about it (Gualtieri, 2013). They are; How Hadoop store files and How Hadoop process the data. Imagine that you have a large file and it is so large that it did not fit into your PC's capacity. At the end of the day, you will not be able to store the file. But with Hadoop, you can store files that are bigger than what can be stored on one particular node or server. You will be able to store very large files or even many files that you want.

The second characteristic for Hadoop is the ability to process data or at least provide a framework for processing data. This is called Map/Reduce. Map/Reduce, rather than taking the conventional step of moving data over a network to be processed by software uses a smarter approach tailor made for big data sets. For a really large data sets, moving the data over a network can be very slow. Imagine opening a very large file on laptop. This will take a very long time or forever for some old laptops. Worst case is laptop will stop working and reboot itself. MapReduce, rather than moving the data to the software, it moves the processing software to the data.

In another article, Hadoop was explained by defining the modules in it. Hadoop is made up of "modules", which each module carries out a particular task that is essential for a computer system designed for big data analytics (Bernard Marr, 2014). These four modules are Distributed File-System, MapReduce, Hadoop Common and Yarn. Bernard and Gualtieri both agreed that Distributed File System and

MapReduce are the two most important or fundamental thing in Hadoop. Distributed File System or HDFS (Hadoop Distributed File System) serves the purpose in allowing data to be stored in an easy accessible format, across a large number of linked storage devices. The method called “file system” is used by a computer to store data so it can be easily found and used. This is usually determined by the computer’s operating system. Hadoop system however, uses its own file system that sits “above” file system of the host computer – it can be easily accessed using any computer running any supported operating system.

Hadoop Common, is the other module in Hadoop which provides the tools in Java that is needed for the user’s computer system such as Windows or Unix. This is to read the data stored under the Hadoop file system. The final module in Hadoop is YARN. YARN serves the purpose of managing resources of the systems storing the data and running the analysis. Numerous other procedures, libraries or features have come to be considered part of the Hadoop “framework” over the years, but HDFS (Hadoop Distributed File System) , MapReduce, Hadoop Common and YARN are the principle four (Bernard Marr, 2014). According to Ravi in his article “What is a “Hadoop”? Explaining Big Data to the C-Suite” the big deal about Hadoop is the flexibility, scalability and economics.

By definition, according to dictionary.com, flexible is susceptible of modification or adaption; adaptable. Hadoop is flexible as we can store any data and can run any analysis. Scalability according to Wikipedia is the ability of a system, network, or process to handle an increasing amount of work in a capable manner or the ability for it to be enlarged to accommodate the growth. Hadoop can be considered as a scalable ecosystem as the ability for Hadoop to accommodate growth of data can start at 1 Terabyte per 3-nodes and can grow up to petabytes per 1000s nodes. Economical or economics according to the definition in the free dictionary site is sparing in the amount of items involved to achieve a desired result. Hadoop is considered economics as the cost per Terabytes at a fraction of traditional options.

2.3 THINGS TO DO WHEN SELECTING DATABASE

Before selecting database model, there are several things that need to be considered. According to Jnan Dash (2013), the two most important things one needs to consider when selecting a database are characteristics of data and the volatility of the data model. In case that the selected data consist of a simple tabular structure, it should be adequate to use the relational model. For example, data like an accounting spreadsheet does not need to use the model-less (NoSQL) approach.

Data that might have multiple levels of nesting tend to be more complex. This type of data normally modelled into relational tables. But a problem occurs when these data are not fit to the two-dimensional row-column structure of relational tables. Some examples for multiple levels of nesting type of data are geo-spatial, molecular modelling and engineering parts. NoSQL database should be considered as an option as the multi-level nesting and hierarchies will be easily presented in the JavaScript Object Notation format used by some NoSQL products.

Data volatility according to article in its bldrdoc website is the rate of change in the amount of stored data over a period of time. It is important to know whether the data model is going to be changed or evolved easily over a period of time or is it going to stay the same. The facts about the data model are unknown at the design time is known and so flexibility is very important. In many companies that use the MySQL database system, they will spend many hours cautioning their users to design the right schema the first time. Later revisions made after that will slow or stop the database from operating. Any potential changes needed to be done must be minimal to avoid the slowing or stopping process of the database. The “get it right first” approach might not have worked in the new world of dynamic schema where changes are made daily, hourly even every minute to fit the ever changing data model. It might work in the old static schema. So a great flexibility type of database is required and NoSQL is the answer.

As for Stephen Pimentel (2014) data model, query language, scalability, concurrency control and consistency model are the things that needed to be considered when selecting database. Data model can be considered as one of the fundamental issues

that have to do with the “structure” of your data. Some might just be expressed in normalized tables and for some data centered on structured objects that will be converted easily into the embedded JSON documents. For some data they might consist of nodes and links. As for the simple one, there might just be like a dictionary with values and keys. Application data can be categorized on any of these categories; one or more. Thus it is hard to make decision on which database solution needed because most of the database solution only support a single data model; graph, document, key-value or relational. Things for the queries will get complicated if the data model is poorly matched and it will also harm the overall performance.

A query language might not be a concern if the applications used are very well-suited to access the data through application programming interface (API). Because not all application that were used requires a separate query language. Applications often concurrent with each other, they don't just perform a single write. They need to perform the whole sets of reads and writes as determined by the business logic. Some systems may support single row or documents only. The strongest form of concurrency control is to support transactions with full Atomicity, Consistency, Isolation, Durability (ACID) properties over numerous keys on any node in the cluster. However, even systems claiming to offer transactions vary significantly in their performance and support for Atomicity, Consistency, Isolation, Durability (ACID) properties.

Operational database normally have numerous clients performing updates concurrently. They definitely need a mechanism that can control and handle potential conflict updates. The mechanisms used by these systems varies widely. There are systems that can offers only the weak consistency guarantees. For an example, eventually consistent system guaranteed that a key will return the most recent updated value after a sufficient period of inactivity. The conflicting values then will be managed by application clients. In contrary, a system with a strong level of consistency will guarantee that any write acknowledged to a client will be understandable and readable by all clients.

Terry Orsborn (2013) has the same idea with Stephen Pimentel that scalability is important when selecting database. According to Stephen Pimentel (2014) application will grow over time. In older times, people might want to replace the

server with the larger one which is expensive. Sometimes this approach is referred to “vertical” scaling. The latest alternative introduced is the “horizontal” scaling. It allows user to add commodity hardware to a distributed cluster that makes it cheaper than “vertical” scaling. It is very challenging to provide transactions over distributed cluster when multi-client operations rely on transactional guarantees for the database operations. Majority of the NoSQL systems does not have the jettisoned support for the transactions to provide a simpler distributed designs and it will appear simpler for the database designers but not for the programmers.

According to Terry Osborn (2013) internal users and external customers, they don’t readily differentiate between local and remote assets and they come to expect a near-instantaneous reply from technology. For an example, user usually assumes that when they want to retrieve data from the internet, they will get it quickly as it is when they retrieved data from the data hosted on their computer. A Relational Database Management System (RDBMS) underpins most enterprise applications, where any sluggishness within this layer will inevitably bubble up and interrupt the rest of the information-processing environment. A scalable Relational Database Management System (RDBMS) are able to support ever-growing quantities of data, users and transactions without involving any new hardware costs. Several best methods for increasing scalability according to Terry Osborn (2013) is to include data compression, in-memory processing, database virtualization, and exploiting the power of multi-processor servers.

2.4 WHY CHOOSE NOSQL OVER RDBMS

NoSQL does not require a tight schema binding up front. When handling data that the schema is not properly defined or it is expected to change over time, the traditional RDBMS with rigid schema will make things be a little bit difficult. Sure, we can make things work but it will require an extra effort front and then extra effort for the life of the application. With NoSQL, that problem will go away.

2.5 NOSQL DATABASE DEPLOYMENTS

2.5.1 SESSION STORE

As the application have grown in scale, it is very hard for the web application developers to handle session information using the relational technology. A global session store (one that manages session information for each user who visited a website) is the right approach. NoSQL begin to be one of the best options in storing web application session information. This is due to the unstructured nature of session data that make it easier to store data in a schema-less document that in a structured or more rigid RDBMS record. It is critical to have the low-latency to session data to ensure a great user experience

2.5.2 USER PROFILE STORE

Ability to login and user profiles are needed for all web applications. Another example where the key value characteristics of NoSQL comes into play is the global user profile store. User preference, user ID, multiple ID mappings and any additional user information can be stored using NoSQL database so that the application can quickly search for a user and authenticate access. The “always on” and the scale-out characteristics of NoSQL are very essential to any web application. Recently, TuneWiki drafted a BlogSpot on how they use NoSQL as a user profile store.

2.5.3 CONTENT AND METADATA STORE

Integration of different learning tools into a single platform from text-heavy data such as articles, digital content and eBooks is needed by companies like McGraw-Hill. Metadata in content-driven application is the most heavily accessed data because the response time is slow. To build custom content-driven application using NoSQL, document databases particularly gives the flexibility to store a wide variety of content and provide fast access to it.

2.5.4 MOBILE APPLICATIONS

It is very critical for the developer to update and enhance mobile applications without any service disruption. NoSQL database comes in handy as it allows developer to store user information and application content in a schema-less format. This allows developers to make modification or changes to the mobile applications quickly and without major database infrastructure changes. User will not experience any interrupt problem when using the application. Kobo and Playtika that serve millions of users across the globe are some examples of popular companies that use NoSQL and take it as advantage for their mobile applications.

2.6 HADOOP AND NOSQL

Hadoop refers to an ecosystem of software packages, including HDFS, MapReduce, and a whole host of other software packages to support the import and export of data into and from HDFS (Hadoop Distributed File System). By having a Hadoop cluster,

it literally means a cluster of machines all running in general ecosystem with a large distributed file system to support the large scale computation. NoSQL refers to non-relational or at least non-SQL database solutions such as MongoDB, HBase (also a part of the Hadoop ecosystem), CouchDB, Cassandra, Riak, etc.. Hadoop is a computing framework and NoSQL is Not Only SQL databases.

2.7 SCENARIOS FOR USING HADOOP

When a user types in a query, it is not that practical to exhaustively scan millions of items. It does make sense instead to create an index and use it to rank items and then find the best matches. A distributed indexing capability is provided by Hadoop. According to an article in IBM site, Solr offers the indexing capabilities and other powerful search capabilities such as central configuration system, reliability and scalability and failover and recovery. Solr is an enterprise search tool from the Apache Lucene project. Hadoop runs on a cluster or collection of commodity, shared-nothing x86 servers. Servers in a Hadoop cluster (sizes from 50, 100 to even 2000+ nodes) can be added or removed at will as the system will detect and compensates for system or hardware problems on any server. In spite of system changes or failure, Hadoop can deliver data and can run large-scale, high-performance processing batch jobs as it is a self-healing and fault tolerant.

2.7.1 THREE DISTINCT SCENARIOS FOR HADOOP

Hadoop as an ETL(Extract, Transform and Load) and Filtering Platform

Extracting valuable signal from lot of noise became one of the biggest challenges with high volume data sources. It is a good way to go to load large, raw data into a MapReduce platform for initial processing. Raw data can be read by Hadoop platform, appropriate filters and logic can be applied, and structured summary or refined data set can be set as an output. This output (e.g., hourly index refreshes) can be further analysed or serve as an input to a more traditional analytic environment like SAS. Any business problem required a small percent of a raw data feed. Hadoop becomes a great tool for extracting these pieces.

Hadoop as an exploration engine

Using tools to analyse data where it sits makes sense once the data is in the MapReduce cluster. New data can be added to the existing pile without having to re-index all over again as the refined output is in a Hadoop cluster. In other words, new data can be added to existing data summaries. The data can be loaded into corporate systems once it is distilled so users have wider access to it.

Hadoop as an Archive.

Most of the historical data doesn't need to be accessed and kept in a SAN environment. The historical data is usually archived by disk or tape to secondary storage or sent offsite. When these data is needed for analysis, it is painful and costly to retrieve back and load it back up. Most people don't even bother using the historical data for their analytics. With cheap storage in a distributed cluster, a lot of data can be kept "active" for a continuous analysis. Hadoop is very efficient as it allows the generation of different index types in one cluster by allowing better utilization of hardware.

2.8 HADOOP REAL WORLD EXAMPLES

LinkedIn

LinkedIn is an enormous data accumulation whose value is connections. It currently computes more than 100 billion personalized recommendations every week and powering an ever growing assortment of products, including Groups You May Like, Jobs You May be Interested In, Ad Targeting, and News Relevance. LinkedIn leverages Hadoop by using knowledge aggregated to transform raw data to rich features from LinkedIn's 125 million member base. The company then uses Lucene to do a real-time recommendations, and in addition to Lucene on Hadoop to bridge offline analysis with user-facing services. The streams of user-generated information, referred to as a "social media feeds", may contain valuable, real-time information on the LinkedIn member opinions, activities, and mood states.

CBS Interactive

Hadoop is used by CBS Interactive as the web analytics platform, processing one Billion weblogs daily as it grows from 250 million events per day from hundreds of web site properties. CBS Interactive are the online division for the broadcast network CBS. They are the largest premium online content network and listed as in top 10 global web property. Some of the brands include: CNET, TV.com, Last.fm, CBS Sports, 60 Minutes, etc. To crunch web metrics, CBS Interactive migrated its processing from a proprietary platform to Hadoop. The goal was to achieve fault-tolerance and scalability, more robustness, and significant reduction of processing time to reach SLA (Service-Level Agreement); over 6 hours of reduction so far. They built an

ETL framework called Lumberjack, built based on python and streaming to enable this.

Explorys and Cleaveland Clinic

Explorys, founded in 2009 in partnership with the Cleveland Clinic, is one of the largest clinical repositories in the United States with 10 million lives under contract. The Explorys healthcare platform is based upon a massively parallel computing model that enables subscribers to search and analyse patient populations, treatment protocols, and clinical outcomes. With billions of clinical and operational events already curated, Explorys helps healthcare leaders leverage analytics for break-through discovery and the improvement of medicine. HBase and Hadoop are at the centre of Explorys. Already ingesting billions of anonymised clinical records, Explorys provides a powerful and HIPAA compliant platform for accelerating discovery.

Foursquare

Foursquare is a mobile, location and social networking start-up aimed at letting your friends in almost every country know where you are and figuring out where they are. As a platform foursquare is now aware of 25+ million venues worldwide, each of which can be described by unique signals about who is coming to these places, when, and for how long. To reward and incent users foursquare allows frequent users to collect points, prize “badges,” and eventually, coupons, for check-ins.

Foursquare is built on enabling better mobile + location + social networking by applying machine learning algorithms to the collective movement patterns of millions of people. The ultimate goal is to build new services which help

people better explore and connect with places. Foursquare engineering employs a variety of machine learning algorithms to distil check-in signals into useful data for app and platform. Foursquare is enabled by a social recommendation engine and real-time suggestions based on a person's social graph.

Matthew Rathbone, foursquare engineering, describes the data analytics challenge as follows:

“With over 500 million check-ins last year and growing, we log a lot of data. We use that data to do a lot of interesting analysis, from finding the most popular local bars in any city, to recommending people you might know. However, until recently, our data was only stored in production databases and log files. Most of the time this was fine, but whenever someone non-technical wanted to do some ad-hoc data exploration, it required them knowing SCALA and being able to query against production databases.

This has become a larger problem as of late, as many of our business development managers, venue specialists, and upper management eggheads need access to the data in order to inform some important decisions. For example, which venues are fakes or duplicates (so we can delete them), what areas of the country are drawn to which kinds of venues (so we can help them promote themselves), and what are the demographics of our users in Belgium (so we can surface useful information)?”

To enable easy access to data foursquare engineering decided to use Apache Hadoop, and Apache Hive in combination with a custom data server (built in Ruby), all running in Amazon EC2. The data server is built using Rails, MongoDB, Redis, and Resque and communicates with Hive using the ruby Thrift client.

CHAPTER 3: METHODOLOGY/PROJECT WORK

Methodology used in this project is waterfall methodology. It is also referred to as a linear-sequential life cycle model. It is quite simple to understand. Each phase in a waterfall model must be completed before the next phase can begin and there is no overlapping in the phases. Waterfall model is the earliest SDLC approach that was used for software development. The waterfall model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

3.1 WATERFALL MODEL DESIGN

In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

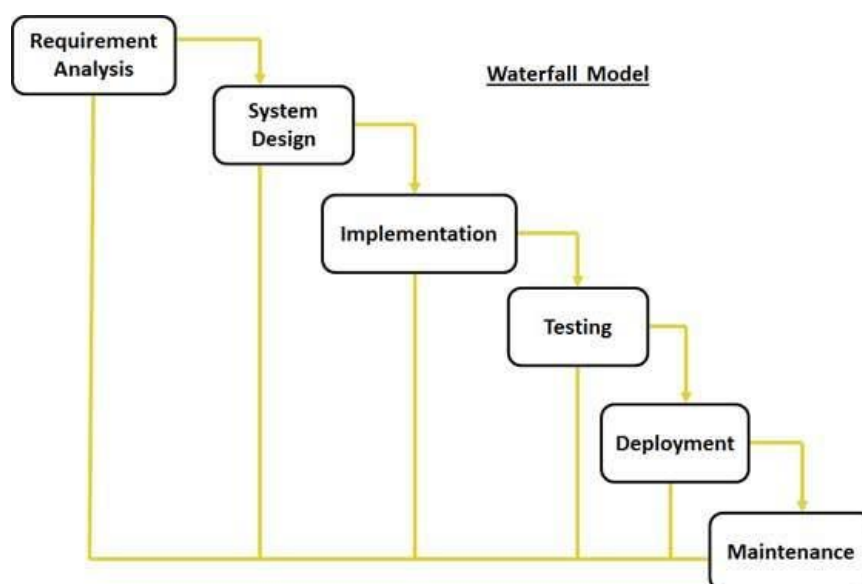


Figure 3.1 : Waterfall Model

3.2 PHASES IN WATERFALL MODEL

3.2.1 REQUIREMENT ANALYSIS

All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc. For this project, MATLAB and Metocean data are the basic requirement that were identified. The Metocean data were used is a sample of a coordinate taken from the real database. The data consists of more than 440000 rows.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
CCYYMM	DDHhmm	WD	WS	ETOT	TP	VMD	ETTSea	TPSea	VMDSea	ETTSw	TPSw	VMDsw	MO1	MO2	HS	DMDIR	ANGSPR	INLINE
195607	10000	31.516	2.71	0.001	1.389	171.213	0.002	2.368	8.001	0.001	3.359	251.024	0	0.01	0.007	100.111	0.38	0.642
195607	10100	34.607	2.492	0	0.185	55.379	0.002	0.026	165.599	0	3.306	53.252	0.004	0.011	0.055	108.732	0.296	0.482
195607	10200	95.886	0.261	0.002	2.249	30.312	0	1.423	125.197	0.001	0.548	247.464	0.003	0.003	0.177	232.304	0.75	0.089
195607	10300	139.985	2.628	0.002	1.446	75.547	0	1.022	111.116	0.001	0.738	190.271	0.002	0.002	0.064	71.918	0.312	0.264
195607	10400	58.54	2.987	0.001	1.27	108.972	0	1.107	218.621	0.001	1.674	14.133	0.002	0	0.082	10.308	0.282	0.006
195607	10500	81.892	1.641	0	2.294	195.096	0.001	0.253	126.327	0.001	1.849	263.675	0.005	0.002	0.197	308.784	0.519	0.412
195607	10600	142.186	2.167	0.003	1.567	280.599	0.001	0.641	38.954	0.001	1.789	49.186	0.003	0.009	0.108	267.776	0.394	0.434
195607	10700	51.829	3.881	0.002	1.62	254.113	0	1.454	72.265	0.001	0.422	63.824	0	0.003	0.122	145.305	0.653	0.521
195607	10800	92.471	2.013	0.002	2.179	118.081	0	1.869	188.61	0.002	1.304	187.699	0.005	0.005	0.133	279.611	0.78	0.106
195607	10900	177.162	0.218	0.002	1.939	149.746	0	0.459	215.17	0.001	1.932	275.706	0.003	0.001	0.19	208.194	0.125	0.419
195607	11000	2.485	1.88	0.001	0.929	32.373	0.001	0.5	292.13	0	2.167	271.255	0.004	0.004	0.098	109.664	0.112	0.486
195607	11100	115.02	1.974	0.002	2.25	8.451	0	0.747	253.825	0.002	0.196	200.282	0.003	0.011	0.062	197.967	0.29	0.095
195607	11200	147.104	2.713	0.002	0.191	272.196	0	1.141	0.394	0.001	0.571	15.923	0.003	0.005	0.149	311.933	0.028	0.314
195607	11300	39.378	2.3	0.001	2.361	116.173	0.001	1.633	0.052	0.002	0.145	122.231	0.004	0.009	0.137	94.128	0.721	0.024
195607	11400	19.587	3.21	0.001	1.997	303.572	0.001	0.314	303.613	0	0.175	27.755	0.001	0.007	0.179	83.062	0.757	0.187
195607	11500	50.623	2.494	0.001	1.406	110.886	0	1.314	211.179	0.001	2.033	261.828	0.003	0.006	0.176	239.553	0.316	0.34
195607	11600	5.409	1.239	0.002	2.262	221.702	0	1.178	252.298	0.002	1.62	222.196	0.004	0.005	0.077	199.448	0.677	0.69
195607	11700	17.946	2.644	0	1.17	25.39	0	0.296	71.859	0.002	1.579	51.331	0.001	0.005	0.054	133.586	0.499	0.064
195607	11800	73.762	1.299	0.001	1.465	126.409	0	0.716	87.974	0	0.589	313.301	0.002	0.004	0.127	182.674	0.003	0.461
195607	11900	83.684	2.581	0.001	0.698	81.959	0	0.226	259.08	0	0.322	3.555	0.003	0.002	0.126	126.312	0.39	0.673
195607	12000	56.083	1.533	0.002	0.47	199.602	0	0.743	315.903	0.001	1.804	159.675	0.002	0.001	0.02	180.368	0.761	0.415
195607	12100	30.175	2.723	0	0.638	296.706	0	0.368	220.668	0.001	0.22	101.836	0.001	0.003	0.129	220.342	0.578	0.634
195607	12200	51.261	1.582	0	0.865	131.366	0	0.48	0	0.001	0.593	28.068	0	0.005	0.142	35.441	0.39	0.554
195607	12300	143.6	1.865	0	1.117	171.962	0	0.186	0.427	0	1.482	76.752	0	0.003	0.06	284.05	0.305	0.591
195607	20000	1.665	1.45	0.001	0.633	23.385	0	0.322	4.323	0.001	1.649	253.36	0.001	0.002	0.077	15.581	0.713	0.482
195607	20100	79.565	1.256	0.001	0.12	290.291	0	0.473	1.659	0.001	1.227	61.67	0.002	0.001	0.077	88.907	0.728	0.369
195607	20200	87.79	0.174	0	1.064	21.875	0	0.189	233.307	0.001	0.226	96.375	0.002	0.002	0.055	43.907	0.836	0.04
195607	20300	60.431	0.182	0	0.848	8.163	0	0.326	315.994	0.001	0.162	114.044	0	0.002	0.041	202.641	0.368	0.123
195607	20400	52.409	0.593	0.001	0.147	141.402	0	0.907	179.105	0	2.732	230.296	0.001	0	0.046	291.74	0.551	0.461

Figure 3.2 : Metocean data sample

3.2.2 SYSTEM DESIGN

The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. In this phase, the hardware used in this project are laptop and its components. Since that mapreduce is running from the desktop, no extra database storage needed since the data had been saved manually in the computer's hard disk.

3.2.3 IMPLEMENTATION

With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing. In this phase, the system is built from the maximum function that consists of a script, mapper and reducer function. The system will then be tested and if there is no error, other analysis such as minimum and mean will be developed and tested.

3.2.4 INTEGRATION AND TESTING

All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures. After all the analysis have been developed and tested, it will then be implemented in the Metocean Analysis interface to test it as a system.

3.2.5 DEPLOYMENT OF SYSTEM

Once the functional and non-functional testing is done, the product is deployed and shown to supervisor.

3.2.6 MAINTENANCE

Maintenance will be done if there is any updates or bugs that need to be fixed

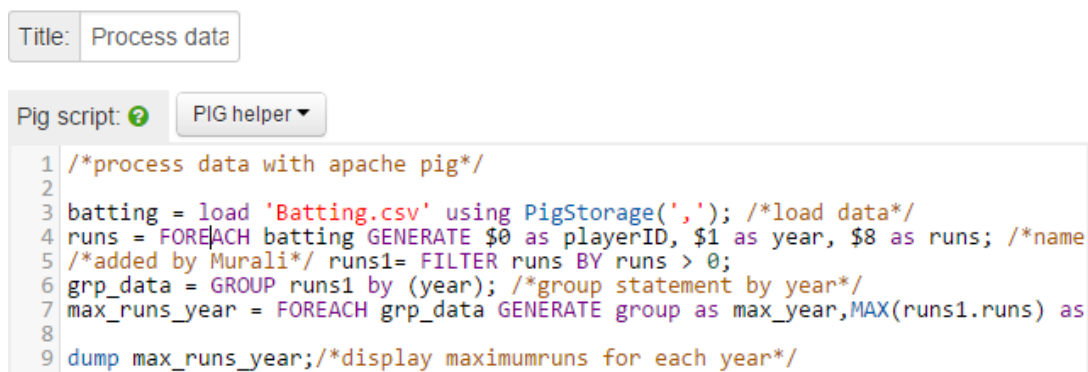
All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 HORTONWORKS SANDBOX

Sandbox is a personal, portable Hadoop environment that comes with a dozen interactive Hadoop tutorials. Sandbox includes many of the most exciting developments from the latest HDP distribution, packaged up in a virtual environment. It is the easiest way to get started with enterprise Hadoop. Sandbox comes with a dozen hands-on tutorials that will be the guide through the basics of Hadoop. The Sandbox includes the Hortonworks Data Platform in an easy to use form. Our own datasets can be added, and it can be connected to the existing tools and applications. New functionality can be tested with the Sandbox before placing it into production. These screenshots are the example of tested data using Hortonworks Sandbox.

4.2 PROCESS DATA WITH APACHE PIG



The screenshot shows a web-based interface for editing a Pig script. At the top, there is a text input field with the value "Process data". Below it, there are two buttons: "Pig script: ?" and "PIG helper ▾". The main area contains a code editor with the following Pig script:

```
1 /*process data with apache pig*/
2
3 batting = load 'Batting.csv' using PigStorage(','); /*load data*/
4 runs = FOREACH batting GENERATE $0 as playerID, $1 as year, $8 as runs; /*name
5 /*added by Murali*/ runs1= FILTER runs BY runs > 0;
6 grp_data = GROUP runs1 by (year); /*group statement by year*/
7 max_runs_year = FOREACH grp_data GENERATE group as max_year,MAX(runs1.runs) as
8
9 dump max_runs_year;/*display maximumruns for each year*/
```

Figure 4.1 Lines of codes for processing data using Apache Pig

This lines of code will load data from a Comma Separated Values (CSV) file using PigStorage. The file is stored in the file browser. The second line of code will generate column 0, 1 and 8 that contains the details of the player id, year and number of runs. After that it will filter number of runs that is greater than 0 only. The next

line of codes indicates that the year will be grouped since that there are multiple rows that displayed the same year with different value of runs. The 5th line will generate the maximum batting number of run for each year and lastly the result will be dump.

```

Title: operation

Pig script: PIG helper
1 /*process data with apache pig*/
2
3 batting = load 'Batting.csv' using PigStorage(','); /*load data*/
4 runs = FOREACH batting GENERATE $0 as playerID, $1 as year, $8 as runs; /*name the field*/
5
6 runs1= FILTER runs BY runs > 0; /* this will filter run that is only greater than 0*/
7
8 grp_data = GROUP runs1 by (year); /*group statement by year*/
9
10 /*find maximum run for each year*/
11 max_runs_year = FOREACH grp_data GENERATE group as max_year,MAX(runs1.runs) as max_runs;
12
13 /*find total number of run for each year*/
14 sum_runs_year = FOREACH grp_data GENERATE group as sum_year, SUM(runs1.runs) as sum_runs;
15
16 /*count number of runs for each year*/
17 count_runs_year = FOREACH grp_data GENERATE group as count_year, COUNT(runs1.runs) as count_runs;
18
19 result_runs_year = FOREACH grp_data GENERATE group as sum_year, SUM(runs1.runs) as sum_runs/FOREACH
20 grp_data GENERATE group as count_year, COUNT(runs1.runs) as count_runs;
21
22 dump count_runs_year;
23 dump max_runs_year;
24 dump sum_runs_year;

```

Figure 4.2 Count, Maximum and Sum operation

Figure 4.2 shows the line of codes using count, max and sum operations.

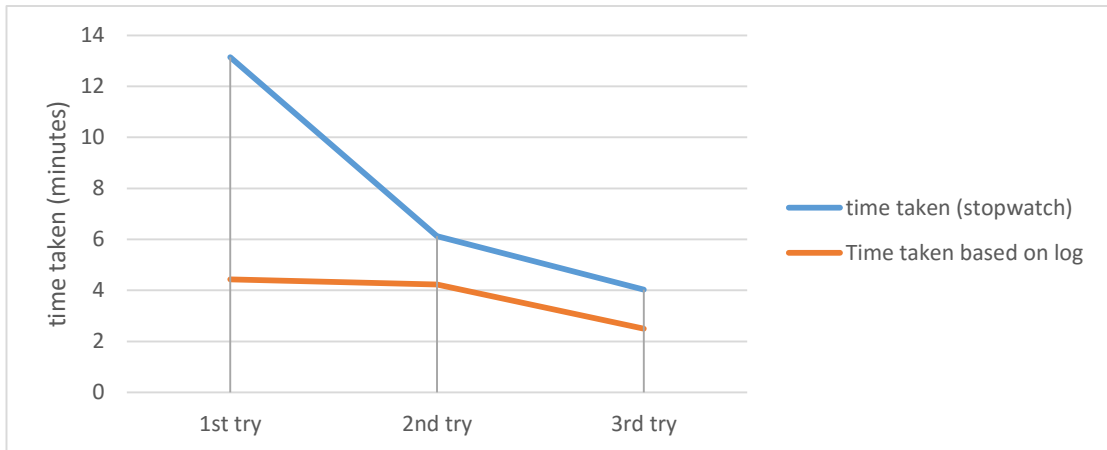


Figure 4.3 Time taken to complete task

The time taken to complete each task is faster from the 1st try to the 2nd try.

4.1.2 EXPLORE DATA WITH APACHE PIG FROM THE GRUNT SHELL

In this tutorial, the author learn on how to lead data file into the Hadoop Distributed File System (HDFS), learn about “FOREACH” and “FILETER” with examples, howto store values into Hadoop Distributed File System (HDFS) and the Grunt shell’s command.

```
hadoop fs -put movies.txt /user/hue
```

```
/*user need to type pig to go into grunt shell*/
```

```
grunt> Movies = LOAD '/user/hadoop/movies.txt' USING PigStorage(',') as (id,name,year,rating,duration);
```

```
DUMP Movies; /*This line of codes will display all the content in the file
```

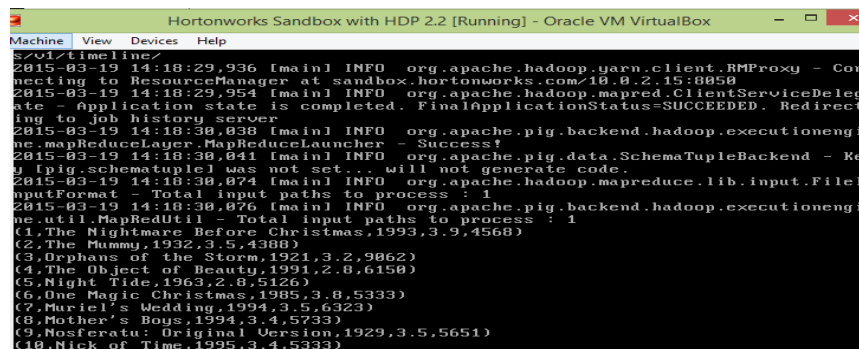
```
Describe Movies; /*this line of code will describe the type of content that is saved in the file (e.g. int, string,etc.)
```

```
grunt>movies_greater_than_three_point_five = FILTER Movies BY rating>3.5; /* this command will filter movies that is greater than 3.5
```

```
grunt> foreachexample= foreach movies_greater_than_three_point_five generate year,rating,name;
```

```
grunt> dump foreachexample;
```

```
grunt> STORE movies_greater_than_three_point_five INTO '/user/hadoop/movies_greater_than_three_point_five' USING PigStorage (',');
```



```
Hortonworks Sandbox with HDP 2.2 [Running] - Oracle VM VirtualBox
Machine View Devices Help
~/vtimeline/
2015-03-19 14:18:29,936 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at sandbox.hortonworks.com/10.0.2.15:8050
2015-03-19 14:18:29,954 [main] INFO org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server.
2015-03-19 14:18:30,038 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2015-03-19 14:18:30,041 [main] INFO org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set, so will not generate code.
2015-03-19 14:18:30,074 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2015-03-19 14:18:30,076 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(1,The Nightmare Before Christmas,1993,3.9,4568)
(2,The Mummy,1932,3.5,4388)
(3,Orphans of the Storm,1921,3.2,9862)
(4,The Object of Beauty,1991,2.8,6150)
(5,Night Tide,1963,2.8,5126)
(6,One Magic Christmas,1985,3.8,5333)
(7,Muriel's Wedding,1994,3.5,6323)
(8,Mother's Boys,1994,3.4,5733)
(9,Nosferatu: Original Version,1929,3.5,5651)
(10,Nick of Time,1995,3.4,5333)
```

Figure 4.4 Dump movies

```

grunt> describe movies
2015-03-19 14:22:46,471 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 10
03: Unable to find an operator for alias movies
Details at logfile: /root/pig_1426774135799.log
grunt> Describe Movies
Movies: {id: int,name: chararray,year: int,rating: float,duration: int}
grunt> _

```

Figure 4.5 Describe movies

```

Hortonworks Sandbox with HDP 2.2 [Running] - Oracle VM VirtualBox
Machine View Devices Help
s/v1/timeline/
2015-03-19 14:40:14,001 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Con
necting to ResourceManager at sandbox.hortonworks.com/10.0.2.15:8050
2015-03-19 14:40:14,026 [main] INFO org.apache.hadoop.mapred.ClientServiceDeleg
ate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirect
ing to job history server
2015-03-19 14:40:26,582 [main] INFO org.apache.hadoop.yarn.client.api.impl.Time
lineClientImpl - Timeline service address: http://sandbox.hortonworks.com:8188/w
s/v1/timeline/
2015-03-19 14:40:26,592 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Con
necting to ResourceManager at sandbox.hortonworks.com/10.0.2.15:8050
2015-03-19 14:40:26,618 [main] INFO org.apache.hadoop.mapred.ClientServiceDeleg
ate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirect
ing to job history server
2015-03-19 14:40:27,147 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MapReduceLauncher - Success!
2015-03-19 14:40:27,167 [main] INFO org.apache.pig.data.SchemaTupleBackend - Re
q [pig.schematuple] was not set... will not generate code.
2015-03-19 14:40:27,223 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileI
nputFormat - Total input paths to process : 1
2015-03-19 14:40:27,229 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.util.MapRedUtil - Total input paths to process : 1
(1993,3.9,The Nightmare Before Christmas)
(1985,3.8,One Magic Christmas)
grunt> _

```

Figure 4.6 Display movies greater than 3.5

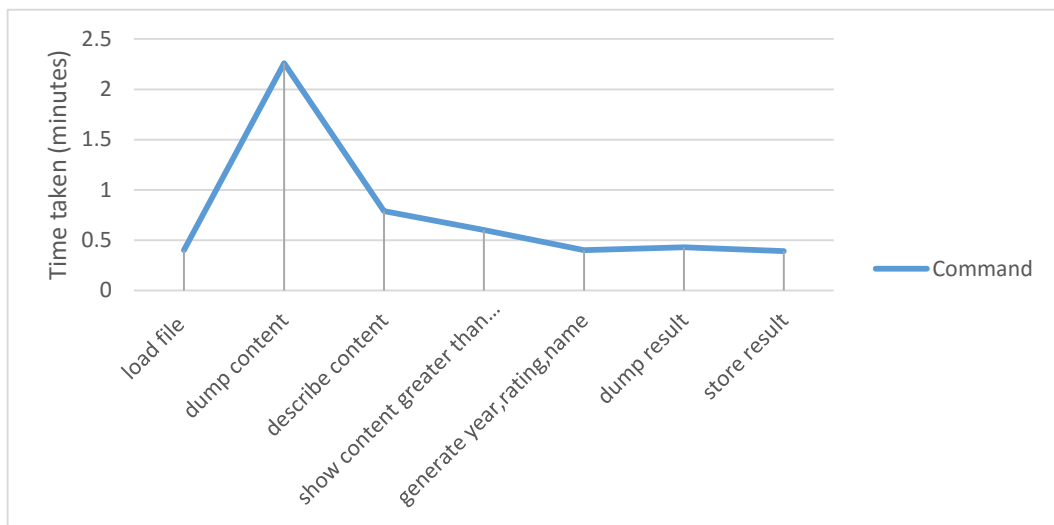


Figure 4.7 Time taken to run command

Dump content command takes longer time approximately around 2 minutes.

4.2 MATLAB AND MAPREDUCE

To do analysis using MapReduce with MATLAB, user have to create a file that contain the data and add it to the path. In my case, Metocean Data file for one coordinate were created in CSV format. Some coding need to be done to retrieve the data and run the analysis using MapReduce. MATLAB provides a slightly different implementation of the MapReduce technique with the MapReduce function.

MapReduce uses a datastore to process data in small chunks that individually fit into memory. Each chunk goes through a Map phase, which formats the data to be processed. Then the intermediate data chunks go through a Reduce phase, which aggregates the intermediate results to produce a final result. The Map and Reduce phases are encoded by *map* and *reduce* functions, which are primary inputs to MapReduce. There are endless combinations of map and reduce functions to process data, so this technique is both flexible and extremely powerful for tackling large data processing tasks.

MapReduce lends itself to being extended to run in several environments. The utility of the MapReduce function lies in its ability to perform calculations on large collections of data. Thus, MapReduce is not well-suited for performing calculations on *normal* sized data sets which can be loaded directly into computer memory and analysed with traditional techniques. Instead, use MapReduce to perform a statistical or analytical calculation on a data set that does not fit in memory.

Each call to the map or reduce function by MapReduce is independent of all others. For example, a call to the map function cannot depend on inputs or results from a previous call to the map function. It is best to break up such calculations into multiple calls to MapReduce.

4.2.1 MAPREDUCE ALGORITHM PHASES

MapReduce moves each chunk of data in the input datastore through several phases before reaching the final output. The following figure outlines the phases of the algorithm for MapReduce.

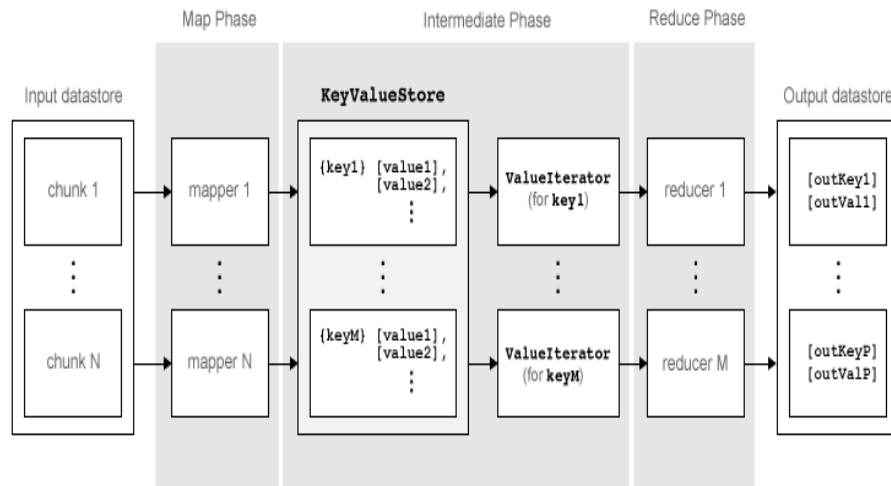


Figure 4.8: MapReduce algorithm

The algorithm has the following steps:

1. mapreduce reads a chunk of data from the input datastore using `[data,info] = read(ds)`, and then calls the map function to work on that chunk.
2. The map function receives the chunk of data, organizes it or performs a precursory calculation, and then uses `thead` and `addmulti` functions to add key-value pairs to an intermediate data storage object called a `KeyValueCollection`. The number of calls to the map function by mapreduce is equal to the number of chunks in the input datastore.
3. After the map function works on all of the chunks of data in the datastore, mapreduce groups all of the values in the intermediate `KeyValueCollection` object by unique key.
4. Next, mapreduce calls the reduce function once for each unique key added by the map function. Each unique key can have many associated

values. `mapreduce` passes the values to the reduce function as a `ValueIterator` object, which is an object used to iterate over the values. The `ValueIterator` object for each unique key contains all the associated values for that key.

5. The reduce function uses the `hasnext` and `getnext` functions to iterate through the values in the `ValueIterator` object one at a time. Then, after aggregating the intermediate results from the map function, the reduce function adds final key-value pairs to the output using `thadd` and `addmulti` functions. The order of the keys in the output is the same as the order in which the reduce function adds them to the final `KeyValueStore` object. That is, `mapreduce` does not explicitly sort the output.

4.3 MAPREDUCE PLATFORM/OPTION

MATLAB has numerous capabilities for exploring and analyzing big data sets. Among them is MapReduce, a powerful, and established programming technique for applying filtering, statistics and other general analysis methods to big data.

The MapReduce functionality built into MATLAB enable user to analyze data that does not fit into memory. By running your MapReduce based algorithms in parallel (using Parallel Computing Toolbox), the processing resources on the desktop can be utilize better without changing any of the algorithms.

To analyze data in MATLAB using MapReduce:

- Specify the data that is going to be analyzed using `datastore`
- Create map and reduce functions in MATLAB
- Execute map and reduce functions using `mapreduce`

While MATLAB MapReduce is optimized for array-based analysis, it is fully compatible with Hadoop MapReduce.

4.4 CODE SAMPLE

In this section, there will be code sample for interface, the mapper and reducer function.

4.4.1 INTERFACE CODE

```
function varargout = metoceanInterface(varargin)
% METOCEANINTERFACE MATLAB code for metoceanInterface.fig
%   METOCEANINTERFACE, by itself, creates a new METOCEANINTERFACE or raises the
existing%   singleton*.
%   H = METOCEANINTERFACE returns the handle to a new METOCEANINTERFACE or the
handle to
%   the existing singleton*.
%
%   METOCEANINTERFACE('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in METOCEANINTERFACE.M with the given input arguments.
%
%   METOCEANINTERFACE('Property','Value',...) creates a new METOCEANINTERFACE or
raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before metoceanInterface_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to metoceanInterface_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help metoceanInterface

% Last Modified by GUIDE v2.5 17-Aug-2015 00:23:04

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @metoceanInterface_OpeningFcn, ...
'gui_OutputFcn',  @metoceanInterface_OutputFcn, ...
'gui_LayoutFcn',  [], ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before metoceanInterface is made visible.
function metoceanInterface_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to metoceanInterface (see VARARGIN)
```

```

% Choose default command line output for metoceanInterface
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

%function bgroup1_SelectionChangeFcn(hObject, eventdata, handles)

set(handles.t1,'String','')
set(handles.t2,'String','')
set(handles.t3,'String','')
set(handles.t4,'String','')
set(handles.t5,'String','')
set(handles.t6,'String','')
set(handles.t7,'String','')
set(handles.t8,'String','')
set(handles.t9,'String','')
set(handles.t10,'String','')
set(handles.t11,'String','')
set(handles.t12,'String','')
set(handles.t13,'String','')
set(handles.t14,'String','')
set(handles.t15,'String','')
set(handles.t16,'String','')
set(handles.t17,'String','')

% --- Executes on key press with focus on pushbutton2 and none of its controls.
function pushbutton2_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTROL)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)

% --- Executes when selected object is changed in bgroup1.
function bgroup1_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in bgroup1
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
met = get(handles.bgroup1,'SelectedObject')
out1 = get (met,'String')

switch out1
    case 'Maximum'
%-----
%maximum value Wind Speed
ds.SelectedVariableNames = 'WS';
type maxMetDataMapperWS.m
type maxMetDataReducerWS.m
maxDataWS = mapreduce(ds, @maxMetDataMapperWS, @maxMetDataReducerWS);

%-----
b = readall(maxDataWS)
set(handles.t2,'String', b.Value)

    case 'Minimum'

%minimum value Wind Speed
ds.SelectedVariableNames = 'WS';
type minMetDataMapperWS.m
type minMetDataReducerWS.m
minDataWS = mapreduce(ds, @minMetDataMapperWS, @minMetDataReducerWS);

```

```

%-----
am = readall(minDataWS)

set(handles.t1,'String', am.Value)

    case 'Mean'

%mean value Wind Speed
ds.SelectedVariableNames = 'WS';
type meanMapperWS.m
type meanReducerWS.m
meanDataWS = mapreduce(ds, @meanMapperWS, @meanReducerWS);

%-----

a1 = readall(meanDataWS)

    case 'Maximum documentation'
        %
        publish('MaximumValue.m','pdf');
        winopen('html/MaximumValue.pdf');
    case 'Minimum documentation'
        %
        publish('MinimumValue.m','pdf');
        winopen('html/MinimumValue.pdf');
    case 'Mean documentation'
        %
        publish('MeanValue.m','pdf');
        winopen('html/MeanValue.pdf');
    case 'Mean by group documentation'
        %
        publish('meanByGroup.m','pdf');
        winopen('html/meanByGroup.pdf');
    case 'None'
        %
end

```

4.4.2 MAPPER CODE SAMPLE

```

function maxMetDataMapperWS (data, info, intermKVStore)
partMax = max(data.WS);
add(intermKVStore, 'WindSpeedMaximum',partMax);

function minMetDataMapperWS (data, info, intermKVStore)
partMax = min(data.WS);
add(intermKVStore, 'WS',partMax);
function meanGroupMapperWS(data, ~, intermKVStore)

dat = data.WS;
day = data.MM;
notNaN = ~isnan(dat);
day = day(notNaN);
dat = dat(notNaN);

% find the unique days in this chunk
[intermKeys,~,idx] = unique(day, 'stable');

% group delays by idx and apply @grpstatsfun function to each group
intermVals = accumarray(idx,dat,size(intermKeys),@countsum);
addmulti(intermKVStore,intermKeys,intermVals);

function out = countsum(x)
n = length(x); % count

```



```

s = sum(x); % mean
out = {n, s};

function meanMapperWS (data, info, intermKVStore)
data(isnan(data.WS),:) = [];

% Record the partial counts and sums and the reducer will accumulate them.
partCountSum = [length(data.WS), sum(data.WS)];
add(intermKVStore, 'PartialCountSum',partCountSum);

```

4.4.3 REDUCER CODE SAMPLE

```

function minMetDataReducerWS(intermKey, intermValIter, outKVStore)
minVal = inf;
while hasNext(intermValIter)
minVal = min(getnext(intermValIter), minVal);
end
% The key-value pair added to outKVStore will become the output of mapreduce
add(outKVStore, 'WS Minimum',minVal);

function meanGroupReducerWS(intermKey, intermValIter, outKVStore)

n = 0;
s = 0;

% get all sets of intermediate results
while hasNext(intermValIter)
intermValue = getNext(intermValIter);
n = n + intermValue(1);
s = s + intermValue(2);
end

% accumulate the sum and count
mean = s/n;
% add results to the output datastore
add(outKVStore,intermKey,mean);

function meanReducerWS(intermKey, intermValIter, outKVStore)

% intermKey is 'PartialCountSumDelay'
count = 0;
sum = 0;
while hasNext(intermValIter)
countSum = getNext(intermValIter);
count = count + countSum(1);
sum = sum + countSum(2);
end

mean = sum/count;
% The key-value pair added to outKVStore will become the output of mapreduce
add(outKVStore, 'MeanWS',mean);

function maxMetDataReducerWS(intermKey, intermValIter, outKVStore)
maxVal = -inf;
while hasNext(intermValIter)
maxVal = max(getnext(intermValIter), maxVal);
end
% The key-value pair added to outKVStore will become the output of mapreduce
add(outKVStore, 'Wind Speed Maximum',maxVal);

```

4.5 INTERFACE

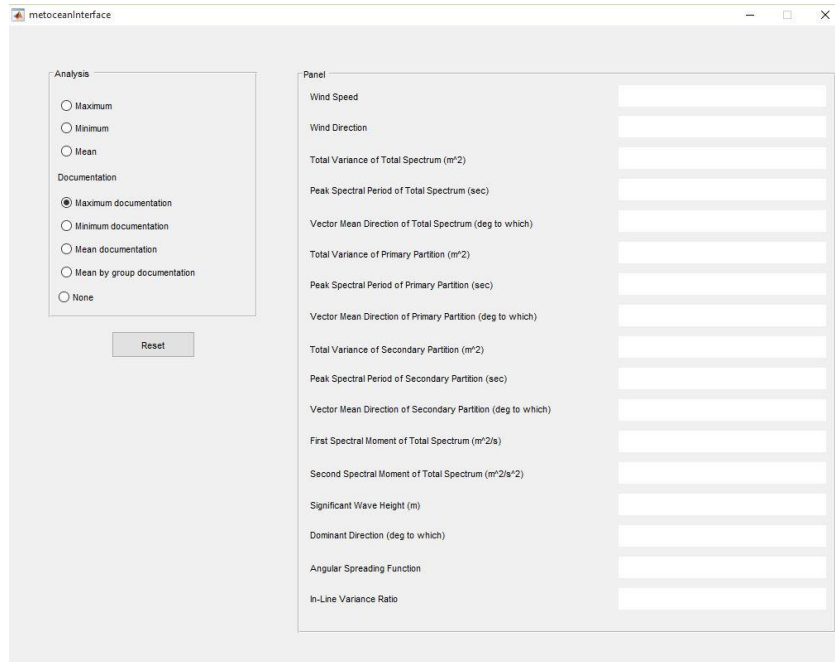


Figure 4.9 : Metocean analysis interface

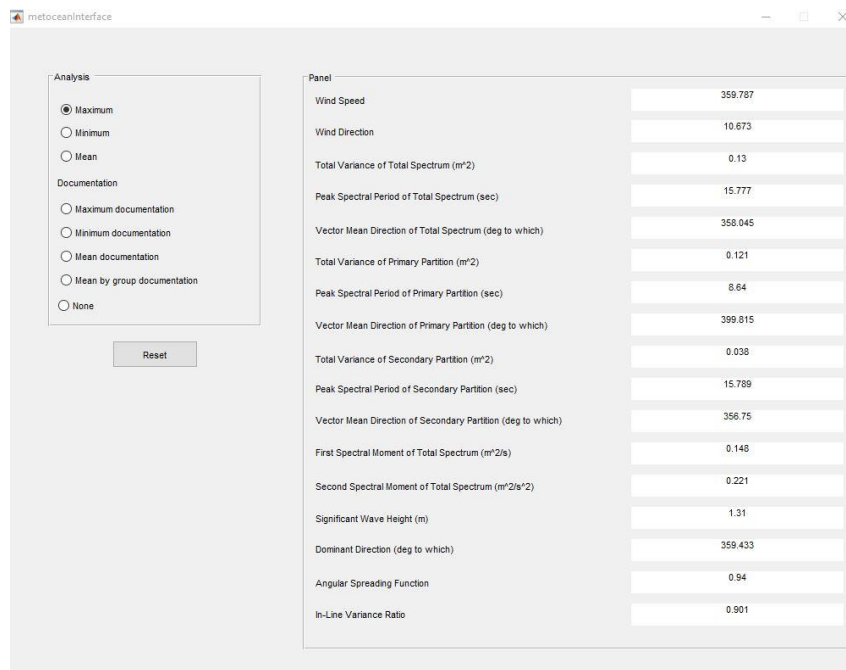


Figure 4.10 : Result when maximum analysis is being selected

```

maxVal = -inf;
while hasNext(interValIter)
    maxVal = max(getnext(interValIter), maxVal);
end
% The key-value pair added to outKVStore will become the output of
mapreduce
add(outKVStore, 'INLINE Maximum', maxVal);

Parallel mapreduce execution on the parallel pool:
*****
*      MAPREDUCE PROGRESS      *
*****
Map  0% Reduce  0%
Map 33% Reduce  0%
Map 66% Reduce  0%
Map100% Reduce 50%
Map100% Reduce100%

a =
      Key      Value
-----
'Wind Direction Maximum' [359.7870]

b =
      Key      Value
-----

```

12

```

      Key      Value
-----
'Wind Speed Maximum'    [10.6730]

c =
      Key      Value
-----
'ETOT Maximum'         [0.1300]

d =
      Key      Value
-----

```

Figure 4.11: Result in pdf when documentation is selected

CHAPTER 5: CONCLUSION AND RECOMMENDATION

Hadoop is an interesting thing to learn. The demand for someone who have knowledge on how to handle big data nowadays are increasing by day. So for people who have knowledge in Hadoop, they have an added advantage. It takes time to learn something new and it can be frustrating sometimes in learning Hadoop. There are times where you arrive at dead end and you need to start over again. But in the end, when things goes right and the expected result were achieved, there are no words that can describe how satisfied you are.

5.1 HORTONWORKS SANDBOX AND MATLAB IN PROCESSING BIG DATA

Hortonworks Sandbox and MATLAB, both tools allow user to process their big data using Hadoop. It is up to individual to choose which environment to use. Based on my experience using both tools, Hortonworks Sandbox is “friendlier” to beginner in Hadoop. Sandbox comes with dozens of tutorials and they did separate the tutorials based on category for the ease of using it. The tutorials categories are for developers, administrators, data scientists and analysts and partner tutorials. MATLAB also comes with their own tutorials on their site; Mathworks. But it is quite hard for someone who are not familiar with MATLAB to understand the tutorial.

Sandbox interface is very attractive and symbols used can be easily understand by the user. For beginner to learn Hadoop it is recommended for them to use Sandbox. But for those who want to play with some coding, they can choose MATLAB.

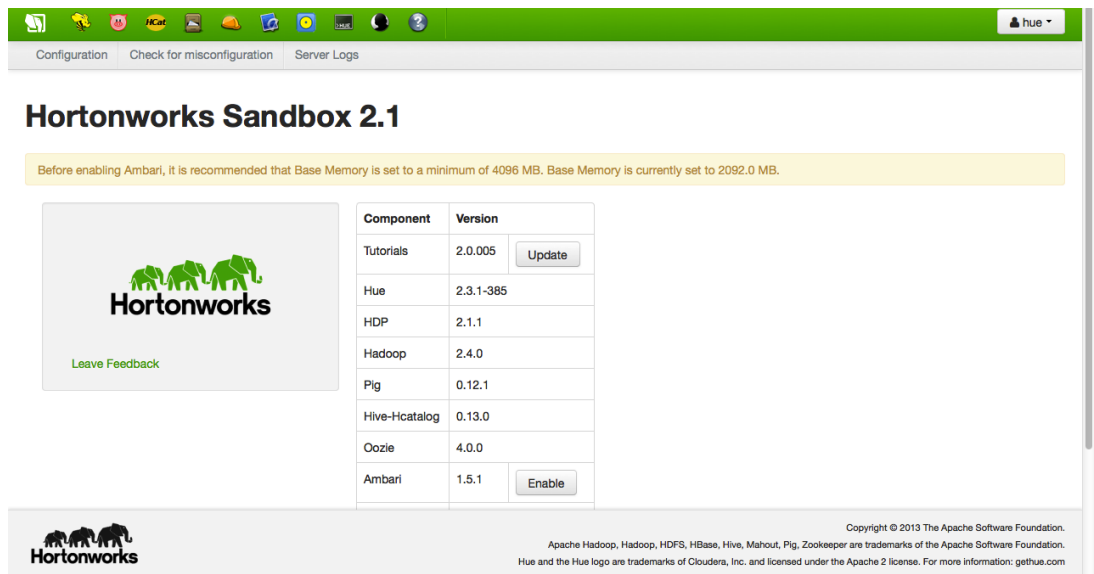


Figure 5.1: Hortonworks Sandbox interface

5.2 HADOOP CHALLENGES

5.2.1 TALENT GAP

Hadoop is a relatively new technology. It is quite difficult to find an entry-level programmers that have sufficient Java skills to be productive with MapReduce. The providers are trying their best to place SQL technology on Hadoop because of the talent gap that exist. It is easier to find those with SQL skills rather than MapReduce.

5.2.2 DATA SECURITY ISSUE

Even though new technologies and tools are surfacing, another challenge centers on the fragmented data security issues in Hadoop. A great step to

make Hadoop a secured environment is the Kerberos authentication protocol. Hadoop does not have the easy to use, full feature tools for data management, data cleansing and metadata. The most important tools that are lacking are standardization and data quality.

5.2.3 INEFFICIENCY

MapReduce is not the best solution for all problems. For simple request for information and problems that can be break into each independent unit, Hadoop can be considered good. But the problem is, it can be inefficient when dealing with iterative and interactive analytic tasks. MapReduce is file-intensive. Iterative algorithms will require multiples map-shuffle or sort reduce phases to be completed because nodes don't intercommunicate except through sorts and shuffles. This will create multiples files between the MapReduce phases and will result in inefficiency for advanced analytic computing.

Hadoop based analytic complexity grows as data mining, predictive modeling and advanced statistics become the norm. Usage growth is driving the need for more analytical sophistication.

Hadoop's framework brings a new set of challenges related to the compute infrastructure and underlined network architectures. As Hadoop graduates from pilots to a mission critical component of the enterprise IT infrastructure, integrating information held in Hadoop and in Enterprise RDBMS becomes imperative.

Finally, adoption of Hadoop in the enterprise will not be an easy journey, and the hardest steps are often the first. Then, they get harder. Weaning the IT organizations off traditional DB and EDW models to use a new approach can be compared to moving the moon out of its orbit with a spatula... but it can be done.

REFERENCES

- Definition-of. (n.d.) *Metocen*. Retrieved February 2, 2015, from Definition –of website: <http://www.definition-of.com/Metoccean>
- Preimesberger, C., (2013) Retrieved February 5, 2015, from <http://www.eweek.com/database/slideshows/nosql-database-deployments-10-real-world-examples>
- Dash, J., (2013) Retrieved February 10, 2015 from <http://www.zdnet.com/article/rdbms-vs-nosql-how-do-you-pick/>
- Pimentel, S., (2014) Retrieved February 10, 2015 from <http://blog.foundationdb.com/5-things-to-consider-when-choosing-database-technology>
- Osborn, T., (2013) Retrieved February 10, 2015 from <http://sapinsider.wispubs.com/Assets/Articles/2013/April/5-Factors-To-Consider-When-Selecting-A-Relational-Database>
- Heng, C., (n.d.) Retrieved February 10, 2015 from <http://www.thesitewizard.com/faqs/what-is-mysql-database.shtml>
- (n.d.) *Data volatility*. Retrieved February 12, 2015 from http://www.its.blrdoc.gov/fs-1037/dir-010/_1460.htm
- Digital Ocean. (n.d.) *Understanding SQL and NoSQL Databases and Different Database Models*. Retrieved February 25, 2015 <https://www.digitalocean.com/community/tutorials/understanding-sql-and-nosql-databases-and-different-database-models>
- Laerd Statistics. (n.d.) *Descriptive and Inferential Statistics*. Retrieved March 5, 2015 from <https://statistics.laerd.com/statistical-guides/descriptive-inferential-statistics.php>

Investopedia (n.d.) *Descriptive Statistics*. Retrieved March 12, 2015 from http://www.investopedia.com/terms/d/descriptive_statistics.asp

Scrum Guides (n.d.) *The Scrum Guide*. Retrieved March 20, 2015 from <http://www.scrumguides.org/scrum-guide.html>

Scrum Alliance (n.d.) *Learn about scrum*. Retrieved March 22, 2015 from <https://www.scrumalliance.org/why-scrum>

SAS (n.d.) *Hadoop why it is and why it matters*. Retrieved March 25, 2015 from http://www.sas.com/en_us/insights/big-data/hadoop.html

IBM (n.d.) *What is MapReduce?* Retrieved April 2, 2015 from <http://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/>

MathWorks (n.d.) *Getting started with MapReduce*. Retrieved April 2, 2015 from http://www.mathworks.com/help/matlab/import_export/getting-started-with-mapreduce.html?s_iid=disc_trial_ml_cta1

Search Cloud Computing (n.d.) *Hadoop*. Retrieved April 5, 2015 from <http://searchcloudcomputing.techtarget.com/definition/Hadoop>

Bertolucci, J., (2013) Retrieved April 5, 2015 from <http://www.informationweek.com/big-data/software-platforms/how-to-explain-hadoop-to-non-geeks/d/d-id/899721>

Marr, B., (2014) Retrieved April 25, 2015 from <http://www.smartdatacollective.com/bernardmarr/205456/what-s-hadoop-here-s-simple-explanation-everyone>

Kalakota, R., (2011) Retrieved April 27, 2015 from <https://practicalanalytics.wordpress.com/2011/11/06/explaining-hadoop-to-management-whats-the-big-data-deal/>

Dictionary.com (n.d.) *flexible*. Retrieved April 27, 2015 from <http://dictionary.reference.com/browse/flexibility>

Wikipedia (n.d.) *scalability*. Retrieved May 5, 2015 from <https://en.wikipedia.org/wiki/Scalability>

The free dictionary (n.d.) *economical*. Retrieved May 15, 2015 from <http://www.thefreedictionary.com/economical>

Stack overflow (n.d.) *Difference between Hadoop and NoSQL*. Retrieved May 20, 2015 from <http://stackoverflow.com/questions/12401468/difference-between-hadoop-and-nosql>

Bisciglia, C., (2009) Retrieved May 25, 2015 from <http://blog.cloudera.com/blog/2009/05/5-common-questions-about-hadoop/>

Tutorialspoint (n.d) *SDLC – Waterfall Model*. Retrieved August 22, 2015 from http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm