OPTIMIZING B-TREE SEARCH PERFORMANCE OF BIG DATA SETS

MOHSEN MARJANI

University of Malaya

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

2017

OPTIMIZING B-TREE SEARCH PERFORMANCE OF BIG DATA SETS

MOHSEN MARJANI

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

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

2017

University of Malaya

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: MOHSEN MARJANI

Registration/Matric No: WHA110067

Name of Degree: Doctor of Philosophy

Title of Project Paper/Research Report/Dissertation/Thesis (―Optimizing B-Tree Search Performance of Big Data Sets"):

Field of Study: Big Data (COMPUTER SCIENCE)

I do solemnly and sincerely declare that:

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

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

University of Malaya

ABSTRACT

Many applications continuously produce large amounts of various data every day, which exceeds the limit of conventional data storage tools. Such data typically includes a large amount of data with different formats that becomes very difficult to query using traditional indexing technologies. Indexing is used for data retrieval to improve efficiency and accuracy of the results of queries. However, current indexing techniques have low efficiency and poor real-time performance in an actual query when involving big data. Also, current indexing techniques are not supporting all characteristics of big data and they have weaknesses when they have to index a variety of data along with high velocity and volume. B-tree indexing technique is one of the most popular techniques that is used by many database systems including the one that can handle big datasets. Every time search process is running against indexed data using B-tree technique, the process traverses all left child nodes of a node to find lowers values or traverses the right side child nodes for finding bigger values. Repetition of search tasks for later queries with same or overlap conditions causes repeating same algorithmic traverse and consuming same resources including time and computation power in order to retrieve the result of the search process. This study proposes an optimized B-tree search method to improve the execution time of the search tasks and to optimize the performance of the B-tree search process. In this new method, every node has a new element storing a min-max summarization which helps search process checks availability of the value inside the sub-tree of the node, then start traversing it to find the location of the value. However, during every search task, a history value is added to every traversed node to mark the history of last search operation to be used for next search operation. The results of the experimental

University of Malaya

analysis show that our new proposed search method decreases the execution time of the search tasks and it improves the search performance several times better than B-tree search performance for same query and same dataset. Moreover, the history value improves the performance of the later queries up to 52%. This research contributes in optimizing data retrieval for big data sets and gives direction to researchers towards a novel approach of indexing and searching big data in order to improve query processing and search performance.

University of Malaya

ABSTRAK

Banyak aplikasi secara berterusan menghasilkan sejumlah besar pelbagai data setiap hari, yang melebihi had alat penyimpanan data konvensional. Data tersebut biasanya termasuk sejumlah besar data dengan format yang berbeza yang menjadi sangat sukar untuk pertanyaan menggunakan teknologi pengindeksan tradisional. Pengindeksan digunakan untuk mendapatkan semula data untuk meningkatkan kecekapan dan ketepatan hasil pertanyaan. Walau bagaimanapun, teknik pengindeksan semasa mempunyai kecekapan yang rendah dan prestasi masa nyata yang lemah dalam pertanyaan sebenar apabila melibatkan data yang besar. Juga, teknik pengindeksan semasa tidak menyokong semua ciri-ciri data besar dan mereka mempunyai kelemahan apabila perlu mengindeks pelbagai data bersama-sama dengan hadlaju tinggi dan jumlah teknik pengindeksan B-pokok adalah salah satu teknik yang paling popular yang digunakan oleh banyak sistem pangkalan data termasuk satu yang boleh mengendalikan set data yang besar. Setiap kali proses carian berjalan terhadap data berindeks menggunakan teknik B-pokok, proses merentasi semua nod anak di sebelah kiri untuk mencari nilai-nilai yang lebih rendah atau merentasi nod anak di sebelah kanan untuk mencari nilai-nilai yang lebih besar. Pengulangan tugas carian untuk pertanyaan seterusnya dengan syarat yang sama atau bertindih menyebabkan pengulangan traverse algoritma sama dan memakan sumber yang sama termasuk masa dan kuasa pengiraan untuk mendapatkan hasil daripada proses carian. Kajian ini mencadangkan satu kaedah carian B-pokok yang dioptimumkan untuk meningkatkan masa pelaksanaan tugas mencari dan untuk mengoptimumkan prestasi proses pencarian B-pokok. Dalam kaedah baru ini, setiap nod mempunyai elemen baru menyimpan satu rumusan min-max yang membantu mencari sekiranya ada nilai dalam sub-pokok nod, kemudian mula

University of Malaya

merentasinya untuk mencari lokasi nilai tersebut. Walau bagaimanapun, semasa setiap tugas carian, nilai sejarah ditambah kepada setiap nod dilalui untuk menandakan operasi mencari terakhir yang digunakan untuk operasi mencari akan datang. Keputusan analisis eksperimen menunjukan yang kaedah carian baru dicadangkan mengurangkan masa pelaksanaan tugas carian dan ia meningkatkan prestasi carian berbanding prestasi carian B-pokok untuk pertanyaan yang sama dan set data yang sama. Selain itu, nilai sejarah meningkatkan prestasi pertanyaan terkemudian sehingga 52%. Kajian ini menyumbang dalam mengoptimumkan semula data untuk set data yang besar dan memberikan arahan kepada penyelidik ke arah pendekatan pengindeksan dan pencarian data besar yang novel untuk meningkatkan pemprosesan pertanyaan dan prestasi pencarian.

University of Malaya

ACKNOWLEDGEMENTS

I wish to give my gratitude to the almighty Allah for giving me the opportunity to complete the thesis. My sincere appreciation goes to my supervisors, Prof. Dr. Abdullah Bin Gani and Dr. Fariza Hanum Binti Md Nasaruddin, Faculty of Computer Science and Information Technology for taking their time to guide and thoroughly go through each and every line of the thesis despite their tight schedules. I believed the constructive comments of my supervisors have significantly improved the quality of the thesis which could have not being so without their inputs.

Special thanks to my parents and family who supported me during this journey. The words are not capable enough to represent my appreciations to them.

Last but not least, great thanks to all my friends especially Dr. Ibrahim Abaker Targio Hashem who helped and supported me to fulfill all the requirements of my Ph.D. journey.

University of Malaya

TABLE OF CONTENTS

University of Malaya

University of Malaya

University of Malaya

University of Malaya

LIST OF FIGURES

University of Malaya

University of Malaya

University of Malaya

LIST OF TABLES

University of Malaya

University of Malaya

LIST OF SYMBOLS AND ABBREVIATIONS

IT	:	Information Technology
IDC	:	International Data Corporation
DBMS	:	Database Management System
CPU	:	Central processing unit
IEEE	:	Institute of Electrical and Electronics Engineers
ACM	:	Association for Computing Machinery
IoT	:	Internet of things
GPS	:	Global Positioning System
M2M	:	Machine-to-Machine
RFID	:	Radio-frequency identification
DNA	:	Deoxyribonucleic acid
EMR	:	Electronic medical records
MGI	:	McKinsey Global Institute
SQL	:	Structured Query Language
SDS	:	Single document summarization
MDS	:	Multi-document summarization
QS	:	Query-oriented summarization

University of Malaya

LIST OF APPENDICES

University of Malaya

CHAPTER 1: INTRODUCTION

This chapter provides background information on the research work carried out in this thesis. First, the background and motivation to undertake the research are presented. Then, the statement of problem and objective of the research, and proposed methodology are also presented. Finally, the outline of the thesis is highlighted.

The rest of this chapter is as follows. Section 1.1 presents the motivation of this research. Section 1.2 states the identified research problem. Section 1.3 highlights the aim and objectives of the research. The proposed methodology is described in Section 1.4. Finally, Section 1.5 gives a short description of the layout of this thesis.

1.1 Motivations

Every day many applications such as social media, health care, transactional, and banking applications and also many devices such as sensor devices are producing big amount of data with different formats such as text, image, audio, and video. Big Data has become one of the buzzwords in Information Technology (IT) during the last couple of years. Initially it was shaped by organizations which had to handle fast growth rates of data like web data, data resulting from scientific or business simulations or other data sources. Some of those companies‘ business models are fundamentally based on indexing and using this large amount of data. The pressure to handle the growing data amount on the web e.g. leads Google to develop the Google File System and MapReduce (Dean & Ghemawat, 2010).

More than 2.5 quintillion bytes (exabytes) of data are generated every day (Zhou, et al. 2013). 90% of the total data has been created just in the past few years alone. To contain such a massive amount of data, storage continues to grow at an explosive rate (52% per year) (Zikopoulos, et al. 2012). By the end of 2020, the size of the total

University of Malaya

generated data will exceed 35 zettabytes (ZB), which has proven to be too conservative (Reed, et al. 2012). Whether the term is remained big data or not, the Volume and Variety of data, plus Velocity of producing data will always be growing and it is required to index the data to speed up data retrieval along with accuracy of the retrieved data.

Indexing technique plays a key role in data retrieval by allowing database systems to store and retrieve data efficiently based on the users‘ queries. Just a few solutions have found their respective places in a database system and still there is a big need for more efficient and concise indexing structures (Bühlmann, 2013; Hellerstein, Naughton, & Pfeffer, 1995; Idreos, Kersten, & Manegold, 2007; O'Neil & Quass, 1997; Sidirourgos, 2014).

1.2 Statement of Problem

Searching queries in B-tree based indexed data is consuming a lot of time, and computing resources when comparing on keys stored at nodes of the B-tree. These search comparisons are applied against the keys stored in the nodes of the sub-trees under each non-leaf node, even the target is not in the range of the minimum and the maximum values of the sub-trees. Searching for a key in a tree based index structure; the search algorithm has to traverse the tree from root to leaf, making comparisons with keys stored in the nodes of the tree and deciding, based on the comparison, to continue searching in the left or right sub-trees. This process repeats every time even next queries are same or have overlap with earlier queries.

For instance, as system searches for a key in a tree, the process traverses the tree from root to leaves, making comparisons to keys stored in the nodes of the tree and deciding, based on the comparison, to continue searching in the left or right sub-trees.

University of Malaya

This process repeats every time even next queries are same or have overlap with earlier queries. Therefore, repetitive queries are consuming same amount of time, energy, memory, CPU power, and other resources. And also, the time and other resources that are used for earlier queries are not used for later queries even though they are the same or they have overlaps. In addition to rapidly growing volume of data, the response time for data retrieval and needed resources such as memory, CPU power, and energy are increasing.

1.3 Statement of Objectives

This research aims to optimize resource consumption in data retrieval for big data by using query summarization concept and reusability approach. In order to achieve our aim, we seek undertaking the following steps.

- To study the current big data indexing techniques and identify the key issues with respect to B-tree search method.
- To investigate the problem of the current B-Tree search method.
- To propose a solution to optimize the performance of the B-Tree search method for big data sets.
- To evaluate the performance of the proposed search method by validating it with the performance of the B-tree search method.

1.4 Proposed Research Methodology

Our proposed methodology for this research work is based on six phases as follows. The first phase is review stage in which we review literature and credible studies related to big data and big data indexing structures and techniques to achieve insight into the area of big data and the methods and technologies that are proposed by previous researchers. In this phase, recent publications from online scholarly databases such as

University of Malaya

ACM, IEEE, Elsevier and web of science are collected to have trustworthy of the literature.

In the second phase, the investigation of the research carried out. We explore big data indexing requirements and investigate big data indexing techniques to determine significant shortcomings and weaknesses of the current indexing methods. A simulation engine is created to simulate data retrieval process by applying different pre-defined queries on different sizes of datasets and we monitor resource consumption and particularly the response time of each attempt. Then we analyze the trend of the results analytically in order to find out how the search method can be optimized for big data sets and how earlier queries results can effect on later queries. At the end of this phase, a benchmarking is used to validate the result of the analytical analysis and also demonstrate the significance of our research problem.

The third phase is a proposal in which a search method based on B-tree search method by using summarization concept and also reusability theory is proposed to address our research problem. This proposal aims to optimize resource consumption in data retrieval by using summarization techniques upon every query processing and reusing the result of the earlier queries to reduce resource consumption specifically execution time of the later queries.

To implement our proposed solution, we design and program a modified indexing structure based on data structure B-tree which is a popular and default data structure used in most of the current databases. To simulate data retrieval by using our implemented model, a simulation is created to generate index structure and index the stored data then to apply four different pre-defined queries on different sizes of a data set and to record the results of resource consumption namely index creation time, index

University of Malaya

size and queries response time. We repeat applying the queries on the data ten times and record every response time.

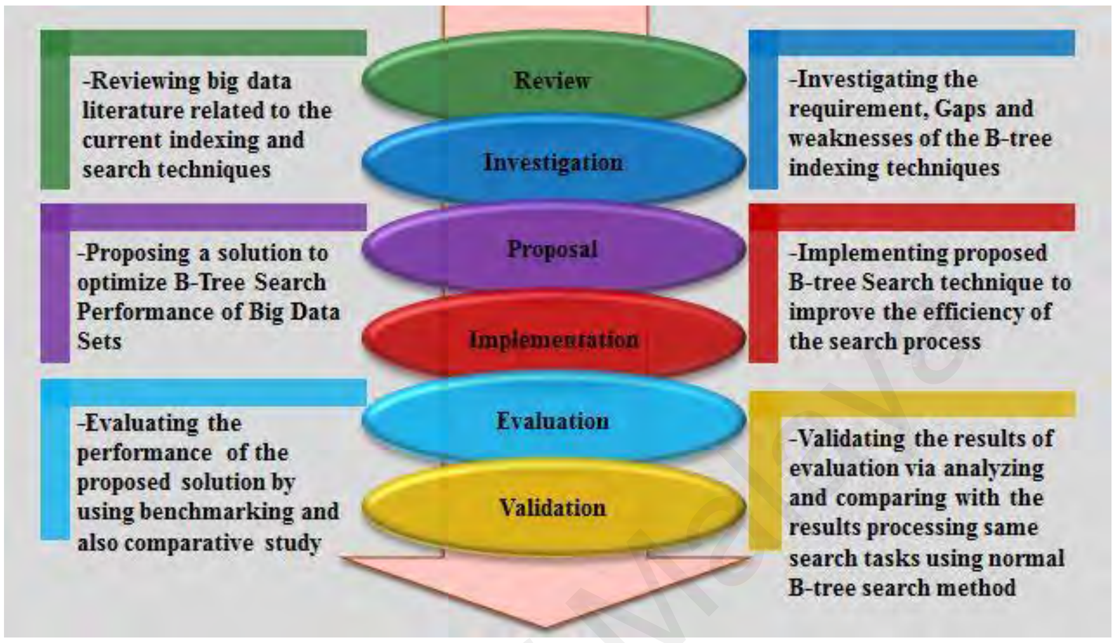
-Reviewing big data literature related to the current indexing and search techniques
Review
-Investigating the requirement, Gaps and weaknesses of the B-tree indexing techniques
Investigation
-Proposing a solution to optimize B-Tree Search Performance of Big Data Sets
Proposal
-Implementing proposed B-tree Search technique to improve the efficiency of the search process
Implementation
-Evaluating the performance of the proposed solution by using benchmarking and also comparative study
Evaluation
-Validating the results of evaluation via analyzing and comparing with the results processing same search tasks using normal B-tree search method
Validation

Figure 1.1: Proposed research methodology stages

In the next phase, we evaluate our proposed model by using benchmarking and also comparative study. Based on Sieve benchmark, we use ten benchmarks to run a performance evaluation for our proposal. The sufficiency of this number of benchmarks for evaluating the performance of computing systems by using Sieve benchmark is already proven (Jain, 2008). For the comparative study, we demonstrate the performance of our proposed model in comparison with the related model, specifically with normal B-tree technique.

In the last phase, we validate the results of the performance evaluation stage via analyzing and comparing the results of execution time of processing search tasks using our proposed search method with the results of execution time of processing same search tasks using normal B-tree search method.

University of Malaya

In this study, the concept of reusability and query summarization were integrated into indexing procedures by creating node base summary of data in order to reuse the resources used for earlier queries and help to minimize the response time of later queries. However, it will optimize data retrieval for big data and give direction to researchers towards the novel approach of indexing for a verity of big data in order to improve query processing.

1.5 Layout of Thesis

This thesis includes 7 chapters as it is illustrated in Figure 1.2. Also, a short summarization of each chapter is presented in Table 1.1. The rest of this thesis is organized as follows:

Chapter 2 reports a detailed review of the state-of-the-art research from literature and provides the previous works that support this study and related big data indexing concepts and techniques. Moreover, it identifies the open problem related to this research. A comparison of the current indexing techniques based on the requirements of big data indexing which are extracted from previous related works is given in the following of this chapter. Also, B-tree and some of the B-tree search method are highlighted. Furthermore, the chapter states some open research challenges and highlights the problem which is addressed in this thesis.

In Chapter 3, the aim is to demonstrate the importance of the identified problem by analyzing the results of benchmarking experiments. This chapter reports how our investigation is conducted and what are the activities in establishing the essence of our research problem. One of the most popular database systems is selected which is capable of dealing with big datasets and using B-tree as default indexing technique for indexing and searching large datasets. Then we apply four different queries on six

University of Malaya

different sizes of a datasets starting for few megabytes to more than one terabyte and repeat this process by a simulation engine for a number of times and analyze resource consumption upon data retrieval mainly response time in order to monitor relation between resource consumption of later queries and earlier queries. Then, we present our findings which support identifying our research problem.

Chapter 4 explains our proposed solution to address the research problem. It reports how our proposed method for solving the problem identified in chapter 3 is modeled. The model which is in a form of simulation, clearly explains our approach. To support our simulation model, we also present a comparative study to prove validity of our solution.

Data collection and the activities to gather data is explained in chapter 5. We stress a comprehensive explanation about what data is collected, how the data is collected and also how the collected data is processed. Moreover, this chapter reports the requirements of the simulation model and presents the results of the benchmarking experiments in the form of several tables.

Chapter 6 elaborates the obtained results from data benchmarking experiments and discusses and interprets the results to highlight the advantages and weaknesses of our proposed solution. To clarify the meaning of the result, we illustrate our points by using tables and graphs and direct the reader to our points by adding more explanations about those diagrams and tables. In this chapter, the results of our evaluation on the performance of our proposed model are presented. These results are collected based on analyzing execution time of processing of four predefined queries with range of simple to complex query using seven sizes of the selected datasets.

University of Malaya

Table 1.1: Overview of chapters and contents reported in this thesis

Chapter	Highlights	Description
Chapter 1 Introduction	Big Data Overview	To present an overview of big data
	B-tree Search	To introduce B-tree technique and its search method
	Statement of Problem	To state the research problem identified for this thesis
	Statement of Objectives	To state the aim of the thesis and objectives to attain the aim
	Research Methodology	To state the steps taken to achieve the aim and objectives
	Thesis Layout	To demonstrate the structure of contents presented in thesis
Chapter 2 Literature Review	Big Data	To introduce big data and its related terms and characteristics
	Big Data Indexing	To introduce indexing techniques for big data and its requirements
	B-tree Indexing Technique	To introduce B-tree technique and its search method
Chapter 3 Problem Analysis	Analysis of the identified issues	To report how the investigation is conducted to establish the research problem.
	Analytical Verification of the identified problem	To verify the research problem by using analysis on the result of applying queries on data set via running benchmarking experiments using a simulation engine
Chapter 4 Proposed Model	Graphical Presentation	To schematically present our model developed for evaluating our proposed method
	Modified Algorithm	To introduce the part of the algorithm that is changed in our method
	System Design	To discuss system design of the proposed method
	Search Query	To describe the process of searching query in the proposed method
Chapter 5 Evaluation	Datasets	To describe the datasets used in this study
	Benchmarking Modelling	To explain the requirement of the designed model for evaluating the performance of the proposed model
Chapter 6 Result & Discussion	Evaluation Results	To present the result of the performance evaluation of the proposed model analytically and graphically
	Validation Results	To report the results of proposed model validation analytically and schematically
	Discussions	To discuss the findings of the evaluation and the validation
Chapter 7 Conclusion	Research Aim and Objectives	To explain the aim and objectives of this research
	Scope and Limitations	To report scope and limitations of this study
	Significance and Contributions	To highlight contributions and importance of this research
	Future Direction	To propose future direction of the research work

University of Malaya

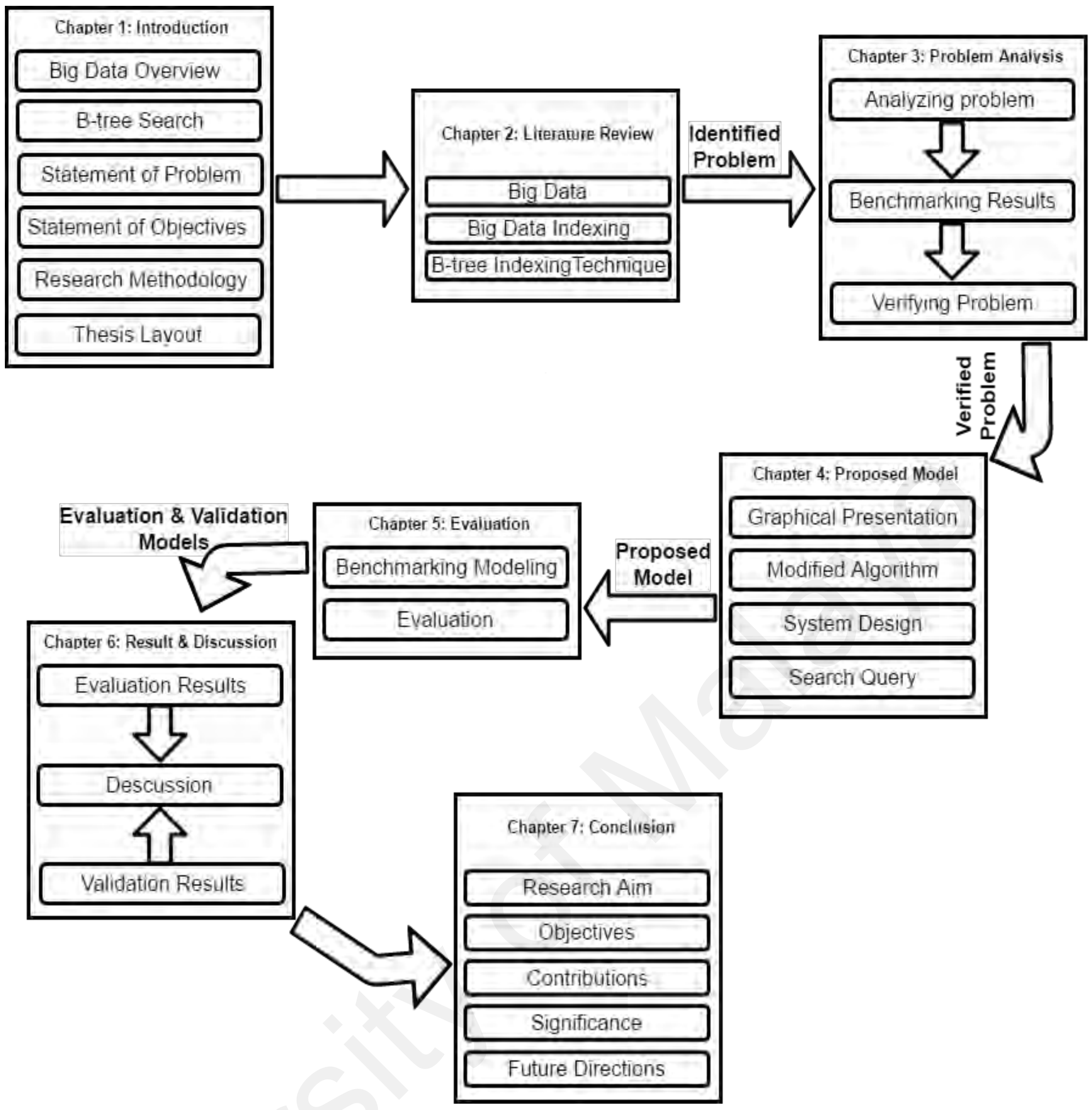

Figure 1.2: Illustration of chapters and contents of this thesis

Every search task is repeated 10 times and the execution time of every search operation is captured. We also validate the result of evaluation by benchmarking analysis. Finally, to show the validity of our proposed method, the benchmarking and comparative analysis results are presented.

In Chapter 7, a conclusion of this thesis describes all efforts and activities to fulfill the aim of our research briefly. This chapter explains how the objectives of this research are done and what are the significance and contributions, strength and weaknesses of our proposed solution.

CHAPTER 2: OVERVIEW OF BIG DATA INDEXING

2.1 Introduction

This chapter offers a review of big data including its characteristics, trends, and challenges with respect to indexing and B-tree based indexing techniques. The main goal is to describe and discuss about big data indexing based on the format of big data, structures and requirements of big data indexing along with its evaluation metrics in data retrieval. However some of the current indexing techniques are analyzed and discussed based on the requirements of big data indexing. Finally, B-tree indexing technique and its mechanism are discussed.

The remainder of the chapter is as follow: Section 2.2 introduces big data and presenting the characteristics, trends, the source of big data and the existing challenges from an indexing perspective. Section 2.3 provides the concept of big data indexing and discusses the structures and requirements of indexing. Section 2.4 introduces B-tree indexing technique and discusses the flow of its search method in data retrieval. Section 2.5 concludes the chapter.

2.2 Big data

As we enter the new era of big data, the continuous increase in computational capacity has recently produced an overwhelming flow of data, thereby exceeding the limit of conventional processing tools. The term ―big data" is mainly used to describe volumes, variety, the velocity of the data (M. Chen, Mao, & Liu, 2014). Such data typically include large amounts of unstructured data formats that are extremely difficult to store, process, and analyze with traditional database technologies. According to McKinsey Global Institute (MGI), *big data refer to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze. This*

University of Malaya

definition is intentionally subjective and incorporates a moving definition of how big a dataset needs to be in order to be considered big data. (James Manyika et al., 2011).

The MGI's definition for ‗big data' put its focus on the size and volume of the data. Based on this notion, there is a misleading and ‗big data' is not a new problem. Handling a large amount of data in a specific situation is an old existing topic of database research and it directs us to the initiation of parallel database systems with ‗shared-nothing' architectures (DeWitt & Gray, 1992.). Hence, there must be more about ‗big data' rather than just the size of data to be considered as ‗big data'. Other discussions in later publications widen the MGI's definition for ‗big data'. For instance, IDC's ‗The Digital Universe' study, has given following definition (Maier, Serebrenik, & Vanderfeesten, 2013a):

IDC defines Big Data technologies as "a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data by enabling high-velocity capture, discovery, and/or analysis. There are three main characteristics of Big Data: the data itself, the analytics of the data, and the presentation of the results of the analytics". (Gantz & Reinsel, 2012).

The above definition uses the 3V's model proposed by Doug Laney in 2001 (Laney, 2011). Although there is no term ‗big data' in Laney's model, but he highlighted an e-commerce trend in which data management will be more difficult and important in future by facing challenges of managing volume, velocity, and variety of data. In 3V's model, data volume refers to the size of data, data velocity means the speed of generating new data, and variety of data refers to different sources of data that can produce structured, semi-structured, or unstructured data. On the other hand, in the academic area, we cannot find such a reliable definition for ‗big data' except using the exactly same definition as Laney's or a modified version of it. For example, ‗big data' is

University of Malaya

defined as"*the amount of data just beyond the technology's capability to store, manage and process efficiently*" in (Stephen Kaisler, Frank Armour, J. Alberto Espinosa, & William Money, 2013b). This research described variety and velocity of data as additional characteristics. Or Sam defined ‗big data' as ―*too big, too fast, or too hard*"(Madden, 2012). Obviously, ‗too big' refers to Volume of data, and ‗too fast' refers to velocity. Also ‗too hard' points to data that is hard to be fit into existing tools and this meaning is very similar to variety of data (Maier et al., 2013a).

2.2.1 Sources of Big Data

Big data has the potential of using different data sources and it is uneasy to integrate and combine in a unified format, which required for analyzing data. However, the rise of social media from several years ago and also the rise of Internet of Things (IoT), and other communication and telecommunication approaches, the public availability and text-centric data sources are rapidly increasing. Data sources such as community pages, blog posts, social networks messages and images, sensors' data, mobile phones and GPS data (Marz & Warren, 2013) accompany the other data resources and make a variety of data types with different formats and structures. For instance, some of the companies need to integrate their transactional sales and customer master data with sentiment analysis of social media sources in order to optimize their marketing activities. Therefore, variety of data is caused by the general diversity of data sources which is resulting in rapid growth of the volume of data as well as structural differences among those sources (Maier, Serebrenik, & Vanderfeesten, 2013b). Big data sources are classified into different categories to better understand their characteristics. (Soares, 2012)categorized resources of big data into five groups including, the web and social media, machine-to-machine data, big transaction data, biometrics and human-generated data as described below:

University of Malaya

(a) Web Data & Social Media: The web is a very diverse source of data and it is valuable for analytics. As the user interest in using social media and web-based tools is growing in recent years, the researcher's interest in social media analysis is increasing as well by focusing on weblogs (Denecke & Nejdl, 2009). Analyzing social media data from weblog communities is one research aspect (Hsu, King, Paradesi, Pydimarri, & Weninger, 2006; Kumar, Novak, Raghavan, & Tomkins, 2005). Web resources such as crawling of web pages, online articles, and blogs are initially developed for human usage (Y. C. Jinchuan Chen, Xiaoyong Du, Cuiping Li, Jiaheng Lu, Suyun Zhao, and Xuan Zhou., 2013).

Information, knowledge, content, public ideas and opinions can be directly extracted from these web resources. But the point is the major part of these resources produces semi-structured and unstructured data such as images and videos along with textual data. Beside it, these resources mostly are connected to each other via hyperlinks and also they may offer some categorization using tag clouds (Hassan-Montero & Herrero-Solana, 2006; Kumarasamy, 2015; Maier et al., 2013a; Sinclair & Cardew-Hall, 2008).

In addition to web-based tools that are designed to serve human, there is another type of web-based content and tools offering machine-readability (Knuth, Waitelonis, & Sack, 2016). Applications can use machine-readable web-based content to access to data and understand the mtovisa their semantics. These capabilities enable applications to discover new information by integrating data from different sources, generating new understandable metadata and publishing them into web pages using data formats from the semantic web standard (Burgos, 2011; W3C, 2013). Whereas the type the data is mostly graph-shape, it can be categorized in semi-structured data group (Maier et al., 2013b).

University of Malaya

The other type web data are semi-structure and captured from navigational activities of users in the web. This type of data is gathered from click streams and logs (Gerken, Bak, Jetter, Klinkhammer, & Reiterer, 2008; Kammenhuber, Luxenburger, Feldmann, & Weikum, 2006) recorded by web applications and tools. The information extracted from this kind of data reveals how users navigate and interact with the web. Thus, the knowledge driven from this type data can benefit companies and firms to optimize their services based on the user‘s behaviors.

By the rise of communicational and social media web-based services, another type of web data is coming to the scene which is so called social media data or communications data and captured from social media sites' interactions and activities in messaging services. The data resulted in activities such messaging and status updates through these tools and services typically include unstructured text and images. However, there is another type data created by using social media sites and messaging services representing the relationships and connections among users. This type of data is mostly graph-shape and therefore it is categorized in the semi-structured data group and it is so-called social graph data or social network data (Ko, Cheek, Shehab, & Sandhu, 2010).

As it is mentioned above, there are different types of data that can be categorized as web data and social media data, but they might have several overlaps. Generally,it can be claimed that any data resulted by doing web-based activities, messaging communications and social media interactions which are mostly semi-structured and unstructured, can be categorized into the web and social media data.

(b) Machine-to-Machine Data: Machine-to-Machine data include any data generated or exchanged in a Machine-to-Machine (M2M) communication, which refers to communication between electronic devices or systems via any wired or wireless communication channel without human interposition. The main aim of this type of

University of Malaya

communications is sharing information among those systems autonomously (Booysen, Gilmore, Zeadally, & Van Rooyen, 2012; Niyato, Xiao, & Wang, 2011). This ability helps to control remote and distributed mobile system autonomously (Booysen et al., 2012). Hence, M2M communication especially the wireless type of it, is gaining much more attention in both industry and academia and it is recently deployed in the eras such as home networking, vehicular networking, smart grid, health care, etc. (Chan, 2011; H.-C. Chen, Wu, Sun, & Feng, 2016; de Mattos & Gondim, 2016; Lu, Li, Liang, Shen, & Lin, 2011; Yan Zhang et al., 2011).

M2M communications often are useful for a remote sensors system that needs to transmit data to a data center or to a central system. The data is mostly related to measurements of properties such as temperature or movements of physical objects. These objects can be Radio-frequency identification (RFID) chips that automatically identify and track attached tags to the objects by using electromagnetic fields (Zhu, Khullar, & Chi, 2016) or they can be sensors or Global Positioning System (GPS) receivers (Hoffman, Hoffman, Hoffman, & Doe, 2016).

For instance, to bring intelligence concept into human life environment, some sensors can be embedded in smart meters and cars for technical diagnosis. But the volume of data generated by this kind of systems can be huge. For example, the BMW group predicts this volume of data that come from its cars using such a system can be about one petabyte per day in 2017 (Sergey Melnik & Tolton, 2011). As it in mentioned by some researchers (Ashton, 2009; de Mattos & Gondim, 2016; Paul & Rho, 2016; Seo, Jeon, Lee, & Lee, 2016)(Alam, Nielsen, & Prasad, 2013), Internet of Things (IoT) is one of the main approaches that are going to become a major stream technology uses M2M technology and rapidly will produce large amount of data. The data generated by M2M technology is typically semi-structured (Maier et al., 2013b).

University of Malaya

(c) Transactional data: Every day, many transactions data are generated due to very large number of deals, agreements, purchases, payments, etc. It can be a purchase item via e-commerce systems, a payment transaction of a credit card, or a call detail captured by a telecommunication company. A mush as the dimensions of a system grow up, the transactional data become bigger as well to record those data and another amount of operations conducted by the system (Jinchuan Chen et al., 2013).

Transactional data typically are structured or semi-structured. But sometimes big transactional data can be unstructured data. For example, the data recorded by an insurance agent about an accident can include some personal notes and description about the accident, or in a health care transaction, a doctor may add some diagnosis notes and treatment description to the data (Goswami & Chandra, 2015).

*(d) Biometrics Data:*Any human physiological or behavioral characteristics can be considered as biometrics and it can be universal which every person should have it or it can be uniquely for only one person (Jain, Bolle, & Pankanti, 2006). Biometrics data generally describe biological organism of human or sometimes non-human individuals to be used in identifying idiosyncratic anatomical, behavioral characteristics, traits, and properties (Nagpal, Nagpal, & Malhotra, 2012).

In computer science, biometric authentication or biometric identification such as fingerprints, and retinal scan can be used for access control (Wayman, Jain, Maltoni, & Maio, 2005). However, it can be used for identifying individuals in under surveillance groups (Hu, Tan, Wang, & Maybank, 2004). Deoxyribonucleic acid (DNA), handwriting, and other behavioral characteristics can be analyzed as biometric identification and related data can be in a textual format, image, video, etc. Therefore, biometrics data can be structured, semi-structured, or unstructured. Scientific

University of Malaya

applications use large amounts of biometric data for genomic analysis (Maier et al., 2013b).

(d) Human-generated data: As it understandable from the title of this section, all data generated by human are so called human-generated data such as emails, paper documents, Electronic medical records (EMRs) which are excellent improvement over paper records, physicians note, etc. (Cottle et al., 2013; Maier et al., 2013a; Soares, 2012). Whereas the data produced by human have diverse formats, human-generated data can be structured, semi-structured, or unstructured. This type of data has clear overlaps with another type of data such as web data and transaction data.

Table 2.1 gives a summary on the typical categorization of the five different sources of big data based on their data structures.

Table 2.1: Data type categorization of five big data sources

Big Data Source	**Data Type Categories**		
	Structured	**Semi-Structured**	**Unstructured**
Web Data & Social Media		√	√
Machine-to-Machine Data		√	
Transactional Data	√	√	
Biometrics Data	√	√	√
Human-generated data	√	√	√

2.2.2 Characteristic of Big Data

The terms ―Volume", ―Variety" and ―Velocity" as shown in Figure 2.1 are originally introduced by Gartner‘s Doug Laney 2001 to describe elements of big data challenges. These three attributes shape the three ―Vs".

(1) Volume means a large amount of all types of data generated from different sources and continue to expand. The benefit of gathering a large amount of data is a way in which it can be used to create value through data analysis. Laurila et al. (2012), provide a unique collection of data longitudinal from the smart mobile device in order

University of Malaya

to be available to the research community. This initiative was called mobile data challenge which motivated by Nokia. Whilst, the authors point out that collecting such data requires a great deal of effort, as well as underlying investments even though, mobile data challenging, addressed an interesting result similar to examinations of predictability of human behavior patterns or away to share data based on human mobility and visualization techniques for complex data.

(2) Variety refers to different types of data collected via sensors, Smartphones or social networks, such as video, image, text, audio, data logs, etc., whether this data is in structured or unstructured format. Most of the data generated through mobile apps are unstructured data. For example text messages, online games, blogs, and social media generate different types of unstructured data through mobile devices and sensors. On the other hand, people on the internet also generate an extremely diverse set of structured and unstructured data

University of Malaya

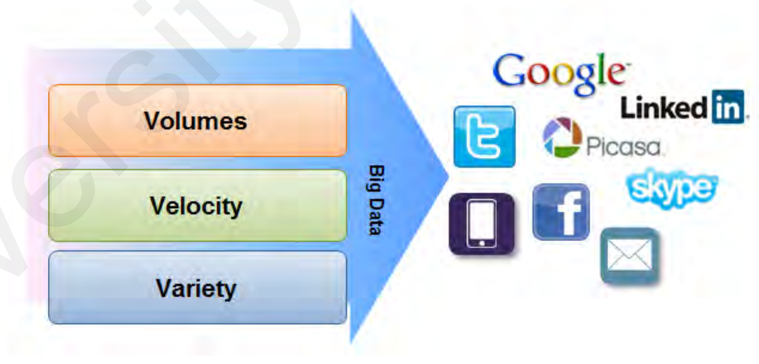

Figure 2.1: 3Vs of Big Data

(3) Velocity refers to the speed of the data being transferred. The contents of the data are constantly changing, through the absorption of complementary data collections, through the introduction of previously data or legacy collections, and the form streamed data arriving from multiple sources (Berman, 2013).

2.2.3 Big Data Trends

Many important future challenges in managing and analyzing big data arise from the nature of big data as it has a large volume, diverse formats and high velocity (V. Gopalkrishnan, D. Steier, H. Lewis, & Guszcza, 2012). Rapid growth of the volume of data requires more infrastructures and faster technologies to store and analyse generating data in order to extract useful information for decision making. Moreover, current aging equipment and technologies which are still used by data centres and other enterprises and companies that are dealing with large data sets, must be updated by new hardware and technologies.

Many applications such as social-network-based recommenders, business process optimizers, healthcare analyzers require efficient ability of analyzing big datasets. It is predicted that the growth of the volume of generated data significantly will effect on the cost and massiveness of storage technologies, required computational power for processing data, and energy consumption as well. For instance, the traffic of data between years 2002 to 2009 was grown 56 times and computation power was increased 16 times. In the other side, the size of data centers was enlarged by a rate of 173% per year from year 1998 to year 2005. Based on these trends, within about 13 years, computing power will increase about 1000 times. However, over the same period of time, this is not expected that energy efficiency to gain a growth rate by a factor of more than 25. These statistics show that an intense mismatch of about 40 times rise in the energy consumption for data analytics processes will be generated. (Kambatla, Kollias, Kumar, & Grama, 2014).

2.2.4 Big Data Challenge

Big data is passing through its early stage of growth. Since the present big data methods and techniques are very less to solve the real big data challenges completely.

University of Malaya

Big data in computational sciences has always been a critical issue (Wing, 2008), and is still of increasing research interest (NTER & MILL, 2012). Researchers have not yet unified around the essential landscapes of big data.

According to some researchers and practitioners, big data is the data that we are not able to process using current technologies, methods, and theories. Consequently, the world is becoming a helplessness age due to incalculable data being generated by science, business, and society. Particularly the issue of data retrieval, as procedures and standard tools are not designed to search large unstructured datasets. Once dealing with huge datasets, organizations face teething troubles in being able to create, manage and manipulate this rapid growing tsunami of information, big data exploding, growing 10X each five years, replication system have security weaknesses, multiple copies equals multiple everything, data governance and policy challenges for defining the data that will be stored, analyzed, and accessed, along with determining its relevance. For example, information such as entities and relationships can be extracted from textual data by using some available technologies in eras of text mining, machine learning, natural processing, and information extraction. But it is required to develop technologies to extract images, videos, and other information from other non-text formats of unstructured data (Divyakant Agrawal, Franklin, Labrinidis, & Kenneth Ross, 2012).

A general challenge of data integration and required techniques is highlighted in a long-lasting research endeavor regarding data integration (Rakesh Agrawal et al., 2009). The challenge shows up when it is required to integrate different sources of textual data, some extractors are used to make all textual data from those resources structured. They need to be harmonized and converted to some schema or structure that are usable for discovering connections among those textual data come from different sources. It is also

University of Malaya

expected that text mining will typically be conducted by applying several specialized extractors on the same text. Hence, managing and integrating different results of the extractions from a certain data source need another technique (Rakesh Agrawal et al., 2009).

Data integration and information extraction can be conducted in two points. The first point is before analysis process where pre-processing tasks are running in which unstructured data will be transformed into structured or semi-structured data, then information extraction and data harmonization will be conducted and the results will be stored in a graphical store or in an RDBMS. This method improves and optimizes the runtime of the analysis process. In contrast, some information might be lost because the information extractors that are used in pre-processing step only store the information that they are built for. Therefore, there might be some valuable information that is not extracted by those extractors' techniques (Maier et al., 2013b).

The second point is conducting information extraction and data harmonization at runtime of the analysis process. This method is more flexible since it can use dedicated extraction techniques befitting for the analysis process. The weak point of this method is worse runtime performance rather than the first method. A solution can be using a combination of both methods in which pre-processing task includes data integration and information extraction. The result will be stored in a structured format and also the original data are kept available and accessible at runtime of the analysis process just in case if the extracted information is inadequate for any analysis task. The advantage of this combined method is preventing both losing valuable data and also insufficient extracted information. In contrast to this improvement, this combined approach causes increasing required storage space (Maier et al., 2013b).

University of Malaya

The other challenge appears in metadata creation process in data transformation and information extraction processes when it is required to give some information to users about the source of data and how it is reliable by tracing the source of data and get metadata included it along with some information about the way that data are recorded, the semantics of data, how it is manipulated in analysis process, the applied information extractors, etc. (Maier et al., 2013b; Rakesh Agrawal et al., 2009). Some of these challenges are discussed in the subsequent below.

(a) Optimal Architecture for Analytics Systems: Analytics systems should have an optimal architecture in order to deal with historical data and real-time data simultaneously. Although there are proposed solutions to address this challenge, yet there is a need to have a better optimal architecture for analytics systems and tools. The Lambda Architecture (N. Marz & Warren, 2013; Namiot, 2015) is one of the proposed architectures to address this challenge. It facilitates simplified business processes and increases the speed of business data integration. The Lambda Architecture approach offers to create real-time processing applications on the top of MapReduce and Storm or similar systems. This Architecture decomposes the problem of computing arbitrary functions on arbitrary data in real-time into three layers including batch, serving, and speed layers. It uses Hadoop (Olson, 2010) for supporting batch layer and Storm (Mera, Batko, & Zezula, 2014) for supporting speed layer in one system and creates a combined system. This combined system is scalable, extensible, robust, and fault tolerant. It also provides capabilities of easy debugging ad hoc queries. Moreover, it minimizes maintenance (Fan & Bifet, 2013).

Another architecture is a software architecture pattern called Kappa (Azim, 1988; Shung et al., 1991) in which there two layers including serving layer and speed layer rather than Lambda architecture with three layers. Therefore, Kappa architecture is a

University of Malaya

simplification of Lambda architecture (Nasir, 2016). It can use relational databases such as SQL (Kriegel, 2011) or it can communicate with a key-value store such as Cassandra (Lakshman & Malik, 2010). The Kappa architecture system stores data in an unalterable log format that can be only appended. Then the system streams logged data into a computational system and supply them to secondary stores for serving.

Zeta architecture (Alloui & Oquendo, 2002; He, Elnikety, Larus, & Yan, 2012) is another solution that aims to be high-level enterprise architectural and to address this challenge. Zeta architecture consists of seven pluggable components including a global and dynamic resource management component at the center of the architecture to manage other components which are consist of real-time data storage, pluggable execution engine, container management system, distributed file system, solution architecture, and enterprise applications. The Zeta architecture reduces costs of applications deployment and maintenance. It brings simplifications into its system by using a distributed file system. Also, it has less duplication and movement of data with no data transformation required except in a case that it is specifically called (Franks, 2012; Scott, 2016).

(b) Statistical Significance: Accuracy of statistical results is another challenge in big data science. As it is mentioned in (Efron, 2010) about Large Scale Inference, with vast data sets and thousands of questions that need to be answered, the probability of achieving wrong results is not low. There many types of research that highlighted the significance of addressing statistical significance problem (Bühlmann, 2013; Franks, 2012; Ritter & Muñoz-Carpena, 2013; Whitaker, 2014).

(c) Distributed Mining: Performing real-time distributed mining on huge data streams is a must for big data analytics (Yu Zhang, Sow, Turaga, & van der Schaar, 2014). Many data mining techniques are not trivial to paralyze(Benatallah et al., 2016;

University of Malaya

Bhardwaj & Johari, 2015; Fan & Bifet, 2013; Lawal, Zakari, Shuaibu, & Bala; J. Singh, 2014). It is required to do theoretical and practical analysis researchers to develop new distributed data mining techniques (Amma, 2016; Bhardwaj & Johari, 2015; Lawal et al.; S. Mishra, Dhote, Prajapati, & Shukla, 2015; J. Singh, 2014). Proposing such distributed mining methods raises another challenge such as real-time processing of enormous amounts of diverse data, real-time adaptation to arriving data characteristics, data access and communication limitation between distributed learners, etc. (Yu Zhang et al., 2014).

(d) Time-Evolving Data: Big data tools especially data mining methods required to be adopted with the probable evolvement of data over time and sometimes the changes must be detected before running main data mining tasks. For instance, there are very powerful data mining techniques in the field of data stream mining (Gama, 2010). There are studies conducted to deal with time-evolving data (G. Chen & Luo, 2015; Faloutsos, Kolda, & Sun, 2007; Golab, Prahladka, & Ozsu, 2006; Reddy & Raju, 2012). Time-evolving data are generated in many areas such as in cancer research where to monitor the efficiency of medications, genes must be measured at the different time frequently. Or in the field of Computer Vision, video streams include time-indexed series of images. However, in Network Security field, the behavior of users can be changed over time. Thus, HTTP and HTTPS connections are recorded continually at different timestamps. Therefore, dynamic techniques that can be adapted to the evolving nature of data are needed to deal with such those scenarios (Vogt et al., 2015).

(e) Compression: The quantity of big data is huge and it is rapidly growing up. So,a huge amount of space needed to store whole data. To address this challenge, two main strategies are available. The first approach is to use compression methods. In this way, they are not losing any part of data and it may take more time and less space. So, it can

University of Malaya

be considered as time transformation to space.The second strategy is using sampling methods in which they have to choose and keep only more representative data and lose some information, but it saves more storage space in return (Fan & Bifet, 2013). For instance, in one study (D. Feldman, M. Schmidt, & Sohler, 2013), Coresets which are small sets that approximate the original data for a given problem, are used to reduce the complexity of Big Data problems (Fan & Bifet, 2013).

(f) Visualization: Visualization is always a major task in big data analysis. Data visualization provides quick identification of important patterns and interesting events which are not easy to be recognized without data visualization (Tam & Song, 2016). Finding user-friendly visualization of the large volume of data is not easy and new frameworks and techniques are required to reveal the stories hidden inside the data (R. Smolan & Erwitt, 2012). The new methods of big data visualization must improve processing, analyze, and visualize of the enormous amount of complicated data (Agrawal, Kadadi, Dai, & Andres, 2015). Visualization is effective for demonstrating significant information in the huge volume of data and also for doing complex analyses (Keim, Qu, & Ma, 2013).

(j) Hidden Big Data: Leveraging big data tools and technologies is not only managing characteristics of big data such as volume, velocity, complexity, heterogeneity, and variety. It is also about analyzing the data to discover hidden information that is non-trivial and useful (Motta, Puccinelli, Reggiani, & Saccone, 2016). Generating a large amount of untagged and unstructured data in recent years causes losing quantities of valuable data (Bhardwaj & Johari, 2015; Fan & Bifet, 2013; Kaur, 2015; S. Mishra et al., 2015). According to a research about issues and challenges of big data analysis (Bhardwaj & Johari, 2015), 97% of the potentially valuable data is still untagged and even more is not analysed yet and if they do tag on digital universe

University of Malaya

data generated till 2012, the useful data would be around 643 Exabytes which 23% of the total data (Bhardwaj & Johari, 2015; J. Gantz & Reinsel., 2012).

(h) Timeliness: Analysis of big data must be faster and faster especially in some urgent cases such as health care emergency conditions or in a suspected transaction of a fake credit card before the worst result takes place. But the size of data sets that are required to be processed is continuously growing up and processing time is increasing as well. Doing full analysis on whole data in such cases is not feasible and wastes much time. Therefore, it is required to do analysis on a small portion of new data and achieve a partial result. So, it consumes lower computation resources and less time to develop quick demonstration (Jaseena & David, 2014). In addition to achieving results as fast as possible, the accuracy of the results is also very important. Designing an efficient system to process a specific size of data and return the results is a major concern (Lawal et al.). However, the speed here is not same as the meaning of Velocity as one of the characteristics of big data (Lawal et al.; B. S. P. Mishra, Dehuri, & Kim, 2016; Mohanty, Jagadeesh, & Srivatsa, 2013).

2.3 Big Data Indexing

The simplest definition of big data indexing is that process of converting a collection of data into a format suitable for easy search and retrieval. Indexing is needed to perform optimized retrieval process in big data which are massive and mostly complex especially in scalable and distributed storage in environments such as cloud computing (Chen et al., 2013). Because performing non-indexed explorations on a huge amount of data is impractical and using proper indexing techniques according to the structure of data would benefit us by optimizing the performance of query processing on big data (Wang, Holub, Murphy, & O'Sullivan, 2013). Hence, appropriate indexing techniques ease accessing big data effectively.

University of Malaya

Many indexing techniques are used by researchers to make big data retrieval faster and accurate as well. For instance, using a file index technique on large text collections can make event stream indexing efficient in cloud computing (Cambazoglu, Kayaaslan , Jonassen , & Aykanat, 2013), or to have optimized and fast searches in the cloud environment, a semantic indexing based approach is highly recommended by some researchers (Miguel Ángel Rodríguez-García, Rafael Valencia-García, Francisco García-Sánchez, & J Javier Samper-Zapater, 2014b) and also R-tree indexing method would help to provide multi-dimensional indexing in the cloud environment (Wang, Wu, Li, & Ooi, 2010).

In fact, using indexing techniques facilitates to balance amount of consumed resource such as computational power with the performance of data retrieval (Paul, Chen, Bharanitharan, & Wang, 2013). Analysing big data with minimized cost and time is strongly required in vital areas such as a complex clinical field. So, using data indexing methods would reduce time consumption (Dijkman et al., 2013; Gani, Siddiqa, Shamshirband, & Hanum, 2016). However, decreasing high costs need to be addressed in developing indexing techniques. Moreover, any efficient data indexing proposal has to satisfy all big data requirements (Chen et al., 2013) such as Volume, Velocity, Variety, Veracity, Value, Variety, and Complexity (Gani et al., 2016). These requirements are explained in Section 2.8.

Although there are many proposed indexing techniques described in the existing literature, rarely we can find a state-of-the-art survey that investigates the consequences and performance of those techniques. In fact, there is a need for having a guidance for helping researchers to compare and select appropriate indexing technique in order to solve indexing on Big Data issues (Gani et al., 2016). Therefore, comparative studies on big data indexing are necessary.

University of Malaya

Based on the above description, indexing approach is required to speed up accessing to big data which can be the format of records, objects, etc. Clearly, any indexing technique must be able to support basic functions of the database system and efficient enough in searching and querying the data.

2.3.1 Big Data Indexing Taxonomy

Big data indexing are classified into different categories to better understand the taxonomy. Figure 2.2 shows the several categories of big data indexing. The classification is important because of large-scale data in the unstructured format. Based on the previous studies conducted by (Athanassoulis, Yan, & Idreos, 2016; Che, Safran, & Peng, 2013; Gani et al., 2016; Y. Wu et al., 2015)(Delbru, Campinas, & Tummarello, 2012; Manolopoulos, Theodoridis, & Tsotras, 2009; Shang, Yang, Wang, Chan, & Hua, 2010; Zhu, Huang, Cheng, Cui, & Shen, 2013)(Wu, Shoshani ,& Stockinger, 2010) , the classification is based on four aspects: (i) content format, (ii) structures, (iii) requirements, and (iv) data retrieval.

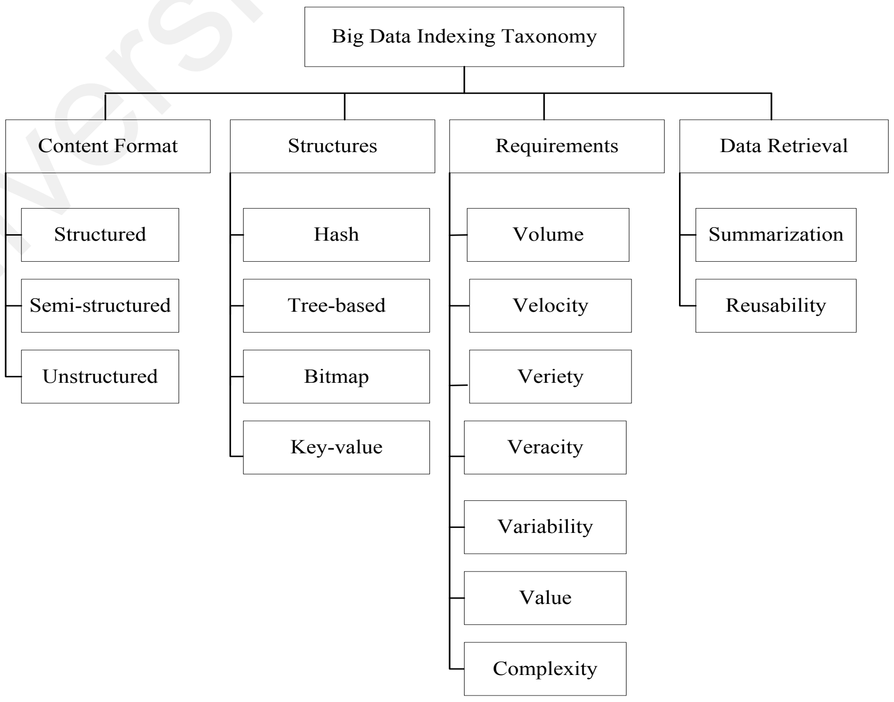

Figure 2.2: Big data indexing taxonomy

University of Malaya

2.3.1.1 Big Data Indexing Content Format

Variety, as one of the major aspect of big data characterization, is resulted from the growth of virtually unlimited different sources of data. This growth certainly leads to the great heterogeneity of big data. Data from different sources naturally has a great many different types and representation forms, and is significantly interconnected incompatible data formats and inconsistently represented (Che et al., 2013).

(a) Structured data: Structured data are often managed using Structured Query Language (SQL) – a programming language created for managing and querying data in relational database management systems. It has the advantage of being easily entered, queried, stored and analyzed. Examples of structured data include numbers, words, and dates.

(b) Semi-structured data: Semi-structured data has a logical flow and format to it that can be understood, but the format is not user-friendly.

(c) Unstructured data: Unstructured data refer to data that does not follow a specified format (e.g., text messages, location information, videos, and social media data). Unstructured data is everywhere. In fact, most individuals and organizations conduct their lives around unstructured data. However, most data is at semi-structured formats.

2.3.1.2 Big Data Indexing Structures

This subsection describes the indexing structures and its importance to big data below:

(a) Hash-Based: A hash-based indexing technique is designed to be search-efficient in the context of high-dimensional data (Zhu et al., 2013). This technique is aimed to perform faster search operation and obtain results by representing high-dimensional data with compact binary code (Gani et al., 2016). The hashed-based technique is

University of Malaya

recommended for approximate similarity searches in high-dimensional data. Hence, many applications in the areas such as document analysis, image retrieval, and near duplication detection use hash-based methods for indexing their data (Shang et al., 2010).

(b) Tree-based index: Tree-based indexing method is useful for ordering entries by using the concept of a tree which consists of root and nodes. Data entries can be inside the leaves of the tree or even inside each node of the tree. Data entry can be actual data or pairs of the search key and value. This data structure is good for range queries. There are some indexing techniques such as B+-tree and R-tree that use the concept of the tree-based data structure. B+-tree is recommended for single-dimensional range and the R-tree is recommended for multi-dimensional ranges (Manolopoulos et al., 2009).

(c) The bitmap Index: Bitmap index structure is the most efficient indexing method for range queries on append-only data (Wu, Shoshani, & Stockinger, 2010). Bitmap indexing technique uses bulk index data consists of sequences of bits to answer queries by utilizing those sequences of bits in bitwise logical operations. Bitmap indexing has been used in different analytics solutions such as Spark, Druid. As the result, bitmap indexing makes query processing very fast. Moreover, bitmap indexing is flexible in conducting Boolean operations in retrieving data (Y. Wu et al., 2015).

Many commercial and open database systems use bitmap indexing techniques. It makes fast read performance in processing particular sorts of queries such as equality and selective range queries. In contrast, update operation in bitmap indexing technique is costly. The reason is that bitmap indexes are compressed to reduce the storage consumption, and every time any update operation is applied, it required to decode and encode a bitvector which causes expensive cost (Athanassoulis et al., 2016).

University of Malaya

Therefore, bitmap indexing techniques are mostly recommended for read-only systems such as data warehouses that require fast query processing and also require joining larger dimension table to smaller dimension tables (Lemire, Kaser, & Aouiche, 2010). It is also not recommended for online transaction processing applications (Lemire et al., 2010).

2.3.1.3 Big Data Indexing Requirements

Big data indexing requirements including Volume, Velocity, Variety, Veracity, Variability, Value, and Complexity are discussed as following:

(a) Volume: The term ‗big data' specifically points to the word ‗big' referring to the volume of data. The current size of generated data is in Petabytes and it is expected to reach to Zettabytes in near future (Gani et al., 2016; Katal, Wazid, & Goudar, 2013). The most obvious challenge in the big data area is how to manage and index the huge volume of data (J. Chen et al., 2013).

(b) Velocity: Velocity refers to the speed of generating and updating data. The second challenge with big data is handling data as fast as they created or updated (J. Chen et al., 2013). Recent business technologies such as e-commerce need to deal with the speed of data as well as the abundance of data (Gani et al., 2016; Stephen Kaisler, Frank Armour, J Alberto Espinosa, & William Money, 2013a).

(c) Variety: Big data are continuously being produced in different formats such as text, images, audio, etc. and they can be gathered from different sources such as social media applications, Web pages, Web log files, health care applications, e-education systems, digital documents, emails, sensor devices, mobile applications, etc. The way of indexing and analyzing a variety of big data is the third challenge (C. P. Chen & Zhang, 2014; Gani et al., 2016; Kaisler et al., 2013a; Yang et al., 2014).

University of Malaya

(d) Veracity: The term ‗veracity' points to trustworthy and accuracy of big data. Especially in the decision-making process, it is required to know up to what extent can data be trusted, how much the data is accurate and not corrupted, and what is the source of data. Achieving this knowledge about the data is not easy. This important issue which is known as big data veracity (Gani et al., 2016; X. Wang, Luo, & Liu, 2015), arises the fourth challenge in big data science.

(e) Variability: Inconsistencies and heterogeneities in big data flow cause another challenge for dealing with big data. It is difficult to maintain data loads, especially in some environments with high use rate and peaks in data loads in certain events such social media. This challenge is so-called big data variability(Gani et al., 2016; Katal et al., 2013).

(f) Value: The usefulness of big data in crucial business processes such decision making is another challenge of big data that is called big data value. Knowing how much is large data valuable and beneficial is an important issue as it is highlighted in some literature that ‗the purpose of cloud computing is insight, not numbers' (Gani et al., 2016; Kaisler et al., 2013a; LaValle, Lesser, Shockley, Hopkins, & Kruschwitz, 2011).

(g) Complexity: Complexity is the seventh challenge in big data management that deal with the degree of interconnectedness and interdependencies in big data structures (Gani et al., 2016; Kaisler et al., 2013a). Connecting, correlating relationships and multiple data linkages of big data from different sources are very important and require significant activities such as cleansing, matching, linking, transformation across systems, etc. Without complexity in term of above activities, organizing big data is not easy (Barbierato, Gribaudo, & Iacono, 2014; Gani et al., 2016).

University of Malaya

Based on those seven challenges which are also entitled as characteristics of big data or big data requirements, the effectiveness of big data indexing techniques can be compared. Establishing big data requirements and developing a system require identifying the characteristics of the indexing technique that is going to be used in the system. The basic measures that present the efficiency of an indexing technique are accuracy and timeliness (Gani et al., 2016).

Different indexing techniques are proposed to fulfill the requirements of big data. But, just a few solutions have found their respective places in a database system and still, there is a big need for more efficient and concise indexing structures (Idreos et al., 2007; Sidirourgos, 2014). Based on a survey conducted on indexing techniques for big data (Gani et al., 2016), different indexing methods have different advantages and disadvantages and only can support some of the requirements of big data. Most of them focus more on handling the volume and less on a variety of big data.

2.3.1.4 Data Retrieval

The rapid growth of data in recent years has raised a challenge of distributing information from a vast plethora of data (Jamil & Ibrahim, 2009). Different types of data are generated by data sources. For example, Web-based applications are rapidly producing millions of documents embedding metadata in the type of semi-structured data every day. This huge amount of data needs efficient search and retrieval tools to explore this semi-structured data and retrieve exploitable and useful information for human as well as for machines (Delbru et al., 2012).

Retrieving information without using any indexing methods means searching whole data and when it comes to large volume of data will waste a lot of time and computational power. In contrast, applying appropriate indexing technique on the data

University of Malaya

before searching process facilitates information retrieval and decreases the cost. Figure 2.3 shows the location of index mechanism in a data retrieval model.

(a) Summarization: Nowadays, extracting useful information out of the massive amount of data available in different data sources such as web-based and social media applications is very attractive and interesting for business firms. Summarization is very important approaches for gathering valuable information from numerous big data source in different fields such as health care, e-education, Internet of Thing (IoT), signal data, etc.

Different definitions for a summary have been given by researchers. Some researchers pointed that it is a text extracted from one or more text which consists of significant parts of the original texts and it should be equal to or less than half of the content of original texts. (García-Hernández et al., 2009; Hovy & Lin, 1998; Rahman & Borah, 2015; Subramaniam & Dalal, 2015). Providing relevant summarization from original data sources according to the system's requirements or user's query is a major challenge (Rahman & Borah, 2015). Moreover, the summarization must be updated for every time that the original sources of data are updated.

The main aim of query-based summarization is providing proficient respond for complex questions. Also, the student can use summarized format of past materials to save their time with an efficient action or people can use summarization results to find out whatever they have missed from important meeting or news.

Summarizations can be categorized based on different criteria. For instance, a summarization that must present main findings of some original textual information according to the requirements of a system or of a user in order to save time and other cost and also probably to ease quick-decision making can be categorized as Query-

University of Malaya

based summarization. Followings are summarization categories presented by some researchers(Rahman & Borah, 2015; Tang, Yao, & Chen, 2009):

- Single document summarization (SDS): This approach creates a summary based on only a single document. It is considered as a hard task (Rahman & Borah, 2015).
- Multi-document summarization (MDS): This approach aims to extract a summary according to the content of multiple documents. The challenge here is about what percentage of each document should be used.
- Query-oriented summarization (QS): This approach is also called as query-based summarization and generates a summary according to a given query. Many text mining application normally used this type of summarization. One of the challenges here is about how to make the returned results digestible for the query applied.

Many studies have used summarization concept and proposed methods and techniques to apply summarization on different media. For example, chat summarization method (Jones, 2007; Uthus & Aha, 2011), Multi-Document text summarization (Agarwal, Gvr, Reddy, & Rosé, 2011; Radev et al., 2004), decision summarization (Bui, Frampton, Dowding, & Peters, 2009; Fernández et al., 2008; L. Wang & Cardie, 2011), and abstractive summarization (Balaji, Geetha, & Parthasarathi, 2016; Sun, Wang, Ren, & Ji, 2016) are some of the proposed methods that used summarization concept in order to optimized related processes. These methods are aimed to retrieved information with optimized approaches. So, integrating summarization approaches in data retrieval mechanism would optimize retrieval process and would save the cost.

(b) Reusability: Reusability has been defined in different researches in different fields. For instance, Singh and Chana pointed out that reusability in software engineering means re-using different form of existing assets such as software

University of Malaya

components, code, designs, and documentations in the process of developing software products (S. Singh & Chana, 2012). This approach reduces the amount of code that needs to be tested and analyzed and also makes the process of development faster and the cost of the product lower. Hence, reusability offers high productivity and better quality of software products along with facilitated maintainability (Ahmaro, Abualkishik, & Yusoff, 2014; Goyal & Gupta, 2014; Sattar & Zhang, 2014).

Facilitating discovery of information is the main aim of most information retrieval technologies and much knowledge work is required to find and re-use earlier extracted information (Dumais et al., 2016). According to some studies reported that the rate of revisiting web pages by users is about 58-81% (Dumais et al., 2016). However, many studies reported overlaps and similarities among web searches and information retrieval queries which caused overlapped or same results.

Stuff I‘ve Seen (SIS) is one of the solutions that integrated reusability approach in data retrieval process (Dumais et al., 2016). This system is designed to use reusability approach in information retrieval This system creates an index of viewed information such as web pages, emails, documents, etc. Then the system uses rich text-based indications according to the information that has been seen by the use. The results show that finding information is easier by this system and it causes less use of search tools.

Based on the above explanation about Stuff I‘ve Seen (SIS), integrating reusability approaches into the cycle of data retrieval would optimize retrieval processes and leads information retrieval system to high performance, faster responsiveness, and lower cost. So, it is attractive and interesting for researchers as well as business expert to work more on this integration.

University of Malaya

2.3.2 Big Data Indexing Evaluation Metrics

Different Metrics have been used in previous studies (Campbell, Santos, & Hindle, 2016; Gani et al., 2016; Qu, Zhang, Yao, & Zeng, 2016; Schäler et al., 2013; K. Wang et al., 2014). Based on our investigation on previous literature, the important factors for evaluating big data indexing techniques are as following:

- **Index size**: This is the size of storage space that is needed for storing index data during implementation of the indexing process.
- **Index Creation Time:** This is the time taken to implement indexing mechanism.
- **Query Response Time:** This is the time taken to apply queries and retrieve results.
- **Indexing Overhead**: This increases the time required to perform indexing before submitting the first query
- **Index Traversing Time**: This is the time taken to traverse the index structure
- **Index Hit Ratio/Index Miss Ratio:** This is fraction of queries served by Index, Index Miss Ratio is vice versa
- **Size Overhead**: This is the size of overhead resulted when indexing is applied
- **Search Performance**: It is the percentage of difference in Search Time between no indexed and indexed searches

The following section provides analysis and discussion on some of the current big data indexing techniques.

2.3.3 Analysis and Discussion of Current Indexing Techniques

This section evaluates some of the current indexing techniques studied in (Gani et al., 2016). The authors examined indexing methods to measure the amount of data volume

University of Malaya

while maintaining its performance. They analyzed selected indexing methods based on the performance of each technique in handling different types of data, various data formations and also based on updating speeds. However, they used more measures such as accuracy, the degree of interconnectedness, implication on decision-making, security, and inconsistency for evaluation (Gani et al., 2016).

Our analysis of the indexing techniques is based on above study and other related literature of different techniques and identified some of the facts. The strength of each indexing technique detailed to ensure its viability for big data. Table 2.2 lists some of the current indexing methods and presents which technique can address which big data indexing requirements.

It is very clear in Table 2.2 that all of the called techniques support volume for big data dataset (Sellis, Roussopoulos, & Faloutsos, 1987; Sellis, Roussopoulos N, & Faloutsos, 1987; Weng & Chuang, 2011) except Lazy indexing and Support Vector Machine. Lazy indexing technique cannot satisfy big data dataset in term of volume (Stefan Richter, JorgeArnulfo, Quian´eRuiz, Stefan Schuh, & Dittrich, 2012). And also the status of satisfaction of Support Vector Machine technique for volume is unknown (Chakrabarti, Pathak, & Gupta, 2011).

The Table 2.2 shows that the techniques satisfy velocity in many cases such as Red-Black Tree, Randomize Interval Labelling, Collaborative filtering, Graphic Query, Bitmap, Geometric Hashing and Hierarchical Tree (Cheng, Ke, Fu, & Yu, 2011b; Dong, 2010; Kaushik, Umarani, Gupta, Gupta, & Gupta, 2013). Hashing, Semantic Indexing, and Suffix Tree techniques cannot support velocity for the big dataset (Miguel Ángel Rodríguez-García, Rafael Valencia-García, Francisco García-Sánchez, & J. Javier Samper-Zapater, 2014a). The rest of called techniques have not been evaluated in velocity term.

University of Malaya

Hashing, Semantic Indexing, Collaborative Vector Technique, Collaborative learning and Shortest Path techniques support variety based on previous related work (Fu & Dong, 2010; Rodríguez-García et al., 2014a; Weng & Chuang, 2011). The rest of techniques are not able to support variety for big data dataset.

In the matter of variability, the following techniques are supported for big data dataset: R+- Tree, Semantic Indexing, Randomized Interval Labelling, Shortest Path Tree, B-Tree, Bitmap and Geometric Hashing. The rest of the called techniques cannot support variability for big data (Maier, Rattigan, & Jensen, 2011; Sellis, Roussopoulos, et al., 1987; Wei L-Y, HsuY-T, PengW-C, & LeeW-C, 2013).

The figure shows that Red-Black Tree, Semantic Indexing, Lazy Indexing, Support Vector Indexing, Shortest Path, B-Tree technique and Collaborative Semantic satisfy value for big data dataset (Maier et al., 2011; Sheng-Cheng, Su, Chen, & Lin, 2013; Weng & Chuang, 2011). In contrast, the bitmap technique cannot support value (Gundema & Armaganb, 2006) and all the rest of technique have not been clarified that support value or not.

Regards of complexity, just some of the techniques satisfy it for big data dataset. According to the Table 2.2, the techniques are as following: Red- Black Tree, Inverted Index, Randomized interval labeling, Graphic Query Tree, Bitmap, Geometric Hashing and Hierarchical Tree (Cheng et al., 2011b; Sheng-Cheng et al., 2013).

Retrieving information without using any indexing methods means searching whole data and when it comes to large volume of data will waste a lot of time and computational power. In contrast, applying appropriate indexing technique on the data before searching process facilitates information retrieval and decreases the cost. Figure 2.3 shows the location of index mechanism in a data retrieval model.

University of Malaya

Table 2.2: Some of the current indexing techniques versus big data indexing requirements

Indexing Technique	Authors	Big Data Indexing Requirements						
		Volume	Velocity	Variety	Veracity	Variability	Value	Complexity
SmallClient	(Aisha, et al., 2016)	Yes	Yes	Yes	NA	NA	NA	NA
R+-tree	(L.-Y. Wei, Hsu, Peng, & Lee, 2014)	Yes	NA	No	NA	Yes	NA	NA
Red–Black tree	(Yeh, Su, Chen, & Lin, 2013)	Yes	Yes	No	Yes	NA	Yes	Yes
Hashing	(X. Zhu, Z. Huang, H. Cheng, J. Cui, & H. T. Shen, 2013)	Yes	No	Yes	No	NA	NA	NA
Inverted index	(B. B. Cambazoglu, Kayaaslan, Jonassen, & Aykanat, 2013)	Yes	NA	No	NA	NA	NA	Yes
Semantic Indexing	(Rodríguez-García et al., 2014b)	Yes	NA	Yes	Yes	Yes	Yes	NA
	(Done, Khatri, Done, & Draghici, 2010)	Yes	No	Yes	Yes	NA	NA	NA
Support Vector Machine	(Dittrich, Blunschi, & Salles, 2011)	NA	NA	No	Yes	NA	Yes	NA
Lazy Indexing	(Richter, Quiané-Ruiz, Schuh, & Dittrich, 2012)	No	NA	No	Yes	NA	Yes	NA
Randomized interval labeling	(Yıldırım, Chaoji, & Zaki, 2012)	Yes	Yes	No	Yes	Yes	NA	Yes
Collaborative filtering technique	(M.-F. Weng & Chuang, 2012)	Yes	Yes	Yes	Yes	NA	NA	NA
	(Huang, Lu, Duan, & Zhao, 2012)	NA	NA	No	Yes	No	Yes	No
Collaborative learning	(Fu & Dong, 2012)	Yes	NA	Yes	NA	NA	NA	NA
Graph Query Tree	(Cheng, Ke, Fu, & Yu, 2011a)	Yes	Yes	NA	NA	NA	NA	Yes
Shortest Path Tree	(M. Maier, M. Rattigan, & D. Jensen, 2011)	Yes	NA	Yes	Yes	Yes	Yes	NA
B-Tree	(Li, Yi, & Le, 2010)	Yes	NA	NA	NA	Yes	Yes	NA
Bitmap	(K. Wu, Shoshani, & Stockinger, 2010)	Yes	Yes	No	NA	Yes	No	Yes
Geometric hashing	(Mehrotra, Majhi, & Gupta, 2010)	Yes	Yes	No	Yes	Yes	NA	Yes
Collaborative Semantic	(Leung & Chan, 2011)	Yes	NA	No	Yes	NA	Yes	NA
	(Dieng-Kuntz et al., 2006)	Yes	NA	No	NA	NA	Yes	NA
	(Gacto, Alcalá, & Herrera, 2010)	NA	NA	NA	Yes	NA	NA	NA
Fuzzy	(Dittrich et al., 2011)	Yes	Yes	No	No	NA	Yes	Yes

University of Malaya

Table 2.2: Continued

Manifold learning	(Elleuch, Zarka, Ammar, & Alimi, 2011)	Yes	Yes	Yes	Yes	NA	NA	Yes
Hierarchical Tree	(Lazaridis, Axenopoulos, Rafailidis, & Daras, 2013)	Yes	Yes	No	Yes	NA	NA	Yes
Suffix Tree	(Russo, Navarro, & Oliveira, 2011)	Yes	No	NA	NA	NA	NA	NA
Incremental collaborative filtering	(Komkhao, Lu, Li, & Halang, 2013)	Yes	NA	NA	Yes	NA	No	NA
Collaborative annotation	(Elleuch et al., 2011)	Yes	NA	NA	Yes	NA	NA	NA

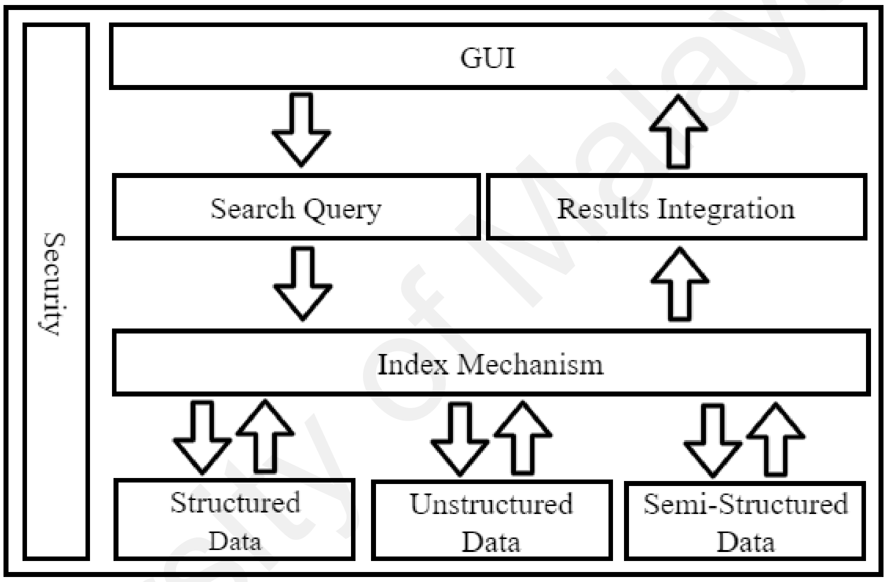

Figure 2.3: Index mechanism in a data retrieval model

In this model, different types of data may need to be searched. So, proper index structure must be applied on different types of data or their metadata if they exist. Then the results of all search tasks require being integrated and returned to the user.

Various indexing techniques are proposed to make data retrieval faster (Boubekeur & Azzoug, 2013; Kathuria, Datta, & Kaul, 2013; Ramakrishna & Rani, 2013; L. Wei & Shen, 2016) in environments such as data centers and data warehouses but finding a proper indexing technique for a particular query type is not easy (Jamil & Ibrahim, 2009).

University of Malaya

2.4 B-tree Indexing Technique

This section introduces B-tree indexing technique and discusses the flow of the search method that is used in B-tree technique.

B-trees are indexing techniques that use tree-based data structure. The algorithm of B-tree performs vigorous indexing, but when it requires to index online data stream, it consumes huge computing resources (Gani et al., 2016). B-trees are balanced search trees designed to work with a database based on nodes and many children. Moreover, every modern Database Management System (DBMS) contains some variant of B-trees plus other index structures for special applications.

B-trees generalize binary search trees in a natural manner. A common variant on a B-tree, known as a B+-tree, stores all the satellite information in the leaves and stores only keys and child pointers in the internal nodes, thus maximizing the branching factor of the internal nodes. B+-tree is a dynamic index structure. For such data structure, all leaf nodes are at the same height and it means the number of edges between the root node and leaf nodes is always same. So, each node has either zero or two child. However, all data are stored in the leaves of the tree. If any new insertion or deletion happens, B+-tree reorganizes itself automatically and it is not required to reorganize the whole file (Orsborn, 2007).

2.4.1 History of B-tree

In the late 1960s, different IT based research groups and institutions were trying to develop a file system so called "access methods" to use in their machines. H. Chiat and other developers at Sperry Univac Corporation implemented a system to perform insert and search operations in which the process flow was similar to the process flow of same operations in the B-tree method. However, B. Cole and other developers at Control Data Corporation developed a similar system. Then, R. Bayer and E. McCreigh at Boeing

University of Malaya

Scientific Research Labs introduced B-tree technique as an external indexing mechanism in which the cost of the basic operations such as insertion, deletion, and search was very low (Bayer, 1971; Beyer & McCreight, 1972; Comer, 1979).

There is no specific explanation about the origin of "B-tree" in the literature. There is more than one term such as "Balanced tree", "Bushy tree", or "Broad tree" (Comer, 1979) which are reported by author that the "B-tree" might be named. In the same time, the modern history of B-tree starts from Boeing Scientific Research Labs, some of the researchers suggest that "B" comes for the word "Boeing". But, some believes that the "B" stands for Bayer who was one of the main developers of B-tree mechanism at Boeing Scientific Research Labs.

2.4.2 B-tree Mechanism

A tree based structure is an upside down tree in which the root of the tree is at the top and the leaves are at the bottom. The last down level of nodes in a B-tree is so called the leaf level. Every node is the leaf level is called leaf node and each leaf node does not have any node bellow itself. All upper level nodes above the leaf level are so called index nodes or directory nodes. The nodes between root node and leaf nodes are called internal. The root node of B-tree structure is known as level 1 of the B-tree. Lower levels below the root node have the larger number successively to the leaf level which is the lowest level but it has the largest level number. This largest number for leaf level is also called the depth of the B-tree. Figure 2.4 shows a B-tree structure with depth of 4.

Whereas the B-tree is a balanced tree, the lengths of all paths from the root node to a leaf node are same. In a B-tree based structure, every node can contain some keys and some pointers. The keys are the values of a specific attribute that the tree structure is generated according to that attribute. Pointers are kind of a link to the location of the record that contained the keys. A B-tree of degree d has a tree structure in which every

University of Malaya

node can have up to 2d keys and 2d+1 pointers. Every node always must have at least half of the maximum number of its keys. Based on (D. Comer, 1979), each node of a B-tree structure with order d can contain up to 2d keys and 2d+1 pointers.

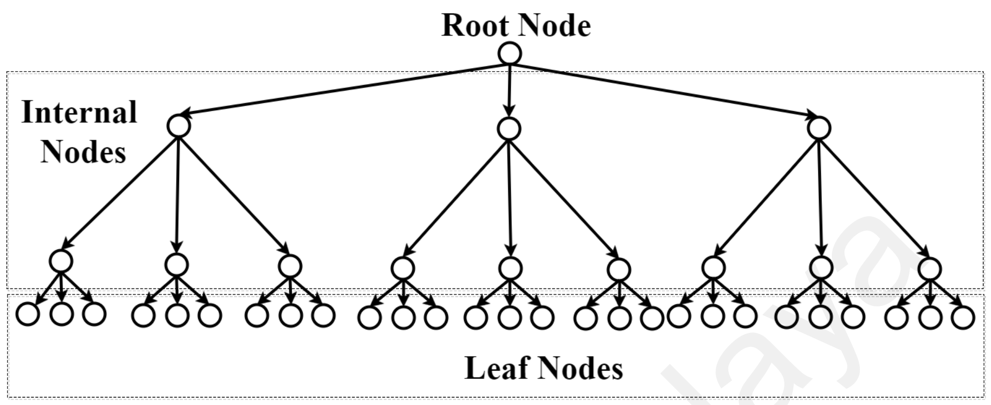

Figure 2.4: A B-tree structure with depth 4

In a B-tree of order m, the root node must have at least two children. Every node has at most m children. However, every internal node has up to m/2 children and if an internal node contains k children, it must have (k-1) ordered keys in which the first key of the internal node is greater than the keys of the leftmost child and the last key of the internal node is less than the keys of the rightmost child.

2.4.3 B-tree Main Operations

In this section, the main database operations including insertion, deletion and search actions using B-tree indexing structure are described.

2.4.3.1 Insertion

The purpose of using insertion in B-Tree is to be implemented in a way that keeps the tree perfectly balanced in the sense that all the leaf nodes of the tree are on the same level of the tree (Graefe, 2016). B-tree insertion is a complicated process than normal insertion process. The initial insertions begin at a leaf node. Then new element search the tree to find the leaf node where the new element that required to be added. Before

University of Malaya

new element inserts it, if the node contains fewer than the maximum number of elements, then there is room for the new element. Insert the new element in the in the node, keeping the node‘s element ordered. However, the node can split into two nodes if the node is full. In the case, a single median is chosen from among the leaf‘s element and the new element. Values less than the median are put in the new left node and values greater than the median are put in the new right node, with the median acting as a separation value (Comer, 1979). Suppose that a key k needs to be inserted in the sub-tree rooted at y in a B-tree T. If y is a leaf, insert the key. If not, find a child in which the key should go to and then make a recursive call with y set to the child.

2.4.3.2 Deletion

Deletion in B-tree is a set of operations to find the proper node based on the deleted resides in a leaf or the key resides in a non-leaf node. Deletion of non-leaf demands that the discovered key to be identify and then changed into the vacated position in order to find the result efficiently. There are two popular strategies for deletion from a B-Tree. First, is to locate and delete the item, and then restructure the tree to regain its invariants. Second, a single pass down the tree is done, but before entering a node, the tree needed to be restructured so that once the key to be deleted is encountered; it can be deleted without triggering the need for any further restructuring (Comer, 1979). Deletion can be preform in a multiple steps, for example, searching for the value to delete, if the value is in a leaf node, it can simply be deleted from the node. Then if underflow happens, the sibling will be checked to either transform a key or fuse the siblings together. If deletion is preformed from the right child retrieve the max value of left child if there is no underflow in left child.

University of Malaya

2.4.3.3 Search

To search a B-tree structure T of order m for finding a key K among the keys of the nodes (key_i $0 \le i < m$) inside the B-tree structure, the B-tree search algorithm traverses the nodes of the B-tree T from the root to the leaves and compares the key K with every key (key_i $0 \le i < m$) of the traversed node. If K is equal to key_i, it returns $pointer_i$ ($0 \le i < m$) as the result which is the pointer that points to the location of the records of datasets in physical storage that include key K. But if key K is less than key_i and greater than key_{i-1} (if key_{i-1} is existed) of the node, it traverses the left child of the node linked to the left side of key_i and repeats the comparison process. However, if the target key K is greater than a key_i and less than key_{i+1} (if key_{i+1} is existed) of the node, it traverses the right child of the node that is linked to the right side of key_i and repeats the comparison process. This process continues until the search algorithm finds the match key or there are no more nodes to be traversed. In this case no match key with the target key K is found and hence the result is empty. Figure 2.5 demonstrates an example of B-tree search operation.

University of Malaya

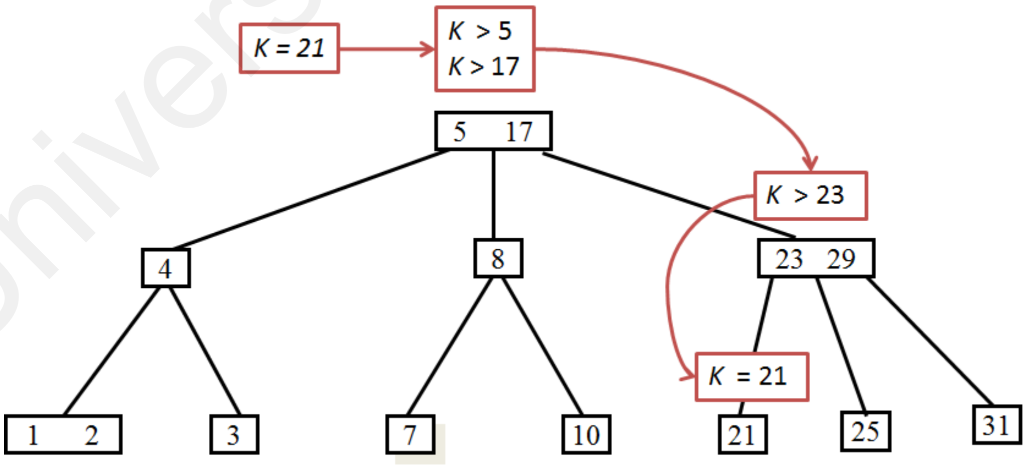

Figure 2.5: Example of a B-tree search operation

According to the nature of the B-tree search algorithm, the comparisons between the target key K with the keys of the nodes of the B-tree structure regardless of availability of the key *K* in the range of the minimum and maximum value of the keys of the nodes inside the B-tree structure. It means, the search algorithm is not capable of avoiding performing numbers of comparisons when the final result is empty. Hence, we can conclude that the time and computing resources consumed for searching the B-tree in the case that the key *K* is not in the range of keys of the nodes inside the B-tree are wasted. While, checking availability of the target key *K* among the keys of the nodes before traversing the B-tree structure, offers resource consumption optimization.

2.4.4 Existing Modified B-tree Indexing Techniques

B-Tree indexing technique is more adaptable to increasing the volume and variety of data (Siddiqa, Karim, & Chang, 2016). Based on the standard B-tree indexing technique, some modified indexing techniques are proposed. Current modifications of B-tree indexing technique use flash memory optimization or main memory (Chi, Lee, & Xie, 2014; Jin, Cho, & Chung, 2014; Siddiqa et al., 2016). SmallClient (Siddiqa et al., 2016) is one of the recent work which proposes a block level indexing mechanism based on the B-tree indexing technique to minimize indexing overhead and improve the performance of search execution tasks with optimized aggregation of computing performance.

FB$^+$-tree is another work which tries to speed up building index structure for large data storage systems using multi-level key ranges (Yu & Boyd, 2014). But, this work is using B$^+$-tree not B-tree. It is designed to be a main memory resident. Another study proposed indexing techniques and uses B-tree indexing structure and summarization concept. These summaries are not based on the values stored inside the nodes of the B-tree structure. In fact, this work offers extracting statistical summaries such as quantiles, frequent items, and various sketches of the records in the query (Yi, Wang, & Wei, 2014).

University of Malaya

2.5 Conclusion

Several kinds of literature related to big data and big data indexing are reviewed. Different structures of data namely structured, semi-structured, and unstructured data along with related challenges are described. Then various sources of big data are explained. Also, some of the challenges that researchers need to deal with in current time and future are elaborated. After that, big data indexing and its role in big data area followed by some of the main indexing structures were explained. We also discuss B-tree based indexing techniques.

However, this study focused on big data requirements that are required to be handled by indexing techniques. Some of the current indexing techniques are analyzed based on previous studies and then we discussed the analysis. Moreover, we discuss data retrieval, query summarization, and reusability concepts to investigate how we can bring those concepts into big data indexing techniques in order to facilitate data retrieval and enhance the performance of query processing in big data. Lastly, we described B-tree indexing structure and main database operations.

In next chapter, the problem that this study focuses on is analyzed. In fact, we measure resource consumption during data retrieval and query processing via a database system which uses B-tree indexing technique. We study the effect of increasing the volume of data on consuming resources, particularly, the trend of time consumption during data retrieval process. We also study the impact of processing earlier queries on process of later queries especially when they have overlaps and similarities.

University of Malaya

CHAPTER 3: PERFORMANCE ANALYSIS OF B-TREE INDEXING TECHNIQUE

3.1 Introduction

This chapter aims to provide problem analysis of resource consumption in data retrieval when querying a large amount of data, particularly with respect to the time consumption of later queries in data retrieval. That is being said, using performance analysis is derived from establishing the problem and identifying the time consumption model. The problem analysis is carried out based on time consumption in data retrieval. Some queries are applied against the data stored inside the database to retrieve desired data from the datasets stored in MongoDB database (MongoDB, 2016) in two modes, with and without using indexing method. Then, we used default indexing method of MongoDB which uses a B-tree data structure. A simulation engine is developed to apply and repeat queries. This simulation calculates and records the response time for each search operation. The findings of the analysis are verified using initial experimental analysis to measure the performance of the B-tree search method.

The remaining parts of this chapter are organized as follows. Section 3.2 presents an analysis of the B-tree in indexing. Section 3.3 provides time consumption analysis of B-tree method in indexing. Section 3.4 presents a retrieving time performance evaluation, which describes the process flow of the time consumption that being used for simulation engine to apply queries and measure the response time. Section 3.5 offers execution time analysis and discussion. Section 3.6 concludes the chapter.

3.2 Analysis of B-tree Indexing Technique

Time consumption of B-tree in indexing is an essential part of this study since it is the basics that the rest of the work and other algorithms proposed are based on. It is one of the most widely accepted methods for analysis optimization problem in indexing. B-

University of Malaya

tree is the default indexing technique in MongoDB (MongoDB, 2016), which is a balanced search tree which is intended to work well on magnetic disks and other direct access secondary storage devices (Cormen, 2009) with an acceptable minimization in disk I/O operations. B-tree has the "branching factor" which ability to have multiple children in every node.

The data structure of B-tree can be represented by $\log N$ as following:

$$\boldsymbol{O}\ \mathbf{log}\,\boldsymbol{N}\ (1)$$

Where N is donated to the number of keys of the B-tree. All leaves are at the same depth in the tree. Also, we can use B-trees to implement many dynamic-set operations in time O(lg n) (Cormen, 2009). Figure 3.1 illustrates a simple B-tree in which the keys in each node are from English alphabet.

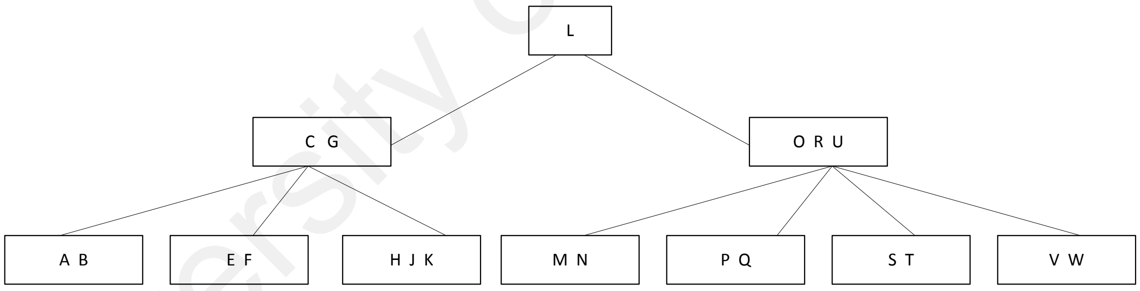

Figure 3.1: A B-tree structure with keys from English alphabet

As shown from the figure the each level represent a row of indexing strategies. Thus, the tree can be formulated as follows:

$$\boldsymbol{g} = \{\sum \boldsymbol{N}\} \rightarrow \sum_{\boldsymbol{g-1}}^{\boldsymbol{1}} \boldsymbol{N} - \mathbf{1}\ (2)$$

Where N donated to nodes and g donated to the number od descsions in B-tree, which is based on comparisons with the $g[N]$ keys of node N. Thus, each nodes are

University of Malaya

facing a multiway branching decsion in accordance with the number of children of the node. In addition, searching key (k) in B-tree (T) takes a pointer to the node (N), which is the root of the subtree presented as following:

$$\boldsymbol{T_s\ (R[T], k)} \quad (3)$$

Where T_s is the B-tree search and R is donated to the root of the tree. Figure 3.18 shows sample code for B-Tree-Search algorithm 1.

Algorithm 1: B-tree-search procedure

Required

B-tree search (N, K)

```
1. For$i \leftarrow 1$ & $k \geq k_i$
2. While $i \geq g[N]$
3. Do $i \leftarrow i+1$
4. If$i \geq g\ [N]$& $k = k_i[N]$
5. Then return $(i\ , N)$
6. If leaf $[N]$
7. Then return Null
8. Else DISK-READ ( $C_i[N]$
9. Return B-tree-search $B - tree - search\ (C_i[N], K$
```

If the search operation can find key (k) in the B-tree (T) the operation returns an orders pair with form (N, i) in which (N) is the node that the key (k) has been found in and $(i)$is the index of the key.

$$\boldsymbol{K_i = K \rightarrow B - tree - search\ \neq k\, , k\ \in T, k_i = 0} \quad (4)$$

In this procedure, a recursion is used for doing comparisons between k and the keys inside the B-tree and the nodes are encountered form a path starting from the root of the B-tree downward to the leaves. Therefore, the number of disk pages accessed by B-tree-search can be represented as following:

$$\boldsymbol{O(h) = O\ (\log n)}(5)$$

University of Malaya

Where h is the height of the B-tree and n is the number of keys in the B-tree. Since $g[N] < 2_t$, the time taken by the while loop as it is shown in Figure 3.18 in line 4 for doing comparisons among the keys within each node is $O\ (t)$, and the total CPU time (Cormen, 2009) shown as:

$$\boldsymbol{Total\ CPU\ time} = \boldsymbol{O}\ (\boldsymbol{t_h}) = \boldsymbol{O}\ (\boldsymbol{t}\ \log \boldsymbol{n}) \quad (6)$$

Based on the above explanations, repeating any number of this search operation to find key k in the B-tree T, consumes same amount of time and other resources such as computational power as the first search operation and there is no capability in B-tree structure to help later search operations to benefit from earlier search operations.

3.3 Query Processing Time Consumption Analysis

To analysis time consumption for each query in data retrieval, a cost best model is used to measure the time. Following the approach used in (Banker, 2011), the time consumption for each query, which shown as follow:

$$\boldsymbol{Total\ time} = \boldsymbol{T_{stop}} - \boldsymbol{T_{start}} \quad (7)$$

Where T_{stop} donated to stop time, which is the time taken from second check point at the end of the processing the query and T_{start} donated to start time, which is the time taken from first check point at the beginning of the processing a query.

Some queries are applied against data stored inside the database to retrieve desired data from the datasets stored in MongoDB databases in two times, with and without using indexing method. Thus, for the first time, indexing techniques are not used and just the four queries as they are listed in Table 3.1, are applied. Then, default indexing method of MongoDB which uses a B-tree data structure is used (MongoDB, 2016). For this purpose, four different query statements of different levels of complexity are used.

University of Malaya

Firstly, each query statement is applied on each dataset without using indexing technique. The simulation engine repeated each query transaction 20 times and returned the results and related response time. The retrieval time for each query transaction was calculated and recorded. The whole steps are repeated and in this time the dataset is indexed inside the database using the shell environment that MongoDB provided for its users and then the simulation is started again to repeat the whole steps and processes that have been done in the first time. Figure 3.2 illustrates the query process model that measures the response time.

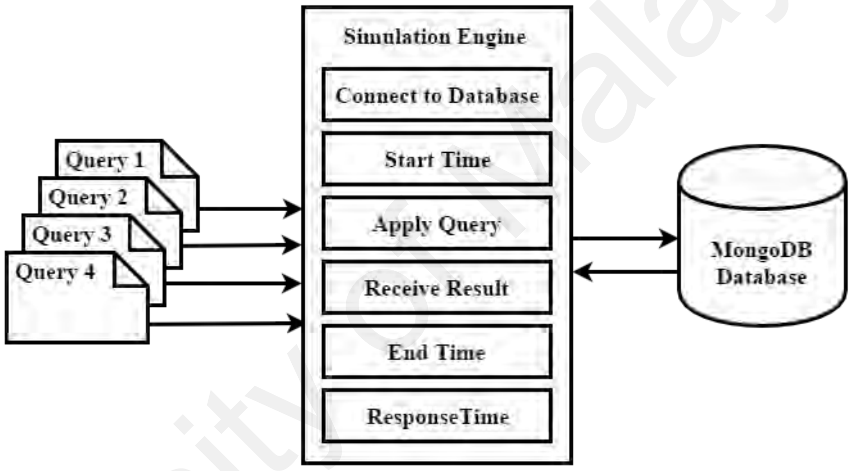

Figure 3.2: Query processing model for measuring response time

The processes that are shown in Figure 3.2 start after functioning connecting the simulation system to the database which is created by MongoDB system. However, the datasets are stored into the MongoDB database before simulation engine runs those functions.

The simulation engine captures the system time in millisecond and stores it in a variable named startTime. This the first check point. Then, the system calls the MongoDB java library and it uses appropriate java classes and functions to apply appropriate query statement against dataset inside the database and receive the results.

University of Malaya

Table 3.1: Queries for the experiments

Query	Description
Query 1	Select all records in which attribute ―country" is equal to ―MY"
Query 2	Select all records in which country is Malaysia and data. Humidity is greater than 80 and temperature is greater than 300
Query 3	Select all records in which country is Malaysia and data. Humidity is less than 50 and wind. Speed is greater than 3
Query 4	Select all records in which country is Malaysia and data. Humidity is greater than 90 and data. Pressure is greater than 1000

3.4 Retrieving Time Performance Evaluation

In this section, the process flow of the time consumption is described. This process is used in simulation engine to apply queries and measure the response time. We also declare how the engine calculates response time. Then, all the recorded results via simulation engine are reported in different tables followed by some analysis and discussions on the results.

Table 3.2: Response times for processing Query 1 without indexing (milliseconds)

No.	240 MB	740 MB	2.1 GB	51.75 GB	438.21 GB	1275.20 GB (1.245 TB)
1	93	7571	20646	508776	4308230	12233892
2	91	7784	20714	510452	4322420	12175173
3	92	7481	20496	505080	4276930	12542829
4	93	7329	19829	488643	4137746	12037903
5	91	7356	20277	499683	4231231	12309877
6	89	7406	21016	517894	4385439	12558513
7	89	7494	21599	532261	4507094	12322288
8	87	7326	20110	495568	4196382	12208494
9	88	7443	21139	520925	4411105	12833185
10	89	7389	20026	493498	4178854	12157499
11	88	7445	19968	492069	4166751	12122288
12	88	7501	20489	504908	4275469	12438579
13	89	7708	20136	496209	4201808	12224278
14	89	7486	20551	506435	4288407	12476219
15	89	7613	20972	516810	4376257	12331802
16	89	7334	21023	518067	4386899	12762763
17	91	7601	21086	519619	4400046	12501009
18	90	7705	20203	497860	4215789	12264953
19	90	7618	19633	483813	4096846	12418914
20	89	7579	20256	499166	4226848	12297128

University of Malaya

Four queries with different complexities are used to evaluate the response time in data retrieval and query processing while no indexing technique was used inside the database. This process repeated 20 times and all response times are recorded. Table 3.2 lists the response times of applying Query 1 for twenty times without using indexing technique.

However, the response times for processing Query 2, Query 3, and Query 4 without using any indexing technique are presented in Table 3.3, Table 3.4, and Table 3.5 respectively.

Table 3.3: Response times for processing Query 2without indexing (milliseconds)

No.	240 MB	740 MB	2.1 GB	51.75 GB	438.21 GB	1275.20 GB (1.245 TB)
1	84	186	415	10227	85599	242941
2	83	187	417	10276	85016	243155
3	85	186	410	10104	84555	248905
4	87	178	423	10424	86268	256797
5	86	180	424	10449	88477	257404
6	83	181	409	10079	85347	248298
7	85	179	413	10178	86181	250726
8	84	182	412	10153	85973	250119
9	86	185	415	10227	86599	251941
10	84	184	411	10128	85764	249512
11	84	186	421	10375	87851	255583
12	84	179	400	10327	83469	242834
13	84	180	403	10218	84095	244656
14	85	186	424	10449	88477	257404
15	84	181	407	10030	84929	247084
16	85	184	421	10375	87851	255583
17	84	178	410	10104	85555	248905
18	84	180	409	10079	85347	248298
19	87	182	422	10399	88059	256190
20	83	178	405	9980	84512	245870

After running the simulation and applying queries on the datasets inside the database without using any indexing technique, an index structure on the datasets is created by using MongoDB system via its shell environment. MongoDB indexes use a B-tree data

University of Malaya

structure(MongoDB, 2016). Table 3.6 shows the list of response times for processing Query 1 when default indexing technique of the MongoDB system is used.

Table 3.4: Response times for processing Query 3 without indexing (milliseconds)

No.	240 MB	740 MB	2.1 GB	51.75 GB	438.21 GB	1275.20 GB (1.245 TB)
1	38	172	373	9192	79834	222443
2	37	180	388	9561	78965	233549
3	39	177	361	8896	75330	219158
4	37	183	363	8945	75748	221372
5	38	171	372	9167	77626	224836
6	38	181	373	9192	77834	223443
7	37	172	364	8970	75956	221979
8	38	178	367	9044	76582	222800
9	38	182	362	8921	75539	219765
10	38	178	358	8822	74704	217337
11	38	171	361	8896	75330	219158
12	39	180	363	8945	75748	220372
13	38	175	363	8945	75748	220372
14	38	172	362	8921	75539	219765
15	38	179	394	9709	80217	239192
16	38	182	374	9216	78043	227050
17	37	179	391	9635	80591	237371
18	38	177	385	9488	79339	233728
19	39	174	378	9315	78878	229478
20	38	180	371	9143	77417	225229

Table 3.5: Response times for processing Query 4 without indexing (milliseconds)

No.	240 MB	740 MB	2.1 GB	51.75 GB	438.21 GB	1275.20 GB (1.245 TB)
1	26	87	199	4904	41526	129810
2	26	89	218	5372	45490	129309
3	26	91	224	5520	46742	129423
4	27	83	213	5249	44447	129512
5	26	87	239	5890	49872	124563
6	25	84	204	5027	42569	125842
7	26	90	209	5150	43612	126881
8	26	87	210	5175	43821	127488
9	26	85	213	5249	44447	129309
10	26	87	212	5224	44238	128702
11	25	89	205	5052	42778	124453
12	26	93	222	5471	46325	126773
13	26	84	210	5175	43821	127488
14	26	87	217	5348	45282	125738
15	27	94	208	5126	43404	126274
16	26	88	204	5027	42569	123845

University of Malaya

Table 3.5: Continued

17	28	87	215	5298	44864	124523
18	27	93	208	5126	43404	126274
19	27	94	210	5175	43821	127488
20	27	90	211	5200	44030	128095

Table 3.7 and Table 3.8 report the response times for processing Query 2, and Query 3 respectively. The results are for the time that default indexing technique in MongoDB system is used and queries are applied via simulation engine.

Table 3.6: Response times for processing Query 1 - using indexing (milliseconds)

No.	240 MB	740 MB	2.1 GB	51.75 GB	438.21 GB	1275.20 GB (1.245 TB)
1	22	6463	19099	470654	3985416	11294730
2	31	6431	19151	471935	3996267	11626299
3	28	6380	20370	501975	4250637	11466336
4	31	6454	19036	469101	3972269	11556484
5	24	6415	18974	467574	3959332	11518844
6	24	6378	18669	460058	3895687	11333683
7	25	6209	18800	463286	3923023	11413211
8	23	6385	19110	470925	3987711	11301408
9	23	6390	18925	466366	3949107	11489097
10	22	6395	19636	483887	4097472	11420735
11	23	6305	18528	456583	3866264	11248084
12	23	6244	18546	457026	3870020	11259012
13	22	6421	18601	458382	3881497	11292401
14	23	6393	18766	462448	3915928	11392571
15	23	6297	18725	461438	3907373	11367680
16	23	6289	18469	455129	3853953	11212266
17	30	6143	18581	457889	3877324	11280260
18	29	6359	18898	465701	3943473	11472706
19	34	6566	18252	449781	3808671	11580528
20	26	6656	18439	454390	3847692	11494053

And finally, Table 3.9 that shows query processing response times for applying Query 4 against the datasets that are indexed by using default indexing method of MongoDB which is B-tree.

University of Malaya

Table 3.7: Response times for processing Query 2 – using indexing (milliseconds)

No.	240 MB	740 MB	2.1 GB	51.75 GB	438.21 GB	1275.20 GB (1.245 TB)
1	26	91	254	6259	53003	154200
2	29	87	293	7220	61141	157876
3	27	85	279	6875	58219	159377
4	22	85	298	7344	62184	150912
5	21	88	253	6235	52794	153593
6	24	86	251	6185	52377	152379
7	21	85	292	7196	60932	157269
8	21	87	299	7368	62393	161519
9	25	86	253	6235	52794	153593
10	21	87	252	6210	52585	152986
11	21	87	252	6210	52585	152986
12	21	90	252	6210	52585	152986
13	21	85	269	6629	56133	153306
14	23	86	290	7146	60515	156055
15	23	84	266	6555	55507	151485
16	26	85	261	6432	54463	158449
17	26	86	256	6309	53420	155414
18	22	87	258	6358	53837	156628
19	25	85	256	6309	53420	155414
20	26	84	266	6555	55507	151485

Table 3.8: Response times for processing Query 3 – using indexing (milliseconds)

No.	240 MB	740 MB	2.1 GB	51.75 GB	438.21 GB	1275.20 GB (1.245 TB)
1	15	81	359	8847	74913	207944
2	14	90	367	9044	76582	212800
3	14	87	351	8650	73244	213087
4	14	80	367	9044	76582	202800
5	14	80	355	8748	74078	215515
6	14	81	347	8551	72409	210659
7	14	81	350	8625	73035	212480
8	15	79	323	7960	67401	196089
9	15	81	325	8009	67818	197303
10	16	86	332	8181	69279	201552
11	15	80	314	7738	65523	200625
12	15	79	317	7812	66149	202446
13	15	81	327	8058	68236	198517
14	15	80	329	8108	68653	199731
15	15	88	330	8132	68862	200338
16	14	84	334	8231	69696	202767
17	14	80	361	8896	75330	209158
18	14	79	335	8255	69905	203374
19	14	80	345	8502	71992	209445
20	15	82	352	8674	73452	203694

University of Malaya

Table 3.9: Response times for processing Query 4 – using indexing (milliseconds)

No.	240 MB	740 MB	2.1 GB	51.75 GB	438.21 GB	1275.20 GB (1.245 TB)
1	17	61	175	4310	36518	108240
2	17	60	173	4269	36100	105026
3	16	62	174	4288	36309	109633
4	16	60	173	4263	36100	105026
5	16	59	174	4288	36309	105633
6	16	60	173	4263	36100	115026
7	16	60	173	4263	36100	105026
8	17	60	172	4239	35891	104419
9	16	59	173	4263	36100	105026
10	16	60	173	4263	36100	105026
11	16	59	174	4288	36309	105633
12	16	60	173	4263	36100	109026
13	16	60	173	4263	36100	105026
14	16	59	173	4263	36100	105026
15	16	60	176	4337	36726	108847
16	16	58	176	4337	36726	106847
17	16	60	175	4313	36518	106240
18	16	59	174	4288	36309	109633
19	16	60	175	4313	36518	106240
20	17	62	174	4288	36309	105633

Based on the results recorded from the experiments in the condition that no indexing technique is used, some visualization diagrams are created to have a clear understanding of the performance of the database system in term of query processing speed when the volume of the data is growing up. Moreover, by using these diagrams we can have a better analysis of the time consumption as one of the resources used for data retrieval process for big data. Figure 3.3 illustrates the performance of the MongoDB system for processing Query 1 without using any indexing technique.

Also, visualization of the response times for processing Query 2, Query 3, and Query 4 without using any indexing technique are imaged in Figure 3.4, Figure 3.5, and Figure 3.6 respectively.

University of Malaya

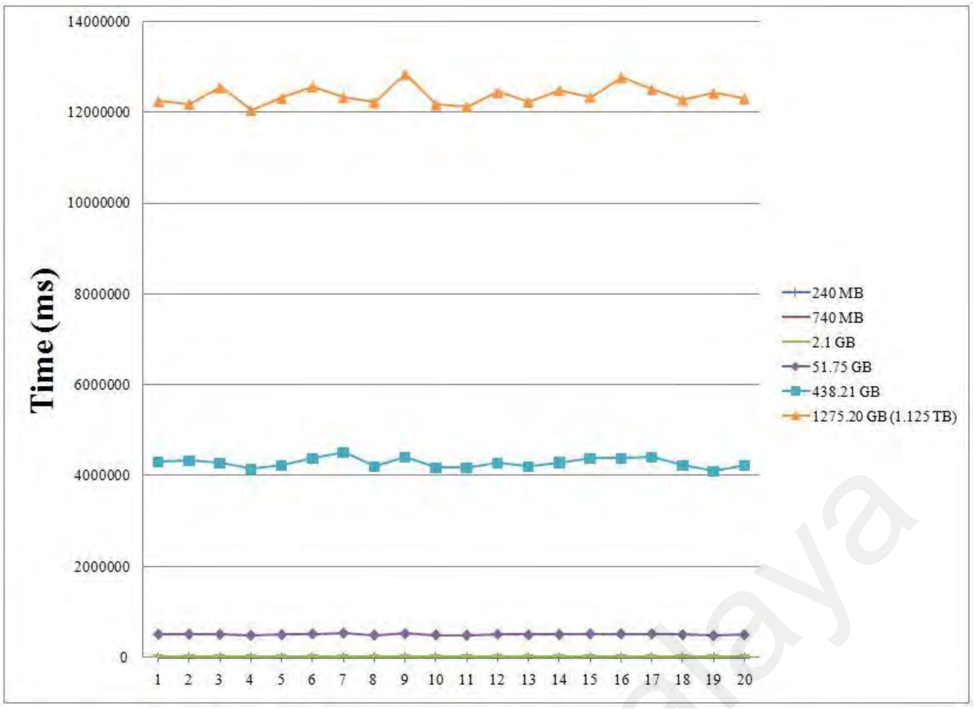

Figure 3.3: Response time for 20 times processing of Query 1 without using indexing

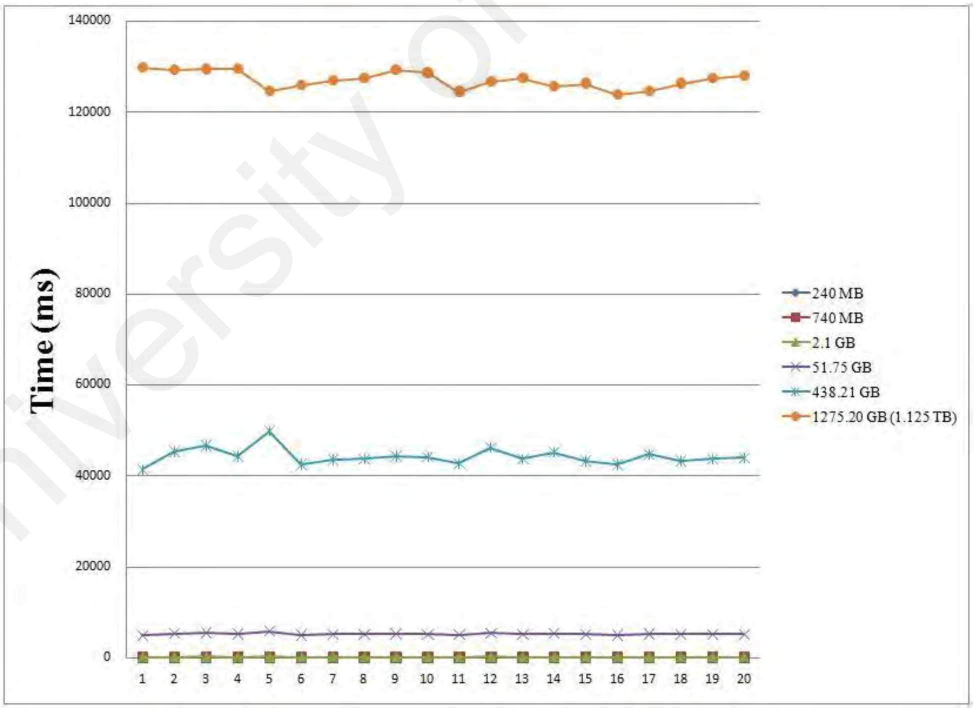

Figure 3.4: Response time for 20 times processing of Query 2 without using indexing

University of Malaya

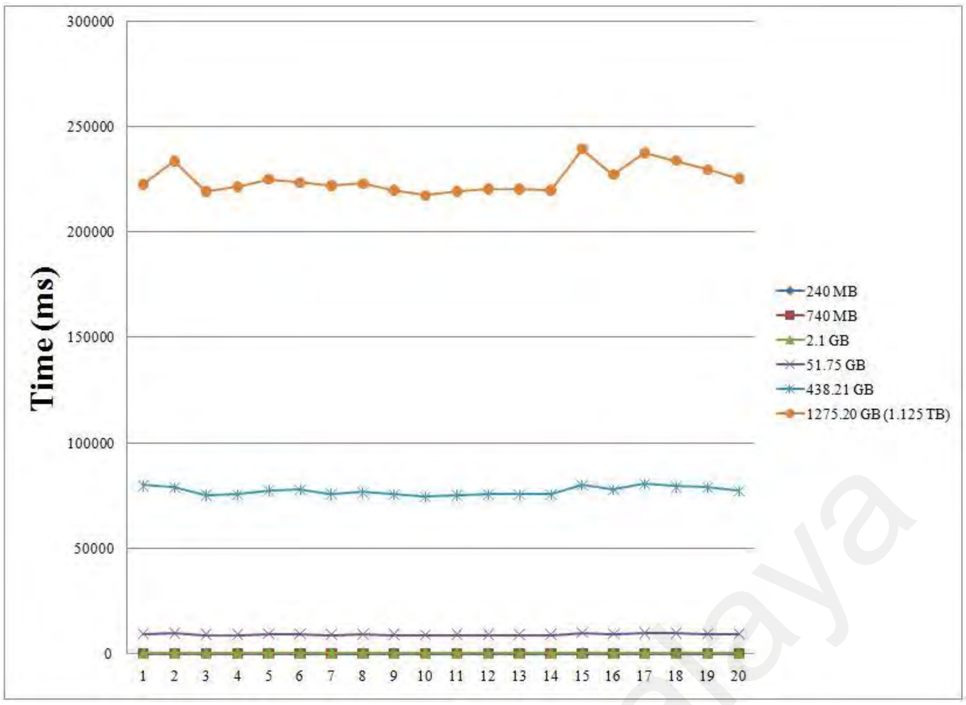

Figure 3.5: Response time for 20 times processing of Query 3 without using indexing

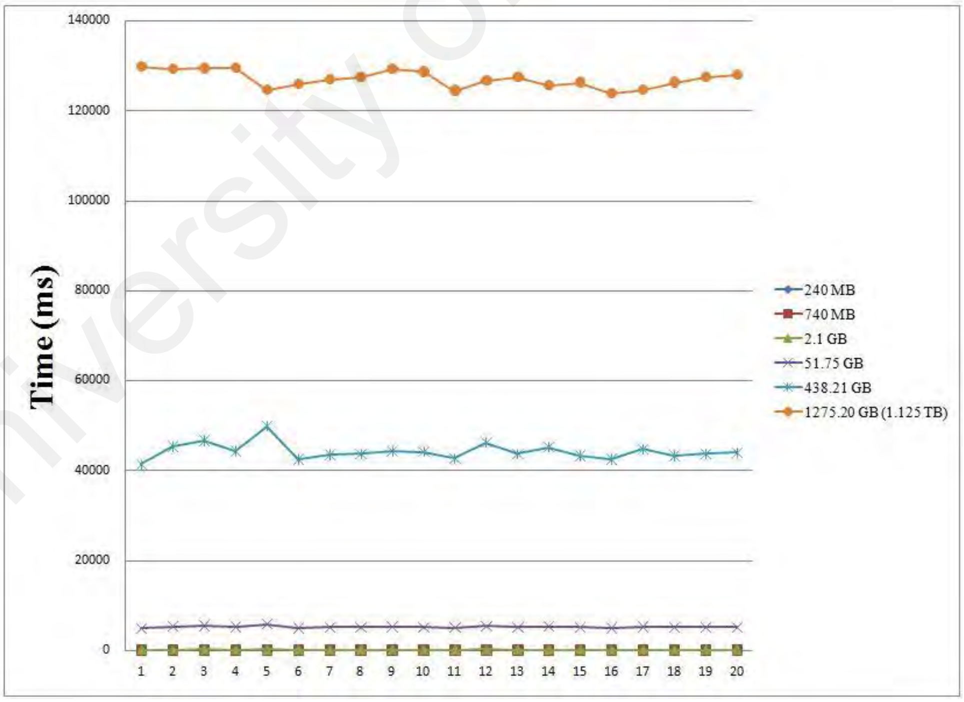

Figure 3.6: Response time for 20 times processing of Query 4 without using indexing

As it can be seen in Figure 3.4, Figure 3.5 and Figure 3.6, every time the simulation engine processes every query, the response time is mostly same and there are no much

University of Malaya

changes in the time that are consumed for processing the query to return the result. However, when the volume of the data is growing up, although the response time for the bigger dataset is increasing, but still the response times' trend for every dataset is linear and the response times are almost same. It means, for each dataset with a specific volume, same queries are applied twenty times, but the time consumption in data retrieval is nearly same. Therefore, the resources, particularly the time that is consumed for processing earlier queries are not reused for processing later queries to be faster and processing later queries consumes same resources specifically the time which is the metric that is used in this study. Therefore, no matter how similar are the queries, the resources like time needed to process them are almost same and when it comes to big data era, based on the above diagrams, it is shown that the data volume grows from 240 MB to 1.245 TB (1275.20 GB or 1305804.8 MB) and in contrast for processing Query 1, the average of response times increases from 89.71ms to 12360879ms. Based on the Equation (3.2) in which V_{Start} is the starting value and V_{End} is the end value, the growth rate of the data volume is about 543985.33% and the growth rate of response time is about 13778608.06%.

$$Percent\ Rate = \frac{(V_{End} - V_{Start})}{V_{Start}} \times 100 \qquad (8)$$

Table 3.10 presents more information about the growth rate of time consumption in data retrieval of our predefined queries in contrast with the growth rate of data volume on our experiments when no indexing technique is used before applying the queries.

As it is in Table 3.10, the growth rate of response time or consumed time for processing Query 1 is about 25.3 times higher than the growth rate of data volume and it means not only the time as one of the resources consumed for processing earlier queries cannot be reused for the later queries even the queries are same (Query 1), but by

University of Malaya

growth of the volume of data, the time needed for processing same queries is potentially and strongly increasing.

Table 3.10: Growth rate of query response time in contrast with data volume (without indexing)

Query	Data Volume (MB)		Average of Response Time (ms)		Growth Rate	
	From	To	From	To	Data Volume	Response Time
Query 1	240	1305804.8	89.71	12360879	5439.85	137786.08
Query 2	240	1305804.8	84.63	250110	5439.85	2954.33
Query 3	240	1305804.8	38.04	224920	5439.85	5911.72
Query 4	240	1305804.8	26.25	127090	5439.85	4840.52

Table 3.11 provides more details about the relation between the growth of data volume and response time needed for processing our predefined queries. Based on the information gathered in this table, we can see as the size of the datasets growing up, the average time consumed for processing the four queries are increasing strongly.

Table 3.11: Response time vs. data volume (without indexing)

Data Volume (MB)	Average Response Time (ms)			
	Query 1	Query 2	Query 3	Query 4
240 MB	89.71	84.63	38.04	26.25
740 MB	7508	182.1	177.15	88.45
2150 MB (2.1 GB)	20508	413.55	371.15	212.55
52992 MB (51.75 GB)	505387	10229	9146	5238
448727.04 MB (438.21 GB)	4279528	85996	77248	44353
1305804.8 MB (1.245 TB)	12360879	250110	224920	127090

To visualize how is the trend of time consumption in data retrieval in processing Query 1 when the volume of data is growing up, Figure 3.9 is created based on the information inside Table 3.11.

Based on figure 3.7, it is clear that when the volume of data is increasing, the time consumption for processing Query 1 is increasing strongly. This behavior is same as the relation between response time of processing Query 2, Query 3, and Query 4. Figure 3.8 illustrates this relationship.

University of Malaya

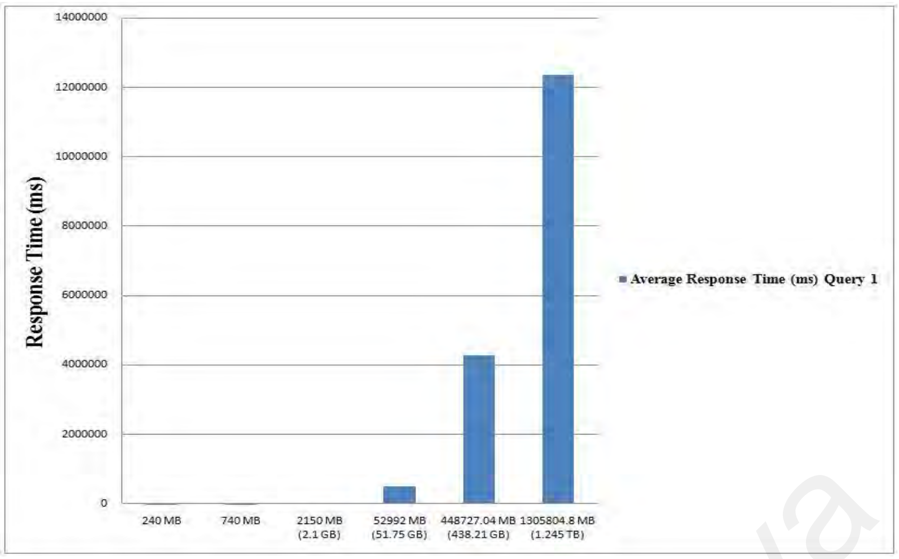

Figure 3.7: Visualization of the relation between response time for processing Query 1 without indexing and data volume

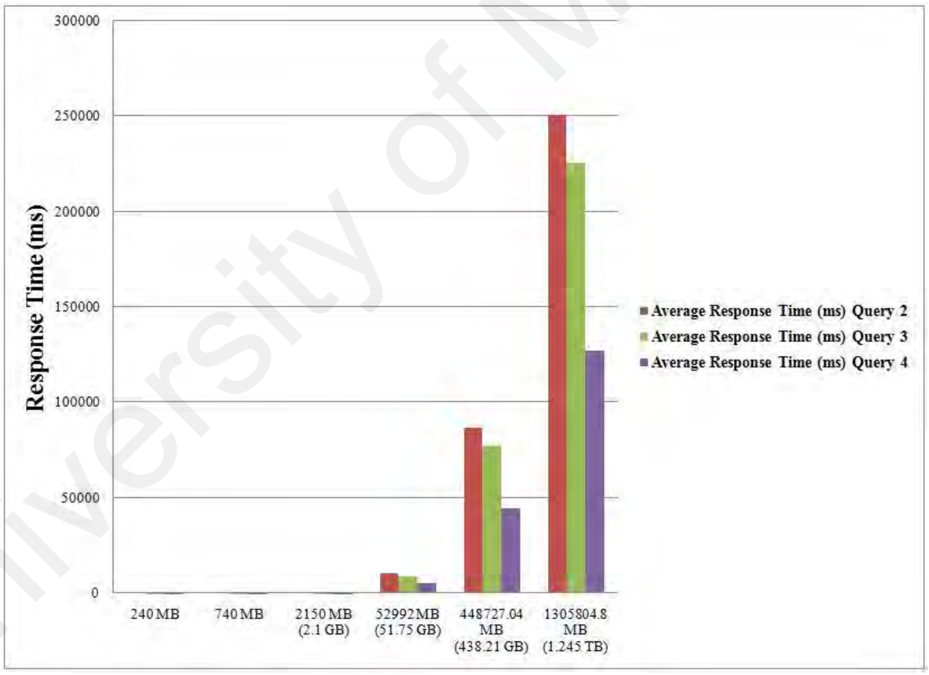

Figure 3.8: Visualization of the relation between response time for processing Query 2, Query 3, and Query 4 without indexing and data volume

As it is explained above, when no indexing technique is used for processing queries, the response time which is the time that is consumed for processing the queries, is almost same for all times that simulation engine applies same queries and the resources particularly the time that is consumed for processing earlier queries do not have any

University of Malaya

impact on the response time of later queries even for same queries. However, by the growth of the size of datasets, the response time is increasing powerfully.

In the first experiments, no indexing technique is used. Now, to figure out what is the impact of using indexing on the consumption of resources particularly on the time consumption in data retrieval for processing our predefined queries, default indexing technique of MongoDB is used inside the database and then all the above processes and steps are repeated. The default indexing technique is B-tree which is the default indexing technique in MongoDB(MongoDB, 2016).

3.5 Execution Time Analysis and Discussion

Base on the results recorded from the experiments in the condition that default indexing technique (B-tree) of MongoDB system is used, some visualization diagrams are created to have a clear understanding of the performance of the database system in term of query processing speed and the trend of time consumption and one of the resources that are used during query processing when the volume of the data is growing up. Figure 3.8 illustrates the performance of the MongoDB system for processing Query 1 in the condition that default indexing technique (B-tree) of MongoDB system is used.

The response time is mostly same with a small variation for each time the simulation engine processes every query, and there is no much change in time consumption for processing the query to return the result. However, when the size of the dataset is growing up, although the response time for the bigger dataset is increasing, but still the response times' trend for every dataset has little variation and the response times are almost same. In another word, same queries are applied twenty times against each dataset with a specific volume, but the time consumption in data retrieval is nearly same. Therefore, the resources, particularly the time that is consumed for processing earlier queries, are not reused for processing later queries to be faster and processing

University of Malaya

later queries consumes same time which is the metric that is used in our study. So, no matter how much the queries are similar, the time as one of the resources needed to process queries are almost same with a little variation and when it comes to big data era, based on the above diagrams, it is shown that the data volume grows from 240 MB to 1.245 TB (= 1275.20 GB or 1305804.8 MB) and in contrast for processing Query 1, the average of response times increases from 25.47ms to 11401019ms. Based on the Equation (3.2) the growth rate of the data volume is about 543985.33% and the growth rate of response time is about 44802898.30%.

Table 3.12 presents more information about the growth rate of time consumption in data retrieval of our predefined queries in contrast with the growth rate of data volume on our experiments when indexing technique (B-tree) is used before applying the queries.

Table 3.12: Growth rate of query response time in contrast with data volume (with indexing)

Query	Data Volume (MB)		Average of Response Time (ms)		Growth Rate	
	From	To	From	To	Data Volume	Response Time
Query 1	240	1305804.8	25	11401019	5439.85	456039.76
Query 2	240	1305804.8	23	154896	5439.85	6733.61
Query 3	240	1305804.8	15	205016	5439.85	13666.73
Query 4	240	1305804.8	16	106812	5439.85	6674.75

Based on the Table 3.12, the growth rate of response time for Query 1 which is the consumed for processing Query 1 is about 83.8 times higher than the growth rate of data volume. Therefore, the time as one of the resources used for processing Query 1 in earlier attempt cannot be reused for a later attempt for processing the same query (Query 1). However, when the volume of data is growing up, time consumption for processing Query 1 is strongly increasing. This behavior is same for other queries. Table 3.13 provides more details about the relation between the growth of data volume

University of Malaya

and response time needed for processing our predefined queries. Based on the information gathered in this table, as the size of the datasets growing up, the average time consumed for processing the four queries are robustly increasing.

Table 3.13: Response time vs. data volume (with indexing)

Data Volume (MB)	Average Response Time (ms)			
	Query 1	Query 2	Query 3	Query 4
240 MB	25	23	15	16
740 MB	6379	86	82	60
2150 MB (2.1 GB)	18879	268	341	174
52992 MB (51.75 GB)	465226	6592	8403	4283
448727.04 MB (438.21 GB)	3939456	55820	71157	36267
1305804.8 MB (1.245 TB)	11401019	154896	205016	106812

Figure 3.9 visualizes the trend of time consumption in data retrieval and processing of Query 1 while the size of the dataset is growing up based on the information inside Table 3.10.

Figure 3.9 shows that by increasing the volume of data, the time that is consumed for processing Query 1 is increasing powerfully. This behavior is same as the relation between response time of processing Query 2, Query 3, and Query 4. Figure 3.10 shows this relationship.

University of Malaya

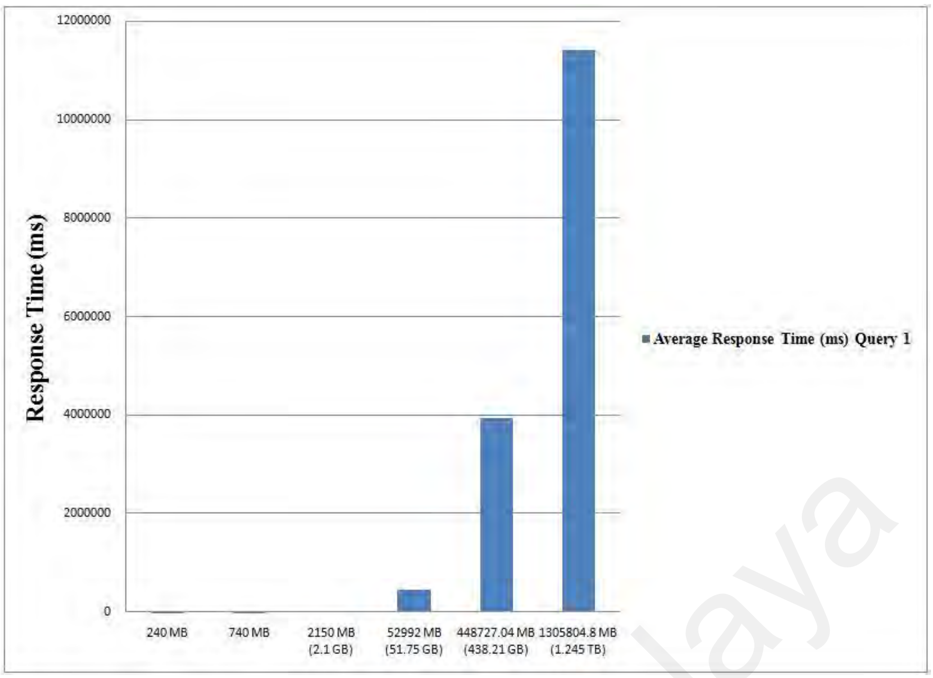

Figure 3.9: Visualization of the relation between response time for processing Query 1 with indexing and data volume

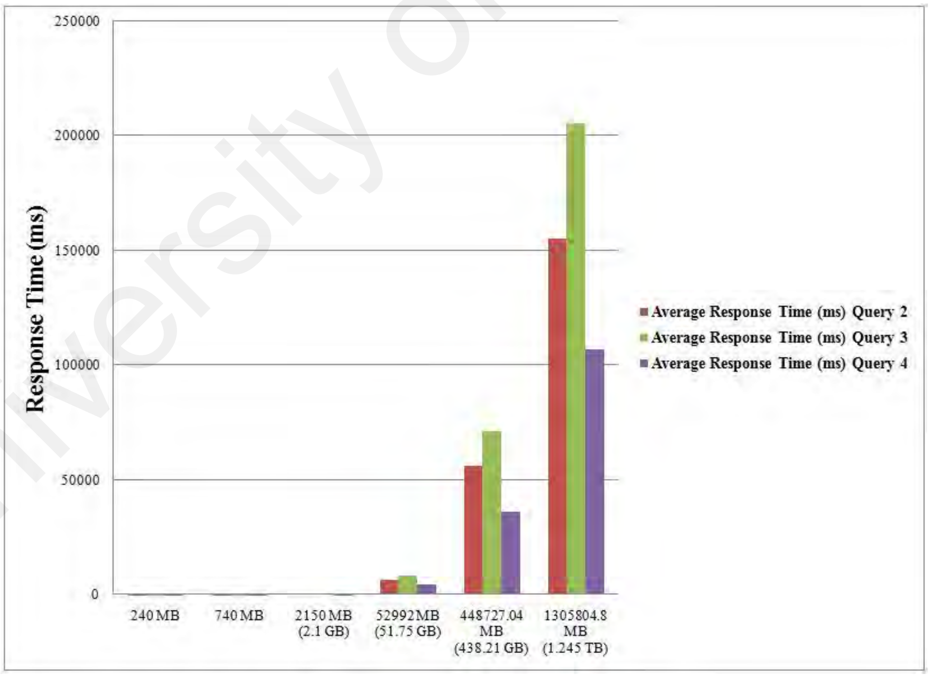

Figure 3.10: Visualization of the relation between response time for processing Query 2, Query 3, and Query 4 with indexing and data volume

So, when the default indexing technique (B-tree) in MongoDB is used for processing queries, the response time which is the time that is consumed for processing the queries,

University of Malaya

becomes better than the time that no indexing technique is used. But every time the same query is applied and the response time is almost same with earlier attempts that simulation engine applies the same query and the resources, particularly the time that is consumed for processing earlier queries, do not have any impact on the resources consumption particularly on the response time of later queries. Moreover, by the expansion of the volume of the datasets, the response time is increasing impressively.

3.6 Conclusion

In this chapter, an experimental analysis and a formal analysis are conducted in order to verify the problem that this study focuses on. In the experimental analysis section the performance of one of the current database systems is analyzed. This database system is able to store and manipulate big datasets and also the effects of multiple applying same and similar queries on the same dataset are studied to find out the capabilities of those systems to reuse the resources such as the time and computational processes used for earlier queries in order to improve data retrieval in later queries. Particularly, we used time among the resources that are used in data retrieval as our measurement. In this regard, our test environment, data collection method, datasets, and also requirements of the experiments are described.

We developed a simulation engine to connect to the selected database system, apply queries, calculate the response time and return it to the user. Four predefined queries and 6 sizes of selected datasets are used. The result shows that the response time which is the time consumed for processing the queries, is almost same for all times that simulation engine applies same queries and the resources particularly the time consumed for processing earlier queries do not decrease the required resources particularly the required time for processing later queries even when the queries and the

University of Malaya

datasets are exactly same.. However, the results show that by the growth of the size of datasets, the response time of every query is increasing impressively.

Moreover, we analyzed B-tree structure formally and resulted that repeating any number of this search operation to find key *k* in the B-tree *T*, consumes the same amount of time and other resources such as computational power as the first search operation and there is no capability in B-tree structure to help later search operations to benefit from earlier search operations.

In the following chapter, a modification in B-tree search method is proposed to speed up the search process and get benefit from earlier queries by adding new elements in the nodes of the B-tree and store some metadata in each node. This metadata is reusable for later queries and minimize the query execution time. Our proposed solution aims to use summarization and reusability concepts inside B-tree structure to reduce resource consumption particularly time consumption of processing queries.

University of Malaya

CHAPTER 4: IMPROVED SEARCH METHOD FOR BIG DATA SETS

4.1 Introduction

This chapter describes in details the improved method for indexing big data using a summarization technique and history value which offers reusability to be added to the search algorithm in order to reduce the number of comparisons required to be done by search algorithm to speed up the search process and address the problem of low efficiency and poor performance of the current indexing techniques, which are used in big data retrieval. The proposed search method is an extension of B-tree search method with the same index structure and uses a Min-Max summarization technique to add a summary of data stored inside the sub-trees under each node. Such approach helps to optimize search process by reducing the number of comparisons required to be done to find the result of the search. Moreover, this method helps later queries to benefit from the results of earlier queries via a history object which is added to each node of the indexing structure upon every search operation.

In this chapter, an improved indexing method based on B-tree structure is proposed to optimize resource consumption in data retrieval for big data. Also, we focus on efficiency of search execution time to improve the overall efficiency of big data indexing procedure.

Section 4.2 elaborates an improved B-tree indexing technique which is an extension of B-tree indexing technique. Also, the architecture of the system that is used for running and evaluating our proposed search method is illustrated. In the following of this section, the components of the system are described. Section 4.3 presents the structure of the improved indexing technique. In Section 4.4, we explain the data retrieval architecture and highlight the layer that our proposed search method is located. Section 4.5 describes the simulation engine that is used for benchmarking experiments.

University of Malaya

The process of search tasks in the proposed search method is detailed in Section 4.6 and finally, Section 4.7 provides the summary of this chapter and also highlights some of the advantages of the proposed method.

4.2 Improved B-tree Technique

Improved indexing technique is an extension of the B-tree indexing technique which performs basic operations such as Insert, Update, Delete, and Search. The proposed system to run and evaluate our proposed method consists of different components. The idea is to add a new element to each node of the tree structure of the indexed data to store the maximum and minimum value of the sub-tree under the node. In this approach, we name this new element as meta-data of a node as it carries some information about the data inside the sub-tree of the node.

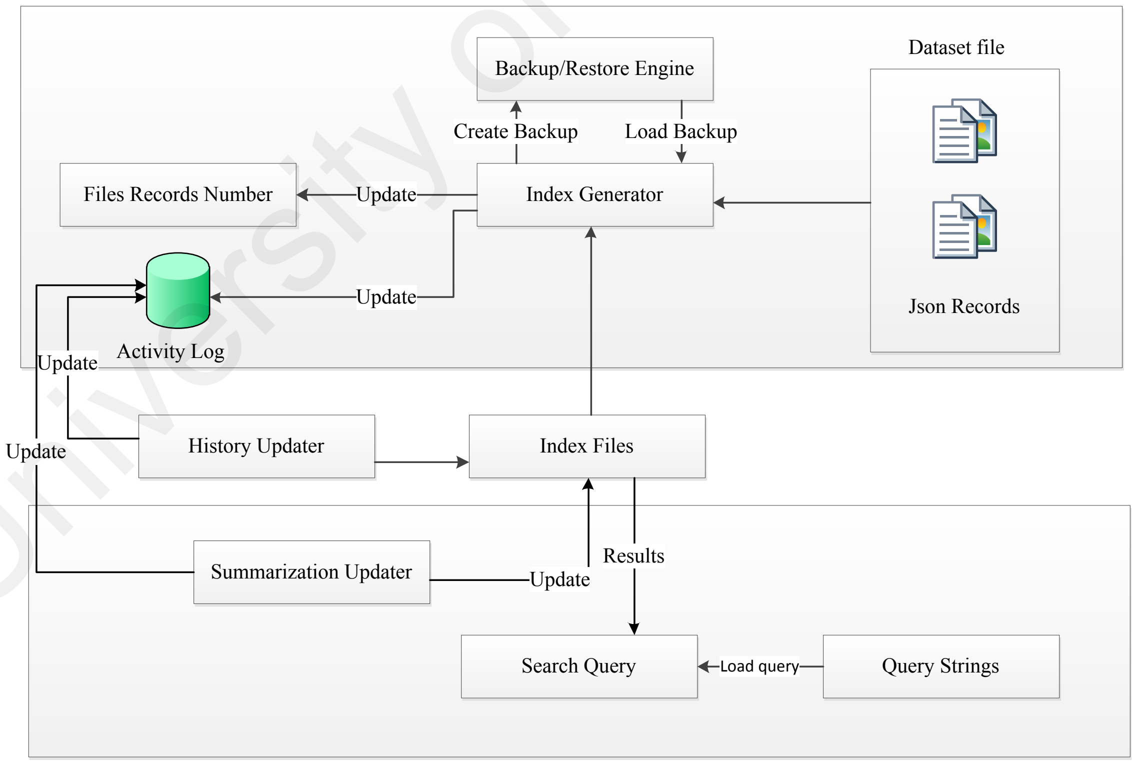

Figure 4.1: Improved big data retrieval system architecture

University of Malaya

The objective of this research is to improve search process in big data based on B-tree structure. Thus, to enhance the performance of the B-tree search, two techniques have been used, first a min-max summarization technique which adds two values to each non-leaf node to store minimum and maximum values of the sub-tree of the node. The second technique is using the last history of the last search to get the benefit of earlier queries for speeding up the next search processes and decreasing the execution time of later queries.

In fact, the first technique which is a min-max summarization is running after index data is ready and before processing the first query. Then during search process, the modified search algorithm checks the result of the min-max summarization process to reduce the number of comparisons. However, every time a query is processed by the modified algorithm, the history updater updates the history value in meta-data of some of the traversed nodes. When search process is traversing the B-tree structure, it checks history value and if the current query is same as previous query or they have some overlaps, it uses the stored pointer of every last right or last left traversed node by previous query and jumps to that node and reduces the number of required comparisons rather than previous query. This procedure is illustrated in Figure 4.5.

An improved data retrieval system with 6 main components including Index Generator, Summarization Updater, Data Storage, Backup Engine, Search Query Engine and History Updater is proposed. This System uses an improved B-tree indexing technique in which we use our proposed search method to speed up the query processing and reduce execution time of the search operations by using Summarization Updater and History Updater to add and update metadata to the node of the structure of the indexed data. However, this system can make a backup of the index data and store it into a file into the physical storage and reuse for further search process in future. This

University of Malaya

feature and also added metadata to the nodes provide reusability of the previous indexing process and also search operations' results. Figure 4.1 illustrates the architecture of our improved big data retrieval system. The descriptions of each component are provided as following.

4.2.1 Index Generator

This engine is responsible to read files from Data Storage and generates index structure based on the predefined attributes. However, it can load a backup of previous index data and update it based on new dataset files instead of regenerating index structure using the old datasets and indexing new dataset files.

4.2.2 Summarization Updater

Min-Max algorithm has been used widely in many areas such as Neural Network, Video Processing, Smart Energy System, and sensor networks. In order to optimize the search process of the B-tree technique, a min-max summarization technique is also used after creating an index over attribute values and before the first time query processing . By using this technique, two values will be added to each node of the B-tree. The first value is representing the minimum value of the sub-trees of the node and the second one is the maximum value of the sub-trees of the node. Figure 4.2 shows a portion of a B-tree structure with the order of 3 before using summarization technique and Figure 4.2 illustrates how min-max summarization values are added to the nodes of the same portion of the B-tree structure.

University of Malaya

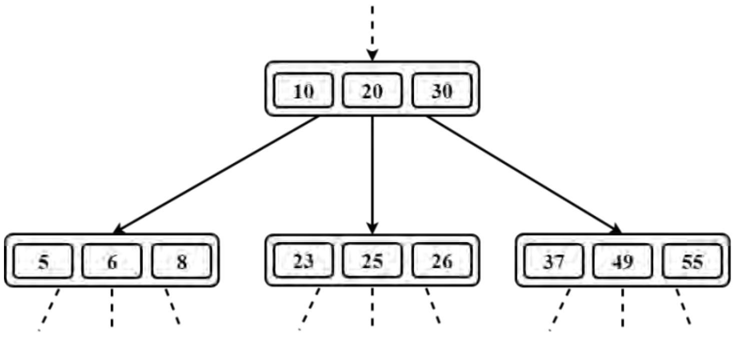

Figure 4.2: A portion of a B-tree structure before running min-max summarization

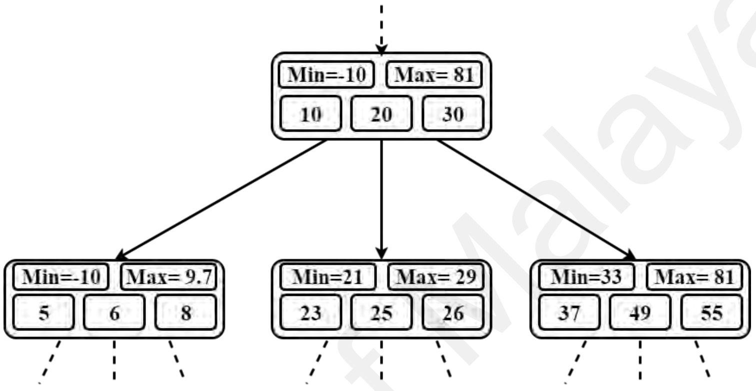

Figure 4.3: A portion of a B-tree structure after running min-max summarization

University of Malaya

The min and max values resulted from running min-max technique will be reused for later queries. It means this summarization technique is running one time before first search task and can be reused many times. If we assume ST_1 is the first search task, $T(ST_1)$ is the time needed for running ST_1, $t_{min\text{-}max}$ is the time required to run min-max algorithm, and t_{Q1} is the time required for searching indexed data to find the results for Query 1, then total consumed time for doing ST_1 and processing Query 1 will be an accumulation of $t_{min\text{-}max}$ and t_{Q1}.

$$T(ST_n) = t_{min-max} + t_{Qn}(n = 1)$$

Also, whereas min-max technique is not running for later queries, $t_{min\text{-}max} = 0$ and $T(ST_1)$ is equal to the time used for processing next queries.

$$T(ST_n) = t_{Qn}(n > 1)$$

Based on the above explanations, the Summarization Updater is responsible of adding a min-max summarization data into each node of the B-tree structure of the index data. This metadata provides minimum and maximum values stored inside the nodes of the sub-tress under every node and reduces the number of comparisons operation of the search algorithm during data retrieval. It fact search algorithm checks the availability of the key that is looking for inside the branch of the tree structure before starting searching the branch‘s nodes.

4.2.3 Data Storage

To test our technique, a simulation engine is developed to read dataset files in which every line is representing a record of a file based database. Every line includes semi-structured data with json format. Before starting reading file and splitting every record, the list of attributes that indexes must be created based on them is defined. For instance, to create an index to search records to find a specific country, we can define an object from a class with name BTree which using B-tree structure.

$$Btree < String > CountryIndx = new\ Btree < String > (OrderOfTree$$

After the system is opening the file, it starts reading the file, line by line using a while loop. If a line is not null, it calls a method with name ReadOneRecord to read a record (line) as string and stores it into an object with name Record in order to split it and extract main parts for indexing purposes.

$$Record = ReadOneRecord(line);$$

However, the position of each record inside the file will be captured as a pointer of the record to be used inside the index for linking the index and related records by calling a method with name setStartPositionInFile.

University of Malaya

$$RecordsetStartPositionInFile\ (LinePosition)$$

$$LinePosition \mathrel{+}= line.ength() + 1$$

Now the system creates an array list with name data and stores extracted parts inside the array list.

$$ArrayList < Data > data = Record.getData();$$

Before starting indexing process for current record (line), the engine captures the current time in Milliseconds.

$$startTime\ \ = System.currentTimeMillis();$$

Then, it starts Indexing process to add the record into the index structure using one of the attributes of the record. For example, to create an index based on the attribute "country".

$$CountryIndex.add(Record.getCity().getCountry, Record.getStartPositionInfile();$$

After this process, the engine calculates the elapse time and adds it to the total elapse time.

$$elapsedTimeTmp \mathrel{+}= System.currentTimeMillis() - startTime;$$

When the while loop is ended, the engine will show the total time consumed for indexing whole datasets. Figure 4.4 presents the flow of the process of reading data from dataset files.

University of Malaya

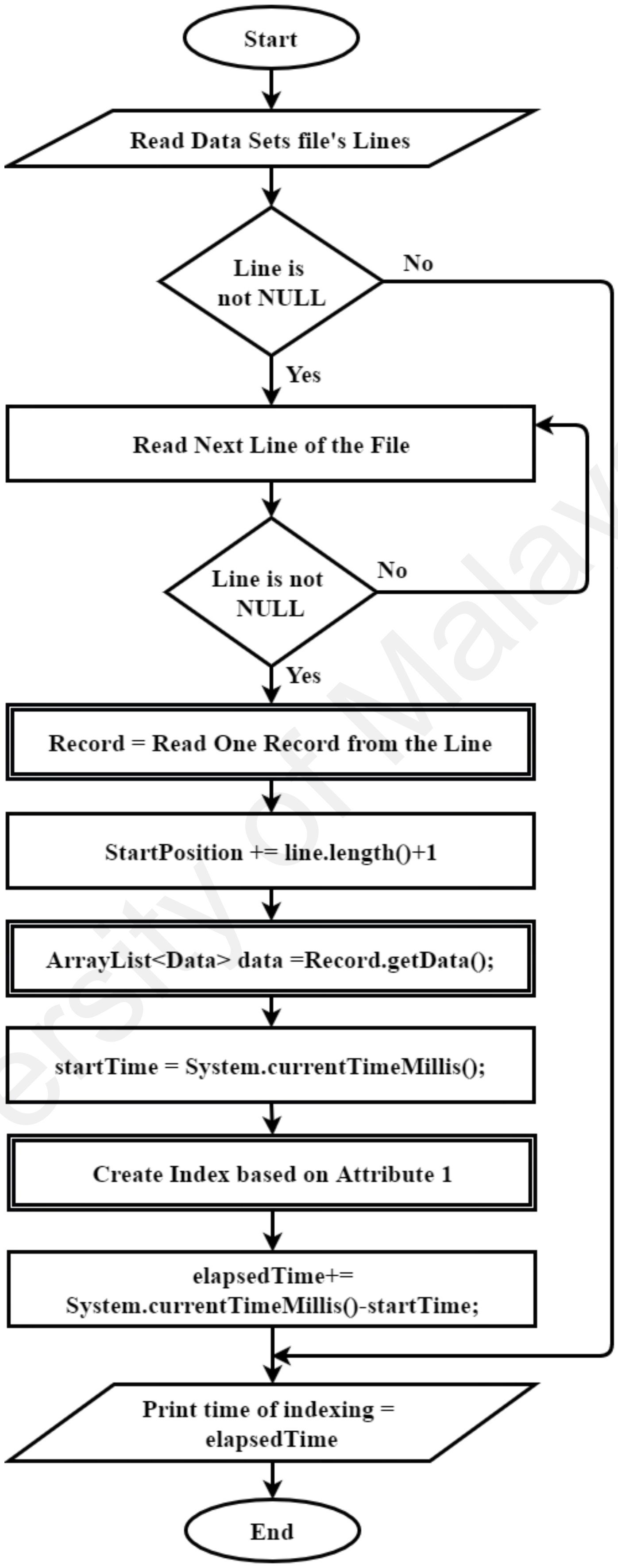
Start
Read Data Sets file's Lines
Line is not NULL
No
Yes
Read Next Line of the File
Line is not NULL
No
Yes
Record = Read One Record from the Line
StartPosition += line.length()+1
ArrayList<Data> data =Record.getData();
startTime = System.currentTimeMillis();
Create Index based on Attribute 1
elapsedTime+= System.currentTimeMillis()-startTime;
Print time of indexing = elapsedTime
End

Figure 4.4: flow of the process of reading data from dataset files

University of Malaya

Algorithm 1: Improved B-tree indexing method

Required

Index attributes Country, Humidity, Pressure, Wind Speed, Day Temperature

```
10. While $(line != Null)$
11. Read $file;
12. If$(line\ isempty)$
13. Break;
14. Else, $set$ line as one recordEnd if
15. $set\ startPosition\ as\ start\ position\ of\ Record\ in\ datasetFile$
16. ArrayList [Data record
17. Set → startTime in milliseconds
18. Set → $endTime\ in\ milliseconds$, Calculate elapsedTime , Print elapsedTime
19. End.
```

4.2.4 Backup Engine

This engine makes backup from the index data after indexing every dataset file is finished. In fact, this engine makes a backup of the index data and stores it into a file into the physical storage. This capability offers reusability of index data for further search processes in the future and reduces the time of generating indexes by eliminating re-indexing the dataset files that are indexed in the past.

4.2.5 History Updater

This function stores and updates an extra value inside a node. This new element is actually a pointer that represents the last history of the search route map and it points to a child node of the current node that during the last traverse of the branch which has that child, the direction of the search path has been changed from left to the right or vice versa. Therefore, in the next search query, if the search key is same, in the normal B-tree search process, the search function traverse whole path that previous search task has done. But, in our proposed method, the search function jumps from current node to one of its child nodes using the history value stored in the node from last search operation. Figure 4.5 illustrates how history updater stores and updates the history value inside some of the traversed nodes during last search process. Blue dashed arrows show the

University of Malaya

path of the search and green dotted arrows show how history values or pointers are pointing the nodes to make shortcuts for next search process.

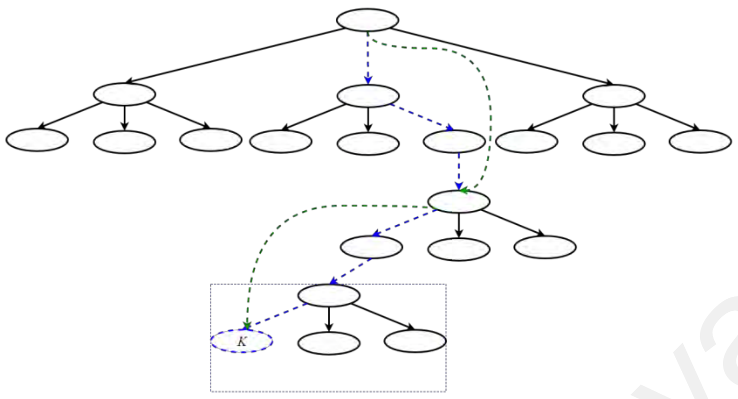

Figure 4.5: History updater stores/updates pointers as shortcuts for next search process

History Updater function is running during every search process and stores or updates the history value HistoryValue and HistoryNode inside related node in order to reduce the number of comparisons of the search algorithm in the next search tasks. In the second search and later on, History Updater updates the history value.

4.2.6 Search Query Engine

The basic idea of the search process in a *n-key* B-tree is comparing the value that a query is looking for with the value of every internal node x of the B-tree to make a decision among an $(n[x] + 1)$*-way* branching decision in order to find out on which branch of the sub-tree under the node x, the target value might be. B-tree search algorithm directly generalized from the Tree search algorithm.

The search algorithm compares the target value k with every value $key_i[x]$ inside the node x, if it is equal ($key_i[x] = k$), the algorithm returns an ordered pair (x, i) in which x is the node and i is the index of the related key. If the target value k is less than $key_0[x]$, then the algorithm start traversing the left branch of the sub-tree at the left side of the key_0. If k is greater than $key_0[x]$, then the search algorithm repeats the above procedure for the next key (key_{n-1}) of the node x. If there is no result found, the algorithm traverses

University of Malaya

the most right branch of the sub-tree bellow the node *x.* The above process is repeating for every internal node of the B-tree and if there is no result fount, the algorithm returns null. Algorithm 2 presents a simple pseudo code of a normal B-tree search process without our modifications in which we are looking for value k in a node x of a B-tree T with order n.

Algorithm 2: Improved B-tree indexing method

```
1. **Begin**
2. **Get** node $x$ of the B-tree $T$ with order $n$
3. $i \rightarrow 0;$
4. **While** $(i < n[x]) \& \; value \; (k \geq key_j[x])$
5. **Set** $\boldsymbol{i \leftarrow i + 1}$
6. $\boldsymbol{if} \; i < n[x] \& \; value \; k = key_i[x]$
7. **Return** *null*
8. **Set**$Get \; c_i[x] \; top \; node \; of \; the \; left \; branch \; under \; key_i$
9. **End**.
```

Algorithm 2 of a B-tree search process without proposed modifications

As you can see in Figure 8, every time a search task is looking for a key in a node *x*, if the key is not among the nodes' keys, it starts traversing branches of the sub-tree regardless of knowing the value is inside the sub-tree of the node. The first part of our modifications on the B-tree search shows its role here by adding a Min-Max summarization to each node indicating that what are the minimum and maximum values of the sub-tree of the node *x* before start first search task. Then, during search procedure, before starting comparing the keys of the node *x* or traversing sub-tree of the node *x*, it checks the min-max values and if key k is between these two values, it starts searching the node *x* and its keys and sub-tree. But, if the key *k* is greater than the maximum value, the search algorithm ignores that node with the whole sub-tree from applying search procedure. Algorithm 3 shows the modified search algorithm using min-max values added to the node *x* upon summarization process.

University of Malaya

In addition to Min-Max summarization checking, our technique provides a history of the last search in every index structure that has been created and used from previous query processing. So, if the current query is same or has overlapped with the previous query, the search algorithm will get benefit from history element s inside some of the nodes inside the index structures that have been used by the previous search process. For example, if the previous query was looking for every record that attributes –Country” is Malaysia and humidity is 90, history element inside indexed data of the country attribute can help for any other queries that are looking for –Malaysia” as a country name. Therefore, our modified search algorithm can get the benefit of history element by jumping from a parent node to one of the child nodes that exactly has the value –Malaysia” or gives the next child that search process should jump in to find the value –Malaysia”. Figure 4.6 images the flowchart of the modified search process.

Algorithm 3: Improved B-tree indexing method

```
1.  Begin
2.  Get node $x$ of the B-tree $T$ with order $n$
3.  $Get\ Min_{node(x)} \& Max_{node(x)}$
4.  If $(Min_{node(x)} \le k \ \& \ Max_{node(x)} \ge k)$
5.  Set $\boldsymbol{i \leftarrow i + 1}$
6.  $\boldsymbol{While}\ i < n[x] \& \ value\ k = key_i[x]$
7.  Set $i \leftarrow i + 1$
8.  If$i < n[x] \& \ value\ k = key_i[x]$
9.  Return Null
10. $\boldsymbol{If}\ node(x)$ is a leaf node
11. Return Null
12. Else
13. Set$Get\ c_i[x]\ top\ node\ of\ the\ left\ branch\ under\ key_i$
14. End, else
15. $Get\ node\ (x)\ parent\ of\ node\ x$
16. $Continue\ search\ process\ in\ node\ p_x\ to\ find\ k$
```

University of Malaya

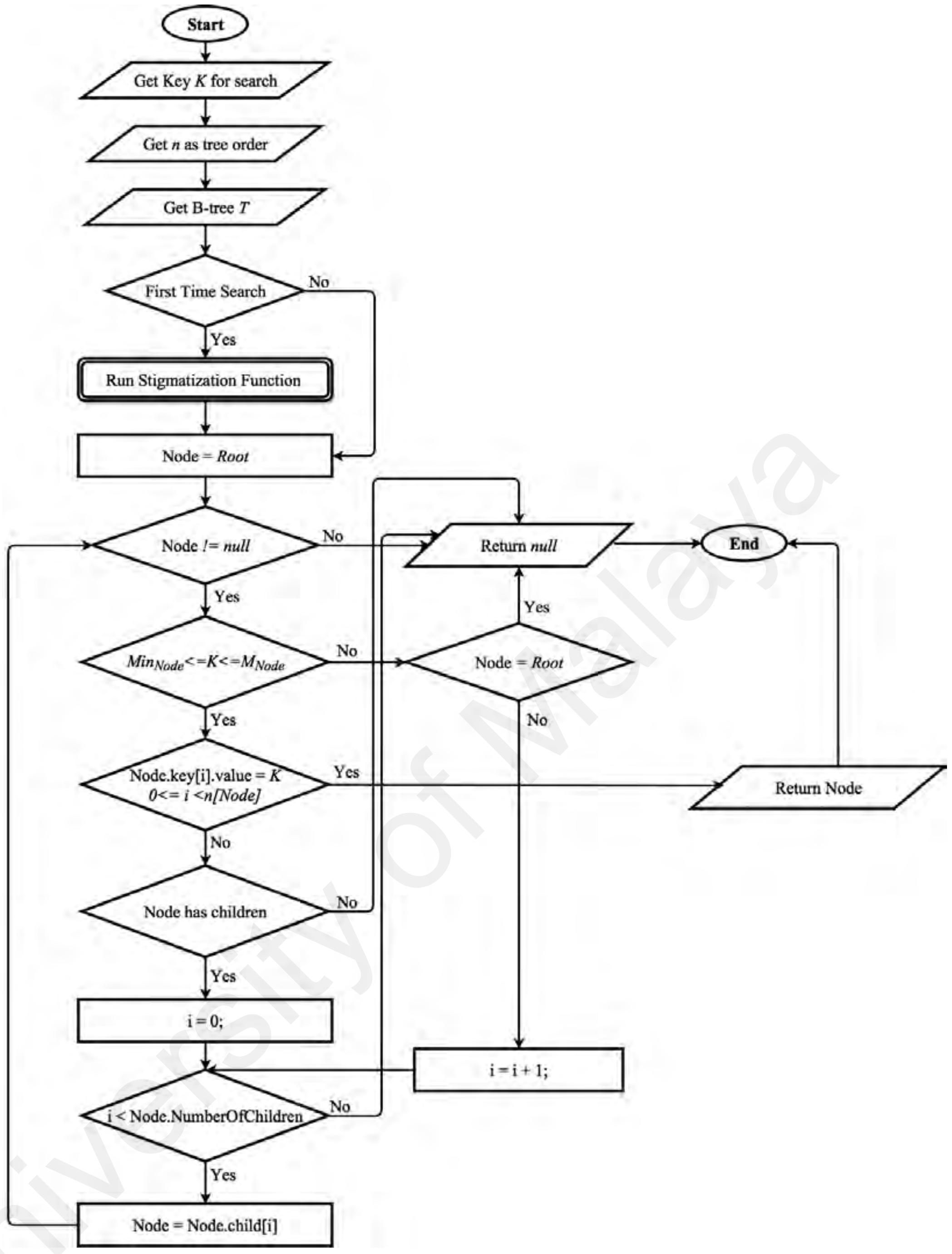

Figure 4.6: Flowchart of our modified B-tree search process

4.3 IB-tree Indexing Structure

In a tree structure, every node consists of two elements. One element is carrying attribute value representing a data record and the other element is storing a pointer that links the data to the location of the data. A B-tree structure is a balanced search tree which works well on disks or other direct-access secondary storage devices. The

University of Malaya

proposed solution of this work is suggesting adding a new element to each node of the tree structure of the index to reduce the number of comparisons that normally must be done by search algorithm. In our proposed method, the metadata will be added to nodes the tree structure of the index right before the first search task operates. Figure 4.7 demonstrates a modified and improved B-tree structure based on the proposed solution of this research. We named this indexing structure IB-tree which stands for Improved B-tree.

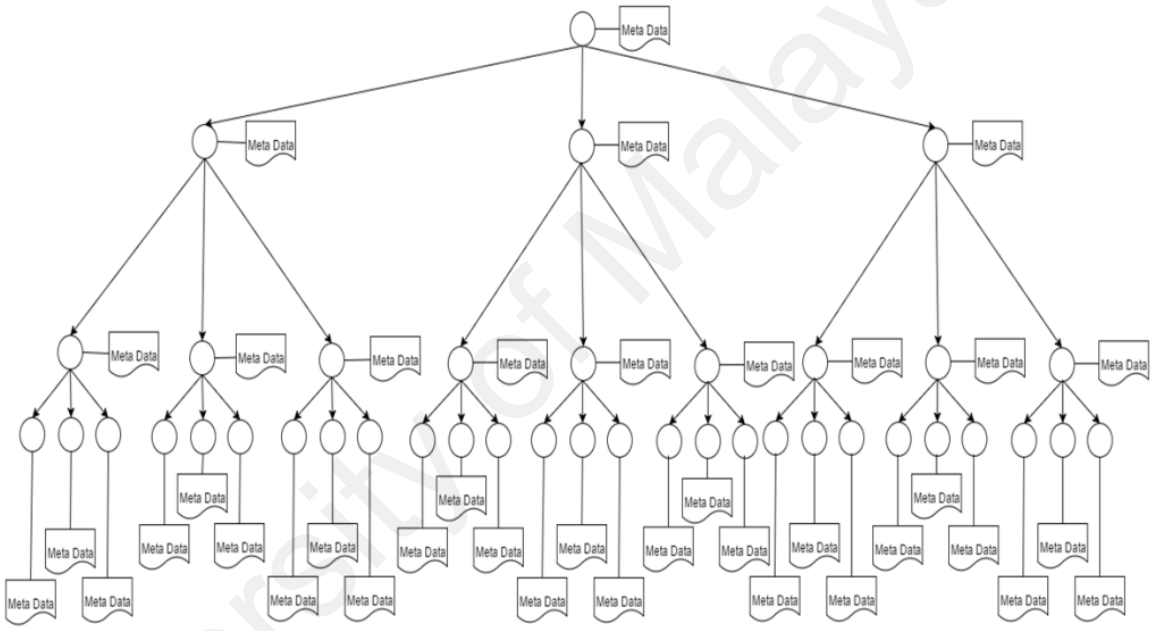

Figure 4.7: IB-tree indexing structure

4.4 Data Retrieval Architecture

Figure 4.8 shows a data retrieval architecture presenting the main layers and components of a data retrieval system. It also highlighted the indexing layer in which our proposed solution is located.

University of Malaya

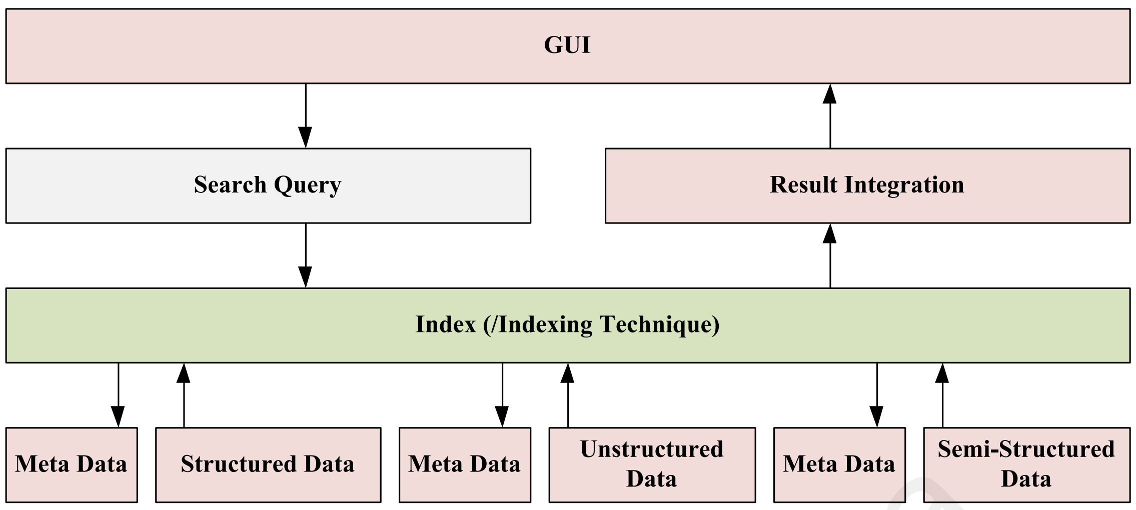

Figure 4.8: Data retrieval architecture

When the user wants search for information from big data. Firstly, the users submit the query via graphic user interface that incorporates with search query engine. Then, the query will be processed using improved B-tree indexing technique. The datasets can be retrieved in a deferent format which content structure, unstructured or semi-structured. The system also stored meta-data that can be used during the indexing process.

4.5 Simulation Engine

The simulation engine is responsible for reading dataset files, splitting each line of the file and extracting internal objects, creating index structure based on the predefined attributes, running queries against indexed data, calculating the elapsed time to execute search process over indexed data, and returning the elapsed time plus the result of the search to the upper layer. Whereas index creation process for our proposed method is same as B-tree search method, every time a search task is requested, simulation engine calls required indexed data and applies queries using two search algorithms; B-tree search algorithm and improved B-tree search algorithm. Figure 4.9 presents the flow of the search process using our simulation engine when no modification is applied.

University of Malaya

However, Figure 4.6 demonstrates the flow of the search process using our proposed method.

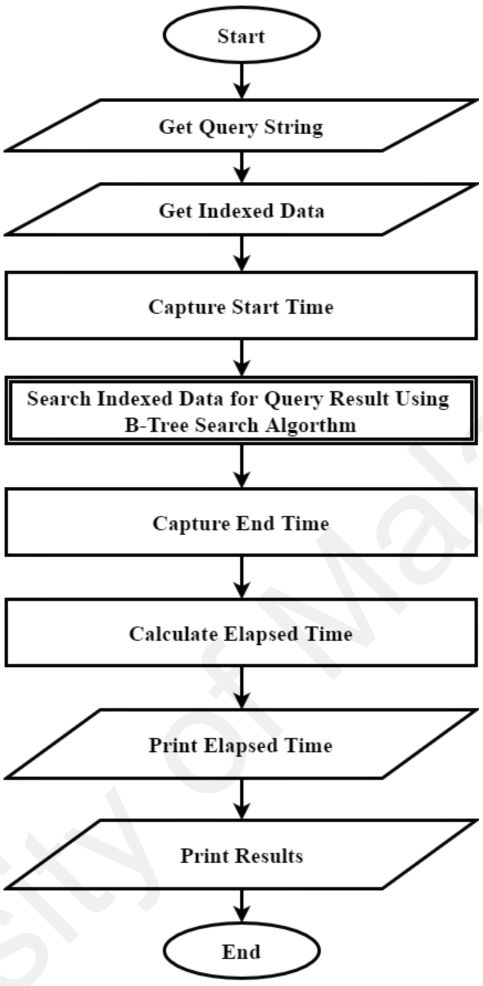

Figure 4.9: Search process flow of the simulation system using B-tree search algorithm

University of Malaya

4.6 Search Operation Flow

Every time the search operation is running to apply a query against data, it checks every node's metadata first before going inside the branch of that node (if there is any). If metadata has key value to help the system to figure out the result is inside that node (as leaf node consist of data) or the branch (sub-tree) under the node (as parent node), the system follow the value and do related process to retrieve the result from the node or its branch (sub-tree), or ignore that node and its branch (sub-tree) and go back to the parent node and check other child nodes/neighbors if there is not any helpful value inside the metadata, the query process continue by searchingthat node and its branch (sub-tree) to see the result is there or not. In this case, the system willupdate the metadata of the node based on the result of the search.

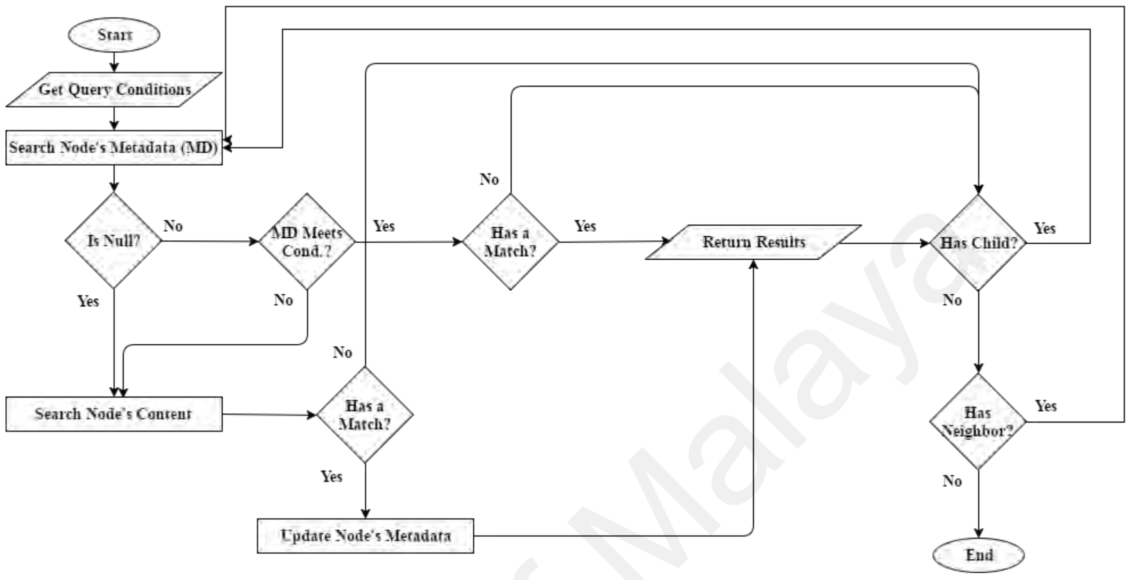

Figure 4.10: Search operation flowchart

Query by query the metadata become more enriched and more useful for next query. If the search is looking for data of specific time (or location or gender), maybe for the first query, we search a branch (sub-tree) of a node that totally is not having data for that specific time, but after system detects the data bellow each node is of which date/location/gender or what is the minimum and maximum values of one attribute under that node and its sub-tree, and updates the metadata, next time queries looking for a data of another specific time, can read metadata of the parent node before go through the branch (sub-tree). Figure 4.10 presents the flow of the search operation using the proposed method.

4.7 Conclusion

In this chapter, the proposed method of this research is illustrated and its components are described. The proposed method is an extension of B-tree structure and uses a Min-

University of Malaya

Max summarization method to add a summary of data stored inside the sub-tree under each node. Min-Max summarization checking, our technique provides a history of the last search in every index structure that has been created and used from previous query processing. So, if the current query is same or has overlapped with the previous query, the search algorithm will get benefit from history element s inside some of the nodes inside the index structures that have been used by the previous search process. Such method helps to optimize search process by reducing the number of comparisons required to be done to find the result of the search. In the next chapter, performance evaluation methods are described. These methods are used for evaluating and validating our proposed solution via benchmarking experiments in which different queries in term of complexity and different datasets in term of volume are used.

University of Malaya

CHAPTER 5: EVALUATION

5.1 Introduction

The objective of this chapter is to provide performance evaluation methods used to evaluate and validate the proposed B-tree based search method. The purpose is to improve the performance of execution time of data retrieval of big data sets. Thus, in order to outline the importance of the proposed method as discussed in Chapter 4, performance evaluation of the proposed search method is provided. Using benchmarking experiments, we collected data of query execution time for searching four predefined queries with the range of simple to complex queries. Every search task is repeated 10 times. The sufficiency of this number of benchmarks for evaluating the performance of computing systems by using Sieve benchmark is already proven (Jain, 2008). Sieve of eratosthenes benchmark (Sieve in brief) is an standard benchmark which is widely used for evaluation performance of computing systems. Sieve algorithm receives benchmark value N and generates all the prime numbers from 1 to N (Bukh & Jain, 1992)

The evaluation results are validated using benchmarking analysis and comparative study of the performance of our proposed search method in contrast to the performance of the B-tree search method. Thus, first, the chapter provides a description of the benchmarks that used for the evaluation of the proposed method. Second, the simulation environment and the datasets used for the experiment are described in details. Finally, the chapter investigates the performance of the proposed method. Moreover, a comparative study is performed to demonstrate the performance of our proposed B-tree based search method in comparison with related search method, particularly with normal B-tree search method.

University of Malaya

The chapter organized as follows. Section 5.2 explains the evaluation methodology of the proposed B-tree based search method using Windows based Server environment. Also, it reports the experimental environment used for the evaluation process. Section 5.3 describes the datasets that is used for performing the experiments. Section 5.4 provides benchmark description that is used for the evaluation of B-tree based search method. At the end a conclusion of this chapter is reported in Section 5.5.

5.2 Evaluation of the Proposed Search Method

In this section, the evaluation methodology of the proposed B-tree is proposed based search method using Windows based Server environment. Also, the experimental environment used for the evaluation process is reported.

5.2.1 Performance Evaluation Testing Environment

This section determines the testing environment approach used for the performance evaluation procedure. This environment is designed to index data and run searching query process in two modes. In the first mode, normal B-tree search algorithm is used and the taken time to execute searching the indexed data to find the results for the four predefined queries from a simple to complex one is measured. In the second mode, our proposed search method is used. This proposed search method uses a modified search algorithm based on B-tree search algorithm and the consumed time for executing search processes to apply the four queries against same indexed data is captured.

5.2.1.1 Testing Environment

We used a powerful Desktop Server Computer with 8 processers. Each processor is an Intel(R) Xeon(R) Central Processing Unit (CPU) E5620 with capacity 2.40GHz. This system uses 32 GB Random Access Memory (RAM), and 1.778 TB Hard Disk Drive

University of Malaya

(H.D.D) to carry out the experimentation of both B-tree and the proposed search methods. By choosing this Desktop Server Computer any corruption caused by network overhead is eliminated, and communication costs or other factors such as transformation and changing of the output are avoided. The Operating System (OS) of this server was a Windows Server 2008 R2 Enterprise64-bit SP1. We used java programming language and developed a simulation engine to read and index the datasets, and run search operations to find the results for the four predefined queries with two search method including normal B-tree search method and also the proposed B-tree based search method. Table 5.1 listed the main hardware specifications of the testing environment.

Table 5.1: Hardware specifications of the testing environment

Hardware	Capacity	Description
CPU	8 x 2.40GHz	Each CPU is a Intel(R) Xeon(R) CPU E5620
RAM	8 x 4GB	Each of ECC DDR3 1333MHz. Indexing and searching large data needs more RAM to make the process faster.
H.D.D	1.778TB	DELL MD32xxi SCSI Disk Device (iSCSI). Indexing and searching hundreds of Giga Bytes of data requires large storage to store data and index files
VGA	60Hz	Standard Monitor 1280x1024 Resolution video adapter and monitor

5.3 Datasets

The study and knowledge of how weather evolves over time in some location or country in the world can be beneficial for several purposes. Such knowledge or information could be used for future predictions. For instance, knowledge of how temperature changing effect on the tourists and precipitation aid in flood planning. The use of terms like weather and climate are sometimes used interchangeable in different situations. Their main difference is that weather prediction refers to a short period (e.g. several days to one week); on the other hand, climate prediction involves the process of predicting the future evolution for months, years, etc. Major data attributes included in

University of Malaya

the collected weather from the National Oceanic and Atmospheric Administration (NOAA) information include year, month, day, temperature, dew point, humidity, Significant Weather, Wind Direction, pressure, Precipitation Snowfall, wind speed, etc.

In this research, seven datasets with different sizes starting from 1 GB to 1 TB are used. The datasets are about historical weather data with an hour interval of 22632 cities from different countries over the world. The format of data is in json format.

Table 5.2: The datasets used in this research

Dataset	Size	Description
Dataset 1	**1.046 GB**	Historical weather data with an hour interval of 22632 cities from different countries over the world
Dataset 2	**2.127 GB**	
Dataset 3	**4.255 GB**	
Dataset 4	**8.509 GB**	
Dataset 5	**17.018 GB**	
Dataset 6	**68.073 GB**	
Dataset 7	**1089.163 GB**	

To collect this data, an Application Programming Interface API provided by OpenWeatherMap ("OWM API," 2015) is used. Through this API they provide historical weather data for 22,632 cities. All services provided by OpenWeatherMap such as maps, tiles, APIs and etc. are distributed under terms of the Creative Commons Attribution-ShareAlike 4.0 International license (CC BY-SA 4.0) (C. Commons, 2013). Data and database are open and licensed under Open Data Commons Open Database License (ODbL) (O. D. Commons, 2013). Based on the terms and conditions of the data

University of Malaya

provider, the products and data for non-commercial or commercial purposes can be used. Table 5.2 provides some information about the datasets used in this study.

5.4 Benchmark Description

Our proposed search method is evaluated using benchmarking experiments for query response time for searching four predefined queries with the range of simple to complex queries. Ten benchmarks are used to run a performance evaluation for our proposal. The sufficiency of this number of benchmarks for evaluating the performance of computing systems by using Sieve benchmark is already proven (Jain, 2008).

A Desktop Server Computer with 8 processers is used. Each processor is an Intel(R) Xeon(R) Central Processing Unit (CPU) E5620 with capacity 2.40GHz. 32 GB Random Access Memory (RAM), and 1.778 TB Hard Disk Drive (H.D.D) are used to provide a powerful experimentation environment. The Operating System of this computer is Windows Server 2008 R2 Enterprise64-bit SP1 version 6.1. Java 1.8.0_111 is installed on this machine to use as the programming language and also the NetBeans IDE 8.0.2 (Build 201411181905) as a powerful editor is used for coding, debugging, and running the code.

In order to evaluate the proposed method, an indexing model is designed. This indexing model consists of five main component including Index Creator to create index structure based on some predefined attributes, Backup Engine to make backup of the index files in order to provide ability of reusing previous index files, Summarization Updater to add summarization to each node of the index structure as a kind of metadata, History Updater to store the last search comparison result to be reused in next query,

University of Malaya

and Query Processor to apply the four predefined queries against different sizes of the datasets. Figure 5.1 illustrates a scheme of the benchmarking setup.

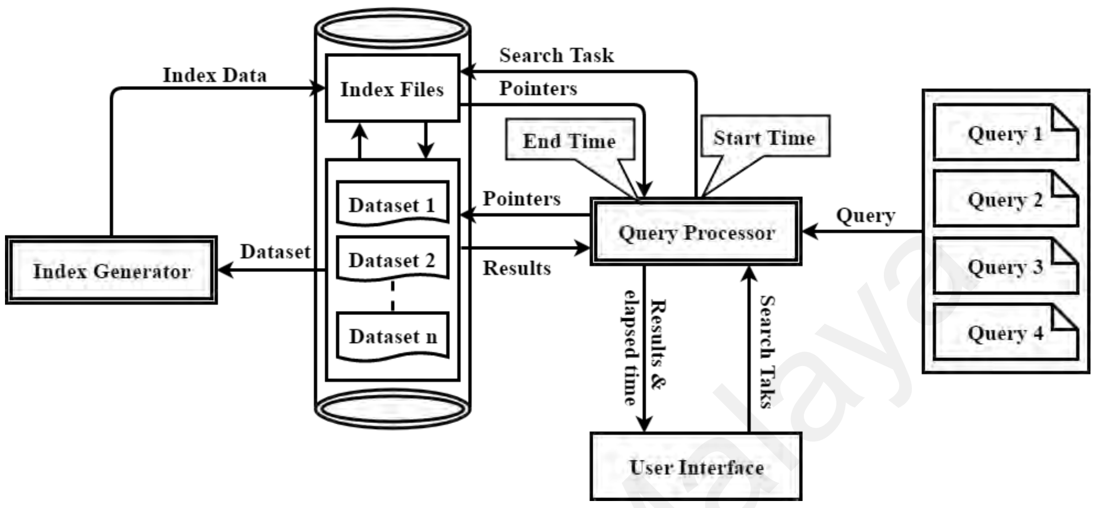

Figure 5.1: Benchmarking Model Scheme

To evaluate the performance of the proposed method in term of search execution time, four different queries with the range of simple query to complex one are defined. Table 5.3 listed these four queries with some description about each of them.

Table 5.3: Queries used in the experiments

Query	Description
Query 1	Select all records in which attribute ―country" is equal to ―MY"
Query 2	Select all records in which country is Malaysia and humidity is greater than 80 and temperature is greater than 300
Query 3	Select all records in which country is Malaysia and humidity less than 50 and speed of wind greater than 3
Query 4	Select all records in which country is Malaysia and humidity is greater than 90 and pressure is greater than 1000

Two different execution modes are used. In the first execution mode that we call it as mode *A*, after indexing a specific size of data, B-tree search method is used to perform search operation to find the results for each predefined queries and every time search execution time for processing each query and every specific size of data that we index

University of Malaya

prior to executing the search tasks is captured. This procedure is repeated 10 times. In the second mode which is called mode ***B*** in this chapter, same procedure is repeated using the proposed search method instead of the B-tree search method.

Every search task is an operation of looking for results of a specific Query through a specific size of data. Each search task is performed 10 times. This procedure must be done in two modes including mode ***A*** in which search task is executed using B-tree search method and mode ***B*** which it runs the search task using the proposed search method. The elapsed time will be calculated based on the difference between the start time and the end time of processing the search task. Table 5.4 presents a summarized report of the workloads when the size of data is about 1.064GB.

Table 5.5 shows the summary report of the workloads for the dataset with size that is used in the experiment. This result is obtained from the two methods B-tree method and the improved that are implemented for data retrieval. As shown in the table the datasets are 2.127GB of weather data collected.

University of Malaya

Table 5.4: Summarized report of the workloads for the dataset with size 1.064GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method
0.37	0.053	199	141	364	174	354	140
0.37	0.005	197	53	365	152	347	116
0.37	0.004	206	64	361	150	340	115
0.37	0.003	222	59	369	141	336	114
0.36	0.004	234	53	363	132	345	115
0.36	0.003	246	61	359	149	343	113
0.37	0.003	236	55	356	135	337	116
0.37	0.004	244	80	360	148	341	113
0.37	0.003	243	70	364	133	340	113
0.36	0.004	239	84	363	145	350	115

Table 5.5: Summarized report of the workloads for the dataset with size 2.127GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method
1.33	0.048	1417	264	2262	351	2165	627
1.23	0.005	1435	249	2255	336	2172	403
1.27	0.004	1552	236	2246	332	2172	264
1.20	0.003	1496	232	2280	317	2163	253
1.26	0.004	1509	190	2254	315	2177	237
1.20	0.003	1587	176	2245	290	2151	217
1.56	0.004	1617	173	2240	271	2184	219
1.40	0.004	1572	165	2222	261	2195	187
1.31	0.003	1545	172	2254	265	2210	263
1.44	0.003	1531	161	2261	244	2178	271

Table 5.6: Summarized report of the workloads for the dataset with size 4.255GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method
1.71	0.033	6221	437	9565	640	8009	514
1.73	0.005	6213	406	9560	562	7992	452
1.69	0.007	6215	312	9529	546	7994	437
1.73	0.004	6827	327	9552	546	7984	406
1.60	0.004	6186	359	9579	484	7998	421
1.61	0.004	6186	343	9591	514	7963	359
1.61	0.005	6209	281	9571	484	7990	343
1.66	0.004	6206	296	9560	483	7990	359
1.63	0.004	6201	265	9568	577	7983	390
1.70	0.003	6205	297	9573	578	8019	359

In Table 5.6 the summary report of the workloads for the dataset with size that is used in the experiment. This result is obtained from the two methods B-tree method and the improved that are implemented for data retrieval. As shown in the table the datasets are 4.255GB of weather data collected.

University of Malaya

Table 5.7: Summarized report of the workloads for the dataset with size 8.509GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method
1.92	0.048	26420	811	41236	1264	36867	1092
1.86	0.005	25444	655	41237	1107	36803	827
1.94	0.004	26491	608	41181	1123	36868	851
1.87	0.004	25383	656	41034	1124	36812	842
1.92	0.004	26935	748	40991	1120	36862	858
1.80	0.003	26920	624	40985	1185	36926	827
1.91	0.003	26931	640	41007	1139	36849	889
1.82	0.006	26915	562	40946	1030	36882	827
1.85	0.003	26937	608	40925	1076	36849	952
1.83	0.003	26942	624	40926	1014	36794	920

Table 5.8: Summarized report of the workloads for the dataset with size 17.018GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method
2.70	0.060	118711	1529	165763	2605	154182	3767
2.63	0.007	118787	1279	165496	2434	154270	2278
2.88	0.006	118726	1389	165807	2449	154198	1638
2.69	0.004	112738	1341	165655	2590	154139	1622
2.66	0.003	119084	1264	167662	2340	154340	1810
2.62	0.004	119108	1373	165779	2542	154331	1622
2.67	0.004	119139	1357	165750	2418	154235	1607
2.53	0.003	119102	1388	165922	2528	154133	1685
2.69	0.003	119071	1233	165800	2277	154268	1622
2.76	0.003	119109	1466	165762	2452	154191	1623

In Table 5.7 the summary report of the workloads for the dataset with size that is used in the experiment. This result is obtained from the two methods B-tree method and the improved that are implemented for data retrieval. As shown in the table the datasets are 8.509GB of weather data collected. In Table 5.8 the summary report of the workloads for the dataset with size that is used in the experiment. This result is obtained

University of Malaya

from the two methods B-tree method and the improved that are implemented for data retrieval. As shown in the table the datasets is 17.018GB of weather data collected.

Table 5.9:Summarized report of the workloads for the dataset with size 68.073GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	**Proposed Method**	**B-tree Method**	**Proposed Method**	**B-tree Method**	**Proposed Method**	**B-tree Method**	**Proposed Method**
4.13	0.066	2165329	4129	3240627	17526	3811579	31803
4.11	0.008	2266765	3454	3235407	16378	3813754	19230
4.07	0.007	2165613	3751	3241487	16479	3811975	13828
4.21	0.004	2130396	3621	3238516	17428	3810516	13692
4.16	0.003	2250315	3413	3277752	15746	3815485	15280
4.53	0.004	2250769	3707	3240940	17105	3815262	13692
4.30	0.004	2251354	3664	3240373	16271	3812889	13566
3.65	0.003	2250655	3748	3243735	17011	3810368	14224
4.15	0.003	2250069	3329	3241350	15322	3813705	13692
4.38	0.003	2250788	3959	3240607	16499	3811801	13701

Table 5.10: Summarized report of the workloads for the dataset with size 1089.163GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	**Proposed Method**	**B-tree Method**	**Proposed Method**	**B-tree Method**	**Proposed Method**	**B-tree Method**	**Proposed Method**
11.69	0.08	1037893853	8906	2535564479	5006298	5107298720	9561356
11.84	0.009	1038538968	7450	2531480360	4678527	5110213732	5781395
11.57	0.008	1038021178	8091	2536237517	4707359	5107828723	4157122
10.02	0.005	1056959986	7811	2533912476	4978383	5105874340	4116516
11.93	0.004	1010827077	7363	2564612197	4497844	5112532491	4593646
10.83	0.005	1011030797	7998	2535809220	4886119	5112234365	4116516
11.38	0.005	1011293936	7904	2535365626	4647772	5109054352	4078447
10.95	0.004	1010979867	8085	2537996594	4859209	5105675589	4276405
10.81	0.004	1010716728	7182	2536130442	4376748	5110147482	4116516
11.44	0.004	1011039286	8539	2535549182	4713125	5107596847	4119054

University of Malaya

In Table 5.9 the summary report of the workloads for the dataset with size 68.073GB that is used in the experiment. Table 5.10 is reporting the execution times for searching the predefined queries against a dataset with size 1089.163GB. This result is obtained from the two methods B-tree method and the improved that are implemented for data retrieval.

For more reports of execution time for searching the predefined queries against datasets with sizes of 34.713 GB, 136.145 GB, 272.291 GB, and 544.582 GB (see Appendix A).

5.4.1 Execution time

In order to perform mapping of the query in big data to the size of data in execution, a matrix is required. There are two ways to obtain the result of execution time, which are the measurement of the size of particular query or the number of recorded to be summarized. In the meantime, statistical analysis defines a number of primitive types. The analysis, determine the performances of the nodes in the cluster, which is the used to produce the execution time estimate. Execution time is chosen for this research to measure the response time for each of the datasets basic task. The response time is mostly same with a small variation for each time the simulation engine processes every query, and there is no much change in time consumption for processing the query to return the result. However, when the size of the dataset is growing up, although the response time for the bigger dataset is increasing, but still the response times' trend for every dataset has little variation and the response times are almost same. In another word, same queries are applied twenty times against each dataset with a specific volume, but the time consumption in data retrieval is nearly same. Therefore, the resources, particularly the time that is consumed for processing earlier queries, are not

University of Malaya

reused for processing later queries to be faster and processing later queries consumes same time which is the metric that is use in our study. For validating the derived execution time models, the second split of the dataset consists of workloads and corresponding execution times is used and the result of measured execution time with the predicted execution time out of the regression model is compared.

5.5 Conclusion

This chapter provides a description of the performance evaluation procedure in terms of execution time and data size using two parts which are benchmarking and comparative study. Moreover, it provides an overview of the datasets used for this research focused on data generation process for execution time. Also this chapter presents some discussions about the experiment procedure based on a powerful Desktop Server Computer with 8 processers to carry out the experimentation of the proposed search method.

Following chapter presents the results of the evaluation the benchmarking experiments using the proposed B-tree based search method for searching different sizes of the data sets based on the execution time, a number of executions and the size of the data. The evaluation results are validated using experimental analysis and comparison. The objective of the next chapter is to evaluating the performance of the proposed improved B-tree method by comparing with current B-tree search method.

University of Malaya

CHAPTER 6: RESULTS AND DISCUSSIONS

6.1 Introduction

This chapter presents the evaluation results of the experiment research of the proposed B-tree based search method for searching different sizes of the data sets based on the execution time, a number of executions and the size of the data. The evaluation results are validated using experimental analysis and comparison. The chapter is fulfilling the objective of evaluating the performance of the proposed improved B-tree method by comparing with current B-tree search method.

The remainder of this chapter is as follows. Section 6.2 presents the results of the performance experiments and reports execution time and the number of the executions as well as the size of the datasets. Section 6.3 discusses the result of benchmarking analysis and the comparative study. Section 6.4 presents a conclusion for this chapter.

6.2 Performance Evaluation Results

In this section, the performance of the evaluation results that obtained from the experiments using the benchmarks is presented. The first results obtained by preforming a number of executions on different sizes of large datasets to capture the execution time of each one. This number of execution times is executed on the bases of four queries depends on the complexity of the query preformed during the experiments. The reason to conduct such experiments shows the performance of each query using the proposed improved B-tree search method. The performance analysis is accomplished to evaluate the proposed method based on the execution time and the number of execution preformed for each query.

University of Malaya

6.2.1 Execution Time

This section presents temporal outcomes of executing the four queries as are listed in Table 5.3. The results are obtained from executing the queries in two modes. The first mode is using B-tree search method and the second mode is using the proposed search method. Execution time data are collected via benchmarking analysis. The results are reported using several tables in Chapter 5 and an impressive number of charts and figures is presented in this section to elaborate the performance of our proposed search method in comparison with the performance of the B-tree search method. Figure 6.1 demonstrates the execution time of processing Query 1 against a dataset with size 1.064GB. As it is explained in Chapter 5, Table 5.3, Query 1 is looking for all records in which country is Malaysia. This is a simple query because it is looking for equality of the attribute ~~–c~~ountry" and Malaysia. However, as the datasets of this research are about the historical weather data for 22,632 cities from different countries over the world, the number of unique values for the attribute ~~–c~~ountry" is limited to the number of countries over the world and it means the height of the tree-based index of this attribute is very low and searching the index of countries' values is very fast.

As it is illustrated in Figure 6.1, the execution time for processing Query 1 via the proposed search method is so much lower than the execution time for processing Query 1 via B-tree search method. This figure shows that using our proposed search method for processing a simple query against a not very large dataset with size 1.064GB is more justifiable than using B-tree search method. This improvement is obtained by using a min-max summarization technique in our proposed search method which adds some information as a kind of metadata to every node of the tree structure of the index and

University of Malaya

help search algorithm to check the availability of the target key inside the sub-trees of the node before starting comparison operation through those sub-trees‘ nodes.

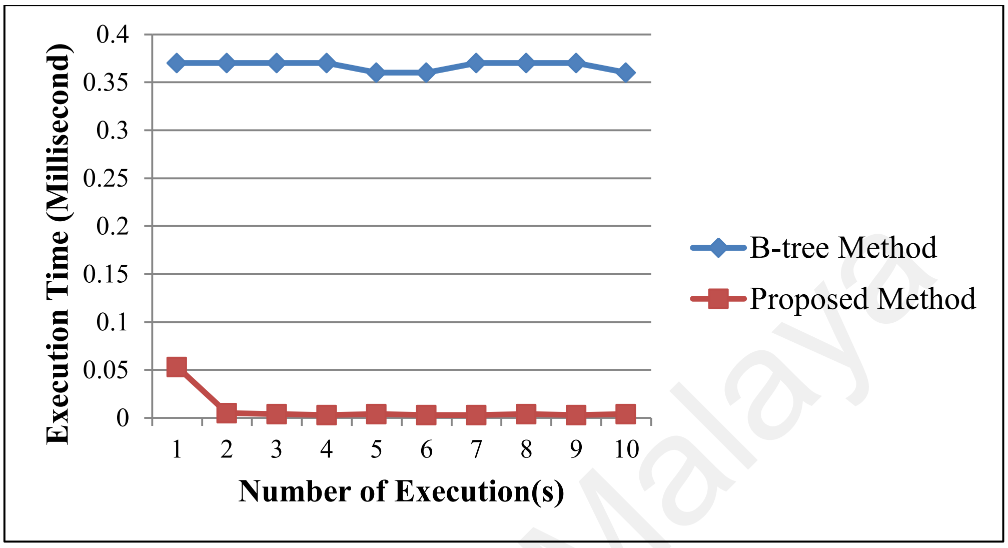

Figure 6.1: Execution time for Query 1 with data size 1.064 GB

Moreover, the trend of the execution time for processing Query 1 when the proposed search method is used, shows that the average execution time of the later search operations after the first one is 14.45 time better than the execution time of the first search task. The reason for this improvement is using history data as the second meta data of the nodes. This history data is added to the nodes after first time query processing. As it is elaborated in Section 4.2.5, in second time applying same query, the search algorithm uses the history data and reduce the number of comparisons by jumping from a node with match history value to the last node of the every traversed path to the left or to the right side branches. Therefore, the modified search algorithm reduces the execution time of the later queries which are same with the first query by reducing the number of comparisons. Hence, the later search operations can get benefit from the first search operation and reduce the execution time.

University of Malaya

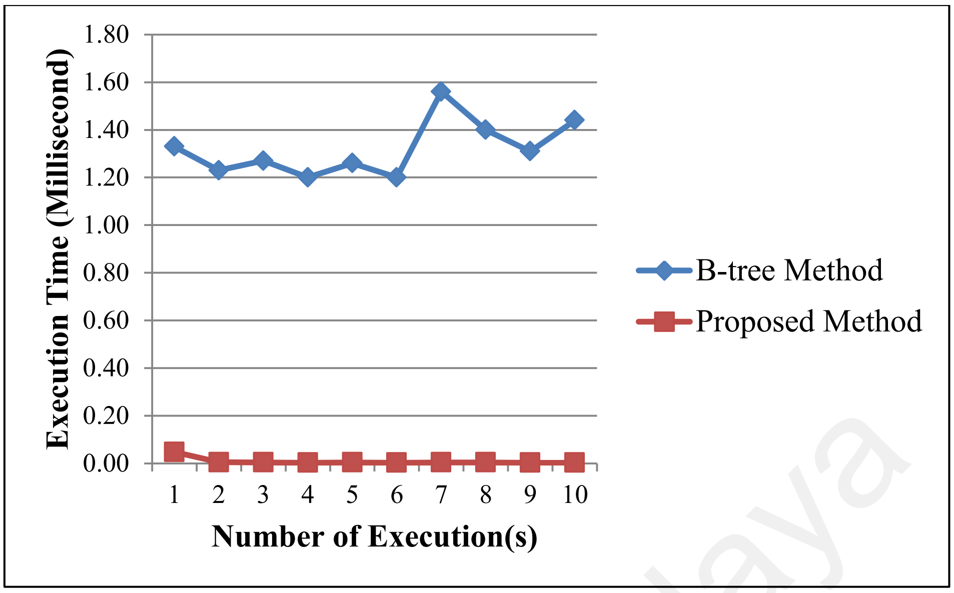

Figure 6.2: Execution time for Query 1 with data size 2.127 GB

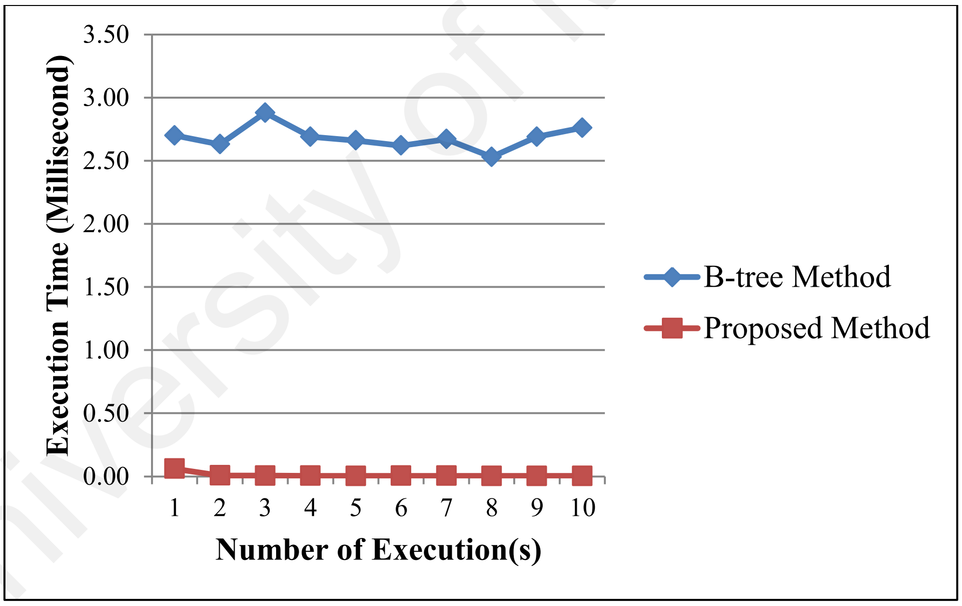

Figure 6.3: Execution time for Query 1 with data size 17.018 GB

Figure 6.2, Figure 6.3, Figure 6.4, and Figure 6.5 present comparable diagrams related to the behavior of the execution time for processing Query 1 when the datasets are 2.127GB, 17.018GB, 68.073GB, and 1089.163GB respectively.

University of Malaya

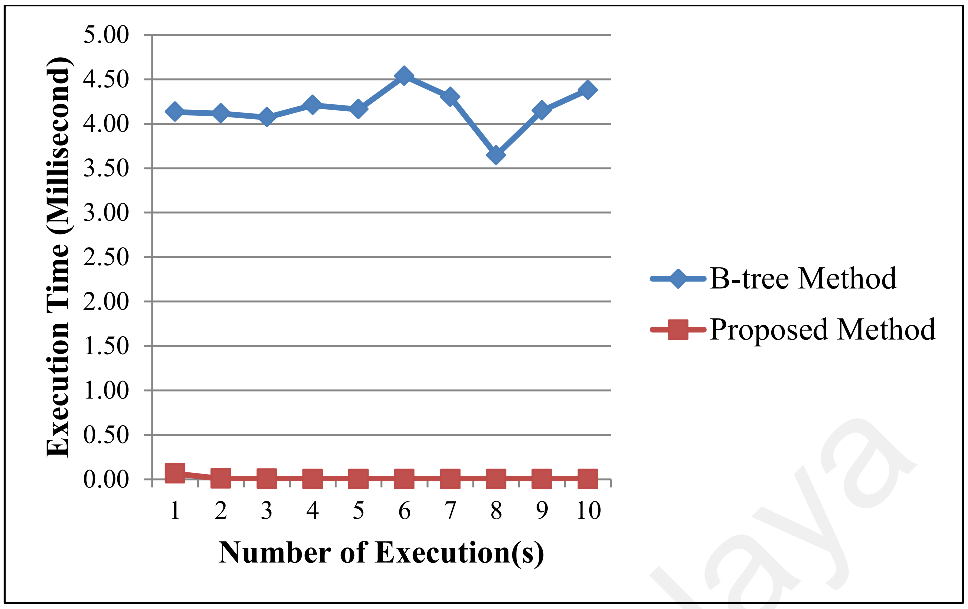

Figure 6.4: Execution time for Query 1 with data size 68.073 GB

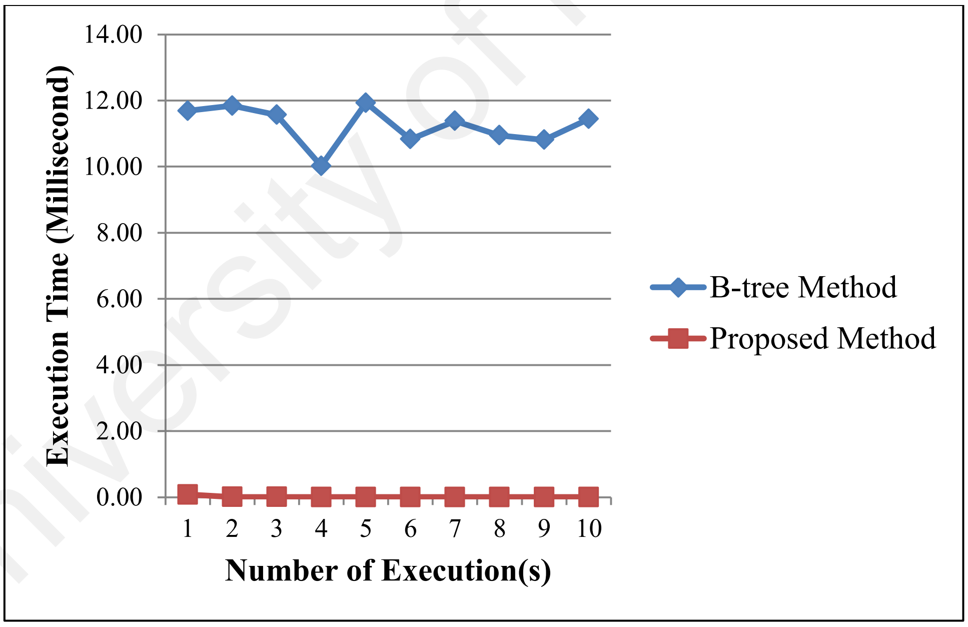

Figure 6.5: Execution time for Query 1 with data size 1089.163 GB

Based on the trend of the execution time for executing Query 1 against different sizes of the datasets, the performance of the proposed search method for processing a simple query like Query 1 is much better than the performance of the B-tree search technique

University of Malaya

for processing same query is same datasets. In fact, according to the results of the benchmarking, this performance up to 353.4 times is improved. Figure 6.2 images the difference of execution time for processing Query 1 against all the datasets when using B-tree search method and the proposed method. For more figures (see Appendix A).

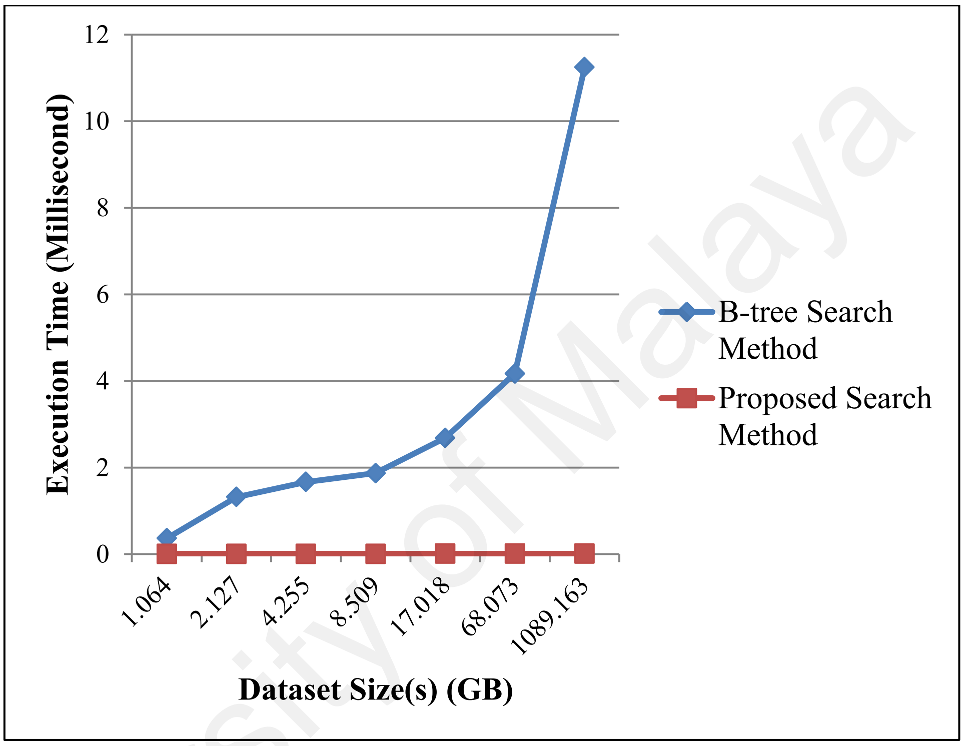

Figure 6.6: Execution time for Query 1 based on all sizes of the datasets

In the second round of the benchmarking experiments, Query 2 is used which is more complex in comparison with Query 1. Based on Table 5.3 in Chapter 5, Query 2 is looking for any record in which the value of attribute ―Country” is equal to ―Malaysia”, the value of attribute ―Humidity” is greater than 80, and the value of attribute ―Temperature” is greater than 300. This query is looking for equality and also a range of values. So, it uses a join clause to find the records that are fulfilling its conditions. Based on the results of benchmarking experiments reported in Chapter 5, Figure 6.6

University of Malaya

compares the execution time of processing Query 2 by using B-tree search method and the proposed search method when the size of the dataset is 1.064GB.

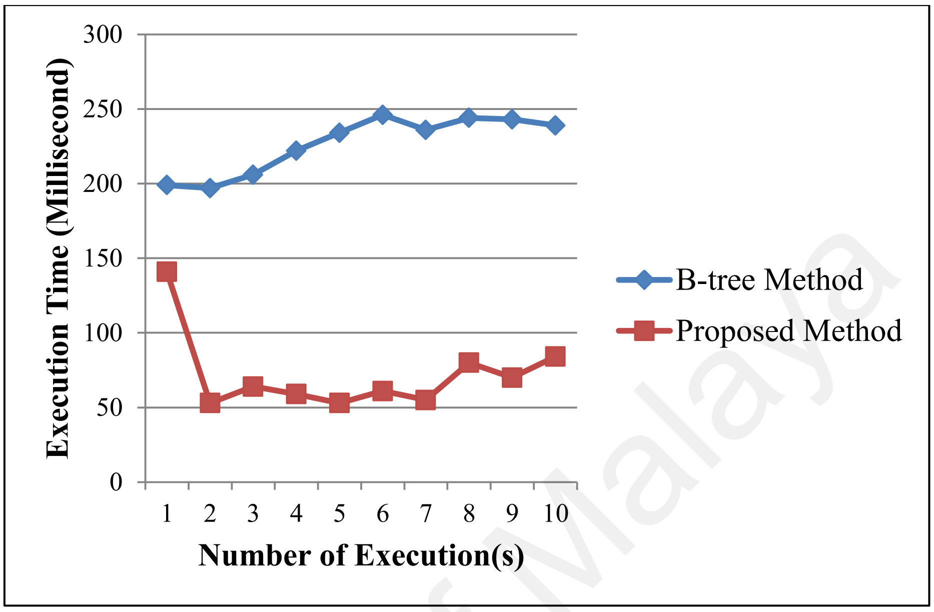

Figure 6.7: Execution time for Query 2 with data size 1.064 GB

As it is shown in Figure 6.7, the execution time for processing the Query 2 significantly is reduced when the B-tree search method is replaced with the proposed search method when the size of the dataset is 1.064GB. The values of the results show that the average of the performance of the proposed search method for processing Query 2 when the size of the dataset is about 1GB is 3.15 times better than the average of the performance of the B-tree search method. However, same as our finding about the impact of adding history value as a metadata to the nodes of the tree structure of the index, the first time of processing Query 2 using the proposed method inherits an advantage of using history metadata for the second and next times of processing same query to reduce the execution time significantly. The results of the experiments show that the execution time of processing the Query 2 via the proposed method up to 53% is reduced.

University of Malaya

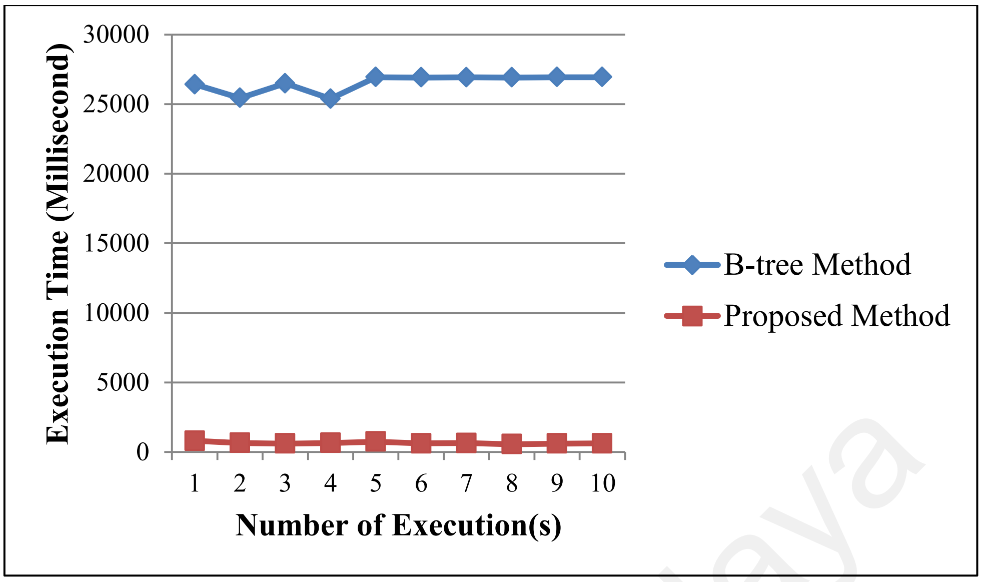

Figure 6.8: Execution time for Query 2 with data size 8.509 GB

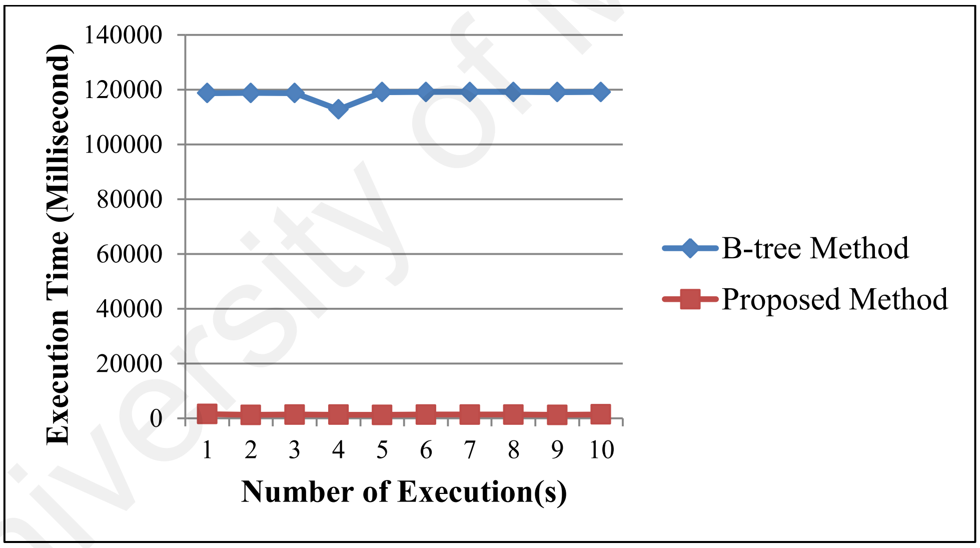

Figure 6.9: Execution time for Query 2 with data size 17.018 GB

University of Malaya

Figure 6.8, Figure 6.9, Figure 6.10, and Figure 6.11 illustrate comparative charts based on execution time values extracted from the experiments which show the difference of the impact of the proposed search method in contrast to the impact of the B-tree search method while looking for the results of Query 2 in four datasets with sizes of 8.509GB, 17.018GB, 68.073GB, and 1089.163 GB respectively. For more figures (see Appendix A).

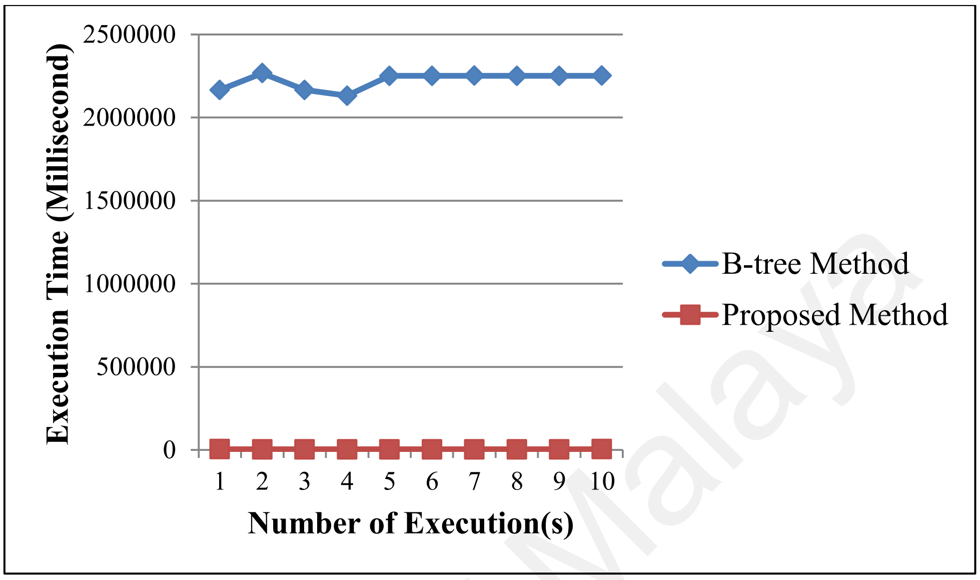

Figure 6.10: Execution time for Query 2 with data size 68.073 GB

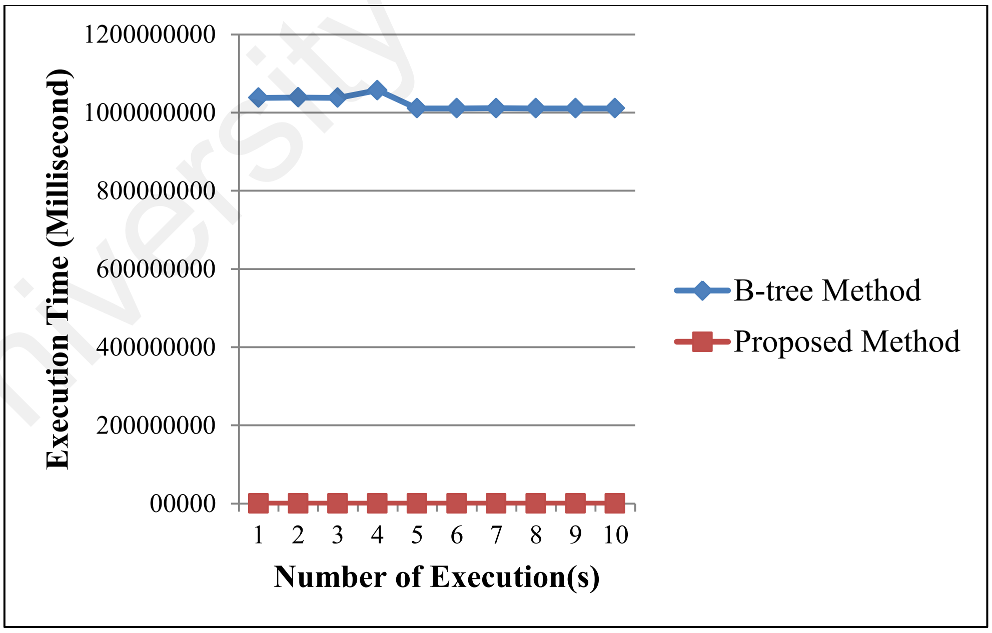

Figure 6.11: Execution time for Query 2 with data size 1089.163 GB

University of Malaya

According to the results of the execution time to apply Query 2 on different sizes of the datasets, when the dataset is growing up to 68GB, the average of the performance of the proposed search method for processing a complex query like Query 2 is increasing up to 377 times better than the average of the performance of the B-tree search method when processing the same query against same datasets. But, when the size of data reach to 1TB, the performance of the proposed search method is so much better than B-tree search method for processing Query 2 and we can mark B-tree search method as out of a performance in comparison with the proposed method. Figure 6.12 presents the difference of the performance of the proposed search method with the performance of the B-tree search method. This diagram shows that when the volume of the data is growing up, the B-tree search execution time for Query 2 is rapidly increasing. In contrast, the execution time of processing Query 2 via the proposed search method is growing with a gentle slope.

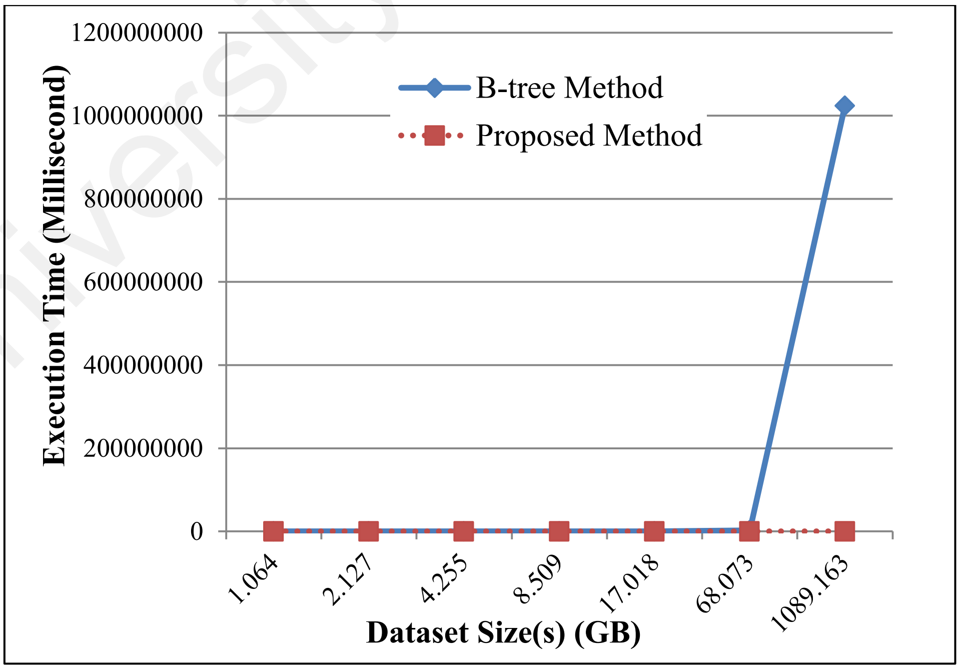

Figure 6.12: Execution time for Query 2 based on all sizes of the datasets

University of Malaya

Figure 6.13 illustrate the result of Query 3 based on 1.064GB data sets. As it is explained in Table 5.3 in Chapter 5, Query 3 is looking for any record in which attribute ―Country" is equal to ―Malaysia", attribute ―Humidity" is less than 50, and attribute ―Wind Speed" is greater than 3.As it can be seen from the figure the execution time of the proposed search method is outperforming the normal search method using same datasets size. The average time of the execution using our proposed method is 2.48 times better compared to the normal B-tree method in terms of the performance. Though, adding history value as a metadata to the nodes of the tree structure of the index is impacting the overall execution time during the process of query 3. As the result, the next query of the datasets utilized the results of metadata obtained by the first query when processing a large amount of data. The result of experiments indicted that the execution time of the processing of Query 3 using proposed algorithm is reduced by 18%.

University of Malaya

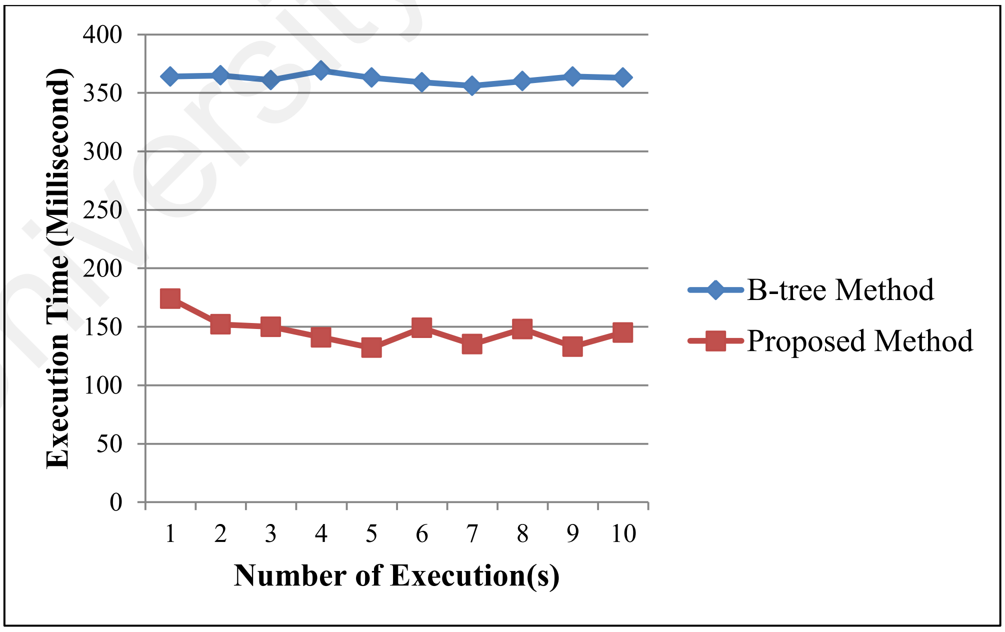

Figure 6.13: Execution time for Query 3 with data size 1.064 GB

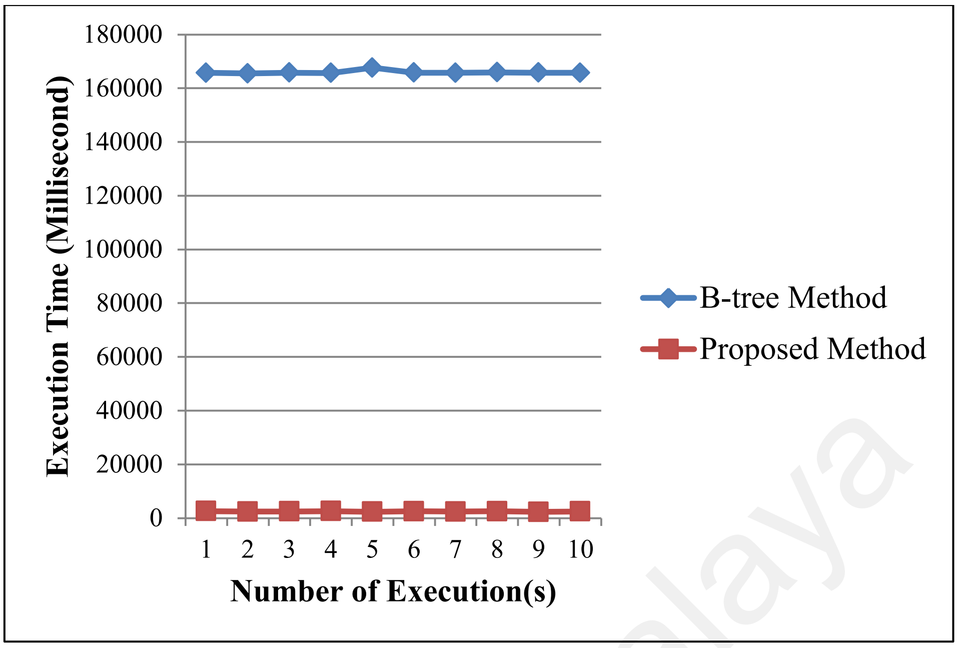

Figure 6.14: Execution time for Query 3 with data size 17.018 GB

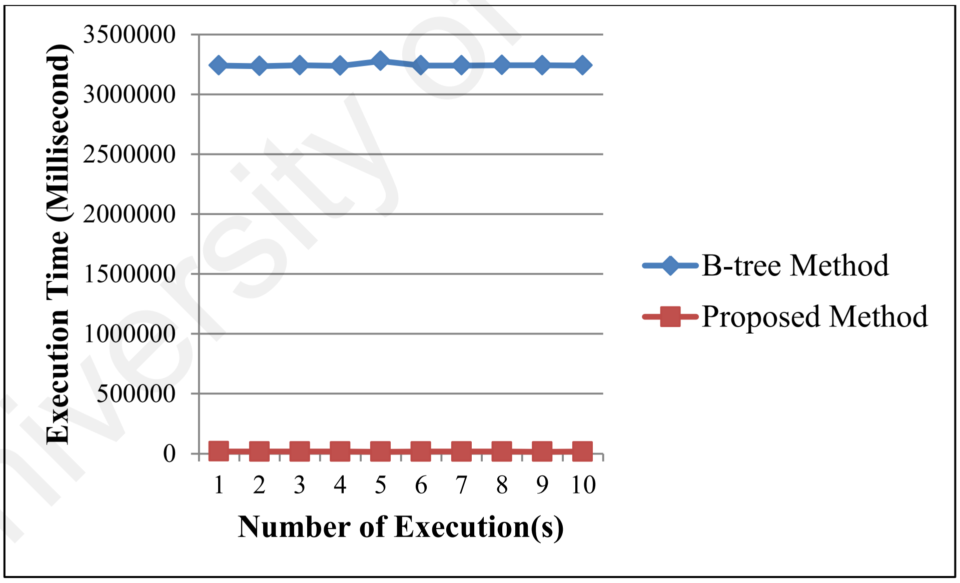

Figure 6.15: Execution time for Query 3 with data size 68.073 GB

University of Malaya

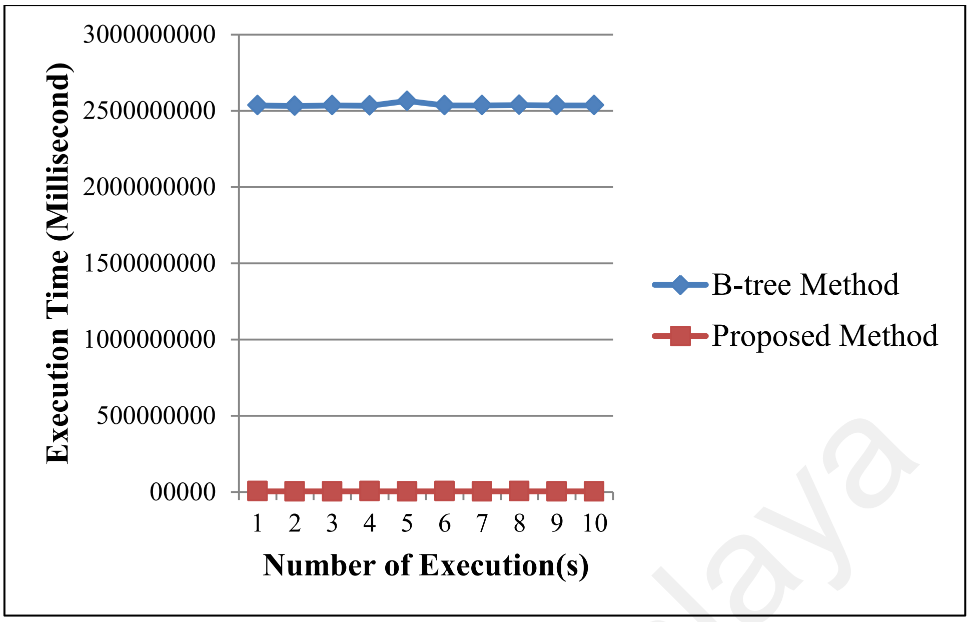

Figure 6.16: Execution time for Query 3 with data size 1089.163 GB

The results of the experiments using datasets 17.018GB (Figure 6.14), 68.073GB (Figure 6.15), and 1089.163 GB (Figure 6.16) have shown that the execution time of the proposed B-tree search method provides better performance compared to normal execution time.

Based on the benchmarking experiments for processing Query 3 against different sizes of datasets from 1.064GB to 1089.163, the average performance of the proposed search method is up to 534 times better than the average performance of B-tree search method for processing the same query on same sizes of data. Figure 6.17 shows the difference of the average performance of B-tree search method with the average performance of the proposed search method upon processing Query3.

University of Malaya

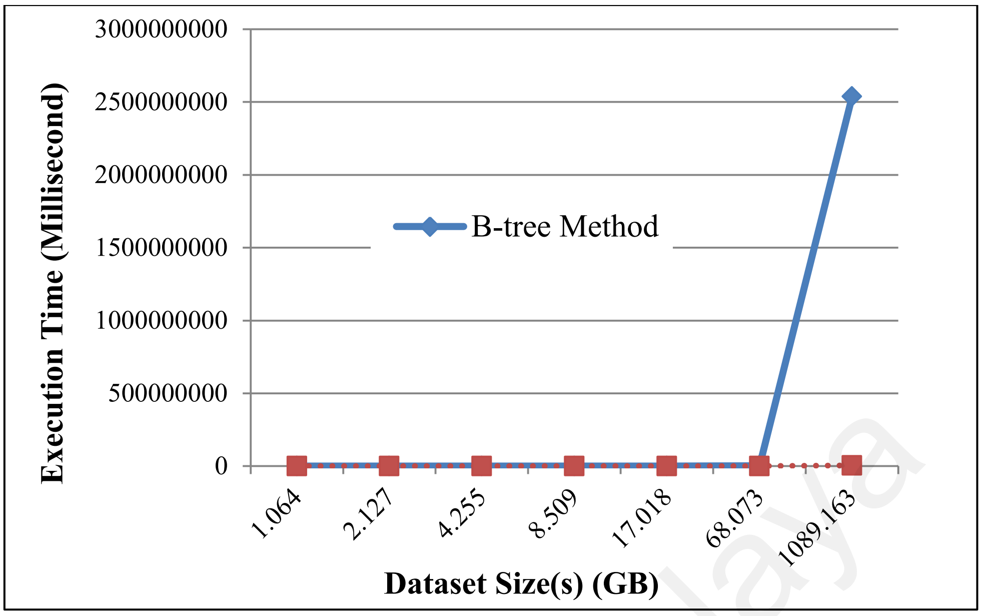

Figure 6.17: Average execution time for processing Query 3

In the last round of the benchmarking experiments, it processes Query 4 by using B-tree search method and also the proposed search method. Query 4 is looking for the records of data in which attribute ―Country" is equal to ―Malaysia", attribute ―Humidity" is greater than 90, and attribute ―Pressure" is greater than 1000. Hence, this query is categorized as a complex query. For more figures (see Appendix A).

Figure 6.18 show the result of Query 4 based on 1.064GB data sets. As it can be seen from the Figure 6.18, the execution time of the proposed search method is outperforming the normal search method using same datasets size. The average time of the execution using our proposed method is 2.93 times better compared to the normal B-tree method in terms of the performance. Though, adding history value as a metadata to the nodes of the tree structure of the index is impacting the overall execution time during the process of query 4. As the result, the next query of the datasets utilized the results of metadata obtained by the first query when processing a large amount of data.

University of Malaya

The result of experiments indicted that the execution time of the processing of Query 4 using proposed algorithm is reduced by 18%.

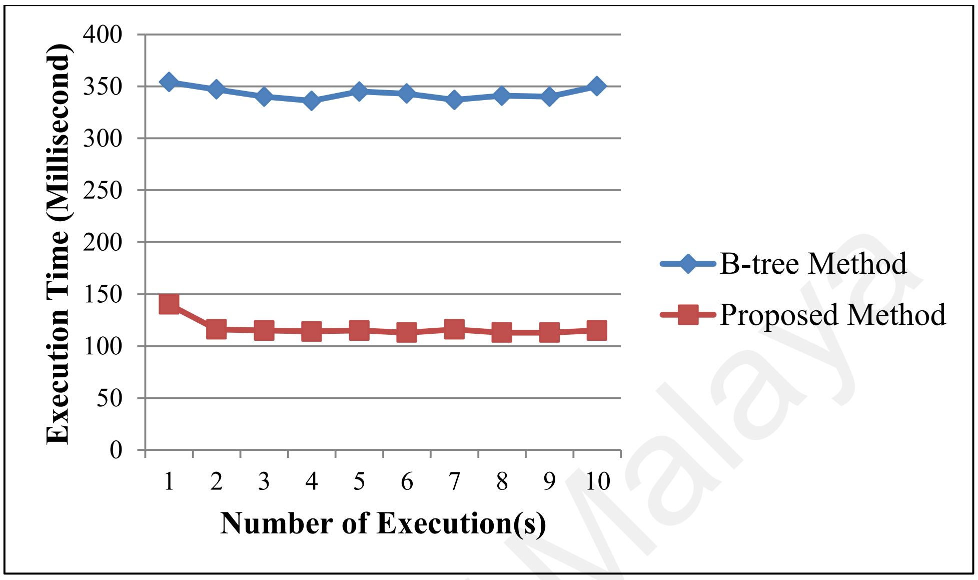

Figure 6.18: Execution time for Query 4 with data size 1.064 GB

As shown in Figure 6.19, Figure 6.20, and Figure6.21, the results are respectively similar due to added history value as a metadata to the nodes of the tree structure of the index during the execution of the first query. The result of an experiment using, 17.018GB, 68.073GB, and 1089.163 GB have shown that the execution time of the proposed B-tree search method provides better performance compared to normal execution time. For more figures (see Appendix A).

University of Malaya

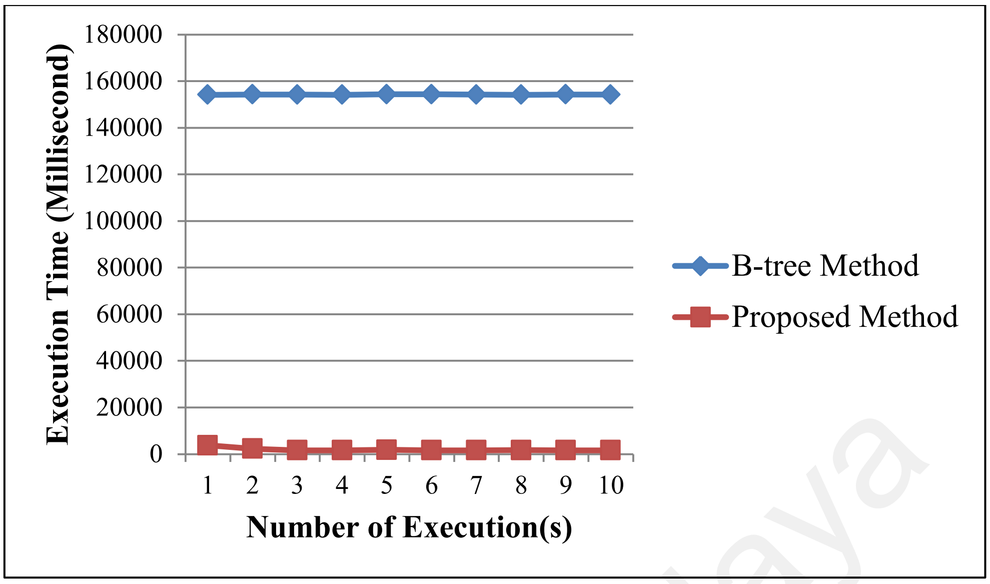

Figure 6.19: Execution time for Query 4 with data size 17.018 GB

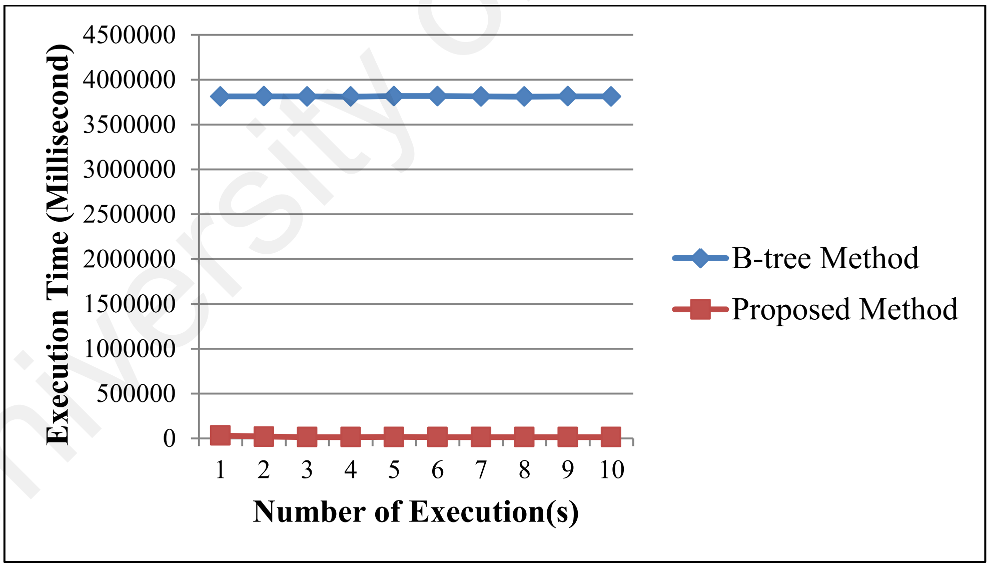

Figure 6.20: Execution time for Query 4 with data size 68.073 GB

University of Malaya

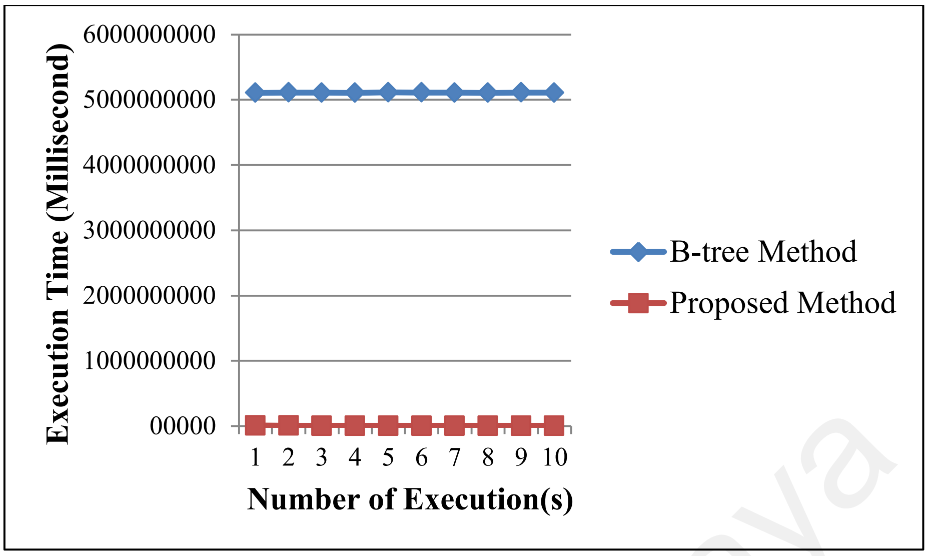

Figure 6.21: Execution time for Query 4 with data size 1089.163 GB

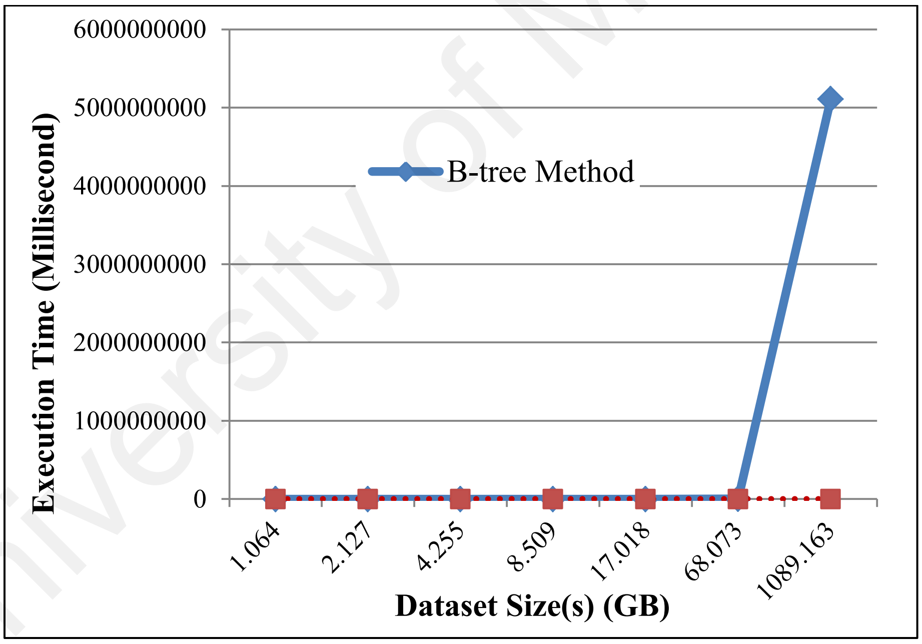

Figure 6.22: Average execution time of processing Query 4

As an overall analysis on the performance of the proposed search method and B-tree search method for executing Query 4, the outcomes of the benchmarking experiments shows that when the size of data grows up from 1GB to 68GB, the performance of the proposed search method is 201 times better than the performance of the B-tree search

University of Malaya

method for searching the same query with using the same dataset. But, when the size of datasets exceeds 1 TB, the performance of the proposed method is 4394 times better than the B-tree search method and the B-tree search method is totally out of a performance in this comparison. Figure 6.22 images this significant efficiency of using the proposed search method rather than the B-tree search method.

6.3 Discussion

Based on the above analysis of the trend of the execution time for processing the four predefined queries with the range of simple query to complex one, it is clear that the performance of the proposed search method is much better than the performance of the B-tree search method for all types of the queries. However, by the raise of the size of the datasets, the superiority of the performance of the proposed search method over the performance B-tree search method strongly increases. Figure 6.23 is an example of visualizing the efficiency of the proposed search method over the performance of the B-tree search method especially when the size of data is growing up.

University of Malaya

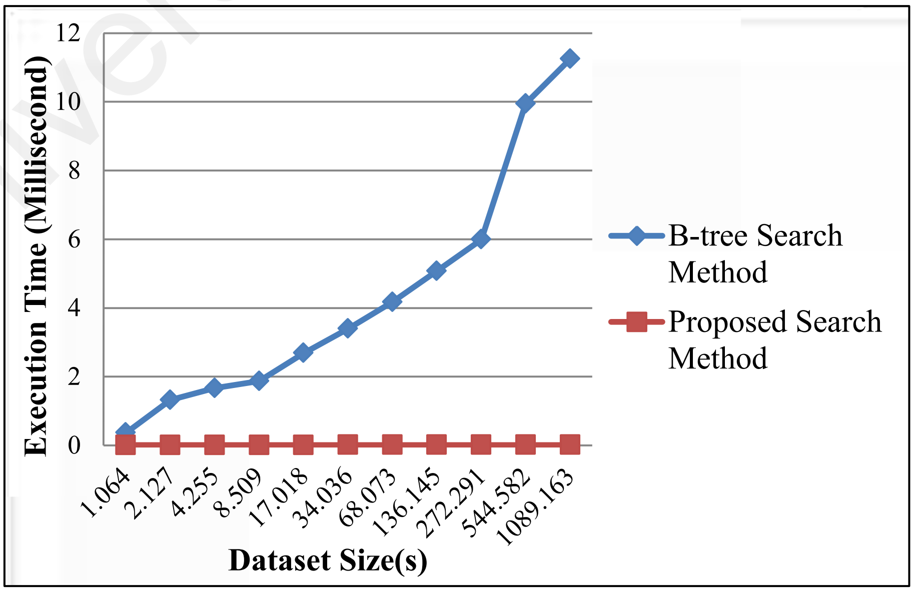

Figure 6.23: Comparison of average execution time of processing Query 1

Moreover, using history value as a part of metadata of the nodes inside the tree structure of the indexed data, speeds up the execution process of later queries by using the results of comparisons in the search algorithm of the earlier queries. This advantage provides reusability of the computation processes of search operations for later search tasks. For instance, Figure 6.24 clearly demonstrates the trend of the execution time of processing Query 1 after the first search operation that strongly decreasing.

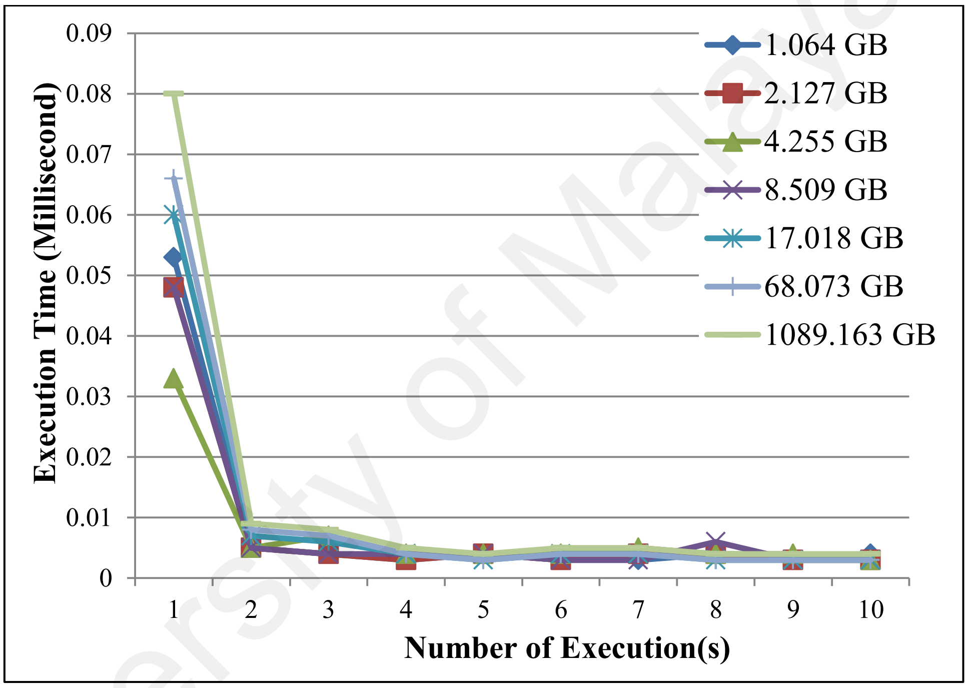

Figure 6.17: Execution time for processing Query 1

University of Malaya

Additionally, as it is illustrated in Figure 4.1 in Chapter 4, storing the data about the index structures inside the physical storage gives the opportunity of reusing the whole index data for further actions.

6.4 Conclusion

This chapter reports the results of evaluating the performance of our proposed search method which were extracted using benchmarking experiments. Several charts and tables are used to illustrate and demonstrate the outcomes of the performance evaluation of the proposed search method. Whereas the proposed method is developed based on the B-tree search method, a comparative study of the performance of both methods is conducted in order to declare justifiability of using proposed method instead of the B-tree search method. Section 6.1 presents the result of the benchmarking experiments. The significance of time-efficiency of the proposed method is demonstrated via analysis of the results of benchmarking experiments in which different sizes of datasets are used and four predefined queries from the range of simple query to complex one are processed multiple times using both B-tree search method and our proposed search method.

The proposed search method is remarkably effective for searching big data sets. Using the proposed search method strongly reduces the execution time of processing both simple and complex queries when the size of data is from few Gigabytes to tens of Gigabytes in contrast with the execution time for processing same query and searching same data and size of data via B-tree search method. However, when the size of data grows up toward Terabyte and exceeds more than Terabyte, B-tree search method is completely out of performance in comparison with the performance of our proposed solution. Additionally, adding history value to the node of the tree-based structure of the data provides reusability of earlier comparisons result of the search algorithm to be reused in later search operations.

The trend of the execution time for processing a simple or complex query from 18% for searching low size datasets and up to more than 90% for searching larger size of datasets via our proposed search method is reduced. Moreover, analyzing the results of the benchmarking experiments shows that the average execution time of the later search

University of Malaya

operations for processing simple queries is up to 14.45 times better than the average of the execution time of the earlier search task. Finally, the history value improves the performance of the later search processes on big data sets up to 52%.

Next chapter concludes this thesis by highlighting aim and adjectives of this research along with the major contributions of the thesis. It also outlines the potential opportunities to further improve or extend the work presented in the thesis. To this end, this thesis stands as a substantial effort to optimize resource consumption in data retrieval for big data by using query summarization concept and reusability approach. Particularly, this research focuses on optimizing B-tree search performance for big datasets.

University of Malaya

CHAPTER 7: CONCLUSION

7.1 Introduction

This chapter concludes the major contributions of the thesis. It also outlines the potential opportunities to further improve or extend the work presented in the thesis. To this end, this thesis stands as a substantial effort to optimize resource consumption in data retrieval in big data by using query summarization concept and reusability approach. Particularly, this research focuses on optimizing B-tree search performance for big data sets.

The rest of this chapter presents the outlines as follow. Section 7.2 reports the efforts that are taken to obtain the aim and objectives of this research. Research scope and limitations of this study are reported in Section 7.3. Section 7.5 highlights the significance and contributions of this work. Future works and directions of this study are elaborated in Section 7.5.

7.2 Aim and objectives of the study

In this thesis, we aimed to achieve an optimized B-tree search method to improve the execution time of search tasks and to improve the performance of the B-tree search process by using query summarization concept and reusability approach. The following presents our verification of the accomplishment of the aim of this thesis via describing the fulfillment of the objectives of this study.

a. ***Study the current big data indexing techniques and identify the key issues with respect to tree-based indexing techniques.***

This objective is accomplished via reviewing the state-of-the-art research from literature and provides the previous works that support this study and related big data

University of Malaya

indexing concepts and techniques. Moreover, it identifies the open problem related to this research. A comparison of the current indexing techniques based on the requirements of big data indexing which are extracted from previous related works is given in the following of this Chapter and demonstrated in Table 2.2. Furthermore, Chapter 2 states some open research challenges and highlights the problem which is addressed in this thesis.

However, we focused on big data requirements that are required to be handled by indexing techniques. Some of the current indexing techniques are analyzed based on previous studies and then we discussed the analysis in the Section 2.3.3 of Chapter 2. Moreover, B-tree indexing technique and its mechanism are discussed in Section 2.4 of Chapter 2. Some of the basic database operations using B-tree indexing technique are explained. Also, the B-tree search method and the flow of its search algorithm are described. At the end, the weakness of the B-tree search method based on the flow of its search algorithm is elaborated.

b. *Investigate the problem of the current B-Tree search method.*

This study investigated the problem of the search algorithm in B-tree indexing technique as the identified problem of this research. In Chapter 3 of this thesis, it is reported how our investigation is conducted and what are the activities in establishing the essence of our research problem. A quantitative analysis based on the time consumption of search process in one of the database systems named Mongo DB (Chodorow, 2013) in which B-tree indexing technique is the default indexing technique is conducted. Time consumption of B-tree in indexing is an essential part of this study since it is the basic that the rest of the work and other algorithms proposed are based on.

University of Malaya

It is one of the most widely accepted methods for analysis optimization problem in indexing. To do time consumption analysis for each query in data retrieval, a cost best model is used to measure the time. Following the approach used in (Banker, 2011). (Chodorow, 2013)

To run our experiments, a simulation engine is created using java programming language. This simulation engine can connect to the database system and apply queries against the datasets inside the database by calling proper libraries, classes, and function. Six different sizes of datasets starting from 240 MB to 1.2 TB are used for searching four predefined simple and complex queries. The search process is repeated 2o times for each query. Based on the results reported in Chapter 3, it is shown that the response time which is the consumed time for processing the queries, always is almost same and the resources particularly the consumed time for processing earlier queries do not have any impact on the response time of later queries even for same queries. However, the results show that by the growth of the size of datasets, the response time of every query is increasing impressively.

Moreover, we analyzed B-tree structure formally and resulted that repeating any number of a search operation to find a key in B-tree based indexed data is consuming the same amount of time and computing resources to perform comparisons on keys stored at nodes of the B-tree. These search comparisons are applying against the keys stored in the nodes of the sub-trees under each non-leaf node, even the target is not in the range of the minimum and the maximum values of the sub-trees. Upon searching for a key in a tree based index structure, the search algorithm traverses the tree from root to leaf, makes comparisons with keys stored in the nodes of the tree and based on the comparisons' results decides to continue searching in the left or right sub-trees. This

University of Malaya

process repeats every time even next queries are same or have overlap with earlier queries.

c. *Propose a solution to optimize the performance of the B-Tree search method for big data sets.*

In this research, a modified indexing structure based on B-tree data structure which is a popular and default data structure used in most of the current databases is proposed, designed, and implemented. Section 4.2 presents details of a modified indexing technique using the modified indexing structure. However, Section 4.5 describe a data retrieval simulation created using java programming language, in which, an index creator engine calls the datasets and starts creating indexes based on some of the predefined attributes according to four predefined queries in order to speed up the search process while is looking for the results of those queries. This engine stores the indexes data into some files to provide the reusability of the index data in future. It means, any time it requires to search more datasets in addition to the previous datasets, the index creator engine is able to load whatever index data is created in the past and then it starts indexing new datasets and adding related indexed data to the previously indexed data.

Also, Every time a backup engine creates a backup of the index files to keep a last copy of the indexed data after indexing every dataset file which consists of semi-structured data in the form of JSON format. We also used a min-max algorithm to add some metadata to each node of the tree to prevent non-necessary comparisons during the search process.

University of Malaya

Moreover, a history updater is developed which is responsible for adding some information about the last applied search into the visited nodes to provided reusability feature for later queries in the case of having similarity and overlap with earlier queries. The flow of the search process is demonstrated in Section 4.6. Finally, a query engine is created to search the indexed data to find ant results for the four predefined queries. This engine captures the response time for ever search process in two modes; (1) using normal B-tree search method, (2) using our proposed B-tree based search method. This engine repeats the search process for a given number of times. Figure 4.1 illustrated how above components are connected and interact.

d. ***Evaluate the performance of the proposed search method by validating it with the performance of the B-tree search method.***

To fulfill this objective, the proposed model is evaluated using benchmarking and also using comparative study. Based on Sieve benchmark, ten benchmarks are used to run a performance evaluation for our proposal. The sufficiency of this number of benchmarks for evaluating the performance of computing systems by using Sieve benchmark is already proven (Jain, 2008). For the comparative study, the performance of our proposed model in comparison with the related model, specifically with the performance of search method of normal B-tree technique is demonstrated.

The results of the performance evaluation stage are validated and discussed by analyzing and comparing the results of the proposed search method with the results of the normal B-tree search method.

University of Malaya

7.3 Research Scope and Limitations

The scope of this research mainly consists of to two parts: (1) analyzing the problem of the search process of B-tree indexing technique and (2) proposing an improved search method for optimizing B-Tree search performance of big data sets. The study in this thesis has limitations from bellow aspects:

- This research concentrates on improving search process of B-tree indexing technique for big datasets. Therefore, other processes and other indexing techniques are not considered in this study.
- The main evaluation metric used in this research is the time consumed for processing queries against large datasets, Search Performance which is the difference in Search Time between no indexed and indexed searches, and also comparing the complexity of the search algorithm of B-tree indexing technique with the complexity of the proposed search algorithm for B-tree indexing technique. Hence, other evaluation metrics are not in used in this study.
- This work used semi-structured data as the datasets for analyzing the problem that this study focused on and also for testing and evaluating the performance of our proposed search method. So, other types of data are not used in this research.

7.4 Significance and Contributions

In this study, the concept of reusability and query summarization were integrated into indexing procedures by creating node base summary of data in order to decrease the search execution time and also reuse the resources used for earlier queries and help to minimize the response time of later queries. This research proposed and implemented an optimize data retrieval system for big data which offers a new search method to the industries and institutions who need a faster data retrieval process and also it gives

University of Malaya

direction to researchers towards a novel approach of indexing diverse types of big data in order to improve query processing and ease data retrieval. The contributions and achievements of this research are briefly described as follows.

Big data indexing taxonomy: As the first contribution of this research, a big data indexing taxonomy is designed based on a classification using four main categories including (i) content format, (ii) structures, (iii) requirements, and (iv) data retrieval. This taxonomy gives a better understanding of areas that are big data indexing concept is dealing with. However, it reports the requirements of big data indexing based on the state-of-the-art existing related literature. Also sufficient elaboration of each category and its subcategories are provided as well. This contribution is presented in Chapter 2.

Comparative analysis of some of the big data indexing techniques based of the requirements of the big data indexing: This analysis of the indexing techniques is based on the current indexing techniques studied in (Gani et al., 2016) and some of the other indexing techniques studied in other related literature. The strength of each indexing technique detailed to ensure its viability for big data. Figures Table 2.2 of Chapter 2 lists some of the current indexing methods and presents which technique can address which big data indexing requirements.

Java based simulation query engine: A simulation engine is developed to connect to MongoDB database (MongoDB, 2016) and process queries. The MongoDB database system (MongoDB, 2016) is a database system that can deal with big data sets. The simulation calculates and records the response time for each search operation. Explanations and related figures are provided in Chapter 3.

University of Malaya

Impact of raising the volume of data on data retrieval response time using B-tree indexing technique: This research contributed to the body of knowledge by identifying and analyzing the impact of raising the volume of data on the response time of processing using B-tree indexing technique via an experimental analysis. The trend of response time in big data retrieval using B-tree indexing technique is investigated and demonstrated. However, the impact of processing earlier queries on processing later queries when the queries and datasets are same is identified. Chapter 3 details more about this achievement.

Improved big data retrieval system: We proposed an improved big data retrieval system with 6 main components including Storage, Index Generator, Summarization Updater, Backup Engine, Search Query Engine and History Updater. Figure 4.1 in Chapter 4 illustrates this system. This System uses an improved B-tree indexing technique in which it uses our proposed search method to speed up the query processing and reduce execution time of the search operations by using Summarization Updater and History Updater to add and update metadata to the node of the structure of the indexed data. However, this system can make a backup of the index data and store it into a file into the physical storage and reuse for further search process in future. This feature and also added metadata to the nodes provide reusability of the previous indexing process and also search operations' results.

Implemented proposed big data retrieval system: Based on our proposed system for big data retrieval, we implemented an application which offers all components of the proposed system. This program of this system is written by java and this system is used for experimental tasks. The detail of this implementation is reported in Chapter 4.

University of Malaya

Optimized B-tree based search method: An optimized search method based on B-tree search method is proposed and it is used it in B-tree indexing technique instead of its default search method. This method reduces the execution time of processing queries by using an added metadata to the nodes of the B-tree indexing structure in which minimum and maximum values of the branch of the node plus a history value of the last search operation help the search algorithm of the proposed method to reduce the number of comparisons for finding the key of conditions of queries. Section 4.2.6 in Chapter 4 gives more information and illustration about this method. This method significantly improves the performance of the search process in comparison with the normal search method of the B-tree indexing technique.

Improved B-tree (IB-tree) indexing structure: Based on our idea for having an optimized B-tree search method, we proposed a new structure of an improved B-tree indexing technique. In this structure, a new element to every node of the B-tree index structure is added and it is named as metadata. This metadata will be check and update every time the search algorithm traverse the tree of the nodes in order to reduce the number of comparisons especially when the size of data is growing up. The elaboration and the demonstration of this structure are presented in Chapter 4.

Proposed search method evaluation and validation: We contributed to the body of the knowledge by evaluating and validating the performance of our proposed search method to demonstrate its significance, reliability and also the validity of its functionality. Chapter 5 describes the details of the system and used data to evaluate and validate our proposed solution. However, Chapter 6 the performance evaluation results and validation outcomes are reported in Chapter 6. The results show that our new proposed search method decreases the execution time of the search tasks and it

University of Malaya

improves the search performance rather than B-tree search performance for same query and same dataset. Also, using this search method improves the performance of the later queries up to 52% by adding history of earlier queries to metadata of the nodes.

7.5 Future Research Directions of the Study

This work proposed an improved search method for the B-tree indexing technique with minimizing the query response time. However, our proposed method does not consider computation overhead issues. Therefore, improving this method in term of computation and other resource consumption is the future work of this research.

Moreover, a min-max summarization technique is used to add a meta-data to each node of the B-tree structure in order to improve the search algorithm. The second future work of this research is using other summarization techniques and comparing the performances of the improved search method while using different summarization techniques.

Also, enriching history data as another meta-data inside the nodes of the B-tree indexing technique is the third future work of this study in order to optimize search performance of the B-tree indexing technique for big datasets.

This study aimed to optimize the search performance of the standard B-tree for big data sets. As it is highlighted in Section 4.2, the proposed solution of this work is an extension of the standard B-tree and only the search algorithm of the standard B-tree is modified and it is shown in Chapter 6 that the performance of the proposed B-tree search method in term of execution time is better than the performance of the standard B-tree search method. Therefore, conducting a comparative study among all other

University of Malaya

modified B-tree and also our proposed solution can be another future research direction of this study.

Finally, this research gives direction to researchers towards a novel approach of indexing diverse types of big data in order to improve query processing and ease data retrieval. This study used semi-structured data as a dataset for analyzing the problem and also for testing and evaluating the proposed method. The forth future work of this research is using the proposed method to index and search structured and unstructured data.

University of Malaya

REFERENCES

Agarwal, N., Gvr, K., Reddy, R. S., & Rosé, C. P. (2011). Towards multi-document summarization of scientific articles: making interesting comparisons with SciSumm. Paper presented at the Proceedings of the Workshop on Automatic Summarization for Different Genres, Media, and Languages (pp. 8-15).

Agrawal, R., Kadadi, A., Dai, X., & Andres, F. (2015). Challenges and opportunities with big data visualization. Paper presented at the Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems (pp. 169-173).

Ahmaro, I. Y., Abualkishik, A. M., & Yusoff, M. Z. M. (2014). Taxonomy, Definition, Approaches, Benefits, Reusability Levels, Factors and Adaption of Software Reusability: A Review of the Research Literature. Journal of Applied Sciences, 14(20), 2396.

Alam, M., Nielsen, R. H., & Prasad, N. R. (2013). The evolution of M2M into IoT. Paper presented at the Communications and Networking (BlackSeaCom), 2013 First International Black Sea Conference on (pp. 112-115).

Alloui, I., & Oquendo, F. (2002). Supporting decentralised software-intensive processes using zeta component-based architecture description language. Enterprise Information Systems III, 3, 97.

Amma, N. B. (2016). Big Data Mining Effective Big Data Management and Opportunities for Implementation (pp. 53-59): IGI Global.

Ashton, K. (2009). That ‗internet of things' thing. RFiD Journal, 22(7), 97-114.

Athanassoulis, M., Yan, Z., & Idreos, S. (2016). UpBit: Scalable In-Memory Updatable Bitmap Indexing. Paper presented at the ACM SIGMOD International Conference on Management of Data (pp. 1319-1332).

Azim, S. K. (1988). Application of silicon compilation techniques to a robot controller design (pp. 267).

Balaji, J., Geetha, T., & Parthasarathi, R. (2016). Abstractive summarization: A hybrid approach for the compression of semantic graphs. International Journal on Semantic Web and Information Systems (IJSWIS), 12(2), 76-99.

Banker, K. (2011). MongoDB in action: Manning Publications Co (pp. 312).

Barbierato, E., Gribaudo, M., & Iacono, M. (2014). Performance evaluation of NoSQL big-data applications using multi-formalism models. Future Generation Computer Systems, 37, 345-353.

University of Malaya

Bayer, R. (1971, November). Binary B-trees for virtual memory. In Proceedings of the 1971 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control (pp. 219-235). ACM.

Bayer, R., & McCreight, E. (2002). Organization and maintenance of large ordered indexes. In Software pioneers (pp. 245-262). Springer Berlin Heidelberg.

Benatallah, B., Sakr, S., Grigori, D., Motahari-Nezhad, H. R., Barukh, M. C., Gater, A., & Ryu, S. H. (2016). Tools, Use Cases, and Discussions Process Analytics (pp. 135-150): Springer.

Berman, J. J. (2013). Introduction Principles of Big Data (pp. xix-xxvi). Boston: Morgan Kaufmann.

Bhardwaj, V., & Johari, R. (2015). Big data analysis: Issues and challenges. Paper presented at the Electrical, Electronics, Signals, Communication and Optimization (EESCO), 2015 International Conference . ISBN 978-1-4799-7676-8, IEEE.

Booysen, M. J., Gilmore, J. S., Zeadally, S., & Van Rooyen, G.-J. (2012). Machine-to-machine (M2M) communications in vehicular networks. available at www.itiis.org.

Boubekeur, F., & Azzoug, W. (2013). Concept-based indexing in text information retrieval. arXiv preprint arXiv:1303.1703.

Bühlmann, P. (2013). Statistical significance in high-dimensional linear models. Bernoulli, 19(4), 1212-1242.

Bui, T. H., Frampton, M., Dowding, J., & Peters, S. (2009). Extracting decisions from multi-party dialogue using directed graphical models and semantic similarity. Paper presented at the Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue.

Bukh, P. N. D., & Jain, R. (1992). The art of computer systems performance analysis, techniques for experimental design, measurement, simulation and modeling: JSTOR, Vol. 22, No. 4, pp. 113-115.

Burgos, J. L. M. (2011). Semantic Web Standards. SNET Computer Engineering. Retrieved from http://www.pdffiller.com/948565-semantic-web-standards_ burgos-Semantic-Web-Standards---SNET-Various-Fillable-Forms-snet-tu-berlin

Cambazoglu, Kayaaslan , Jonassen , & Aykanat. (2013). A term-based inverted index partitioning model for efficient distributed query processing. . ACM Trans Web, 7(3), 1-23. doi:doi:10.1145/2516633.2516637

Cambazoglu, B. B., Kayaaslan, E., Jonassen, S., & Aykanat, C. (2013). A term-based inverted index partitioning model for efficient distributed query processing. ACM Transactions on the Web (TWEB), 7(3), 15.

University of Malaya

Campbell, J. C., Santos, E. A., & Hindle, A. (2016). The unreasonable effectiveness of traditional information retrieval in crash report deduplication. Paper presented at the Proceedings of the 13th International Workshop on Mining Software Repositories. (pp. 269-280). ACM.

Chakrabarti, S., Pathak, A., & Gupta, M. (2011). Index design and query processing for graph conductance search. The VLDB Journal, 20, 445–470.

Chan, C.-Y. (2011). Connected vehicles in a connected world. Paper presented at the VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on. (pp. 1-4). IEEE.

Che, D., Safran, M., & Peng, Z. (2013). From Big Data to Big Data Mining: Challenges, Issues, and Opportunities. In B. Hong, X. Meng, L. Chen, W. Winiwarter, & W. Song (Eds.), Database Systems for Advanced Applications (Vol. 7827, pp. 1-15): Springer Berlin Heidelberg.

Chen, Chen, Du, C, L., Lu, Zhao, & Zhou. (2013). Big data challenge: a data management perspective. Front Comput Sci, 7(2), 157–164. doi:doi:10.1007/s11704-013-3903-7

Chen, C. P., & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences, 275, 314-347.

Chen, G., & Luo, W. (2015). Clustering Time-Evolving Data Using an Efficient Differential Evolution. Paper presented at the International Conference in Swarm Intelligence. (pp. 326-338). Springer International Publishing.

Chen, H.-C., Wu, C.-L., Sun, J.-S., & Feng, H.-M. (2016). Carrier Current Line Systems Technologies in M2M Architecture for Wireless Communication. Journal of Sensors, vol. 2016, Article ID 2652310, 10 pages, 2016. doi:10.1155/2016/2652310 .

Chen, J., Chen, Y., Du, X., Li, C., Lu, J., Zhao, S., & Zhou, X. (2013). Big data challenge: a data management perspective. Frontiers of Computer Science, 7(2), 157-164.

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

Cheng, J., Ke, Y., Fu, A. W.-C., & Yu, J. X. (2011a). Fast Graph Query Processing with a Low-Cost Index. he VLDB Journal, 20(4), 521-539.

Cheng, J., Ke, Y., Fu, A. W.-C., & Yu, J. X. (2011b). Fast graph query processing with a low-cost index. The VLDB Journal, 20(4), 521-539.

Chodorow, K. (2013). MongoDB: the definitive guide: " O'Reilly Media, Inc.".

University of Malaya

Comer, D. (1979). Ubiquitous B-tree. ACM Computing Surveys (CSUR), 11(2), 121-137

Cormen, T. H. (2009). Introduction to algorithms: (Vol. 6). Cambridge: MIT press .

Cottle, M., Hoover, W., Kanwal, S., Kohn, M., Strome, T., & Treister, N. (2013). Transforming Health Care Through Big Data Strategies for leveraging big data in the health care industry. Institute for Health Technology Transformation, http://ihealthtran. com/big-data-in-healthcare.

Commons, C. (2013). Creative commons attribution-sharealike 4.0 international (cc by-sa 4.0). Zugriff au f http://creativecommons.org/licenses/by-sa/4.0/(Zuletzt abgerufen: 02.2015).

Commons, O. D. (2013). Open Database License (ODbL). Retrieved from https://opendatacommons.org/licenses/odbl/

Cormen, T. H. (2009). Introduction to algorithms: MIT press. Retrieved from http://cs.slu.edu/~goldwasser/courses/loyola/comp363/2003_Spring/handouts/course-info.pdf

Cottle, M., Hoover, W., Kanwal, S., Kohn, M., Strome, T., & Treister, N. (2013). Transforming Health Care Through Big Data Strategies for leveraging big data in the health care industry. Institute for Health Technology Transformation, http://ihealthtran.com/big-data-in-healthcare.

D. Feldman, M. Schmidt, & Sohler, C. (2013). Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (pp. 1434-1453). Society for Industrial and Applied Mathematics.

de Mattos, W. D., & Gondim, P. R. (2016). M-Health Solutions Using 5G Networks and M2M Communications. IT Professional, 18(3), 24-29.

Dean, J., & Ghemawat, S. (2010). MapReduce: A Flexible Data Processing Tool. Communications of the ACM, 53(1), 72-77.

Delbru, R., Campinas, S., & Tummarello, G. (2012). Searching web data: An entity retrieval and high-performance indexing model. Web Semantics: Science, Services and Agents on the World Wide Web, 10, 33-58.

Denecke, K., & Nejdl, W. (2009). How valuable is medical social media data? Content analysis of the medical web. Information Sciences, 179(12), 1870-1880.

DeWitt, D., & Gray, J. (1992.). Parallel Database Systems: The Future of High Performance Database Systems. Communications of the ACM, 35(6), 85–98.

Dieng-Kuntz, R., Minier, D., Růžička, M., Corby, F., Corby, O., & Alamarguy, L. (2006). Building and using a medical ontology for knowledge management and

University of Malaya

cooperative work in a health care network. Computers in Biology and Medicine, 36(7), 871-892.

Dijkman, R. M., van Dongen, B. F., Dumas, M., García-Bañuelos, L., Kunze, M., Leopold, H., . . . Weske, M. (2013). A short survey on process model similarity Seminal Contributions to Information Systems Engineering (pp. 421-427): Springer.

Dittrich, J., Blunschi, L., & Salles, M. A. V. (2011). MOVIES: indexing moving objects by shooting index images. Geoinformatica, 15(4), 727-767.

Divyakant Agrawal, P. B., Elisa Bertino, Susan Davidson, Umeshwar Dayal, Michael, Franklin, J. G., Laura Haas, Alon Halevy, Jiawei Han, H. V. Jagadish, Alexandros, Labrinidis, S. M., Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan,, & Kenneth Ross, C. S., Dan Suciu, Shiv Vaithyanathan, and Jennifer Widom. (2012). Challenges and Opportunities with Big Data: A community white paper developed by leading researchers across the United States. Whitepaper, Computing Community Consortium.

Done, B., Khatri, P., Done, A., & Draghici, S. (2010). Predicting novel human gene ontology annotations using semantic analysis. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 7(1), 91-99.

Dong, W. (2010). Collaborative Indexing and Knowledge Exploration: A Social Learning Model. INTELLIGENT SYSTEMS, IEEE.

Dumais, S., Cutrell, E., Cadiz, J. J., Jancke, G., Sarin, R., & Robbins, D. C. (2016). Stuff I've seen: a system for personal information retrieval and re-use. Paper presented at the ACM SIGIR Forum. (Vol. 49, No. 2, pp. 28-35). ACM

Efron, B. (2010). Large-Scale Inference: Empirical Bayes Meth- ods for Estimation, Testing, and Prediction. Institute of Mathematical Statistics Monographs: Vol. 1. Cambridge University Press .

Elleuch, N., Zarka, M., Ammar, A. B., & Alimi, A. M. (2011). A fuzzy ontology: based framework for reasoning in visual video content analysis and indexing. Paper presented at the Proceedings of the Eleventh International Workshop on Multimedia Data Mining. (p. 1). ACM

Faloutsos, C., Kolda, T. G., & Sun, J. (2007). Mining large time-evolving data using matrix and tensor tools. Paper presented at the ICDM Conference. Conference (Vol. 565)

Fan, W., & Bifet, A. (2013). Mining big data: current status, and forecast to the future. ACM sIGKDD Explorations Newsletter, 14(2), 1-5.

Fernández, R., Frampton, M., Dowding, J., Adukuzhiyil, A., Ehlen, P., & Peters, S. (2008). Identifying relevant phrases to summarize decisions in spoken meetings. Paper presented at the INTERSPEECH. (pp. 78-81)

University of Malaya

Franks, B. (2012). Taming the big data tidal wave: Finding opportunities in huge data streams with advanced analytics (Vol. 49): John Wiley & Sons.

Fu, W.-T., & Dong, W. (2010). Collaborative Indexing and Knowledge Exploration: A Social Learning Model. Intelligent System, 27(1), 39-46 IEEE.

Fu, W.-T., & Dong, W. (2012). Collaborative indexing and knowledge exploration: A social learning model. IEEE Intelligent Systems, 27(1), 39-46.

Gacto, M. J., Alcalá, R., & Herrera, F. (2010). Integration of an index to preserve the semantic interpretability in the multiobjective evolutionary rule selection and tuning of linguistic fuzzy systems. IEEE Transactions on Fuzzy Systems, 18(3), 515-531.

Gama, J. (2010). Knowledge Discovery from Data Streams Chapman & Hall/Crc Data Mining and Knowledge Discovery. CRC Press, Boca Raton, FL, 2010. xx+237 pp. ISBN: 978-1-4398-2611-9

Gani, A., Siddiqa, A., Shamshirband, S., & Hanum, F. (2016). A survey on indexing techniques for big data: taxonomy and performance evaluation. Knowledge and Information Systems, 46(2), 241-284.

Gantz, J., & Reinsel, D. (2012a). THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. Study report,. IDC iView: IDC Analyze the future 2007.2012 (2012): 1-16.

Gantz, J., & Reinsel, D. (2012b). The Digital Universe In 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. Study report, IDC iView: IDC Analyze the future 2007.2012 (2012): 1-16.

García-Hernández, R. A., Ledeneva, Y., Mendoza, G. M., Dominguez, Á. H., Chavez, J., Gelbukh, A., & Fabela, J. L. T. (2009). Comparing commercial tools and state-of-the-art methods for generating text summaries. Paper presented at the Artificial Intelligence, 2009. MICAI 2009. Eighth Mexican International Conference on. (pp. 92-96). IEEE

Gerken, J., Bak, P., Jetter, C., Klinkhammer, D., & Reiterer, H. (2008). How to use interaction logs effectively for usability evaluation.. In: CHI 2008 Workshop BELIV ' 08 : Beyond time and errors - novel evaLuation methods for Information Visualization, Apr 2008. BELIV, Apr

Golab, L., Prahladka, P., & Ozsu, M. T. (2006). Indexing time-evolving data with variable lifetimes. Paper presented at the 18th International Conference on Scientific and Statistical Database Management (SSDBM'06).

Gopalkrishnan, V., Steier, D., Lewis, H., & Guszcza, J. (2012, August). Big data, big business: bridging the gap. In Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications (pp. 7-11). ACM.

University of Malaya

Goswami, B., & Chandra, P. K. (2015). The Evolution Of Big Data As A Research And Development. International Journal of Scientific Research and Engineering Studies (IJSRES), 2(3).

Goyal, N., & Gupta, D. (2014). Reusability Calculation of Object Oriented Software Model by Analyzing CK Metric. International Journal of Advanced Research in Computer Engineering & Technology, 3(7), 2466-2470.

Graefe, G. (2016). B-Tree Locking Encyclopedia of Database Systems (pp. 1-6): Springer.

Gundema, & Armaganb. (2006). Efficient storage of healthcare data in XML-based smart cards. computer methods and programs in biomedicine, 8 (1), 26–40.

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

Hassan-Montero, Y., & Herrero-Solana, V. (2006). Improving tag-clouds as visual information retrieval interfaces. Paper presented at the International conference on multidisciplinary information sciences and technologies. (pp. 25-28).

He, Y., Elnikety, S., Larus, J., & Yan, C. (2012). Zeta: Scheduling interactive services with partial execution. Paper presented at the Proceedings of the Third ACM Symposium on Cloud Computing. (p. 12). ACM

Hellerstein, J. M., Naughton, J. F., & Pfeffer, A. (1995). Generalized search trees for database systems (pp. 562-573) September.

Hoffman, M., Hoffman, J., Hoffman, A., & Doe, D. (2016). Personal security and tracking system: Google Patents. U.S. Patent No. 5,742,233. Washington, DC: U.S. Patent and Trademark Office. Hovy, E., & Lin, C.-Y. (1998). Automated text summarization and the SUMMARIST system. Paper presented at the Proceedings of a workshop on held at Baltimore, Maryland: October 13-15, 1998.

Hsu, W. H., King, A. L., Paradesi, M. S., Pydimarri, T., & Weninger, T. (2006). Collaborative and Structural Recommendation of Friends using Weblog-based Social Network Analysis. Paper presented at the AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs. (Vol. 6, pp. 55-60).

Hu, W., Tan, T., Wang, L., & Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 34(3), 334-352.

Huang, Z., Lu, X., Duan, H., & Zhao, C. (2012). Collaboration-based medical knowledge recommendation. Artificial intelligence in medicine, 55(1), 13-24.

University of Malaya

Idreos, S., Kersten, M. L., & Manegold, S. (2007). Database Cracking. Paper presented at the CIDR. (Vol. 7, pp. 68-78).

J. Gantz, & Reinsel., D. (2012). The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. IDC iView: IDC Analyze the future, 2007(2012), 1-16

Jain, A., Bolle, R., & Pankanti, S. (2006). Biometrics: personal identification in networked society (Vol. 479): Springer Science & Business Media.

Jamil, S., & Ibrahim, R. (2009). Performance analysis of indexing techniques in Data warehousing. Paper presented at the Emerging Technologies, 2009. ICET 2009. International Conference on. (pp. 57-61). IEEE

Jaseena, K., & David, J. M. (2014). Issues, Challenges, and Solutions: Big Data Mining. NeTCoM, CSIT, GRAPH-HOC, SPTM–2014, 131-140.

Jin, R., Cho, H. J., & Chung, T. S. (2014, January). A group round robin based b-tree index storage scheme for flash memory devices. In Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication (p. 29). ACM..

Jinchuan Chen, Y. C., Xiaoyong Du, Cuiping Li, Jiaheng Lu, Suyun Zhao, and Xuan Zhou. (2013). Big data challenge: a data management perspective. . Frontiers of Computer Science, 7(2), 157–164. doi:10.1007/s11704-013-3903-7

Jones, K. S. (2007). Automatic summarising: The state of the art. Information Processing & Management, 43(6), 1449-1481.

Kaisler, S., Armour, F., Espinosa, J. A., & Money, W. (2013a). Big data: issues and challenges moving forward. Paper presented at the System Sciences (HICSS), 2013 46th Hawaii International Conference on (pp. 995-1004). IEEE.

Kaisler, S., Armour, F., Espinosa, J. A., & Money, W. (2013b). Big Data: Issues and Challenges Moving Forward. In Proceedings of the 46th Hawaii International Conference on System Sciences, HICSS '13,, 995–1004.

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

Kammenhuber, N., Luxenburger, J., Feldmann, A., & Weikum, G. (2006). Web search clickstreams. Paper presented at the Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (pp. 245-250). ACM..

Katal, A., Wazid, M., & Goudar, R. (2013). Big data: issues, challenges, tools and good practices. Paper presented at the Contemporary Computing (IC3), 2013 Sixth International Conference on (pp. 404-409). IEEE..

University of Malaya

Kathuria, C., Datta, G., & Kaul, V. (2013). Context Indexing in Search Engine Using Binary Search Tree. International Journal on Computer Science and Engineering, 5(6), 514.

Kaur, P. D. (2015). A survey on Big Data storage strategies. Paper presented at the Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on. (pp. 280-284). IEEE.

Kaushik, V. D., Umarani, J., Gupta, A. K., Gupta, A. K., & Gupta, P. (2013). An efficient indexing scheme for face database using modified geometric hashing. Neurocomputing, 116, 208–221.

Keim, D., Qu, H., & Ma, K.-L. (2013). Big-data visualization. IEEE Computer Graphics and Applications, 33(4), 20-21.

Knuth, M., Waitelonis, J., & Sack, H. (2016). I am a Machine, let me understand Web Media! Paper presented at the International Conference on Web Engineering. (pp. 467-475). Springer International Publishing.

Ko, M. N., Cheek, G. P., Shehab, M., & Sandhu, R. (2010). Social-networks connect services. Computer, 43(8), 37-43.

Komkhao, M., Lu, J., Li, Z., & Halang, W. A. (2013). Incremental collaborative filtering based on Mahalanobis distance and fuzzy membership for recommender systems. International Journal of General Systems, 42(1), 41-66.

Kriegel, A. (2011). Discovering SQL: a hands-on guide for beginners: John Wiley & Sons.

Kumar, R., Novak, J., Raghavan, P., & Tomkins, A. (2005). On the bursty evolution of blogspace. World Wide Web, 8(2), 159-178.

Kumarasamy, M. (2015). An Analysis Of Big Data Discovery And Collaboration. International Journal Of Advanced Computer Technology, Volume 4,Number 6

Lakshman, A., & Malik, P. (2010). Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, 44(2), 35-40.

Laney, D. (2011). 3D Data Management: Controlling Data Volume, Velocity and Variety. Technical report, META Group, Inc (now Gartner, Inc.).

Laurila, J. K., Gatica-Perez, D., Aad, I., Blom, J., Bornet, O., Do, T.-M.-T., . . . Miettinen, M. (2012). The mobile data challenge: Big data for mobile computing research. Paper presented at the Proceedings of the Workshop on the Nokia Mobile Data Challenge, in Conjunction with the 10th International Conference on Pervasive Computing.

University of Malaya

LaValle, S., Lesser, E., Shockley, R., Hopkins, M. S., & Kruschwitz, N. (2011). Big data, analytics and the path from insights to value. MIT Sloan Management Review, 52(2), 21.

Lawal, Z. K., Zakari, R. Y., Shuaibu, M. Z., & Bala, A. A review: Issues and Challenges in Big Data from Analytic and Storage perspectives. ISSN:2319-7242Volume –5 Issue -03 March, 2016Page No.15947-15961

Lazaridis, M., Axenopoulos, A., Rafailidis, D., & Daras, P. (2013). Multimedia search and retrieval using multimodal annotation propagation and indexing techniques. Signal Processing: Image Communication, 28(4), 351-367.

Lemire, D., Kaser, O., & Aouiche, K. (2010). Sorting improves word-aligned bitmap indexes Data & Knowledge Engineering, 69(1), 3-28.

Leung, C., & Chan, W. (2011). Semantic music information retrieval using collaborative indexing and filtering Computer and information sciences (pp. 345-350): Springer.

Li, F., Yi, K., & Le, W. (2010). Top-k queries on temporal data. The VLDB Journal—The International Journal on Very Large Data Bases, 19(5), 715-733.

Lu, R., Li, X., Liang, X., Shen, X., & Lin, X. (2011). GRS: The green, reliability, and security of emerging machine to machine communications. IEEE Communications Magazine, 49(4), 28-35.

Madden, S. (2012). From Databases to Big Data. iEEE Internet Computing, 16(3), :4–6.

Maier, M., Rattigan, M., & Jensen, D. (2011). Indexing network structure with shortest-path trees. ACM Transactions on Knowledge Discovery from Data (TKDD), 5(3), 15.

Maier, M., Serebrenik, A., & Vanderfeesten, I. (2013a). Towards a big data reference architecture. EINDHOVEN UNIVERSITY MS thesis.

Maier, M., Serebrenik, A., & Vanderfeesten, I. (2013b). Towards a big data reference architecture: University of Eindhoven.

Manolopoulos, Y., Theodoridis, Y., & Tsotras, J. V. (2009). Tree-based Indexing. In L. Liu & M. T. ÖZsu (Eds.), Encyclopedia of Database Systems (pp. 3172-3173). Boston, MA: Springer US.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity, The McKinsey Global Institute. Available at http://www.citeulike.org/group/18242/article/9341321.

University of Malaya

Marz, N., & Warren, J. (2013). Big Data - Principles and best practices of scalable realtime data systems. Manning Publications Co. Greenwich, CT, USA ©2015 ISBN:1617290343 9781617290343.

Mehrotra, H., Majhi, B., & Gupta, P. (2010). Robust iris indexing scheme using geometric hashing of SIFT keypoints. Journal of Network and Computer Applications, 33(3), 300-313.

Mera, D., Batko, M., & Zezula, P. (2014). Towards fast multimedia feature extraction: Hadoop or storm. Paper presented at the Multimedia (ISM), 2014 IEEE International Symposium on. (pp. 106-109). IEEE

Mishra, B. S. P., Dehuri, S., & Kim, E. (2016). Techniques and Environments for Big Data Analysis: Parallel, Cloud, and Grid Computing (Vol. 17): Springer.

Mishra, S., Dhote, V., Prajapati, G., & Shukla, J. (2015). Challenges in Big Data Application: A Review. International Journal of Computer Applications, 121(19).

Mohanty, S., Jagadeesh, M., & Srivatsa, H. (2013). Big data imperatives: Enterprise ‗Big Data'warehouse,‗BI'implementations and analytics: Apress.

MongoDB. (2016). Indexes - MongoDB Manual 3.2. Retrieved from https://docs.mongodb.com/manual/indexes/

Motta, G., Puccinelli, R., Reggiani, L., & Saccone, M. (2016). Extracting Value from Grey Literature: processes and technologies for aggregating and analyzing the hidden ―big data" treasure of organizations. Grey Journal (TGJ), 12(1).

Nagpal, R., Nagpal, P., & Malhotra, S. (2012). Biometric techniques and facial expression recognition-A Review. Journal of Global Research in Computer Science, 3(11).

Namiot, D. (2015). On Big Data Stream Processing. International Journal of Open Information Technologies, 3(8), 48-51.

Nasir, M. A. U. (2016). Fault Tolerance for Stream Processing Engines. arXiv preprint arXiv:1605.00928.

Niyato, D., Xiao, L., & Wang, P. (2011). Machine-to-machine communications for home energy management system in smart grid. IEEE Communications Magazine, 49(4), 53-59.

NTER, U., & MILL, E. (2012). Agencies rally to tackle big data. Science, 336(6077), 22-22.

O'Neil, P., & Quass, D. (1997, June). Improved query performance with variant indexes. In ACM Sigmod Record (Vol. 26, No. 2, pp. 38-49). ACM.

University of Malaya

Olson, M. (2010). Hadoop: Scalable, flexible data storage and analysis. IQT Quart, 1(3), 14-18.

Orsborn, K. (2007). database technology 1dl124. Uppsala University,Uppsala, Sweden Retrieved from http://user.it.uu.se/~udbl/dbt-sommar07/alt. http://www.it.uu.se/edu/course/homepage/dbdesign/st07

OWM API, (2015). Open Weather Map. Retrieved from http://openweathermap.org/.

P. Zikopoulos, C. Eaton, D. d., T. Deutsch, & Lapis, G. (2011). IBM Understanding Big Data: Analyt- ics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill Companies,Incorporated. ISBN:0071790535 9780071790536

Paul, A., Chen, B.-W., Bharanitharan, K., & Wang, J.-F. (2013). Video search and indexing with reinforcement agent for interactive multimedia services. ACM Trans. Embed. Comput. Syst., 12(2), 1-16. doi:10.1145/2423636.2423643

Paul, A., & Rho, S. (2016). Probabilistic model for M2M in IoT networking and communication. Telecommunication Systems, 62(1), 59-66.

Qu, P., Zhang, J., Yao, C., & Zeng, W. (2016). Identifying long tail term from large scale candidate pairs for big data oriented patent analysis. Concurrency and Computation: Practice and Experience. 28(15), 4194-4208.

R. Smolan, & Erwitt, J. (2012). The Human Face of Big Data. Sterling Publishing Company Incorporated. Vol. 351, Issue 6274, pp. 673. DOI: 10.1126/science.aaf3194

Radev, D. R., Allison, T., Blair-Goldensohn, S., Blitzer, J., Celebi, A., Dimitrov, S., . . . Liu, D. (2004). MEAD-A Platform for Multidocument Multilingual Text Summarization. Paper presented at the LREC.

Rahman, N., & Borah, B. (2015). A survey on existing extractive techniques for query-based text summarization. Paper presented at the Advanced Computing and Communication (ISACC), 2015 International Symposium on (pp. 98-102). IEEE..

Rakesh Agrawal, A. A., Philip A. Bernstein, Eric A. Brewer, Michael J. Carey,, Surajit Chaudhuri, A. D., Daniela Florescu, Michael J. Franklin, Hector Garcia-Molina,, Johannes Gehrke, L. G., Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein,, Yannis E. Ioannidis, H. F. K., Donald Kossmann, Samuel Madden, Roger Magoulas,, Beng Chin Ooi, T. O. R., Raghu Ramakrishnan, Sunita Sarawagi, Michael Stonebraker,, & Alexander S. Szalay, a. G. W. (2009). The Claremont Report on Database Research.. 52(6), 56-65.

Ramakrishna, K., & Rani, D. B. P. (2013). Study of Indexing Techniques to Improve the Performance of Information Retrieval in Telugu Language. International Journal of Emerging Technology and Advanced Engineering, ISSN, 2250-2459.

University of Malaya

Reddy, H. V., & Raju, S. V. (2012). A study in employing rough set based approach for clustering on categorical time-evolving data. IOSR Journal of Computer Engineering (IOSRJCE), 3(5), 44-51.

Richter, S., Quiané-Ruiz, J.-A., Schuh, S., & Dittrich, J. (2012). Towards zero-overhead adaptive indexing in Hadoop. arXiv preprint arXiv:1212.3480.

Ritter, A., & Muñoz-Carpena, R. (2013). Performance evaluation of hydrological models: Statistical significance for reducing subjectivity in goodness-of-fit assessments. Journal of Hydrology, 480, 33-45.

Rodríguez-García, M. Á., Valencia-García, R., García-Sánchez, F., & Samper-Zapater, J. J. (2014a). Creating a semantically-enhanced cloud services environment through ontology evolution. Future Generation Computer Systems, 32, 295-306.

Rodríguez-García, M. Á., Valencia-García, R., García-Sánchez, F., & Samper-Zapater, J. J. (2014b). Creating a semantically-enhanced cloud services environment through ontology evolution. Future Generation Computer Systems, 32, 295–306.

Russo, L., Navarro, G., & Oliveira, A. L. (2011). Fully compressed suffix trees. ACM Transactions on Algorithms (TALG), 7(4), 53.

Sattar, A., & Zhang, H. (2014). Component based Software Development using Reusability Measurement. Software Engineering and Technology, 6(6), 174-178.

Schäler, M., Grebhahn, A., Schröter, R., Schulze, S., Köppen, V., & Saake, G. (2013). QuEval: beyond high-dimensional indexing à la carte. Proceedings of the VLDB Endowment, 6(14), 1654-1665.

Scott, J. (2016). Zeta – Enterprise Architecture | MapR. Retrieved from https://www.mapr.com/solutions/zeta-enterprise-architecture.

Sellis, Roussopoulos, & Faloutsos. (1987). The R+-Tree: A dynamic index for multi-dimensional objects. In VLDB. Proceedings of the 13th VLDB Conference, Brighton 1.

Seo, D., Jeon, Y.-B., Lee, S.-H., & Lee, K.-H. (2016). Cloud computing for ubiquitous computing on M2M and IoT environment mobile application. Cluster Computing, 19(2), 1001-1013.

Sergey Melnik, A. G., Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt, & Tolton, a. T. V. (2011). Dremel: Interactive Analysis of Web-Scale Datasets. . Communications of the ACM, 54(6), 114–123.

Shang, Yang, Wang, Chan, & Hua. (2010). Real-time large scale near-duplicate web video retrieval. In: Proceedings of the international conference on multimedia., 531–540.

University of Malaya

Sheng-Cheng, Su, M.-Y., Chen, H.-H., & Lin, C.-Y. (2013). An efficient and secure approach for a cloud collaborative editing. Journal of Network and Computer Applications, 36(1632), 1641.

Shung, C. B., Jain, R., Rimey, K., Wang, E., Srivastava, M. B., Richards, B. C., . . . Hilfinger, P. N. (1991). An integrated CAD system for algorithm-specific IC design. IEEE transactions on computer-aided design of integrated circuits and systems, 10(4), 447-463.

Siddiqa, A., Karim, A., & Chang, V. (2016). SmallClient for big data: an indexing framework towards fast data retrieval. Cluster Computing, 1-16.

Sidirourgos, E. (2014). Space efficient indexes for the big data era.

Sinclair, J., & Cardew-Hall, M. (2008). The folksonomy tag cloud: when is it useful? Journal of Information Science, 34(1), 15-29.

Singh, J. (2014). Big data analytic and mining with machine learning algorithm. Int J Inform Comput Technol, 4(1), 33-40.

Singh, S., & Chana, I. (2012). Enabling reusability in agile software development. arXiv preprint arXiv:1210.2506.

Soares, S. (2012). Big Data Governance - An Emerging Imperative. MC Press Online, LLC, 10-12,143-209.

Stefan Richter, JorgeArnulfo, Quian´eRuiz, Stefan Schuh, & Dittrich, J. (2012). Towards ZeroOverhead Adaptive Indexing in Hadoop. arXiv preprint arXiv:1212.3480.

Subramaniam, M., & Dalal, V. (2015). Test Model for Rich Semantic Graph Representation for Hindi Text using Abstractive Method. International Research Journal of Engineering and Technology(IRJET). ISSN: 2395-0072, Volume: 02 Issue: 02.

Sun, R., Wang, Z., Ren, Y., & Ji, D. (2016). Query-Biased Multi-document Abstractive Summarization via Submodular Maximization Using Event Guidance. Paper presented at the International Conference on Web-Age Information Management. (pp. 310-322). Springer International Publishing.

Tam, N. T., & Song, I. (2016). Big Data Visualization Information Science and Applications (ICISA) 2016 (pp. 399-408): Springer.

Tang, J., Yao, L., & Chen, D. (2009). Multi-topic Based Query-Oriented Summarization. Paper presented at the SDM. (pp. 1148-1159). Society for Industrial and Applied Mathematics.

Uthus, D. C., & Aha, D. W. (2011). Plans toward automated chat summarization. Paper presented at the Proceedings of the Workshop on Automatic Summarization for

University of Malaya

Different Genres, Media, and Languages. (pp. 1-7). Association for Computational Linguistics.

Vogt, J. E., Kloft, M., Stark, S., Raman, S. S., Prabhakaran, S., Roth, V., & Rätsch, G. (2015). Probabilistic clustering of time-evolving distance data. Machine Learning, 100(2-3), 635-654.

W3C. (2013, 2013-06-19). W3C Semantic Web Activity. Retrieved from https://www.w3.org/2001/sw/

Wang, Holub, Murphy, & O'Sullivan. (2013). High volumes of event stream indexing and efficient multi-keyword searching for cloud monitoring. Future Gener Comput Syst, 29(8), 1943–1962.

Wang, Wu, Li, & Ooi. (2010). Indexing multi-dimensional data in a cloud system. ACM, 591-602. doi:ACM SIGMOD international conference on management of data. (pp. 591-602). ACM.

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

Wang, L., & Cardie, C. (2011). Summarizing decisions in spoken meetings. Paper presented at the Proceedings of the Workshop on Automatic Summarization for Different Genres, Media, and Languages. (pp. 16-24). Association for Computational Linguistics.

Wang, X., Luo, X., & Liu, H. (2015). Measuring the veracity of web event via uncertainty. Journal of Systems and Software, 102, 226-236.

Wayman, J., Jain, A., Maltoni, D., & Maio, D. (2005). An introduction to biometric authentication systems: Springer. Biometric Systems, 1-20.

Wei L-Y, HsuY-T, PengW-C, & LeeW-C. (2013). Indexing spatial data in cloud data managements. . Pervasive Mobile Comput, 1–14. doi:10.1016/j.pmcj.2013.07.001

Wei, L.-Y., Hsu, Y.-T., Peng, W.-C., & Lee, W.-C. (2014). Indexing spatial data in cloud data managements. Pervasive and Mobile Computing, 15, 48-61.

Wei, L., & Shen, J. (2016). Method and system for document indexing and data querying: Google Patents. U.S. Patent No. 9,275,128. Washington, DC: U.S. Patent and Trademark Office

Weng, & Chuang. (2011). Collaborative Video Re-indexing via Matrix Factorization. ACM Trans. Multimedia Comput.Communications. (TOMM), 8(2), 23.

University of Malaya

Weng, M.-F., & Chuang, Y.-Y. (2012). Collaborative video reindexing via matrix factorization. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 8(2), 23.

Whitaker, S. (2014). Big Data versus a Survey. Whitaker, Stephan, Big Data versus a Survey (December 29, 2014). FRB of Cleveland Working Paper No. 14-40. Available at SSRN: https://ssrn.com/abstract=2543571.

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), 3717-3725.

Wu, K., Shoshani, A., & Stockinger, K. (2010). Analyses of multi-level and multi-component compressed bitmap indexes. ACM Transactions on Database Systems (TODS), 35(1), 2.

Wu, Y., Chen, Z., Wen, Y., Cao, J., Zheng, W., & Ma, G. (2015). A general analytical model for spatial and temporal performance of bitmap index compression algorithms in Big Data. Paper presented at the 2015 24th International Conference on Computer Communication and Networks (ICCCN). Las Vegas, NV, 2015, pp. 1-10. doi: 10.1109/ICCCN.2015.7288362

Yang, C., Zhang, X., Zhong, C., Liu, C., Pei, J., Ramamohanarao, K., & Chen, J. (2014). A spatiotemporal compression based approach for efficient big data processing on cloud. Journal of Computer and System Sciences, 80(8), 1563-1583.

Yeh, S.-C., Su, M.-Y., Chen, H.-H., & Lin, C.-Y. (2013). An efficient and secure approach for a cloud collaborative editing. Journal of Network and Computer Applications, 36(6), 1632-1641.

Yi, K., Wang, L., & Wei, Z. (2014). Indexing for summary queries: Theory and practice. ACM Transactions on Database Systems (TODS), 39(1), 2.

Yıldırım, H., Chaoji, V., & Zaki, M. J. (2012). GRAIL: a scalable index for reachability queries in very large graphs. The VLDB Journal—The International Journal on Very Large Data Bases, 21(4), 509-534.

Yu, C., & Boyd, J. (2014, April). FB+-tree: Indexing based on key ranges. In Networking, Sensing and Control (ICNSC), 2014 IEEE 11th International Conference on (pp. 438-444). IEEE.

Zhang, Y., Sow, D., Turaga, D., & van der Schaar, M. (2014). A fast online learning algorithm for distributed mining of bigdata. ACM SIGMETRICS Performance Evaluation Review, 41(4), 90-93.

Zhang, Y., Yu, R., Xie, S., Yao, W., Xiao, Y., & Guizani, M. (2011). Home M2M networks: architectures, standards, and QoS improvement. IEEE Communications Magazine, 49(4), 44-52.

University of Malaya

Zhu, Huang, Cheng, Cui, & Shen. (2013). Sparse hashing for fast multimedia search. ACM Trans Inf Syst, 31(2), 1-24. doi:10.1145/2457465.2457469

Zhu, J., Khullar, S., & Chi, W. (2016). Item Loss Prevention Backpack, Available at https://pdfs.semanticscholar.org/f1fb/78ca4003bda2e38a2de7c4d416102f04fb96.pdf.

Zhu, X., Huang, Z., Cheng, H., Cui, J., & Shen, H. T. (2013). Sparse hashing for fast multimedia search. ACM Transactions on Information Systems (TOIS), 31(2), 9.

University of Malaya

APPENDIX A.
MORE RESULTS AND FIGURES OF EXECUTION TIME FOR SEARCHING QUERIES

Table .1:Summarized report of the workloads for the dataset with size 34.713GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method
4.99	0.063	425019	2656	711366	6257	804648	12014
3.35	0.007	425346	2221	710220	5847	805107	7264
4.94	0.006	425084	2412	711555	5883	804731	5223
3.42	0.004	485415	2329	710903	6222	804424	5172
3.39	0.003	512739	2195	719516	5622	805472	5772
2.06	0.004	512842	2385	711435	6107	805426	5172
1.87	0.004	512976	2357	711310	5809	804925	5125
2.97	0.003	512816	2411	712049	6073	804392	5373
4.19	0.003	512683	2141	711525	5470	805097	5172
2.75	0.003	512846	2546	711362	5891	804695	5176

Table .2:Summarized report of the workloads for the dataset with size 136.145GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method
7.47	0.069	8344631	5746	15670942	57233	22672589	128918
5.01	0.008	8351056	4807	15645700	53485	22685529	77952
7.39	0.007	8345899	5220	15675101	53815	22674941	56051
5.12	0.005	9530417	5040	15660732	56913	22666265	55504
5.07	0.003	10066883	4750	15850470	51420	22695823	61937
3.09	0.005	10068912	5160	15672454	55859	22694499	55504
2.80	0.005	10071532	5100	15669713	53134	22680382	54991
4.44	0.003	10068404	5216	15685973	55551	22665383	57660
6.27	0.003	10065784	4634	15674440	50036	22685235	55504
4.11	0.003	10068996	5509	15670847	53881	804695	5176

University of Malaya

Table .3:Summarized report of the workloads for the dataset with size 272.291GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method
8.81	0.073	38050756	7159	80443579	217908	122412035	433809
5.91	0.008	38080053	5989	80314006	203641	122481902	262308
8.72	0.007	38056539	6504	80464932	204896	122424738	188613
6.05	0.005	43457833	6279	80391168	216693	122377895	186771
5.98	0.004	45904066	5919	81365150	195776	122537478	208418
3.64	0.005	45913318	6429	80451344	212677	122530332	186771
3.30	0.005	45925267	6354	80437271	202302	122454114	185043
5.24	0.004	45911005	6499	80520741	211505	122373131	194025
7.40	0.004	45899055	5773	80461535	190505	122480314	186771
4.86	0.004	45913703	6864	80443094	205147	122419180	186886

Table .4:Summarized report of the workloads for the dataset with size 544.582GB (Milliseconds)

Query 1 (MS)		Query 2 (MS)		Query 3 (MS)		Query 4 (MS)	
B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method	B-tree Method	Proposed Method
10.22	0.076	176857676	7985	438346868	967305	829935379	2235308
9.86	0.009	176993843	6680	437640808	903974	830409068	1351607
10.11	0.008	176884551	7254	438463222	909545	830021505	971876
9.01	0.005	201989451	7003	438061271	961912	829703918	962382
10.93	0.004	213359397	6601	443368620	869063	830785867	1073929
9.22	0.005	213402397	7171	438389179	944085	830737421	962382
9.83	0.005	213457939	7087	438312490	898032	830220669	953482
10.07	0.004	213391647	7249	438767331	938885	829671621	999762
9.58	0.004	213336106	6439	438444711	845665	830398303	962382
10.63	0.004	213404189	7656	438344223	910659	829983825	962976

University of Malaya

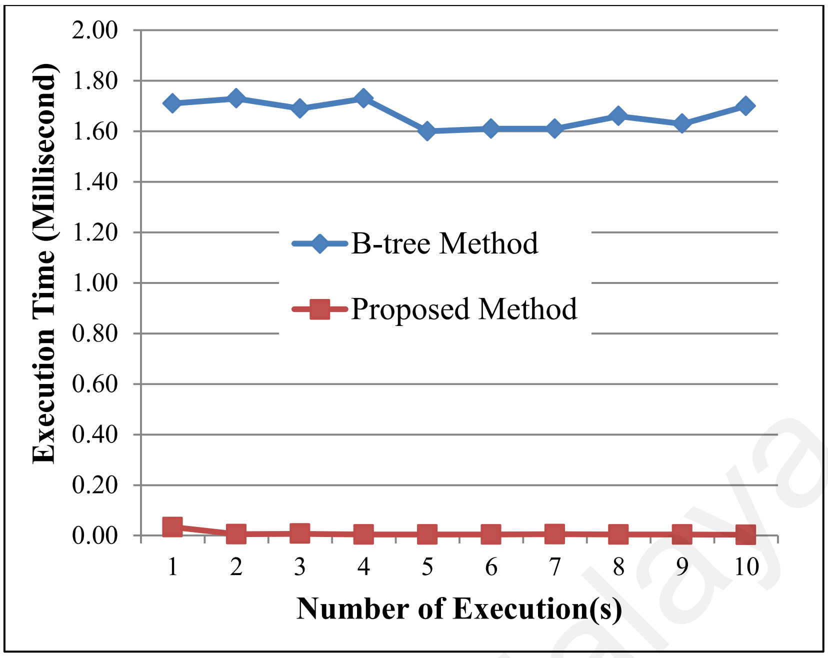

Figure .1: Visualization of execution time for searching Query 1 when the size of data is 4.255GB

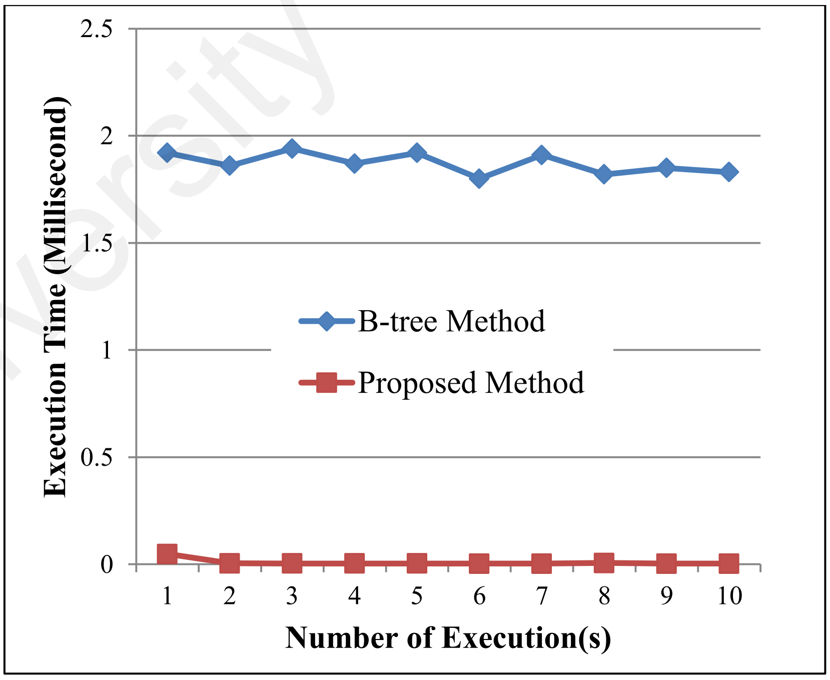

Figure .2: Visualization of execution time for searching Query 1 when the size of data is 8.509GB

University of Malaya

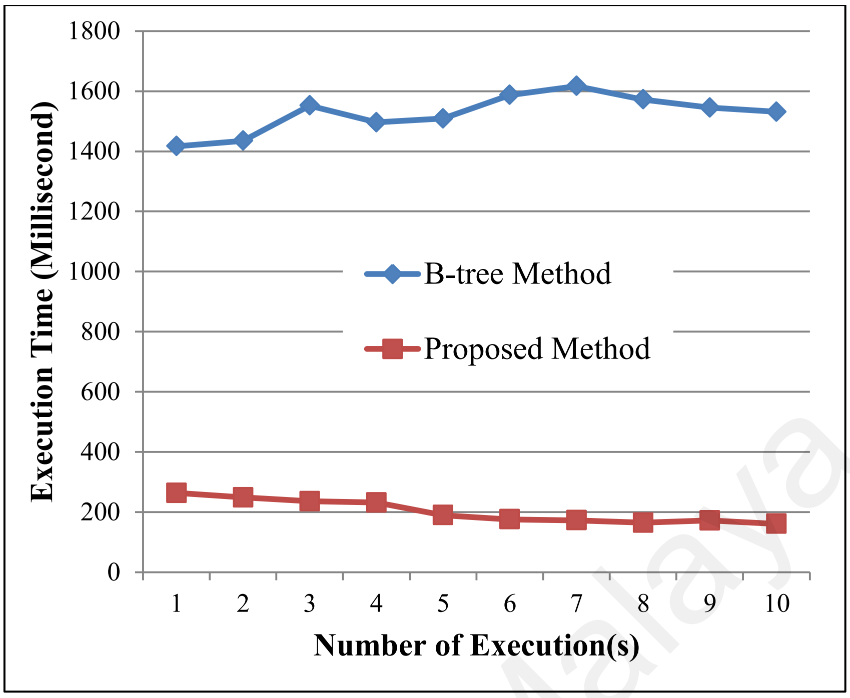

Figure .3: Execution time for Query 2 with data size 2.127GB

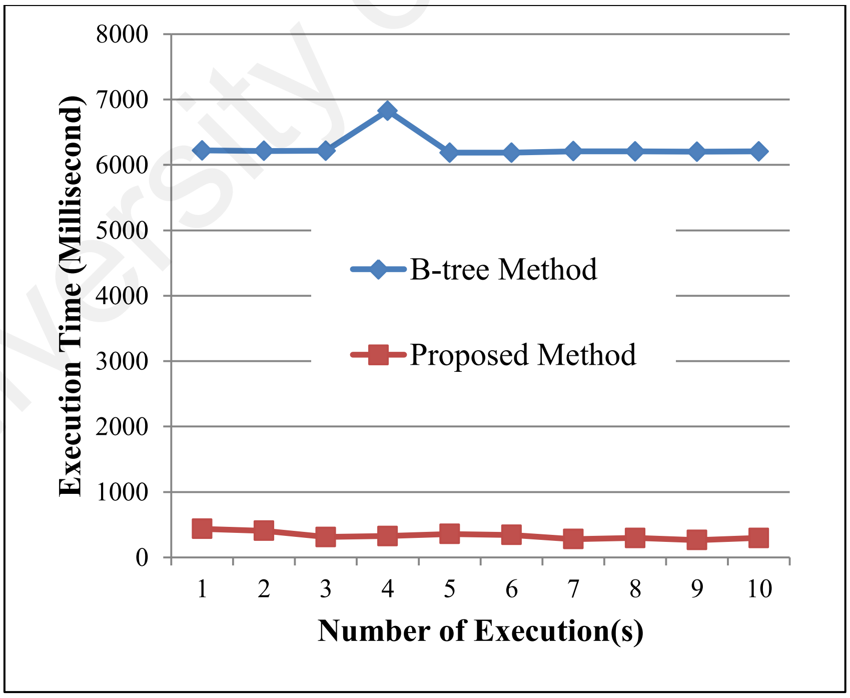

Figure .4: Execution time for Query 2 with data size 4.255GB

University of Malaya

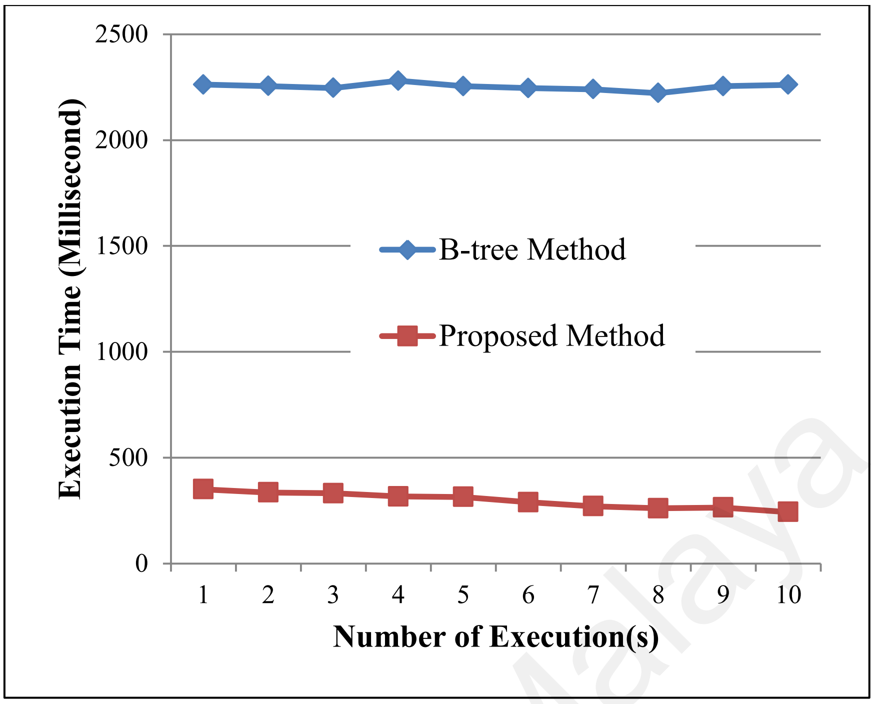

Figure .5: Execution time for Query 3 with data size 2.127GB

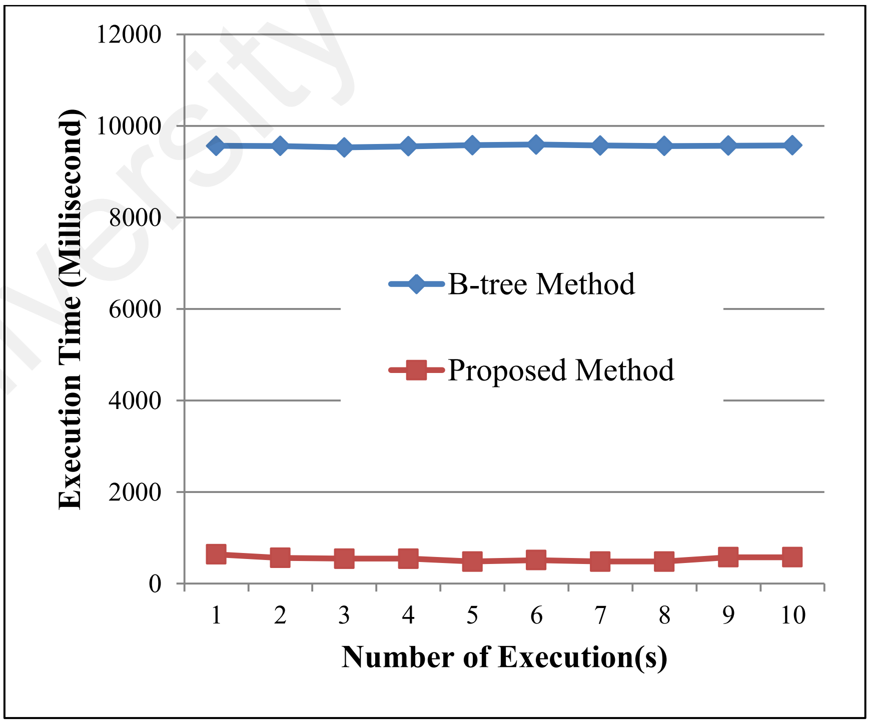

Figure .6: Execution time for Query 3 with data size 4.255GB

University of Malaya

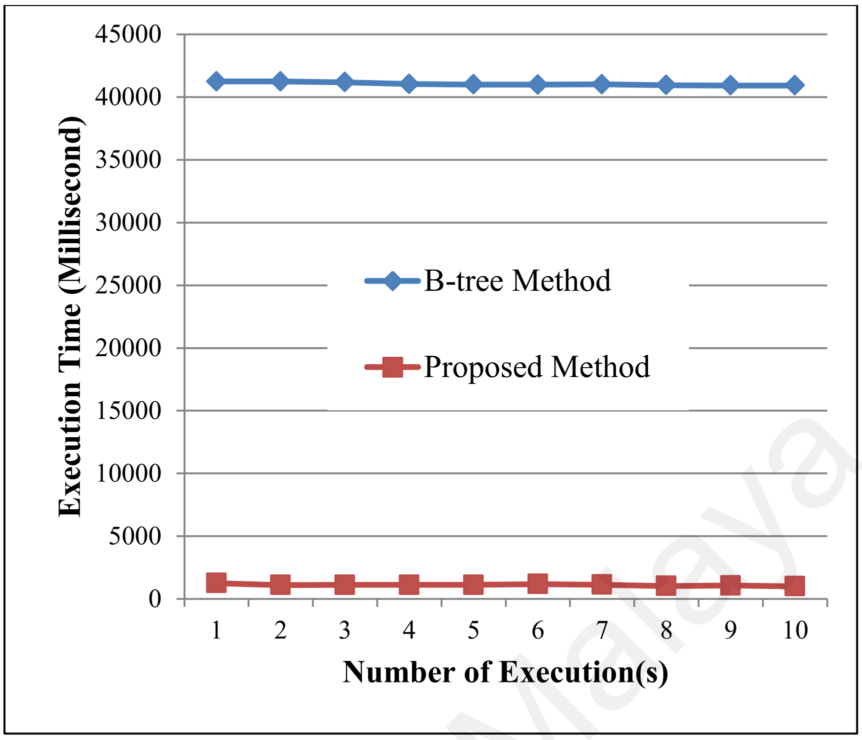

Figure .7: Execution time for Query 3 with data size 8.509GB

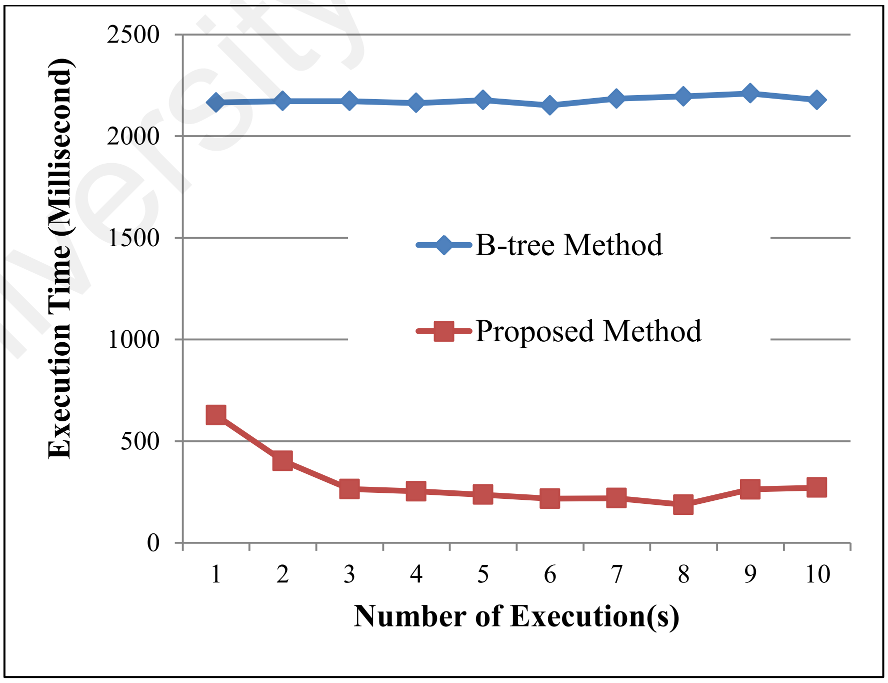

Figure .8: Execution time for Query 4 with data size 2.127GB

University of Malaya

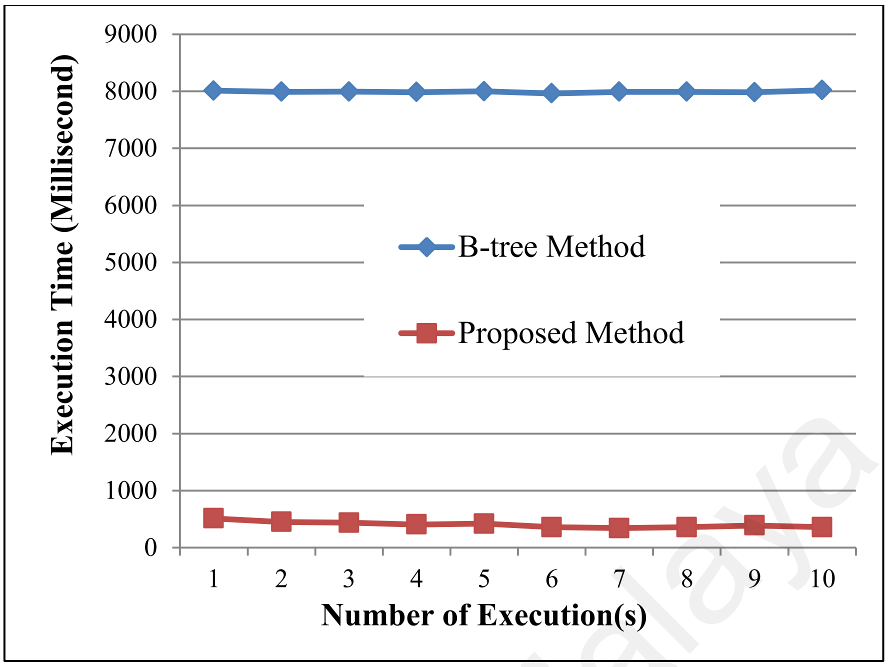

Figure .9: Execution time for Query 4 with data size 4.255GB

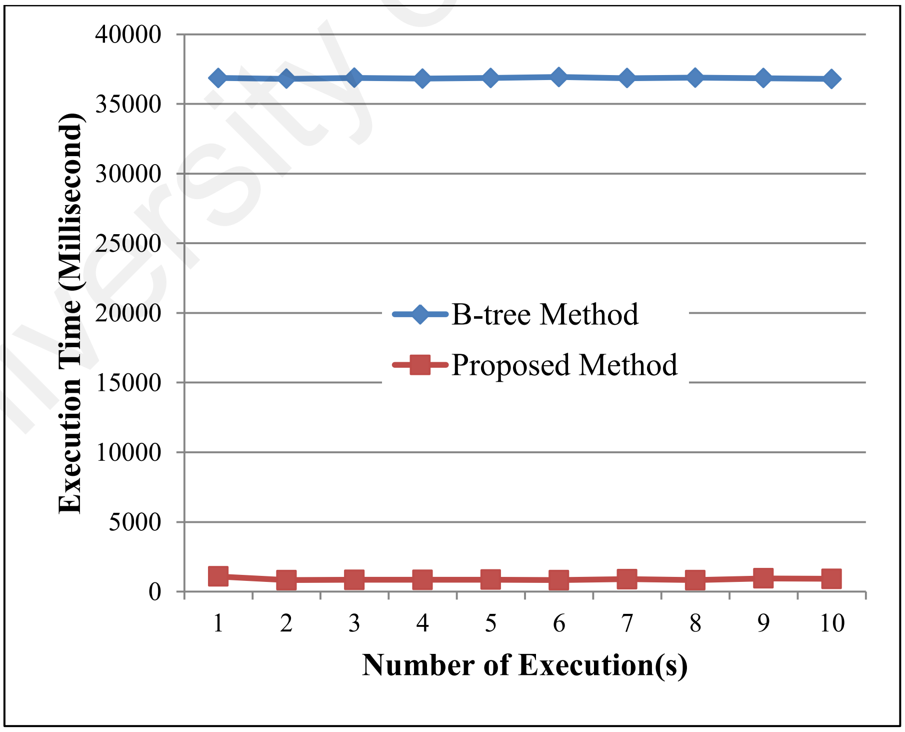

Figure .10: Execution time for Query 4 with data size 8.509GB

University of Malaya

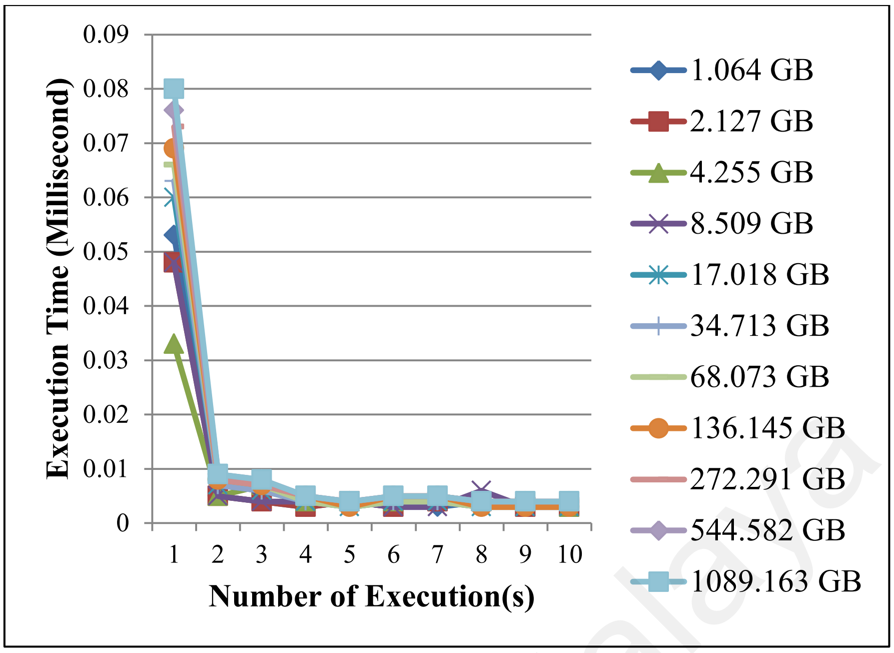

Figure .11: Execution time for processing Query 1 using more datasets

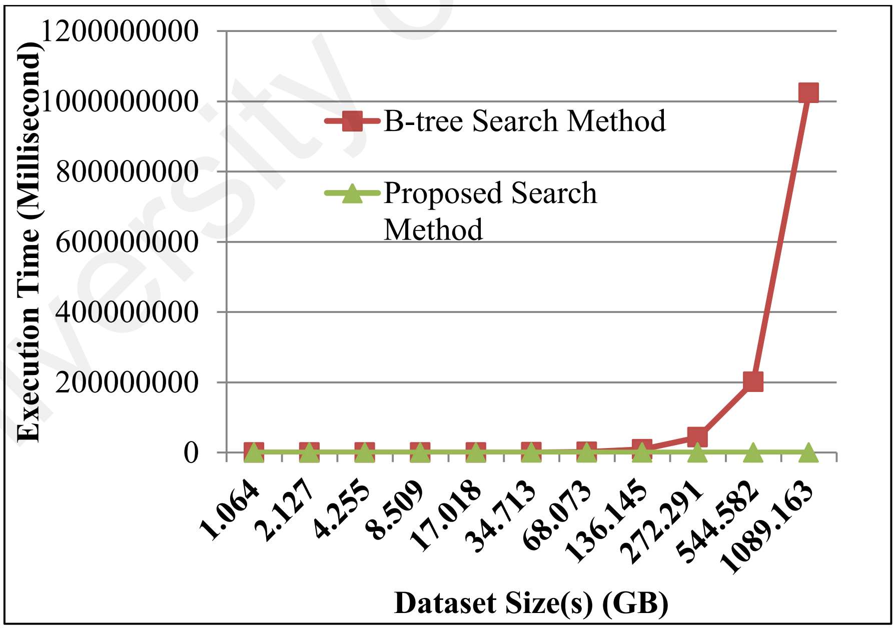

Figure .12: Execution time for processing Query 2 using more datasets

University of Malaya

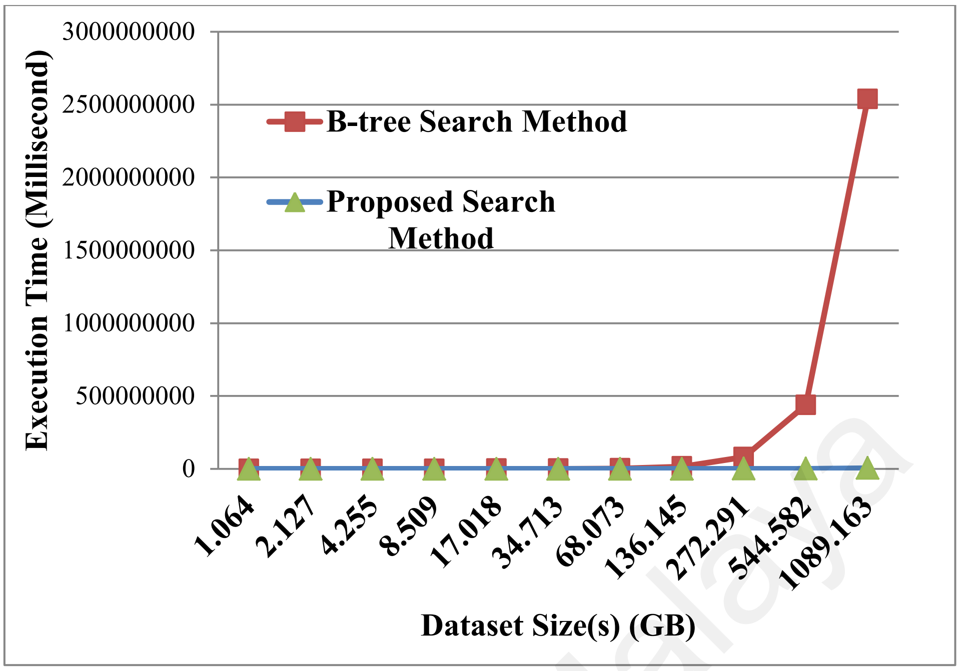

Figure .13: Execution time for processing Query 3 using more datasets

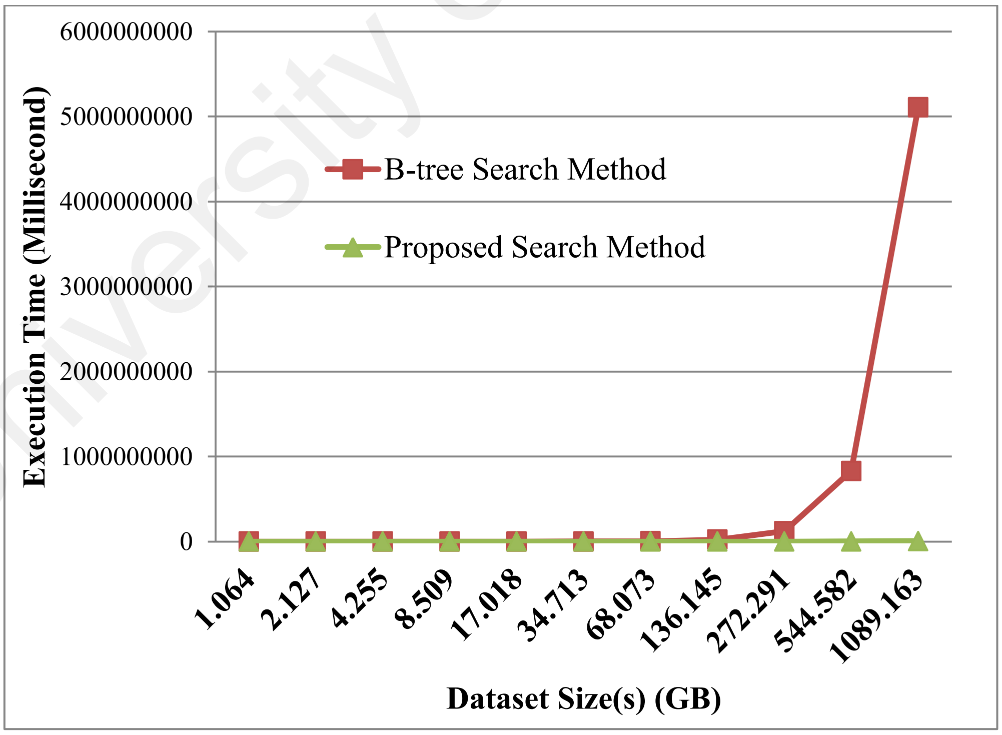

Figure .14: Execution time for processing Query 4 using more datasets

University of Malaya