SLAC-PUB-16460

A Survey On Big Data Indexing Strategies

Fatima Binta Adamu[1], Adib Habbal[1], Suhaidi Hassan[1], R. Les Cottrell[2], Bebo White[2], Ibrahim Abdullahi[1]

InterNetworks Research Laboratory, School of Computing, Universiti Utara Malaysia, 06010 UUM, Sintok, Kedah, Malaysia.[1]

SLAC National Accelerator Lab[2]

adamufatimabinta@gmail.com, adib@uum.edu.my, suhaidi@uum.edu.my, {cottrell,bebo}@slac.stanford.edu, ibrahim@internetworks.my

***Abstract*—The operations of the Internet have led to a significant growth and accumulation of data known as Big Data. Individuals and organizations that utilize this data, had no idea, nor were they prepared for this data explosion. Hence, the available solutions cannot meet the needs of the growing heterogeneous data in terms of processing. This results in inefficient information retrieval or search query results. The design of indexing strategies that can support this need is required. A survey on various indexing strategies and how they are utilized for solving Big Data management issues can serve as a guide for choosing the strategy best suited for a problem, and can also serve as a base for the design of more efficient indexing strategies. The aim of the study is to explore the characteristics of the indexing strategies used in Big Data manageability by covering some of the weaknesses and strengths of B-tree, R-tree, to name but a few. This paper covers some popular indexing strategies used for Big Data management. It exposes the potentials of each by carefully exploring their properties in ways that are related to problem solving.**

***Index Terms*—Big Data; Indexing; Query; Information Retrieval**

I. INTRODUCTION

Big Data is a term used to describe very large data sets that are of different forms or structure (complex), generated at a very high speed, and cannot be managed by traditional database management systems [1]. This definition explains the three (3) main characteristics associated with Big Data: volume, variety and velocity (3Vs), and the value that can be extracted from it is seen as a fourth characteristic (4V's) [2]. Big Data is sourced from so many end devices such as Personal Computers (PC), smart phones, Global Positioning System (GPS) devices[3], sensors, and Radio Frequency Identification (RFID) devices, monitoring devices, etc. Also, online applications such as social networks and applications that involve video streaming are great sources that generate Big Data.

According to Zhou et al. in [4], the total size of data generated will surpass 7.9 Zettabytes (ZB) by the end of 2015, and predicted to reach 35ZB in 2020. Significant interest have been taken in Big Data lately – this is due to insights or great value that can be acquired from huge amounts of data sets, which can be useful in decision making for business or organization. Cisco related that organizations such as Facebook, Yahoo, Google, Twitter, etc., accumulate Big Data and retrieve information for analysis and decision making [5]. The Internet of Things (IoT), monitoring tools, and lots of other devices used by organizations for business operations, also accumulates data to enable analysis, information retrieval, and decision making. These operations are becoming difficult to perform because data keeps increasing in volume as time passes by [1]. Current Relational Database Management systems (RDBMS) were built with a scale in mind and for structured data. Hence, they cannot handle processing and information retrieval on very large amount of unstructured data (Big Data). Yet, International Data Corporation (IDC) predicts that the global Big Data will multiply 50 times in the next decade[6], which suggests the continues growth and accumulation of unstructured data. This should be met with efficient processing strategies capable of handling such huge amounts of unstructured data.

Numerous indexing strategies have been proposed [7], [8], [9], [10], [11], [12], [13] as a solution to the problem. First, indexing strategies can be said to be a non-polynomial process as each relates to the problem it solves. Different Indexing approaches are applied in different domains and on different data types. Hence, a survey on various indexing strategies and how they are utilized for solving Big Data management issues can serve as a guide for choosing the strategy best suited for a problem, and can also serve as a base for the design of more efficient indexing strategies. The aim of the study is to explore the characteristics of the indexing strategies used in Big Data manageability by covering some of the weaknesses and strengths of B-tree, R-tree, to name but a few. The survey highlights some popular indexing strategies used for Big Data management. It exposes the potentials of each by carefully exploring their properties in ways that are related to problem solving. The paper is structured as follows: Section II explains Big Data indexing and it's requirements. It also outlines the basic categories of indexing strategies. Section III elaborates on Artificial Intelligence (AI) indexing approach, and Section IV on Non-Artificial Intelligence (NAI) indexing approach; while Section V concludes the paper.

II. BIG DATA INDEXING

The growth of data and accumulation of complex data collections has become a challenge for information retrieval [14]. A solution to this is in building indexes on data sets. In general, indexes or indices are a list of tags, names, subjects, etc. of a group of items which references where the items occur. With this, Big Data indexes can be said to be a list of tags, names, subjects, etc. of a dataset which references where data can be found. An indexing strategy is the design of an access method to a searched item, or simply put, an index. It also describes how data is organized in a storage system to

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-76SF00515.

facilitate information retrieval. The idea of Big Data indexing is to fragment the datasets according to criteria that will be used frequently in query[14]. The fragments are indexed with each containing value satisfying some query predicates. This is aimed at storing the data in a more organized manner, thereby easing information retrieval.

Complex data are collected with metadata that describes their contents. Such datasets can be queried using the metadata of the contents. Instead of searching the whole database (which can be time consuming), a more efficient approach is to search the appropriate group(s) relating to the query. This results in a decrease in information retrieval time, since the search process considers only the content of a specific group(s). To facilitate information retrieval, a suitable indexing strategy has to be applied to the datasets during processing. This also comes with the advantage of having an organized storage system to ease search and information retrieval. Big Data indexing depends on a solution that utilizes a massively parallel computer or machine that interconnects lots of RAM, CPUs, and disk units. The benefits of this are high throughput for data processing, decreased access time for queries, data replication that results in increased availability and reliability, and scalability of the structure. The design of an access method or the type of indexing strategy to be used in processing a specific dataset depends on the type of queries that will be performed on the dataset, such as similarity queries (nearest neighbor search), range queries, point query, keyword queries, and ad-hoc query. Therefore, the designer must be aware of the type of data to be indexed (e.g. logs, email, audio, video, images, etc.) and the type of query that will be performed on the indexes. According to [15], indexing strategies can be categorized into Artificial Intelligence (AI) approach, and Non-Artificial Intelligence (NAI) approach.

III. ARTIFICIAL INTELLIGENCE APPROACH

Artificial Intelligence (AI) indexing approaches are so called because of their ability to detect unknown behavior in Big Data. They establish relationships between data items by observing patterns and categorizing items or objects with similar traits. Although this gives AI indexing approaches an edge over NAI, the former generally takes more time in information retrieval and are sometimes considered inefficient as compared to NAI indexing approaches [15]. Latent Semantic Indexing (LSI) [8], [16], [7] and Hidden Markov Model (HMM) [17], [18], are two popular AI indexing approaches.

A. Latent Semantic Indexing

Latent Semantic Indexing, LSI for short, is an indexing strategy (retrieval/access method) that identifies patterns between the terms in an unstructured data set (specifically, text). It uses a mathematical approach known as Singular Value Decomposition (SVD) for the pattern or relationship identification. Hence, LSI is not subjected to any language. The main characteristic of LSI is the ability to elicit the conceptual (semantic) content of data sets and to establish relationships between terms with similar contexts as illustrated in Figure 1. In Figure 1, the audiences discuss the program "House of cards" on social media and forums. LSI is used here, to categorize or index comments made by the audience into audience favors, audience expectations, and the shortcomings of the season (by extracting the meaning of each comment). This makes it easier for the director to make decisions towards the improvement of the next season, and so on. LSI makes use of Resource Description Framework (RDF), which is a standard for web resource description. The RDF can describe author, title, date, time, price, definition, and a lot more information of a web page. RDF also uses tags, by adding information about parts of speech such as noun, verb, adjective, etc. to the context of each word.

Along with establishing meaningful relationships between texts, LSI overcomes the problem that comes with keyword queries (synonyms and polysemy) [8], mostly encountered while working with inverted indexes (see Section IV, subsection D). These problems often result in mismatches during information retrieval and can be bad for decision making. In the LSI strategy, text or documents are assigned to categories according to their contextual similarities. During categorization, the contexts of the set of text to be categorized are compared to the contexts of example documents, and categories are assigned based on matching documents. Also, documents can be grouped together based on their contextual similarities, without comparing with example documents. The challenges mostly faced while working with LSI is scalability and performance [7]. LSI strategy demands very high computational performance as well as memory to index Big Data. LSI supports keyword queries on textual data which can be in the form of web contents (images, audio, etc.), documents, emails, or any item that can be converted into text.

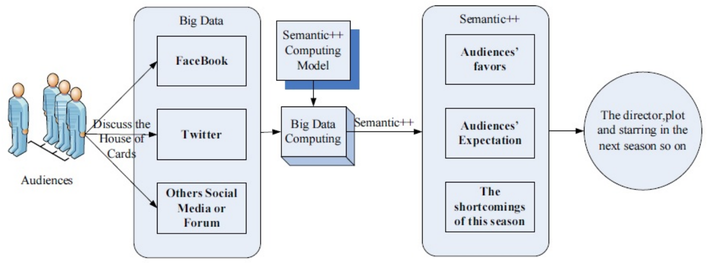

Audiences
Discuss the House of Cards
Big Data
FaceBook
Twitter
Others Social Media or Forum
Semantic++ Computing Model
Big Data Computing
Semantic++
Semantic++
Audiences' favors
Audiences' Expectation
The shortcomings of this season
The director,plot and starring in the next season so on

Figure 1. Latent semantic indexing [16]

B. Hidden Markov Model

The Hidden Markov Model (HMM) indexing approach is an access method developed from the Markov model. A Markov model is made up of states which are connected by transitions, where future states are solely dependent on the present state and independent of historical states. Similar to the LSI strategy, the HMM uses pattern recognition and relationship between data. In the HMM indexing approach, data or characteristics which the states depend on during query, are categorized and stored in advance. The query results are usually predictions of future states of an item, based on the current or present state. The present state is used to predict the future states using the dependent data or characteristics of the states. For example, in the study by Matsui et al.

[17], HMM was used to classify and store motion data used by robots. The classification was based on the acceleration information, consisting of the position information and the pure force. Hence, the prediction of the next series (sequence) of motions was dependent on the position informant and the pure force. The motion data is stored and classified in advance, before the quick search for motion is conducted. Also the study by Widodo et al. [18] used HMM and LSI to classify or index documents. In their work, words are expanded in every document, and stored in advance. Expanding the documents prior to indexing gives more room for prediction and a quicker search on items.

IV. NON – ARTIFICIAL INTELLIGENCE APPROACH

In NAI indexing approach, the formation of indexes does not depend on the meaning of the data item or the relationship between texts. Rather, indexes are formed based on items most queried or searched for in a particular data set. The tree-based indexing strategy (B-tree [9], [10], [19], R-tree [12], [20], and X-tree [21]), inverted indexing approach [22], [23], hash [24], and custom (GiST [25]and GIN [26]) indexing, are NAI indexing approaches covered in this paper.

A. The Tree-based indexing strategies

The Tree indexing structures are the B-tree, R-tree, X-tree, etc.[24]. In the Tree indexing strategy, retrieval of data is done in a sorted order, following branch relations of the data item. This satisfies nearest neighbor queries. According to researches, the Tree indexing strategies are being displaced by other indexing strategies because they are generally outperformed (in terms of speed of information retrieval) by simple sequential scans [24]. The Tree-based indexing strategies are explained as follows:

1) The B-tree: A B-tree works like the Binary tree search, but in a more complex manner. This is because the nodes of B-tree have many branches, unlike the binary tree which has two branches per node. So a B-tree is more complicated than a binary tree [27]. B-tree indexes satisfy range queries and similarity queries also known as Nearest Neighbor Search (NNS), using comparison - operators (<, < =, =, >, > =). In the B-tree, the keys and all records are normally stored in leaves, but copies (of the key) are stored in internal nodes as illustrated in Figure 2. Also, the leaves might include pointers to the next node, showing the path to the searched item.

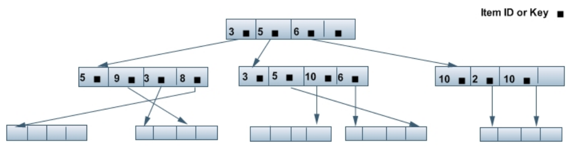


Figure 2. B-tree Indexing

Researches have shown that this strategy is not always fast when searching Big Data and can waste storage spaces since the nodes are not always full [15], [9]. A B-tree scales linearly, but is only suitable for one dimensional access method unlike other tree-based access methods or indexing strategies such as the R-tree. Also, the B-tree algorithm consumes huge computing resources when performing indexing on Big Data [9]. Other variations of the B-tree are the B+tree [10], B*tree, KDB-tree [19], and so on.

2) The R-tree: This is an indexing strategy used for spatial or range queries. It is mostly applied in geospatial systems with each entry having X and Y coordinates with minimum and maximum values [12], [28]. The advantage of using an R-tree over a B-tree is that, the R-tree satisfies multi-dimensional or range queries, whereas the B-tree does not. Given a query range, using the R-tree makes finding answers to queries quick[29]. An example is finding all the hostels within a given campus, or finding all hotels within a given kilometer from a certain location. The idea is to group data items according to their distance from each other, and assign minimum and maximum bounds to them. Each record at the leaf node, describes a single item (with minimum and maximum values). Each internal node describes a collection of items or objects as illustrated in Figure 3 and Figure 4.

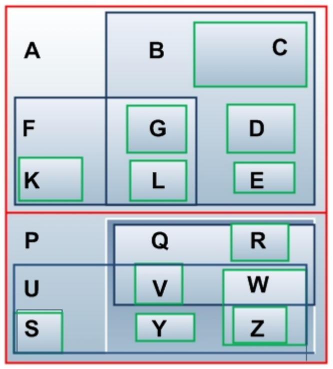


Figure 3. Range Grouping in R-tree

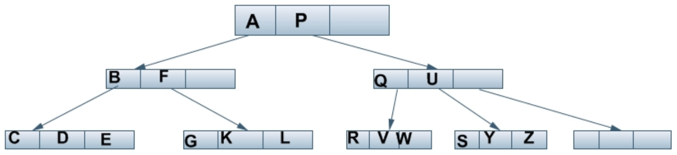


Figure 4. R-tree Indexing

Though the R-tree is preferred over the B-tree in the case of indexing spatial data, the R-tree does not find the exact answer as query results. It merely limits the search space. Also, it consumes memory space because coordinates are stored along with the data [30]. Variants of the R-tree are R*tree, R+-tree [20], etc.

3) The X-tree: This type of indexing strategy, based on the R-tree, satisfies range queries. The X-tree is similar to the R-tree and operates just like the R-tree. Although, unlike the R-tree which satisfies 2-3 dimensional range queries, the X-tree satisfies queries of many dimensions [24], [21]. This implies that the X-tree is a more complicated version of the R-tree. The advantage of the X-tree over the R-tree is that it covers more dimensions, otherwise, the X-tree also consumes memory space due to storage of coordinates.

B. Hash Indexing Strategy

Hash allows for equality comparison. Hash indexing accelerates information retrieval by detecting duplicates in a large dataset [31]. An example of an hash indexing strategy implementation is explained in the study by Giangreco et al (2014) [24]. Giangreco et al designed a strategy that takes hand sketched diagrams and retrieves similar images from a large collection of images, based on hash indexing strategy. Hash indexing strategy is used in password checking systems, DNA sequence match, etc. Hash is used in Big Data indexing to index and retrieve data items (in a dataset) that are similar to the searched item. It uses a hashed key (which is computed by the hash function and usually shorter than the original value) to store and retrieve indexes. For this reason, hash indexing is more efficient than the tree-based indexing in terms of equality or point query [24]. Simply, search on shorter hashed keys can be faster than search on unpredictable length key (found in tree-based indexes). Though, hashing technique works fine with limited data size, it tends to exhibit indexing computational overhead as data size increases [15].

C. Custom Indexing Strategy

Custom indexing supports multiple field indexing based on arbitrary or user defined indices [32]. They are usually based on indexing strategies such as B-tree, R-tree, inverted index, and hash indexing strategy. Two types of custom indexing strategies are Generalized Search Tree (GiST) [25] and Generalized Inverted Index (GIN) [26].

1) GiST: The Generalized Search Tree or GiST indexing strategy, is an indexing strategy based on the B-tree or the R-tree [25]. It allows for the creation of custom or arbitrary fields as indexes. The GiST has the same implementation (for indexing and retrieval) as the R-tree for those based on the R-tree, and as the B-tree for those based on the B-tree. Hence, they support indexing and query on one-dimensional data, as well as multi-dimensional or spatial data. Just like every other tree-based indexing strategy, GiST has root nodes, leaf nodes, pointers, and other characteristics of a balanced tree structure. Despite these similarities between the GiST and the tree-based strategies, the former has an advantage of supporting ad-hoc queries over the latter. Taking GiST based on R-tree for example, each node contains a key-pointer pair, where the key is the searched key, and the pointer points or refers to the corresponding node (or data item, in the case of a leaf node) as illustrated in Figure 5. Also, each node contains minimum and maximum values, with the exception of the root node. The GiST performs well in terms of query search, but is considered generally slower than the GIN [25].

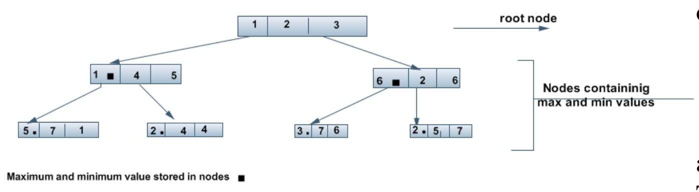

root node
Nodes containinig max and min values
Maximum and minimum value stored in nodes ■

Figure 5. GiST Indexing

2) GIN: Just as in the GiST, the Generalized Inverted Index or GIN indexing strategy (or access method) uses custom or arbitrary fields as indexes [26]. It is designed for specific user requirements. Though the GIN is implemented like the B-tree and has properties of the inverted index, GIN differs from the B-tree which has comparison-based operations that are predefined. GIN is made up of a B-tree index which comprises of Entries Tree or list (ET) and Posting Tree (PT) or Posting List (PL) [26]. In the ET, each entry represents an element of the searched key or indexed value, for example, arrays. The PL is a pointer to a list of items, or a pointer to a B-tree (for leaf nodes), in which case it is called a PT, as illustrated in Figure 6. While the B-tree is good for single-match indexes or range queries, the GIN works best for indexes having many duplicates. This is because the GIN queries data only by point or equality matching. This is often viewed as a limitation.

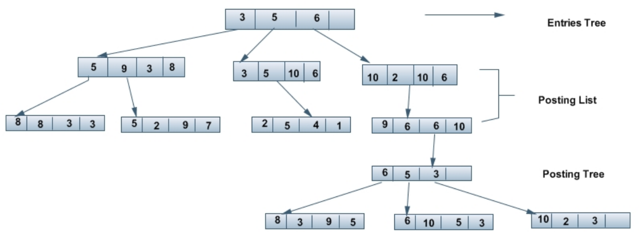


Figure 6. GIN Indexing

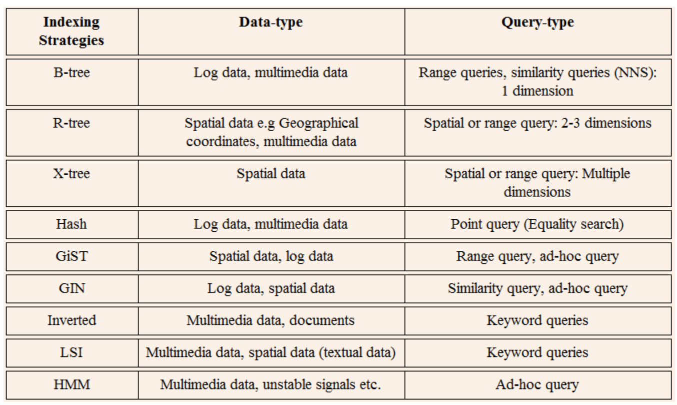
D. Inverted Indexing Strategy

Inverted indexing strategy allows for the design of inverted indices which are used for full - text search like what is obtainable in Google and other search engines [22], [23], [33]. An inverted index is made up of a list of all unique words which appear in documents, and a list of documents in which each word appears. With an inverted index, multiple documents can have the same key as index. Also, multiple keys can be used in indexing a document. For example, a blog post can have multiple tags (as key), and each tag can refer to more than one blog post. Though some inverted indexing strategies use B-tree or can be populated with rows, it is worth noting that not all inverted indexes are based on B-trees. The difference between an inverted index and a B-tree is that B-trees use row-structured data, unlike inverted indexes. Inverted indexing is implemented by storing or indexing a set of key-post list pairs, where key is the searched index, and post list is a collection of documents where the key occurs. The problem with inverted indexes is that, two or more words (keys) might be separate terms, but will appear to the user as the same term. Also, synonyms (of the keys) might not be recognized or retrieved during query search [33].

V. COMPARISON OF INDEXING STRATEGIES

Table I summarizes the various types of indexing strategies, along with the possible type of data and queries they support. Table II outlines the main characteristics of each indexing strategy. The key features and challenges faced in each, are described.

Table I
INDEXING STRATEGIES AND QUERY-TYPES



Indexing Strategies	Data-type	Query-type
B-tree	Log data, multimedia data	Range queries, similarity queries (NNS): 1 dimension
R-tree	Spatial data e.g Geographical coordinates, multimedia data	Spatial or range query: 2-3 dimensions
X-tree	Spatial data	Spatial or range query: Multiple dimensions
Hash	Log data, multimedia data	Point query (Equality search)
GiST	Spatial data, log data	Range query, ad-hoc query
GIN	Log data, spatial data	Similarity query, ad-hoc query
Inverted	Multimedia data, documents	Keyword queries
LSI	Multimedia data, spatial data (textual data)	Keyword queries
HMM	Multimedia data, unstable signals etc.	Ad-hoc query

Table II
CHARACTERISTICS OF INDEXING STRATEGIES

Indexing Strategies	Properties	Challenges
B-tree	- One dimensional access method -Tree structure with nodes and pointers - Scales linearly	- Waste storage space - Not suitable for multidimensional access -Consumes huge computing resources
R-tree	- More scalable than the B-tree - 2 to 3 dimensional access method	- Index consumes more memory space
X-tree	- Multidimensional access method	- Consumes memory space
Hash	- Presents the exact answer (uses '=' operator) - Quick information retrieval	- Computational overhead
GiST	- Arbitrary indexes - Based on the tree structures	- Slower query response
GIN	- Arbitrary indexes - Based on the tree structures	- Longer processing time
Inverted	- Index consumes less space - Full text search (keyword search)	- Longer data processing time - Limits the search space, not necessarily producing the exact answer - Can present wrong answers due to synonyms and polysemy
LSI	- Uses data and meaning of data for indexing - Presents accurate query results (since it uses more information)	- Demands high computational performance - Consumes more memory space
HMM	- Based on the Markov model - Recognizes relationships between data	- Demands high computational performance

The taxonomy of the popular indexing strategies used in Big Data, is as illustrated in Figure 7. The two main categories are AI and NAI. AI utilizes the contextual meaning of data to establish relationships (between the data items) which serves as the bases to index creation. The most popular AI indexing approaches are LSI and HMM. NAI on the other hand, does not detect the unknown behavior of data [15]. The most popular NAI techniques are B-tree, R-tree (and their variants), Hash, Inverted, and custom indexing.

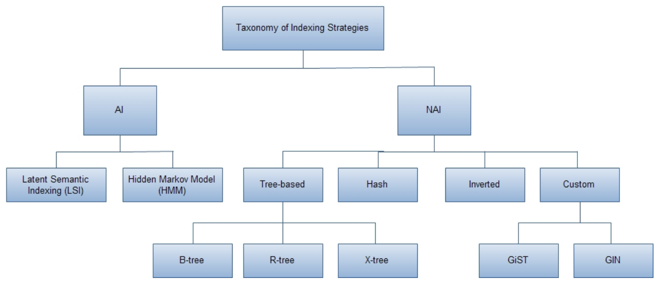

Taxonomy of Indexing Strategies
AI
NAI
Latent Semantic Indexing (LSI)
Hidden Markov Model (HMM)
Tree-based
Hash
Inverted
Custom
B-tree
R-tree
X-tree
GiST
GIN

Figure 7. Taxonomy of indexing strategies

VI. CONCLUSION

This paper puts together popular data indexing approaches for Big Data processing and management. The objective is to review the potentials of the various indexing strategies and how they are utilized for solving Big data management issues. Numerous indexing strategies have been covered which include [7], [8], [9], [10], [11], [12], [13], [34], [35] as a solution to the Big Data indexing problem. The paper concludes as serving as a guide for choosing the approach best suited in solving a specific problem, and can also serve as a base for the design of more efficient indexing strategies.

ACKNOWLEDGEMENT

The authors wish to thank the Ministry of Education, Malaysia for funding this study under the Exploratory Research Grant Scheme (ERGS S/O CODE: 12474).

REFERENCES

[1] J. Li, Z. Xu, Y. Jiang, and R. Zhang, "The overview of big data storage and management. cognitive informatics cognitive computing (icci*cc),," in *IEEE 13th International Conference on, (pp. 510-513*, 2014.

[2] C. Liu, R. Ranjan, X. Zhang, C. Yang, D. Georgakopoulos, and J. Chen, "Public auditing for big data storage in cloud computing -," in *A Survey. Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on, (pp. 1128-1135).*, 2013, Dec.

[3] H. Tan, W. Luo, and L. M. Ni, "Clost: A hadoop-based storage system for big spatio-temporal data analytics.," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (pp. 2139-2143). New York, NY, USA: ACM.*, 2012.

[4] W. Zhou, C. Yuan, R. Gu, and Y. Huang, "Large scale nearest neighbors search based on neighborhood graph," in *Advanced Cloud and Big Data (CBD), 2013 International Conference on*, pp. 181–186, Dec 2013.

[5] H. Nakada, H. Ogawa, and T. Kudoh, "Stream processing with bigdata: Sss-mapreduce," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pp. 618–621, Dec 2012.

[6] T. Chardonnens, "Big data analytics on high velocity streams," Master's thesis, University of Fribourg (Switzerland), June 2013.

[7] F. Amato, A. De Santo, F. Gargiulo, V. Moscato, F. Persia, A. Picariello, and S. Poccia, "Semtree: An index for supporting semantic retrieval of documents," in *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on*, pp. 62–67, April 2015.

[8] Y. Tang and L. Liu, "Multi-keyword privacy-preserving search in personal server networks," *Knowledge and Data Engineering, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[9] V. Alvarez, S. Richter, X. Chen, and J. Dittrich, "A comparison of adaptive radix trees and hash tables," in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pp. 1227–1238, April 2015.

[10] I. Jaluta, “Transaction management in b-tree-indexed database systems,” in *Information Science, Electronics and Electrical Engineering (ISEEE), 2014 International Conference on*, vol. 3, pp. 1968–1975, April 2014.

[11] W. Yang, J. Jhan, D. Chen, K. Lai, and R. Lee, “Quality of service test mechanism and management of broadband access network.,” in *Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific, (pp. 1-4).*, 2014, Sept.

[12] S. Puri and S. K. Prasad, “A parallel algorithm for clipping polygons with improved bounds and a distributed overlay processing system using mpi,” in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pp. 576–585, May 2015.

[13] A. Babenko and V. Lempitsky, “The inverted multi-index,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 37, pp. 1247–1260, June 2015.

[14] K. Fasolin, R. Fileto, M. Krugery, D. Kaster, M. Ferreira, R. Cordeiro, A. Traina, and C. Traina, “Efficient execution of conjunctive complex queries on big multimedia databases,” in *Multimedia (ISM), 2013 IEEE International Symposium on*, pp. 536–543, Dec 2013.

[15] A. Gani, A. Siddiqa, S. Shamshirband, and F. Hanum, “A survey on indexing techniques for big data: taxonomy and performance evaluation,” *Knowledge and Information Systems*, pp. 1–44, 2015.

[16] G. Zhang, J. Wang, W. Huang, C. Li, Y. Zhang, and C. Xing, “A semantic++ mapreduce: A preliminary report,” in *Semantic Computing (ICSC), 2014 IEEE International Conference on*, pp. 330–336, June 2014.

[17] A. Matsui, S. Nishimura, and S. Katsura, “A classification method of motion database using hidden markov model,” in *Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on*, pp. 2232–2237, June 2014.

[18] Widodo and W. Wibowo, “Improving classification performance by extending documents terms,” in *Data and Software Engineering (ICODSE), 2014 International Conference on*, pp. 1–5, Nov 2014.

[19] Y. Yu, Y. Zhu, W. Ng, and J. Samsudin, “An efficient multidimension metadata index and search system for cloud data,” in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pp. 499–504, Dec 2014.

[20] A. Eldawy and M. Mokbel, “Spatialhadoop: A mapreduce framework for spatial data,” in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pp. 1352–1363, April 2015.

[21] H. Xu, N. Yao, W. Hu, H. Pan, and X. Gao, “The design and implementation of image information retrieval,” in *Computer Science Service System (CSSS), 2012 International Conference on*, pp. 1547–1550, Aug 2012.

[22] T. Gollub, M. Volske, M. Hagen, and B. Stein, “Dynamic taxonomy composition via keyqueries,” in *Digital Libraries (JCDL), 2014 IEEE/ACM Joint Conference on*, pp. 39–48, Sept 2014.

[23] R. Grunzke, R. Muller-Pfefferkorn, R. Jakel, J. Hesser, N. Kepper, M. Hausmann, J. Starek, S. Gesing, M. Hardt, V. Hartmann, J. Potthoff, and S. Kindermann, “Device-driven metadata management solutions for scientific big data use cases,” in *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*, pp. 317–321, Feb 2014.

[24] I. Giangreco, I. Al Kabary, and H. Schuldt, “Adam - a database and information retrieval system for big multimedia collections,” in *Big Data (BigData Congress), 2014 IEEE International Congress on*, pp. 406–413, June 2014.

[25] GiST, “Introduction to gist,” March 2015.

[26] PostgreSQL, “Gin indexes,” March 2015.

[27] M. Rouse, “B-tree definition,” 2015 March.

[28] N. Du, J. Zhan, M. Zhao, D. Xiao, and Y. Xie, “Spatio-temporal data index model of moving objects on fixed networks using hbase,” in *Computational Intelligence Communication Technology (CICT), 2015 IEEE International Conference on*, pp. 247–251, Feb 2015.

[29] A. Watve, S. Pramanik, S. Shahid, C. Meiners, and A. Liu, “Topological transformation approaches to database query processing,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 27, pp. 1438–1451, May 2015.

[30] G. Bui Cong and A. Duong-Tuan, “Improving sort-tile-recusive algorithm for r-tree packing in indexing time series,” in *Computing Communication Technologies - Research, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on*, pp. 117–122, Jan 2015.

[31] Teradata, “Hash indexing,” March 2013.

[32] R. Irudeen and S. Samaraweera, “Big data solution for sri lankan development: A case study from travel and tourism,” in *Advances in ICT for Emerging Regions (ICTer), 2013 International Conference on*, pp. 207–216, Dec 2013.

[33] Elastic, “Inverted index,” March 2015.

[34] E. Sungkwang, S. Sangjin, and L. Kyong-Ho, “Spatiotemporal query processing for semantic data stream,” in *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pp. 290–297, Feb 2015.

[35] O. El Midaoui, A. El Qadi, M. Rahmani, and D. Aboutajdine, “A new approach to build a geographical taxonomy of adjacency automatically using the latent semantic indexing method,” in *Intelligent Systems and Computer Vision (ISCV), 2015*, pp. 1–6, March 2015.