Available online at http://icsejournal.com/index.php/JCSE
Journal of Computer Science and Engineering (JCSE)
e-ISSN 2721-0251
Vol. 3, No. 2, August 2022, pp. 71-94

# Big Data Indexing: Taxonomy, Performance Evaluation, Challenges and Research Opportunities

Abubakar Usman Othman[1], Timothy Moses[2*], Umar Yahaya Aisha[3], Abdulsalam Ya'u Gital[4], Boukari Souley[5], Badmos Tajudeen Adeleke [6]

[1,4,5]Abubakar Tafawa Balewa University, University Hostel Rd, 740102, Yelwa, Nigeria
[2]Federal University of Lafia, PMB 146, Maraba Akunza, Obi Road, Lafia, Nasarawa State, Nigeria.
[3]Gombe State University, P.M.B 127, Tudun Wada, 760253, Gombe, Nigeria
[6]Industry and Innovation Institute, Sheffield Hallam University, United Kingdom
[1]othman80s@yahoo.com*; [2]moses.timothy@science.fulafia.edu.ng; [3]aishatuyu@gmail.com; [4]asgital@gmail.com
[5]bsouley2001@yahoo.com; [6]agbeketajudeen@gmail.com
* corresponding author

**ARTICLE INFO**

**Article History:**
Received July 2, 2022
Revised September 5, 2022
Accepted September 6, 2022

**Keywords:**
Indexing
Similarity search
Matching
Big data
Cloud Computing

**Correspondence:**
E-mail:
moses.timothy@science.fulafia.edu.ng

***ABSTRACT***

*In order to efficiently retrieve information from highly huge and complicated datasets with dispersed storage in cloud computing, indexing methods are continually used on big data. Big data has grown quickly due to the accessibility of internet connection, mobile devices like smartphones and tablets, body-sensor devices, and cloud applications. Big data indexing has a variety of problems as a result of the expansion of big data, which is seen in the healthcare industry, manufacturing, sciences, commerce, social networks, and agriculture. Due to their high storage and processing requirements, current indexing approaches fall short of meeting the needs of large data in cloud computing. To fulfil the indexing requirements for large data, an effective index strategy is necessary. This paper presents the state-of-the-art indexing techniques for big data currently being proposed, identifies the problems these techniques and big data are currently facing, and outlines some future directions for research on big data indexing in cloud computing. It also compares the performance taxonomy of these techniques based on mean average precision and precision-recall rate.*

## 1. Introduction

A web-based program called cloud computing offers a shared pool of resources. Mobile devices, like smartphones and tablets, may now be used for a wide range of various purposes thanks to advancements in mobile technology [1]. The accessibility of the internet, through the use of widely available broadband Internet access [2], in combination with these portable (mobile) devices, led to the simple collection of digital information in terms of structured and unstructured data [3], which in turn had contributed to the availability of large volumes of data known as big data.

The massive volume of data created each day has outgrown data processing systems like databases and warehouses. Modern technologies are desperately required to handle this varied volume of data properly. Big data analysis in the cloud requires effective technology or methodologies. Big data indexing in cloud computing aims to provide effective information retrieval from enormous datasets as well as to enhance capacity and capability at runtime without investing in new equipment, purchasing new licenses for software, or hiring new personnel. Through the internet, cloud computing enables consumers to access cloud services on-the-fly and pay as they go [4]. Hardware as a Service (HaaS), Software as a Service (SaaS), Platform as a Service (PaaS), Communication as a Service (CaaS), Infrastructure as a Service (IaaS), Data storage as a Service (DaaS), Security as a Service (SecaaS), and Business as a Service (BaaS) are among these services. Data storage as a Service is utilized for the indexing of huge data on the cloud.

Science research has been significantly altered and affected by big data. Astronomers now utilize the Sloan digital sky survey as a collection of tools and data base [5]. The majority of an



http://dx.doi.org/10.36596/jcse.v3i2.548

astronomer's work in the field of astronomy used to be taking images of the sky, but now that those photos have been catalogued in a database, other astronomers may utilize the objects from the catalogued photos. Data from company purchasing transactions is effectively kept on the cloud. Databases are built so that other biologists and scientists may use the generated biological and scientific data, and biological data and experimental data are saved in a public storage facility.

Today, it is highly challenging to gain access to a very big database where information on a patient's diagnosed ailment may be utilized to track the development of his health. These data might be utilized to develop effective and efficient healthcare practices, starting with diagnosis, prescription, patient monitoring, suggestion, referral, and emergency cases. The growing paradigm of mobile devices that enable cloud-based continuous patient monitoring in their homes through the use of information technology is a satisfying approach to significantly reduce costs. In order for the information retrieved to be utilised by the analysis method, the generated unstructured data must be structuralized [6].

Similar to how it has an impact on decision-making, big data has an impact on urban planning (through the fusion of high fidelity geographic data), intelligent transportation systems (through analysis visualisation of live and detailed road network system data), environmental modeling (through ubiquitous sensor networks collecting data), energy conservation (through revealing patterns of use), and smart materials (through new material genome initiatives) [5]. Big data processing became incredibly challenging, making a highly result-oriented method ideal to maximize the speed of data query processing. For this, effective access to huge data in the cloud requires optimized indexing strategies. Big data is a term used to describe a graph dataset that is many terabytes in size and cannot be handled by DBMSs. Such several graph mining algorithms have been proposed [7].

Researchers have, however, suggested several indexing methods with a focus on huge data in cloud computing. Similarity searches; Approximate Nearest Neighbor (ANN) indexing approaches have been an area of interest for study and tree-based algorithms [8-11] have recently been used for indexing in order to efficiently retrieve huge data on the cloud. [12] suggested R-tree-based indexing as a way to index multi-dimensional data on the cloud. An indexing technique called distributed B-tree allows for high concurrency reading operations while also enabling consistent and concurrent updating [13-14]. In a biometric system, databases are recognized so that a more effective indexing strategy can increase throughput by reducing the search space for query images. The majority of the time, nearest neighbor classifiers are used for form matching and image recognition [15]. KD-tree [16] is a multi-dimensional indexing system that was presented for finding the best matches with less time spent. Trajectory indexing systems have been intensively researched for extracting knowledge from trajectory data [17–18], creating efficient indexing structures [19–20], managing uncertainty, and processing trajectory queries [21–22]. Hash-based indexing techniques are renowned for their efficiency in search and similarity computation as well as their effectiveness in application areas like large-scale vision problems, such as image retrieval [23–24], image search [25], object recognition [26], local descriptor compressing [27], fast multimedia search [28], and image matching [29]. While the $c^2$ [31] is utilized for preserving index items in d-dimensional data, [30] applied approximate similarity search. Many more hashing-based indexing approaches [32–50] were presented for effective big data management, storage needs, and retrieval.

Large data analysis should be quicker and cheaper [51], with effective indexing strategies allowing for faster indexing of big data findings while yet tolerating high costs. Obtaining a structured indexing of the original video material and being familiar with its embedded semantics similarly to humans were the goals of content-based image indexing and retrieval, video indexing, and audio indexing [52].

This paper serves as a menu for choosing indexing approaches, offering scholars a way to comprehend and gain insight into the various indexing techniques and the issues they raise. Big data requirements in terms of volume, velocity, truthfulness, value, variability, diversity, and



http://dx.doi.org/10.36596/jcse.v3i2.548

complexity must be met through an effective indexing strategy. However, these research sought to solve the following problems and provide the following contributions to knowledge.

- Emphasize the cutting-edge methods currently being utilized for indexing huge data.
- Identifies related issues with the suggested indexing methods for huge data.
- Propose solutions to address the shortcomings (inefficiency) of the current big data indexing approaches.
- Identify the difficulties and needs for big data indexing.
- To describe potential prospects for research into indexing methods for massive data in cloud computing.
- Use Mean Average Precision to gauge how well indexing approaches operate.

## 2. Big data indexing requirements and challenges

For effective indexing of huge data in cloud computing, many indexing strategies are created. Different criteria are used to assess how well the established indexing techniques work. The most prevalent and fundamental metrics are the indexing technique's speed and accuracy. An indexing technique's velocity requirements are its speed, while its veracity requirements are its accuracy. Volume, variety, value, variability, and complexity are additional needs. Big data requirements for privacy and usability provide a significant problem, but they are outside the purview of this research and will be investigated in other publications. These huge data issues are necessary for evaluating and contrasting indexing techniques. The following obstacles and needs are listed:

- Volume: is a term that is frequently used to describe size in numerous contexts. The magnitude of large data is a difficulty for effective big data indexing and management. Currently, depending on the application area where big data is employed, data are continually growing from terabytes to zettabytes of dataset. Big data volume increase mostly in the field of research as new discoveries were discovered. Big data cannot be measured. Huge amounts of data, or "big data," have also been tremendously influenced by the accessibility of the internet and portable electronics.
- Velocity: Handling the rate at which new data are generated and current data are updated is a problem [53]. Due to the widespread use of sensing and mobile devices, enormous amounts of data are continuously and often created, and the outdated data may be easily updated thanks to the accessibility of internet services, broadband, and portable devices like smartphones. The storage system responds to the data when new data is created by indexing and storing the updated and newly generated data in the cloud. Data must be indexed at an extremely fast rate. The richness of data, which has greatly risen and is utilized for communication via social networks, as well as its speed both have a significant impact on the telecommunications business. Big data velocity, then, is the pace of data collection and the time it takes to process the data after it has been gathered.
- Veracity: The hallmark of big data veracity is the correctness of the data. It may be quite challenging to determine which data is damaged and which is not, as well as if the created data came from a trustworthy source and can be believed. A very important component of any big data demand is big data truth.
- Value: The importance of data in terms of decision-making is referred to as big data value. Big data should have an impact on potential advantages, business transactions, insight, and communication values.
- Variety: Data are gathered in a variety of forms and models from many sources, including databases. The information may come from emails, sensors, mobile devices, social networking sites like Facebook and Twitter, web pages, blogs, images, and videos, business transactions, RFID readings, and papers from the healthcare and/or aviation industries. The difficulty is in organizing the many forms of data into a dataset and correlating their meanings. Structured, semi-structured, and unstructured data are the many data kinds. Big data variety may be defined as the semantic interpretation, heterogeneity of data kind, and heterogeneity of data format.
- Variability: The rise in mobile device accessibility and affordability, along with the quick rise in broadband accessibility and affordability, have significantly increased the use of



http://dx.doi.org/10.36596/jcse.v3i2.548

sensor devices, including body-warn sensors, sensor networks, social networks, and information retrieval for e-commerce. As these technologies are used more often, network congestion arises, slowing down data downloads and uploads and disrupting the flow of large data.

- Complexity: A feature linked to the complexity of big data is the degree of interconnectedness, interdependencies, and very large datasets [54].

## 3. State-of-the-art indexing techniques

The literature on various indexing algorithms is briefly reviewed in this part, with an emphasis on large data in cloud computing. The primary application of the currently available approaches is to index large amounts of data in the cloud.

Data-independent indexing strategies, commonly referred to as randomised hashing methods, are categorized as hashing-based indexing schemes [30], [40], [42] and [55–56]. The projections used by this class of hashing-based indexing methods are generated at random, and the hash function is created using data distribution via data-dependent binary code embedding techniques [57].

By minimizing the Hamming distance between two binary codes in the original feature space as the code length rises, supervised hash coding using deep neural networks [58] was suggested to enhance retrieval accuracy. This leads to the generation of lengthy codes to reach sufficient performance. By minimizing the reconstructive error between the cosine similarity calculated by the original features and the resultant binary embedding, [59] introduced a unique angular reconstructive embedding (ARE) that learns binary hash codes.

Finding two-dimensional objects represented by discrete points that have undergone an affine translation is possible through the use of geometric hashing [60]. A point-based indexing method called geometric hashing is utilized to index a biometric database. Geometric hashing uses the base pairs of the Speeded-Up Robust Features (SURF) key points as the indexing components. Using Scale Invariant Feature Transform (SIFT) to identify key points and a geometric base indexing approach to transfer the identified key points into a hash table, [61] suggested a robust iris indexing strategy to index iris datasets (Geometric hashing). Two steps comprise geometric hashing; the stages of pre-processing and recognition. A hash table is produced during the pre-processing stage. Features are taken from pictures or objects and stored in the hash table as key points in a database. The Difference of Gaussian (DOG) function, as defined in equations (1) and (2), was employed by the authors to identify prospective interest points [62].

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (1)$$

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (2)$$

Where k is the constant multiplicative factor used to change the scale and x,y are the coordinates of a pixel in image $I$, and $L(x, y, \sigma)$ is determined from equation (2) where $G(x, y, \sigma)$is the Gaussian filter for smoothing the image, and $\sigma$ is defined as the width of the filter. A gradient orientation histogram is used to locate the key points once each key site is given an orientation. Equations (3) and (4) below calculate the magnitude and direction as follows:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y_1))^2} \quad (3)$$

$$\theta(\mathrm{x}, \mathrm{y}) = \tan^{-1}\left(\frac{(\mathrm{L(x+1,y)-L(x,y-1)})}{(\mathrm{L(x+1,y)-L(x-1,y)})}\right) \quad (4)$$

$$P = up_x^i + vp_y^i + p_o^i \quad (5)$$

In equation (5) above, $P = [x, y]$ is the set of key points to be indexed, and $(u, v)$ is the position of point p (the key point) following similarity transformation. $m(x, y)$ and $\theta(\mathrm{x}, \mathrm{y})$ are the magnitude and orientation, respectively [63].



http://dx.doi.org/10.36596/jcse.v3i2.548

Due to the many insertions of image features that have been extracted into the hash table to accommodate every potential rotation and translation, this technique is inefficient because, it leads to high computational and memory costs.

Regarding guiding principles and significant operators capable of mapping linked data with search keys, B-tree indexing-based approaches are connected to an audit of multi-dimensional large data. One of the characteristics of big data is their ability to manage data in massive volumes of various sizes. [64] suggests an effective indexing system for temporal data ranking queries. Top-k searches on temporal data are answered in this study in almost linear time, with projected low I/O costs. The SEB-Tree was created by the authors using the B-tree. They produced a set of l+1 independent samples of the set S, where S is the set of segments shown in equation (6).

$$l = [\sqrt{\log\left(\frac{N}{B}\right)} + \log(\frac{k_{max}}{B})] \quad (6)$$

*N* is the database's number, B is the block size, and k is the query parameter [65-66].
Despite being effective, the approach cannot handle live data streams since their behaviors are unknown.

In order to enable data owners to freely put data into a database at any moment, integrity auditing of outsourced data was recommended [67]. For query authentication, the authors employed a probabilistic technique, which is simpler and more flexible to implement. The authors just add a few tuples to the external database. The tuples are then authenticated using equation (7) to determine if they are genuine or fraudulent.

$$ah = \begin{cases} h(tid \oplus a1 \ldots \oplus a_n) & t\ is\ real \\ h(tid \oplus a1 \ldots \oplus a_n)+1 & t\ is\ fake \end{cases} \quad (7)$$

where $t_i$ is the $t_{th}$ tuple, h is the hashing algorithm, and $a$ is the header column

Because the way these records are formed has a significant influence on the storage performance of the scheme, both the randomized and the deterministic ways for creating the inserted tuple are explored. Absolute accuracy for query authentication is not guaranteed by the technique, which is an issue [64].

[68] suggested an effective indexing method for a face database. The new indexing method was created by the authors of this work using hashing. The hash table and descriptor vector for model recognition were created using the coordinates of the control points. Control points are the elements that give an image its distinctive qualities. Utilizing Speeded-Up Robust Characteristics (SURF), control point features are retrieved. To make the control points invariant to translation, rotation, and scaling, a pre-processing method is utilized. This method comprises mean centering, principal component rotation, and normalization. After that, mean centering is used to reduce the impact of noise. The mean of all the translated control points is zero once each control point has been translated. Utilizing the direction of the control points that stayed constant even when some of the points are not accessible, the rotation of the control points-based primary components is accomplished. Using the Principal Components Analysis (PCA), this is accomplished. By performing a dot product of a point vector with the principal component vector, which gives the projection of the point on the principal component, coordinates are rotated with respect to their mean such that the first and second principal components are aligned along the x- and y-axis of the coordinate system.

The updated geometric hashing is scale-invariant thanks to normalization. To obtain the normalized coordinate values for each control point, the standard deviation of coordinate values is employed as follows:

$$h_x = \frac{1}{\sigma_x} q_x \quad (8)$$



http://dx.doi.org/10.36596/jcse.v3i2.548

$$h_y = \frac{1}{\sigma_y} q_y \quad (9)$$

Where $\sigma_x$ and $\sigma_y$are the standard deviation, $h_x$ and $h_y$ are the normalized coordinate values, $q_x$ and $q_y$ are the coordinate values. Each control point takes up a distinct bin in the hash table as a result of scaling factors multiplied by the normalized coordinate values of the control points. Voting is then utilized to discover the top k best matches against the query from the models kept. The resulting table is searched against control points of a query for recognition so that dissimilar control points to the query's control points are removed [68].

When compared to previous strategies, experimental findings for the suggested technique reveal that both the computational cost and utilization of memory space have been significantly decreased. The approach is redundant in that each normalized point of the altered query model has to be validated to confirm that specific point exists in the hash table, even though memory and computational costs are both decreased.

[69] suggests an indexing method for a biometric database made up of several attributes with a configurable number of dimensions. The research makes advantage of the geometric aspects of the features' principle components such that, after rotating the first two highest principal components to the fundamental axis of the coordinate system, it may insert fewer features into the hash table. Their approach, a two-stage triplet-based indexing strategy, is based on the triangle that triplets of features make. The triangle's angles serve as the features. These features are taken from database models using Speeded-Up Robust Features (SURF), which allows for quick calculation and the production of lower dimensional features. Prior to taking a rectangle window of these found key points, salient points are first determined by employing a Hessian matrix to find key points.

The indexing and searching are phases of the triplet-based indexing approach. Following the extraction of features by SURF, the indexing step performs principal component analysis transformation, triplet construction, and hash table building. To make each model in the database invariant to translation, rotation, and scaling, principal component is employed. The retrieved features are likewise translated from their original positions when the model image is translated. Then, using mean centering, each characteristic of every model image in the database is translated to a different image such that the mean of the translated image is zero.
Mean centering may be calculated by:

$$\overline{f} = \frac{1}{m} = \sum_{i=1}^{m} fi\,, from\ fi \quad (10)$$

$$S_i = \sum_{i=1}^{m} (f_i^1 - \bar{f})(f_i^1 - \bar{f})^T \quad (11)$$

From the list above, m stands for the model's features, $\overline{f}$ for its mean value of the features of model M, and $S_i$ for its scatter matrix.

Following transformation, the major components are then found and calculated using the eigenvalues as in equation (4). The major axes of the coordinate system are formed by rotating the principal components. Each feature takes up a distinct bin in the hash table once the normalized features are multiplied by a scaling factor. A hash table's position is assigned to each triangle in a model image. The coordinates are provided in equation (12).

$$pi = (\propto_{min,} (\propto_{max,} - \propto_A) \quad (12)$$

Where $\propto$ is the angle of a triangle produced by triplet feature points and $pi$ is the $pth$ position of the features in the hash table.
As in equation (13), the triangle is added to the hash table one at a time.

$$H(p_i) = H(p_i)\ U\ (M_{id,}\ \overline{D}) \quad (13)$$



http://dx.doi.org/10.36596/jcse.v3i2.548

Where $M_{id,}$and $\overline{D}$ are the model identity and descriptor vector connected to the triangle's greatest angle, respectively.

The triangles in the model image that do not resemble the triangles in the query are filtered out during the searching phase. In order to access the correct bin, features are retrieved from an image, triangles are generated between triplets with the new coordinate, and these angles are then mapped into the hash. Only triangles whose distance is under a threshold are chosen. The Euclidean distance between the feature descriptor of a query triangle and all the triangles found in the bin is calculated. The remaining query's triangle is subjected to this procedure, and the results are then compiled into the candidate set. The candidate set's occurrences of the model identity are then voted on to determine the top matches.

The triplets are well distributed, as practically every bin occupies an equal amount of triplets, according to experimental results of the suggested approach when compared with the existing techniques in terms of index distribution, which results in quicker indexing and reaction time. Although the plan is effective in terms of speed, it is unable to take data from many sources.

For effective retrieval of skewed geographical data, [70] developed a unique key design based on R+-trees. On this study; indexing spatial data management in the cloud, the authors developed a new indexing method (KR+-index) based on R+-trees that supports effective multi-attribute accesses for skewed data on cloud data management systems (CDMS). By dynamically dividing and merging nodes, the R+-tree creates a balancing search tree and may limit the quantity of items in each node by adjusting the M and m. Key names are created for R+-tree leaves. To split the data, the R+-tree is employed, and the rectangles in the tree's leaf nodes are used as grids.

The objects record for each rectangle, let's say R1, R2, and R3, is kept, and the data required in the scheme are created by the R+-tree with specified M, m. A new data point is added into a node by first looping up the key of the point that corresponds to the model to which the point belongs. To determine if a split would be necessary, the node's current size is checked. The node is divided into two new sub-nodes and the old node is removed when the number of points reached the appropriate limit. The original node's points will be distributed among the new sub-nodes. The method considers cloud data management to effectively extract skewed and spatia data.

The method is effective for accessing data and offers support for both range and nearest-neighbor (NN) searches, but it lacks a mechanism for quick responses to queries regardless of the size of the query and the data being accessed. The parameters of the technique are the order O, the bottom and upper boundaries of the rectangle (M, m). The experimental finding demonstrates that, for skewed data, the new indexing approach, KR+-index, beats the most recent index method, MD-HBase. The system requires a significant amount of memory and lacks secrecy. Because computing requires a lot of processing time, the cost is significant.

[46] suggested a method for keeping track of K-nearest neighbor requests over moving objects. Grid indices are used by the authors to create algorithms for both object's indexing and querying. Additionally, a cost model was created. Continuous monitoring of several K-nearest neighbor (k-NN) queries over moving objects was the major emphasis. In order to achieve high throughput, a two dimensional region of interest is taken into account in this study. Each item in this research has a unique identification $p(t) \in P(t)$, and exists in the unit square $[0,1]^2$. The plane is divided into regular grid cells of equal size, and the coordinates of the objects $p(t)$ are indicated as $\langle p(t)x, p(t)y \rangle$. After that, the items are scanned. To make it possible to create their index, the scanned items are mapped into the correct cells. $(p(t^{'}) \in P(t^{'})$ is used to keep the object's growing list for each cell at time *t*. Equation (14) illustrates how to build mobility and indexing, and equation (15) shows how to calculate the incremental query response time when mobility and cell size are present.

$$\Pr = \begin{cases} 1 - \left(\frac{\sigma}{2v_{max}}\right)^2 \; if \; \sigma \leq v_{max} \\ \frac{v_{max}}{\sigma}\left(1 - \frac{v_{max}}{4\sigma}\right) if \; \sigma > v_{max} \end{cases} \quad (14)$$



http://dx.doi.org/10.36596/jcse.v3i2.548

$$t_{query} = \left(b_0 + b_{1\mu}\sqrt{Npb_2\mu^2 NP}\right).N_Q, \quad (15)$$

$$T = \left(C1\ \frac{(l_{crit}+\delta)^2}{\delta^2} + C2\ (l_{crit} + \delta)^2 N_P\right)N_Q. \quad (16)$$

Where $v_{max}$ denotes the maximum mobility, ($l_{crit}$)denotes the neighbors' distances from the bounding rectangles (width and height), $N_P$ denotes the length of the array, $t_{query}$ denotes the query time, $\sigma$ denotes the size of the cell, $N_Q$ denotes the number of queries, $N_p$ denotes the number of objects, $T$ denotes the overall time, and $\mu$ denotes the deviation.

Equation (16) above demonstrates how query indexing may be utilized to respond to requests by obtaining a bootstrap from k-NNs queries. The scheme is effective in terms of scalability, memory usage, and speed, but it does not support time parameterised queries due to their focus on continuous queries for monitoring rather than a general spatial-temporal indexing method. Experimental results show that index construction requires a linear time and query answering time is nearly constant.

[47] suggest an innovative method for indexing moving objects. The suggested main-memory approach uses frequent snapshots to move items into the index, which supports time-parameterized (predictive) searches and is simultaneously space-, query-, update-, and multi-CPU efficient.

Fundamentally, MOVIES is similar to the method used by a cinematographer: since no camera can record continually changing data in a single frame, a cinematographer must take a sequence of static images at a specific frame rate. An illusion of continuous movement may be produced as long as the frame rate is higher than the inertia of the human eye (i.e., at least 24 frames per second). We take the exact same course of action. By taking into account a dataset of, let's say, N moving objects in a 2-dimensional region of interest, the authors were able to create a short-lived index picture that is only stored in the main memory for a brief amount of time. The domain of the scheme is |X|*|Y|, where |X| stands for the number of different places in the horizontal (or vertical) dimensions. For all incoming queries, the algorithm relies on read-only indexes and index frames. Data and query results are predicted using a timestamp-consistent predictive index and timestamp-consistent query processing. The system is effective, according to experimental results, but it is less effective at dealing with more widespread issues with indexing data streams. Scalability and staleness must be balanced. When processing a huge number of data and changes, the approach is ineffective.

[48] suggests the use of a composite tree index method for the run-time correlation engine to facilitate efficient event indexing and searching. The ability of compact tree data structures is used by the authors so that they may share a single set of event indices for all of the leaf nodes on the B-tree. Additionally, they make advantage of the search index data structure to effectively process timestamp-based searches. Events are promptly stored to the file storage utilizing a container paradigm as they enter the system. A list of keywords is created from the message's extracted content. To index each extracted keyword, a node is created or returned by the tree structure. If an event is received using the RTCE event data format, the engine creates an event object, and the data for the index, which corresponds to the event index. Additionally, the tree structure acts as a lexicon for storing each term included in stream events, after which a reference is made. The root node, the transition node, and the leaf node make up the tree structure of the scheme. The query engine scans the list of requested keywords to execute a query. This approach uses a composite tree-based indexing strategy that is effective in terms of quick response times. The Java Garbage Collector (JGC) must repeatedly stop in order to free up the heap for the incoming events, which results in a reduction in performance, according to experimental data. As a result, the indexing performance is not justifiable.

[49] created an epic system as an integrated framework to accommodate high concurrent OLTP queries and huge scale data analysis activities. Different types of indexes were created and incorporated into the system to enable effective query processing for a range of applications. Run-



http://dx.doi.org/10.36596/jcse.v3i2.548

time Content Accessible Network (CAN) has incorporated an R-tree based indexing technique with a CAN-based routing protocol (RT-CAN). The multi-dimensional data and query processing in a cloud system is called RT-CAN. To accomplish efficient indexing, an integrated CAN-based routing protocol and an R-tree-based indexing system were utilized. To retain index items in d-dimensional data, the author utilizes d-dimensional $C^2$. An R-tree node is inserted into a CAN node using a mapping function. Then, R-tree nodes were indexed for use in indexing operations, and at the same time, the indexes for each subsequent R-tree node were published. In the epic system, a CAN node that has the key receives a query, processes it, searches its buffered global index, and returns the user-retrievable result. Equation (17) and (18) calculates the cost model and the anticipated processing time for KNN queries.

$$D_x \approx \frac{2\sqrt[x]{r\left(\frac{d}{2}+1\right)}}{\sqrt{\pi}} \left(1 - \sqrt{1 - \sqrt[d]{\frac{k}{N}}}\right) \quad (17)$$

$$C(S) = \sum_{n \in S}(c_{FP}(n, Q) + C_M(n, L)) \quad (18)$$

Where *d* is the dimensionality of RT-CAN, *K* are the data items, *S* are the nodes of tree, *N* are the estimated numbers of data in the whole space, $\Gamma(x+1) = x\Gamma(x)$, $\Gamma(1) = 1$ and $\Gamma(\frac{1}{2}) = \frac{\pi}{2}$, and $c_{FP}$ are the false positive and maintenance costs, respectively, and *n* stands for the node.

The epiC system lacks scalability in terms of increased dimensionality [75], despite being effective at managing huge users and massive amounts of data [76-78]. The overlapping of d-dimensional space and range queries with a large number of index items, which results in a high communication cost, are the main causes of throughput declines as dimensionality increases.

[23] provided a quick image search for metrics that were learnt. They discovered a Mahalanobis distance function that effectively encapsulates the underlying connection of the image. The authors encode the learnt metric parameterization into randomised locality-sensitivity hash functions as in equations (19, 20, 21, 22) to enable sub-linear time similarity search under the taught metrics.

$$d_A(x_i, x_j) = (x_i - x_j)^T A(x_i - x_j), \quad (19)$$

$$A_{t+1} = A_t + \beta_t A_t(x_{it} - x_{jt})(x_{it} - x_{jt})^T A_t, \quad (20)$$

$$K_{t+1} = K_t + \beta_t K_t(e_{it} - e_{jt})(e_{it} - e_{jt})^T K_t, \quad (21)$$

$$Pr[h(x) = h(y)] = sim(x, y), \quad (22)$$

Where *t* is an iteration, $\beta_t$ is a projection parameter, $e_{it}$ and $e_{ij}$ are vectors to the $\text{it} - \text{th}$ and $\text{jt} - \text{th}$ standard basis vectors, respectively, and $d_A$ is the distance between the matrix $x_i$ and $x_j$. Equation (22)'s probability of collision and similarity function is denoted by $Pr$, $sim(x, y)$. The system is unreliable and uses an excessive amount of memory and processing time.

[34] provided many indexing methods, including content-based image indexing, content-based multimedia indexing, audio indexing, and video indexing. These methods are together referred to as content-based indexing methodology. The indexing strategy aims to identify the targets speakers in movie dialogs and extract semantically significant movie events. The authors employed a searchable index of the speech material present in digital audio files created using an online audio indexing system. The system looks for the borders of acoustically homogenous segments as data is received and categorizes it into speech, music, and mixing classes. The speech fragments are grouped together to offer consistent speaker identification. The speech and mixing portions were transformed into text format using the ASR technique. After that, the output of the words is time-stamped in XML along with other immediate data. The writers only discuss the various indexing methods and concentrate on the usage of online audio indexing systems. The system cannot handle complexity, diversity, or truth.



http://dx.doi.org/10.36596/jcse.v3i2.548

[50] By using a dynamic threshold to enhance cluster identification of latent semantic indexing, [50] suggest a method that is based on Latent Semantic Indexing (LSI) as part of the strategy. Input selection, pre-processing and indexing, latent semantic indexing, calculating similarity and grouping, and visualization are all included. The input is indexed using the individual words once the input variables have been specified and pre-processed. The latent semantic index is then used to process the pre-processed and index-generated matrix. The approach's text is represented as a term-by-context matrix, or M, which is then broken down using the singular value decomposition method. The heuristic in [72] is utilized to determine the number of dimensions as a result of the size of the document used in equation (23).

$$heu = (m\ x\ n)^{0.2} \tag{23}$$

Where m and n, respectively, stand for the number of contexts and terms. The linked context is clustered using the similarity metric. The distance function is used in R, a programming language designed for data analysis and visualization, to construct these clustered linked contexts. The authors employed dynamic hybrid cut [79], which is renowned for considering the dendrogram's form and building the clusters from the bottom up. According to experimental findings, the dynamic hybrid cutting method significantly increases the ability of LSI to identify issues in source code. Because the dynamic hybrid cutting approach is so good at cutting asymmetric dendrograms, the findings beat the fixed height threshold cutting technique [73-74]. The scheme's flaw is that it only uses one expert per case study, allowing for the use of a priori information that can be exploited to affect performance outcomes.

[51] designed the IG system for graph queries, which is a quick graph query processing with a cheap index. The method allows for quick indexing and effective query processing. The authors built their index using a quick approach for extracting network commonalities based on basic graph statistics. All of the database's data graphs were combined into a single graph and closely adhered to the frequency of the edges in order to achieve high throughput. The integrated graphs (IG) are a small representation of a collection of graphs that have a number of desirable properties and are so inexpensive to build that subgraph isomorphism testing is not necessary. They are also simple to maintain in terms of update database. Query processing in this study comprises query integration, direct incorporation of replies, and project-database filtering. The query response time is as given in equation (24) below:

$$T_{response} = \left(T_{search} + \sum_{q \in Q}(|C_q| * T_{I/O} + |C_q| * T_{verify}\right) \tag{24}$$

Where $T_{verify}$ is the time needed to verify the candidates, $T_{I/O}$ is the disk input and output for obtaining each candidate graph, and $T_{search}$ is the index search time. According to experimental findings, index creation is quicker, uses less memory, and processes queries more efficiently than the state-of-the-art indexes utilized in this study.

To address the tolerant retrieval problem in the shortest possible query time, [81] introduced the compressed permuterm index. The construction of the string, computing $L = bwt(S_D)$, and creating the compressed data structure to facilitate RANK queries over the string L are the three processes that make up the compressed permanent index. The $jump2end$ function, created by the authors, may change backward processes and handle PREFIXSUFIX queries. To discover the rows that are prefixed, the search method *BackPerm_search* scans characters backward. When the *BackPerm_search* ($\alpha\$\beta$) function is used, the number of dictionary strings is returned as $last - first + 1$. Applying $Display\_string$ will return these strings. The *BackPerm_search* ($P$\$)function does the same thing, returning the value of First if First < last, otherwise concluding P ∄ D. Select activates the Display string (i) provided that $1 \leq i \leq m$. According to experimental findings, the plan is effective at addressing issues with tolerant retrieval.



http://dx.doi.org/10.36596/jcse.v3i2.548

## 4. Datasets used in indexing

It is extremely challenging to execute real-time indexing of this complicated information or data, or big data, in cloud computing because of the complex nature of data flowing from many sources as a result of the availability of mobile devices and broad band internet. Volume, velocity, and variety are the elements that make up data. Value [53] and veracity [54] are factors that deal with the utility of data for decision-making, and the degree of data trustworthiness must also be met. The indexing system must be effective in indexing big data and meet requirements for volume, velocity, variety, veracity, variability, value, and complexity. Simulations are performed to verify the effectiveness of the indexing scheme with regard to search accuracy. The simulations don't take place in real time. For non-artificial intelligence (NAI) indexing approaches, the grounds for testing indexing accuracy included charts, images, geographic data, and text. Using a multimedia dataset, the artificial intelligence- and collaborative artificial intelligence-based indexing strategies are assessed. The effectiveness of indexing techniques based on artificial intelligence is also tested using text-based and annotated datasets. Researchers tested the viability and efficacy of the existing indexing techniques using various datasets. Data identification is facilitated by the platforms used for the datasets. For instance, the platform where the graph data will be used can help identify a graph dataset. [55] put a lot of effort into developing various indexing techniques on the iGraph graph dataset structure. Therefore, the implementation platform will aid users in understanding the type of data available in the dataset. Although there are other ways to gather data, it is not discussed in this section. In [56], datasets from the US Forest Service website, Flickr, and the Wikipedia database, respectively, were utilized.

Additionally, two multimedia datasets that were gathered and made publicly accessible from prominent Wikipedia articles are incorporated in [57]. The NUS-WIDE database was used to acquire the second dataset, which is the Flickr data. 2866 image-text pairings from Wikipedia were retrieved in order to assess the efficacy of their technique. Of these, 2173 served as the training set, while 697 served as the test dataset. 186577 image-text pairings total, of which 185577 are utilized as training data and the remaining 1000 pairs as test datasets, are present in the NUS-WIDE database.

## 5. Categorization Methodology

For the purposes of this review, indexing methods suggested by reliable researchers that were made available from highly regarded publishing journals as evidenced by their impact factor were employed. The Journal Citation Report (JCR) determines the rankings. In this study, the effectiveness of an indexing strategy was evaluated in terms of its ability to address big data requirements in cloud computing while also enhancing high recall rates. Additionally, this will help highly esteemed researchers of the highest caliber assess potential offered solutions in order to create effective indexing strategies that meet the needs of big data for big data indexing in cloud computing environments. In light of the performance advantage, the relevance of the methodology utilized in developing an efficient indexing system is examined. Big data factors are used to validate the performance of proposed indexing systems and compare them to other current schemes. One of the most crucial big data factors is volume, which is one of the characteristics that define big data. The 6Vs and C, also referred to as velocity, variety, veracity, value, variability, and complexity, are additional considerations. Some indexing methods just meet the volume requirement for data while others also meet the velocity requirement, which refers to the rate or pace at which new data are generated or current data are transformed.

The ability of bit-scalable deep hashing codes to maintain discriminating powers with short codes is well established. [58] suggests using a deep convolutional neural network to build a supervised learning framework for producing bit-scalable and compact hash codes from raw photos. The use of compact and bit-scalable hashing algorithms in combination with neural networks, according to the scientists, produces outstanding results in similarity searches for picture and person re-identification in surveillance. Additionally, the created codes preserve the ability to discriminate when employing short-length codes. It outperforms previous approaches like DSRH with a



http://dx.doi.org/10.36596/jcse.v3i2.548

significantly increase of 1.67% because of the integrated representation of features and hash algorithms.

Table 1 is an overview of research journals reviewed in this work, along with an explanation of their features and the drawbacks of the current indexing techniques. The table provides a list of the various indexing methods that were utilized to categorize the data in this study. It has six vertical columns that list the author's name, the work's title, the technique utilized, a description of the method, the connected problem and its current solution, and the suggested solution. The table also includes features as optimising search time.

**Table 1.** State-of-the-art techniques

| Author | Title of the work | Method used | Description of the method used | Problem associated with the method used | Proposed solution |
|---|---|---|---|---|---|
| [23] | Fast image search for learned metrics | Hashing | The image's underlying relationship are captured. The learned metric parameterisation are encoded into randomised locality-sensitivity hash function. | There is high computational cost and not suitable for veracity. Data retrieval is very slow. There is high memory consumption. | Deign an algorithm that would be suitable for veracity. Use of geometric hashing of SURF key points to improve the speed of recognition. |
| [32] | Supervise hashing with kernels | Hashing | Minimal amount of supervise information is used for high quality hashing. Supervised information are similar and dissimilar data pairs. The authors utilised the equivalence between optimising the code inner products and the hamming distances. | It cannot handle large spectrum of information such as duplicate document detection. | Design an algorithm that can minimise the minimum information criterion and the hamming distance. |
| [34] | Spherical hashing: Binary code embedding with hyperspheres | Spherical hashing. | Hypersphere-based hashing function is use to map more spatial coherent data points into a binary hash code with a new binary code distance function the spherical Hamming distance suitable to the hypersphere-based coding scheme. The binary code embedding function $H(x)$ maps data in $R^D$ points into the binary hash code. | SHD does not provides significant improvement in terms of accuracy with the generalised spherical hashing. | There should be a similarity embedding term incorporated into the independent hashing functions to improve accuracy. |
| [35] | Compact hashing with joint optimization of search accuracy and time | Compact hashing | The search time is analyse and model to ease the minimisation of the search time. The search time is minimised by balancing the hash bucket. | The scheme cannot handle large scale data set. The use of one hash table degrade performance in terms of recall. | Employ the use of hypersphere to reduce computational complexity by defining tighter closed regions among the data points. |
| [47] | Compact structure hashing via sparse and similarity preserving embedding. | Special structure-based hashing (SSBH) | The sparse reconstructive relationship of data to learn compact hash codes. The information provided by each bit is utilised to obtain desired properties of hash codes. The information theoretic | The computational complexity involve in the objective function, the map matrix, sparse weight matrix degrade performance. | The hash functions should be design using the discriminative similarity information among the data points to |



http://dx.doi.org/10.36596/jcse.v3i2.548

| Author | Title of the work | Method used | Description of the method used | Problem associated with the method used | Proposed solution |
|---|---|---|---|---|---|
| | | | constraint is incorporated into the relaxed empirical fitness as a regularising term to obtain the objective function | | reduce computational cost. |
| [49] | Asymmetric cyclical hashing for large scale image retrieval. | Asymmetric cyclical hashing. | Two hash codes of different length are used for stored images in the database and the queries. The compact hash code is used for the stored images in the database to reduce storage cost while the long hash code is used for the queries for searching accuracy. To retrieve images from the database, the Hamming distance of the long hash code is computed for the query and the cyclical concatenation of the compact hash code of the stored images for better precision-recall rate. | There is long response time and additional computational cost for calculating the Hamming distance of the compact hash code of the stored image and long hash code of the query. There is large storage cost due to the use of long codes. | The hash function designed should be based on the distribution of data for effective short compact hash codes. |
| [52] | Different Indexing Techniques | Content-Based Indexing | Semantically meaningful movie events are extracted from movies. An online audio indexing system is used to create a searchable index speech content contained in digital audio files. Boundaries of the acoustically segment of data are searched and the data is then classified as speech, music or a mixture. | The focus was on online audio indexing system. The system is not suitable for veracity, variety and complexity. | Manifold learning |
| [61] | Robust iris indexing scheme using geometric hashing of SIFT key points. | Geometric Hashing (point based | Local descriptors and relative spatial configuration are used for identity matching. SIFT is used to extract local features from noise independent annular iris image to detect key points. Geometry hashing is then applied to the detected key points for indexing in the database. In the retrieval phase, geometric hash location of query image is used to access the exact bin of the table and a vote is cast and images with certain number of votes are considered. Key point descriptors of possible candidates is matched with the query iris to get the potential match. | The method is redundant in that the features are mapped into the hash table multiple times. The feature points are not normalised. There is high memory consumption and computational cost. | Use SURF extraction technique to extract feature points from images. The feature points should be pre-processed and normalised. DOG should be used to detect interest points. Data number should be reduced in the hash bins to improve the performance of recognition.<br>We employ a technique to evenly distribute features into hash table (DSH). |
| [64] | Top-k queries on temporal data | B –tree | B-tree is used as a building block to design a SEB-tree to support temporal ranking queries. SEB-tree answer a top-k query for any time instance *t* in the optimal number of I/Os in expectation. | The scheme is impractical when dealing with unknown behaviour of online data stream and it is also faced with high computational cost. It | Graph partitioning and B+-tree (hybrid B-tree). |



http://dx.doi.org/10.36596/jcse.v3i2.548

| Author | Title of the work | Method used | Description of the method used | Problem associated with the method used | Proposed solution |
|---|---|---|---|---|---|
| | | | Piecewise linear functions are break into segments and uses the upper envelop U(S) which is made of the portion of the segment visible from $+\infty$ along y-axis. A series of $l+1$ independent samples are created and the query algorithm is designed. | suffer from the curse of dimensionality and cannot deal with large-scale data base because of the memory constraint. | |
| [65] | Indexing in network trajectory flows. | Graph partitioning and B+-tree | The T-PARINET is composed of network model, query model, and the data model. The network model defined for the T-PARINET uses representation from the road network based on the geometric view and topology view. The geometric view captures approximate geographic location of the road network while they topology view uses the graph in order to represent the road sections and intersections. | Consumes vast computing resources when carrying out online data stream indexing. It cannot handle large-scale data base. | Fuzzy |
| [66] | MOVIES: Indexing moving objects by shooting index images. | Fuzzy | Short-lived index images are constructed and kept in the main memory for a very short time. The predictive and non-predictive MOVIES algorithms are designed to support time parameterised predictive queries. The MOVIES indexing algorithm which is based on index frames uses frE-quent Snapshots to index moving objects. | The scheme could not handle general problems of indexing data streams. There is trade-off between scalability and staleness. The scheme is inefficient in handling large volume of data and updates. | Cache aware B+-tree as read-optimised structures. |
| [67] | Integrity auditing of outsourced data | Query Authentication. | Small records are inserted into the outsourced data. Randomised and deterministic approaches for generating the inserted records are both studied. To effectively audit the integrity of the system, the inserted records in the query result are analysed. | There is no guarantee that query authentication is correct. | Authenticated tree-based structures. |
| [68] | An efficient indexing scheme for face database using modified geometric hashing. | Modified Geometric Hashing | It uses the modified geometric hashing. SURF operators are used to extract control points from the face database. A pre-processing method mean centering, principal component, normalisation and rotation are used to make the control points invariant to translation, rotation and scaling. | Redundancy. The scheme does not support indexing of a database that is dynamic (increase and decrease in size) to enable modification. Variant are not uniformly distributed into the hash space. High memory cost. | Employ dynamic geometric hashing technique to support insertion, deletion of feature points, data points, data and updating the bin table. Use prime hash function to maximise the distance of keys with collision and also to ensure uniform distribution of variant into the hash space (bin). |



http://dx.doi.org/10.36596/jcse.v3i2.548

| Author | Title of the work | Method used | Description of the method used | Problem associated with the method used | Proposed solution |
|---|---|---|---|---|---|
| [69] | Use of geometric features of principal components for indexing a biometric database. | Triplet-based hashing | Geometric features of principal components are used to insert fewer features into the hash table. SURF is used to extract features from the database by using Hessian matrix to detect key points. . | The scheme is not suitable for variety and does not support modification of the database. There is high search time and memory cost. | We use compact hash codes to reduce memory cost. Use dynamic geometric hashing technique to support modification of data. There should be minimal number of data points in the hash table to improve the speed of recognition. The number of feature points for each image in hash bins should be equal. |
| [70] | Indexing spatial data in cloud data managemen t | R+-tree | A key names for leaves on an R-tree are designed. R-tree is used to divide the data and the rectangle in the leave nodes are treated as dynamic grids. Data points are inserted into the node. Range query is used to get the geographic coordinates of the overlapped grids. | There is high consumption of space and high response time for large dataset. The search performance drops with data of high dimensionality. | There should be an efficient scheme suitable for velocity and volume |
| [71] | Monitoring k-NNs queries over moving objects. | Fuzzy | Grid indices are used for algorithm formulation to index objects and queries. Objects are scanned and mapped into corresponding cells and their respective index are then constructed. | It does not support time parameterised queries due to their focus on continuous queries for monitoring in contrast to a general spatial-temporal indexing method. Frequent creation of index image consumes time. | Use frE-quent Snapshots to support item parameterised (predictive) queries. |
| [72] | High volumes of event stream indexing and efficient multi-keyword searching for cloud monitoring. | Composit e tree (B-tree) | The solution is built based on the composite index data structure which shares a single list of event indices for all the leave nodes on the B-tree. Search index data structure is used to efficiently process timestamp-base queries. | There is high consumption of computing resources because of many operation involved. The scheme suffers from the curse of data dimensionality. | Hybrid indexing classifier that considers dynamic graph partitioning. Graph partitioning and B+-tree (hybrid B-tree). |
| [81] | The compressed permuterm index | Permuter m index | Compute (First', Last') = BackPerm search($\gamma\$\beta$)<br>• Compute [First'', Last''] = BackPerm search($\beta$ )<br>• For each r ∈ [First', Last'] repeatedly apply Back step until it finds a row which either belongs to [First'', Last''] or to [1,m] (i.e. starts with $). | High memory consumption. | Cross-indexing of binary SIFT codes. |
| [82] | Indexing multi-dimensional data in a | R –tree and content accessibili | Indexes are designed and integrated. The R-tree based indexing scheme and the content accessibility network | An increase in dimensionality result to decreases in throughput due largely to | Compact R-tree to utilised storage |



http://dx.doi.org/10.36596/jcse.v3i2.548

| Author | Title of the work | Method used | Description of the method used | Problem associated with the method used | Proposed solution |
|---|---|---|---|---|---|
| | cloud system | ty network (CAN) | based routing protocol are then integrated as RT-CAN. R-tree node are then inserted to a CAN node. The R-tree nodes are then indexed. A query is directed to a CAN node that contains the key for query processing. | overlapping of d-dimensional space and range query with large number of index items which leads to high communication cost. | |

## 6. Taxonomy of indexing techniques

This study's indexing methods are grouped and/or classified to provide a clear understanding. This is known as taxonomy. Non-artificial intelligent (NAI), artificial intelligent (AI), and Collaborative Artificial Intelligent (CAI) are the three categories on which the taxonomy of indexing approaches is built. These indexing systems are further divided into components for categorization. Non-artificial intelligence, for instance, is further broken down into components like hashing, graphs, and bitmaps. Machine learning (ML), soft computing (SC), as well as Knowledge Representation and Reasoning (KRR), are divisions of artificial intelligence. The sub-division of collaborative artificial intelligence is Collaborative Machine Learning (CML), collaborative soft computing, and collaborative support vector machines.

Non-Artificial Intelligent (NAI): The NAI uses indexing methods including bitmap and hashing [52], as well as tree-based indexing methods like B-tree [59], [19], and R-tree [60-61]. With regard to indexing creation and query response, this category of indexing strategies takes a simple approach. Big data may be retrieved from the cloud relatively quickly and effectively using the indexing techniques included in this group. These indexing strategies fall into a group that is unable to identify large data's unpredicted behavior in cloud computing. The most frequently accessed items in a given data collection are used to generate indexes, not the relationships between texts or the meaning of the data items, as would otherwise be the case.

Artificial Intelligence (AI): AI can recognize the unidentified behavior of massive data, and can apply rule-based automation for recognized patterns. Rule-based information was used in AI indexing approaches to increase the effectiveness of huge data retrieval. These indexing methods fall under a very scientific category. Machine learning, soft computing, knowledge representation, and reasoning are all categories of artificial intelligence. When compared to NAI approaches, they are frequently viewed as being less effective since they construct links between data items by examining the trait or pattern and grouping things with comparable patterns [62]. Latent Semantic Indexing is another indexing method used by AI [63-66].

Collaborative Artificial Intelligent (CAI): The accuracy and search time shortcomings of the AI indexing methods are supplemented by the CAI indexing schemes. It uses artificial intelligence as a foundation to create effective indexing algorithms that outperform AI approaches. The terms Collaborative Knowledge Representation and Reasoning (CKRR) and Collaborative Machine Learning (CML) are used to describe CAI methodologies. When it comes to indexing massive data in a cloud computing context, the CAI and AI behave similarly.

The creation of indexes in NAI indexing methods is based on the objects that are often searched for in the database. For instance, data retrieval is performed in a sorted manner in tree-based indexing approaches, meeting the indexing techniques' reputation for their quick recall rate and storage cost minimization. Data is hashed to increase retrieval precision and decrease memory use. With the help of the feature fusion based hashing approach [67], it is possible to detect large-scale image copies in a huge dataset for big data in cloud computing by making use of the relationship between two feature models. This approach performs effectively in terms of data volume for large data requirements.

Unknown behavior in massive data is found using AI indexing techniques. The indexing methods rely on soft computing, knowledge representation and reasoning, and machine learning. An



http://dx.doi.org/10.36596/jcse.v3i2.548

unstructured dataset can have patterns between its components according to the indexing technique known as Latent Semantic Indexing (LSI), which is based on an artificial intelligence indexing approach. It may build associations between phrases with comparable contexts and extract the semantic meaning of a dataset. ML is an AI-based indexing technique that involves an iterative process of pattern observation, mathematical adjustment, and prediction rating [12]. An AI-based machine learning method is called "manifold learning." [68] suggested a concept of local subspace indexing for image search that enables quick query selection. The program might incorporate several learning algorithms to improve recognition performance.

An indexing strategy based on multi-agent or non-multi-agent systems benefits from the CAI indexing techniques in terms of accuracy and search time. The effectiveness of these strategies also depends on the collaborative power of the adopted method.

**7. Performance Evaluation of hashing techniques based on Mean Average Precision (MAP)**

MATLAB implementation that is freely accessible was used to evaluate some techniques reviewed in this article. The primary assessment metrics for large-scale image retrieval studies are mean average accuracy and precision-recall. Recall rate is a metric for determining and illustrating search accuracy. The ratio of accurately recovered images to the total number of images actually retrieved from a database is known as the recall rate. In this section, five cutting-edge hashing algorithms for high-dimensional closest neighbor search were evaluated for performance. Among the algorithms examined are:

DSH: Density sensitive hashing combines the advantages of data-dependent and data-independent hashing algorithms [80]. It is a semi-supervised based hashing approach. The forecasts are produced using basic concepts. This method creates projections using chosen principles rather than completely random selection [80]. It is a development of LSH.

LSH: Locality sensitive hashing is a hashing-based approach that generates projections independently of the distribution of the data. LSH makes projections at random. The vectors used to generate the projections are chosen at random from a p-stable distribution. It applies to the change detection domain and is an unsupervised approach [30]. SHD is a hashing-based approach that makes use of spherical Hamming distance. KLSH generalizes the Locality Sensitive Hashing to the Kernel Space [56], Shift-Invariant Kernel Hashing for estimating shift-invariant kernels [40]. The SIKH is based on random feature hashing.

## 7 Methods

This section looks at the geometric data features' magnitude structure. The image characteristics are indexed using the results of the quantized hashing. The experiment used a combined strategy that increases both search accuracy and precision while processing binary hash codes using five cutting-edge hashing algorithms. To cut down on storage and computational costs as well as to improve the precision and speed of queries, a dataset including sample data points was indexed. This work addresses the samples of the data points as $x_1, x_2, x_3, \dots, x_N$, where $X$ represents the database. $X = \{x_1, x_2, x_3, \dots x_n, \dots, x_N\} \in R^{d \times N}$ represents the data points contained in the database. In this case, $X$ is the database and R^(d × N) is the dimensional space of size N. Mapping of these data points to k-bit binary hash code is carried out by the hash function model in equation (25)

$$H(x) = \{h_1(x), \dots h_k(x)\} \in \{-1, 1\}^k \quad (25)$$

*Where length of the binary hash code is denoted by k..*

A similarity-preserving term was used to improve search accuracy in a dataset. With a constrained Hamming distance, the similarity preservation term refers to the similarities between the data points Q(y). Comparative geometric feature points of the two data samples' similarities are retrieved as $Q_{ij}$.

Equations (26) or (27) are used to balance the distribution of data points for each bit.



http://dx.doi.org/10.36596/jcse.v3i2.548

$$p_r[h_i(x_i) = 1] = \frac{1}{2}, x \in X, 1 \leq i \leq t \quad (26)$$

$$N_i = \sum_{i=1}^{2^M} N_i \quad (27)$$

Where $N_i$ is the number of training samples in the $i_{th}$ bucket and $M$ is the number of buckets. To achieve independence between two bits given that $x \in X$ and $1 \leq i < j \leq t$ where $i$ and $j$ are the $ith$ and j$th$ data points, and $t$ is the threshold, hash functions are design to be independent and the data points are distributed equally to each hash bucket as in equation (28).

$$p_r\left[h_i(x) = 1, h_j(x) = 1\right] = p_r[h_i(x) = 1] \,.\, p_r\left[h_j(x) = 1\right] = \frac{1}{2} \,.\, \frac{1}{2} = \frac{1}{4} \quad (28)$$

The next stage is to integrate the balanced partitioning sections with the similarity preserving term to speed up and boost search accuracy at the same time. For search precision and the least amount of information, we employ the similarity-preserving term $Q_{(y)}$. The joint optimization component is accountable for the concurrent optimization of search precision and search time, enabling high search precision with short search times. To make optimization easier, a linear function is parameterized and relaxed.

The search accuracy is improved by reducing the Hamming distance between comparable data pieces. Mathematically, this may be written as in equation (29):

$$Q(y) \sum_{i=1,\ldots N} x + \sum_{j=1,\ldots N} x \quad (29)$$

To improve search precision and speed at the same time, balanced partitioning and the similarity preserving term are coupled.

## 7.1 Experimental Results

The dataset was subjected to hash functions, which produced hash codes with lengths of 8, 16, 32, 48, 64, and 96 bits. The mean average accuracy for each technique that was tested using the SIFT 1M data sets is shown in Figure 1. As can be observed, the random projection methods perform poorly when the code length is small while improving as the code length rises in terms of mean average accuracy. When the code length is low, the learning-based spherical hamming distance performs quite well in comparison to MAP, but as the code length rises, no discernible improvement was made. When the code length is short, as seen in Figure 1, the Geo SPEBH reported a high MAP. The approach fails to effectively maintain the similarity between the binary codes and the original data points, which results in the behavior of the SHD. With the exception of the DSH method, the SHD performs better than every other algorithm when the code length is 96 bits. The date-independent based algorithms, on the other hand, perform well with lengthy binary codes due to their capacity to randomly create their hash table while paying close attention to the distribution of data points.



http://dx.doi.org/10.36596/jcse.v3i2.548

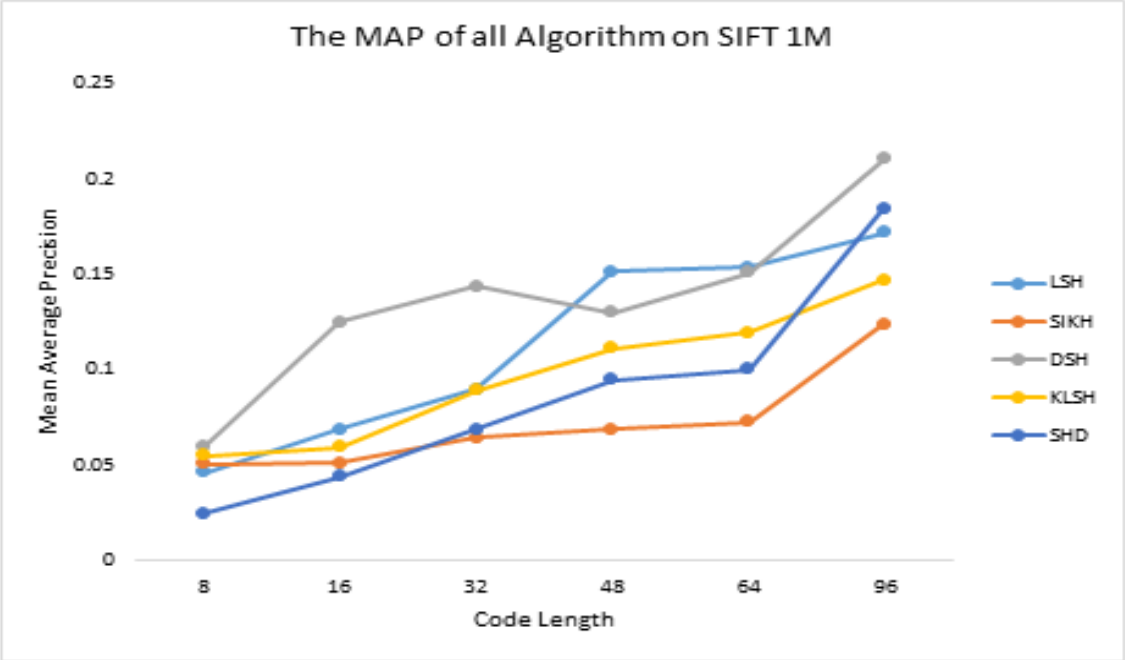


**Fig 1.** The MAP of all algorithms on SIFT IM

The DSH [71] initially quantified the data by dividing up the points into groups using the k-means method. To evaluate the accuracy of the provided quantization, sum of square error (SSE), which is the same as distortion, is expressed as the quantisation result represented as S.

The training and testing times of the approach reduce as the dimension lowers because fewer projections need to be made at random utilizing the geometric information in the data. Since DSH includes calculating each projection's entropy in relation to the entire database, it takes longer as the code length rises. The amount of bits necessary to implement each algorithm (i.e. 16, 32, 64, and 96 bits) is the storage requirement.

The DSH has three parameters: p, which determines the number of k-means iterations, r, which determines the number of r-adjacent groups, and, $\alpha$ which determines the number of groups. The default settings for 64-bit hash codes are p=3, $\alpha$ =1.5, and r=3. As the k-means iteration count varies, DSH's performance also changes. It is evident from the data that when the number of iterations rises, the MAP and the learning time of DSH do as well. An acceptable MAP is obtained after 3 iterations in k-means. Additionally, DSH's performance varies as the number of groups represented by $\alpha$ changes. As a result, the group number produced by k-means likewise alters. As the group size grows, the MAP and the learning time of DSH also increases as $\alpha$ =1.5, which offers a fair balancing when examining efficiency and accuracy of the scheme. Additionally, DSH performance varies with the number of r-adjacent groups, thus when r < 5, high performance was attained.

Because in a d-dimensional space, a near area must be defined by d+1 hyperplanes, d+1 hyperplane are necessary to define a closed area in a hyperplane. DSH continually produces more and more duplicated, less significant forecasts. DSH's performance suffers due to the redundant projections, making it difficult to scale with massive databases. For this, an optimized method is needed to maintain memory cost while balancing the trade-off between search accuracy and search time.

Additionally, DSH creates additional projections that are utilized to separate two distant groups as the number of r-adjacent groups rises. Making more and more forecasts becomes redundant and of less importance. The performance of DSH is negatively impacted by projection redundancy. With large datasets, DSH struggles to scale and ignores search time as a crucial hashing component.



http://dx.doi.org/10.36596/jcse.v3i2.548

Table 2 gives the description of the performance of each hashing technique based on mean average precision. The description in Table 2 helps to assess the effectiveness of algorithms based on MAP and storage evaluation of hashing approaches.

**Table 2.** Performance of hashing techniques based on Mean Average Precision

| **Method of hashing techniques** | ***Authors*** | ***MAP*** | | | ***Storage*** | | |
|---|---|---|---|---|---|---|---|
| | | ***Very Good*** | ***Good*** | ***Average*** | ***Very Good*** | ***Good*** | ***Average*** |
| SHD | **[34]** | | ✓ | | | ✓ | |
| LSH | **[30]** | | ✓ | | | ✓ | |
| DSH | **[80]** | ✓ | | | ✓ | | |
| KLSH | **[56]** | | ✓ | | | ✓ | |
| SIKH | **[40]** | | | ✓ | | | ✓ |

## 8. Discussion and future directions

In relation to large data in cloud computing, many indexing algorithms have been given and analyzed to highlight their merits and flaws.

Data Reduction: A proposed square kilometer array telescope now generates millions of terabytes of data, while petabytes of data are produced by scientific research and modeling results. The majority of these massive data sets are chaotic or unstructured. These noisy and unstructured data need to be repaired, well-organized, and formatted simply. The filters used to filter the experimental and simulation data are designed to prevent the loss of important data. Because of this, the science of data reduction turns into a murky field that has to be investigated by academics.

Data provenance is implied by the ability to comprehend information and move it through the analysis pipeline. In order to trace the connection and transport data provenance across data analysis pipelines, it is preferable that a data system be created.

Data description: Determining the quantity of the stored data and describing the type of data as it is being saved are crucial. To define and comprehend the stored data for these, it is necessary to automatically produce accurate metadata. Researchers should thus focus on developing data systems that can produce metadata and transmit that metadata through data analysis pipelines.

Data Integration: Raw data gathered from sensor devices, student records, health records, x-ray image data, graph data from mathematical and statistical analysis, photographs, and videos cannot be successfully analyzed. In order to collect information from the many sources, arrange it, and put it in a manner that can be analyzed, researchers urgently need to create a strategy.

Data Analysis and Mining: Data mining and analysis are necessary in order to make huge data relevant for decision-making. Relational databases are only partially capable of doing in-depth analyses of massive data in cloud computing. Therefore, it is advised that researchers create a productive big data analysis strategy.

To help users pick the best method for indexing big data in cloud computing, users should take into account search time, a crucial performance measure in assessing the speed necessary to retrieve information from a database. Additionally, while constructing and selecting an indexing technique, the amount of bandwidth needed to move data from source to destination must be taken into account.



http://dx.doi.org/10.36596/jcse.v3i2.548

## 9. Conclusion

The datasets used, as well as the organization, categorization, and comparison of big data storage, management, and indexing strategies, were reported in this study. This study also examines the performance evaluation of indexing strategies based on their categorization. Big data indexing requirements, including volume, velocity, variety, veracity, value, variability, and complexity, were the basis for the evaluation. The study's major goal is to analyze these big data indexing needs and define existing approaches in order to inform famous researchers about the fundamentals that will serve as a framework for developing optimized indexing strategies for specific platforms to support the veracity of big data. The report addressed issues with the approaches already in use and made suggestions on how to fix them (Table 1). The taxonomy includes NAI, AI, and CAI as the indexing methods that are currently available. Future research directions are presented in the debate. In order to assess the effectiveness of algorithms based on MAP, Precision-Recall, and storage, evaluation of hashing approaches was done.

## References

[1] Thilkanathan, Danan, S. C., Surya, N., Rafael, C., & Leila, A. (2014). A platform for monitoring and sharing of generic health data in the cloud. *Future generation computer system, 35*, 102-113. Retrieved April 9, 2017. https://doi.org/10.1016/j.future.2013.09.011

[2] Huang, Z., Heng, T. S., & Shao, J. (2010). Bounded Coordinate System Indexing for Real-time. *ACM Transactions on Information Systems, 10*(10), 1-32. https://doi.org/10.1145/1508850.1508855

[3] Gartner M, Rauber, A., & Berger, H. (2013). Briging structured and unstructured data via hybrid semantic search and interactive ontology-enhanced query formulation. *Knowledge information system*, 1-32. https://doi.org/10.1007/s10115-013-0678-y

[4] Armbrust, M., Fox , A., Griffith, R., Joseph, A. D., Katz, H. R., & Ko, A. (2009, 2). Above the clouds: *Berkeley view of cloud computing, Technical Report UCB/EECS*.Materials Genome initiative for Global Competitiveness. https://inst.eecs.berkeley.edu/~cs10/fa10/lec/20/2010-11-10-CS10-L20-AF-Cloud-Computing.pdf

[5] Agrawal, D., Bernstein, P., Bertino, E., Davidson, S.,& Dayal, U. (2012). Challenges and Opportunities with Big Data. *A white paper prepared for the Computing Community Consortium*, 1-16. https://cra.org/ccc/wp-content/uploads/sites/2/2015/05/bigdatawhitepaper.pdf

[6] Chang, c., Kayed, M., Girgis, M. R., & Shaalam, K. F. (2006). A survey of web information extraction system. *IEEE Transaction on Knowledge and Data Engineering, 18*(10), 1411-1428. https://doi.org/10.1109/TKDE.2006.152

[7] Agrawa, C. C., & Wang, H. (2010). Managing and mining graph data. *Springer publishing company*. https://doi.org/10.1007/978-1-4419-6045-0

[8] Hosagrahar, V. J., Beng, C., Kian-Lee, T., Cui, Y., & Rui, Z. (2005). iDistance: An adaptive B+-tree based indexing method for nearest neighbour search. *ACM Transaction on Database Systems, 30*(2), 364-397. https://doi.org/10.1145/1071610.1071612

[9] Muja, M., & Lowe, D. G. (2009). Fast approximate nearest neighbours with automatic algorithm configuration. *VISAPP*, 331-340.

[10] Jon, L. B. (1975). Multidimensional Binary Search Trees Used for Associative Searching. *ACM, 18*(9), 509-517. https://doi.org/10.1145/361002.361007

[11] Chanop, S.-A., & Richard, H. (2008). Optimised KD-trees for fast image descriptor matching. *IEEE Conference on Computer Vision and Pattern Recognition*, 1-8. https://doi.org/10.1109/CVPR.2008.4587638

[12] Shamshirband, S., Anuar, N., Kiah, M., & Patel, A. (2013). An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique. *Engineering Application of Artificial Intelligent. , 26*(8), 2105-2127. https://doi.org/10.1016/j.engappai.2013.04.010

[13] Aguilera M, K., Golab, W., & Shah, M. A. (2008). A practical scalable distributed b-tree. *Proceedings of the VLDB Endowment, 1*(1), 598–609. https://doi.org/10.14778/1453856.1453922

[14] Jaluta, I. (2014, April). Transaction management in b-tree-indexed database systems. *In Information Science, Electronics and Electrical Engineering. International conference on, 3*, 1968-1975. https://doi.org/10.1109/InfoSEEE.2014.6946267

[15] Frome, A., Singer, Y., Sha, F., & Malik, J. (2008, June). Learning globally-consistent local. *ICCV*. https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/shape/papers/FromeSingerShaMalikICCV07.pdf



http://dx.doi.org/10.36596/jcse.v3i2.548

[16] Friedman, J., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transaction on Mathematical Software, 3*(3), 209-226. https://doi.org/10.1145/355744.355745
[17] Kai, Z. Y., Nicholas, J. Y., & Shuo, S. (2013). Discovering of gathering patterns from trajectories. *ICDE*. https://doi.org/10.1109/ICDE.2013.6544829
[18] Hoyoung, J., Man, L. Y., Xiaofang, Z., Christian, S. J.,& Heng, T. S. (2008). Discovery of convoys in trajectory databases. *VLDBJ*. https://doi.org/10.14778/1453856.1453971
[19] Yu, Y., Zhu, Y., Ng, W., & Samsudin, J. (2014, 12). An efficient multidimension metadata index and search system for cloud data," in Cloud Computing technology and science. *IEEE Transaction on*, 499-504. https://doi.org/10.1109/CloudCom.2014.88
[20] Dieter, P., Christian, S. J., & Yanmis, T. (2000). Novel approaches to the indexing of moving objects trajectories. *VDLB*. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.3875&rep=rep1&type=pdf
[21] Lei, C., Tammer, M. O., & Vincent, O. (2005). Robust and fast similarity search for moving object trajectories. *ICDE*. https://doi.org/10.1145/1066157.1066213
[22] Michail, V., George, K., & Dimitrios. (2002). Discovering similar multidimensional trajectories. *ICDE*. https://doi.org/10.1109/ICDE.2002.994784
[23] Prateek Jain, B. K. (2008, 6). Fast image search for learned metrics. *In proceeding of the IEEE conference on computer vision and pattern recognition*. https://doi.org/10.1109/CVPR.2008.4587841
[24] Xu, H., Wang, J., Li, Z., Zeng, G., Li, S., & Yu, N. (2011). Complementary hashing for approximate nearest neighbor search. *In Proc. ICCV*. https://doi.org/10.1109/ICCV.2011.6126424
[25] Kulis, B., Jain, P., & Grauman, K. (2009). Fast similarity search for learned metrics. *TPAMI, 31*(12), 2143–2157. https://doi.org/10.1109/TPAMI.2009.151
[26] Torralba, A., Fergus, R., & Freeman, W. T. (2008). 80 million tiny images: a large dataset for non-parametric object and scene recognition. *TPAMI, 30*(11), 1958–1970,. https://doi.org/10.1109/TPAMI.2008.128
[27] Strecha, C, A. M., M, M. B., & P, F. (2012). Ldahash: Improved matching with smaller descriptors. *TPAMI, 34*(1), 66-76. https://doi.org/10.1109/TPAMI.2011.103
[28] Zhu, X., Huang, Z., Cheng, H., Cui, J., & Shen, H. T. (2013). Sparse hashing for fast multimedia search. *ACM Transaction on information system, 3*(2), 1-24. https://doi.org/10.1145/2457465.2457469
[29] Avidan, S., & Korman, S. (2011). Coherency sensitive hashing. *In Proceedings of ICCV*. https://doi.org/10.1109/ICCV.2011.6126421
[30] Datar, M., Immorlica, N., Indyk, P., & Mirrokni, P. (2004). Locality sensitive hashing scheme based on p-stable distributions. *In Proceedings of the Symposium on Computational Geometry*, 253–262. https://doi.org/10.1145/997817.997857
[31] Zhou, A. (2005). c^2: a new overlay network based on can and chord. *international journal of high performance computing network, 3*(4), 248-261. https://doi.org/10.1007/978-3-540-24679-4_15
[32] W. Liu, J. Wang, R. J. Y-G. Jang, S-F Chang., (2012). Supervised hashing with kernels. *In computer vision and pattern recognition*. https://doi.org/10.1109/CVPR.2012.6247912
[33] A. Jolly & O. Buisson. (2011). Random maximum margin hashing. *CVPR*. https://doi.org/10.1109/CVPR.2011.5995709
[34] H. Jae-Pil, L. Youngwoon, H. Junfeng, C. Shih-Fu, Y. Sung_Eui, (2015). Spherical Hashing: Binary Code Embedding with Hyperspheres. *IEEE transaction on Pattern Analysis and Machine Intelligent*, 1-14. https://doi.org/10.1109/TPAMI.2015.2408363
[35] J. He, R. Rhadhakrishnan, S-F Chang and C. Bauer. (2011). Compact hashing with joint optimisation of search accuracy and time. *CVPR*. https://doi.org/10.1109/CVPR.2011.5995518
[36] Y. Gong and S. Lazebnik. (2011). Itetrative Quantisation: a procrustean approach to learning binary codes for large-scale image retrieval. *IEEE transaction on pattern analysis and machine inteliigence*. https://doi.org/10.1109/TPAMI.2012.193
[37] A. Torralba, R. fergus, and Y. Weiss, (2008). Small codes and large image dtatabases for recognition. *CVPR*. https://doi.org/10.1109/CVPR.2008.4587633
[38] Y. Weiss, A. Torralba, and R. fergus, . (2008). Spectral Hashing. *in proceedings of NIPS*.
[39] O. Chum, J. Philbin, A. Zisseman, (2008). Near duplicate image detectionmin-hash and tf-idf weighting. *BMVC*.
[40] M. Rangisky and S. Lazebnik. (2009). Locality sensitive binary codes from shift-invariant kernels. *in proceedings od NIPS*, 1509-1517.
[41] R. Salakhutdinov, G. Hinton, (2009). Semantic Hashing. *International Journal of Approximate reasoning*. https://doi.org/10.1016/j.ijar.2008.11.006
[42] Boukari Souley, A. U. Othman (2019). Geometric Similarity Preserving Embedding-Based Hashing for Bid Data in Clou Computing. *International Journal of research and Scientific Innovation*.



http://dx.doi.org/10.36596/jcse.v3i2.548

[43] J. Wang, S. Kumar, S-F Chang,. (2010). Sequential projection learning for hashing with compact codes. *ICML*.
[44] L. Pauleve, H. Jegou, L. Amsaleg, (2010). Locality sensitive Hashing: A comparison of hash function types and queryong mechanism. *Pattern recognition Letters*. https://doi.org/10.1016/j.patrec.2010.04.004
[45] J. Wang, S. Kumar, and S-F Chang, (2010). Semi-supervised hashing for scalable image retrieval. *CVPR*. https://doi.org/10.1109/CVPR.2010.5539994
[46] R-S Lin, D. Rose, J. Yangik, (2010). Spec Hashing: Similarity preserving algorithm for entropy-base coding. *CVPR*. https://doi.org/10.1109/CVPR.2010.5540129
[47] R. Ye, Z. Li, (2016). Compact structure hashing via sparse and similarity preserving embedding. *IEEE transaction on cybernatics, 46*(3), 718-729. https://doi.org/10.1109/TCYB.2015.2414299
[48] H. Zhang, L. Liu, Y. Yong, L. Shao, (2017). Unsupervised deep hashing with pseudo labels for scalable image retrieeval. https://doi.org/10.1109/TIP. 2017. 2781422
[49] Y. Lv, W. Y. Ng Wing, Z. Zeng, S. D. Yeung,and P. K. Patrick (2015). Asymmetric Cyslical Hashing for Large Scale Image Retrieval. *IEEE transaction on multimedia, 11*(8), 1225-1235. https://doi.org/10.1109/TMM.2015.2437712
[50] M. Norouzi and D. J. Fleet. (2011). Minimal Hashing for Compact binary codes. *ICML*.
[51] Kadiyala S, S. N. (2008). A compact multi-resolution indedx for variable length queries in time series database. *Knowledge information system, 15*(2), 131-147. https://doi.org/10.1007/s10115-007-0097-z
[52] Meshram, B. B., & Gaikwad, G. P. (2013, 4). Different indexing techniques. *International Journal of Engineering Research and Application, 3*(2), 1230-1235.
[53] Chen, J., Yuegue, C., Lia, E., Cuiping, I. L., & Jiaheng, U. L. (2013). Big Data Challenges: A data Management Perspective. *Higher education press and springer verlag Berlin Heidelberg, 7*(2), 157-164. http://dx.doi.org/10.1007/s11704-013-3903-7
[54] Kaisler, S.,Armour, F., Espinosa , J. A., & Money, W. (2013). Big data: issues and challenges moving forward. *Hawii international conference on system sciences*, 995-1004. http://dx.doi.org/10.1109/HICSS.2013.645
[55] M. S. Charkar, (2002). Similarity estimation techniques from rounding algorithms. *In Proceedings of Annual ACM Symposium on Theory of Computation*, 380-388. https://doi.org/10.1145/509907.509965
[56] B. Kulis, K. Grauman, (2009, September-October). Kernelised Locality-sensitive hashing for scalable image search. *In Proceedings of IEEE conference on computer vision and pattern recognition*, 2130-2137. http://dx.doi.org/10.1109/ICCV.2009.5459466
[57] B. Souley, A. U. Othman, A. Y. Gital and I. M. Adamu, (2019). Performance evaluation of GSPEBH for big data in cloud computing. *Global Scientific Journal*.
[58] C. Yan, H. Xie, D. Yang, J. Yin, Y. Zhang, (2018). Supervised hash coding with deep neural network for environment perception of intelligent vehicles. *IEEE transaction on intelligent transportation systems, 19*(1), 284-295. https://doi.org/10.1109/TITS.2017.2749965
[59] M. He, Y. Yang, F. Shen, N. Xie, and H. T. Shen, (2017). Hashing with Angular Reconstruction Embeddings. *IEEE Transactions on Image Processing*. 27(5), 545-555. https://doi.org/10.1109/TIP.2017.2749147
[60] Nussinov, R., & Wolfson, H. J. (1991, 12 01). Efficient Detection of three-Dimensional Structural Motifs in Biological Macromolecules by Computer Vision techniques. *Peoceedings of the National Academy of Science America, 88*(23), 10495-10499. https://doi.org/10.1073%2Fpnas.88.23.10495
[61] Mehrotra, H., Majhi, B., & Gupta, P. (2010). Robust ris indexing scheme using geometric hashing of SIFT keypoints. *Journal of Network and Computer Applications, 33*, 300–313. https://doi.org/10.1016/j.jnca.2009.12.005
[62] Lowe, D. (2004). Distinctive image features from scale-invariant key points. *International Journal of Computer Vision* 60, 91–110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94.
[63] Jayaraman, U., Surya, P., & Phalguni, G. (2013). Use of geometric features of principal components for indexing a biometric database. *Mathematical aand Computing Modelling, 58*, 147-164. https://doi.org/10.1016/j.mcm.2012.06.005
[64] Li, F., Yi, K., & Le, W. (2010). Top-k queries on temporal data. *VLDB, 19*(5), 715-733. http://dx.doi.org/10.1007/s00778-010-0186-6
[65] Sandu P, I., Zeitouni, K., Oria, V., Barth , D., & Vial, S. (2011). Indexing in network trajectory flows. *VLDBJ, 20*(5), 643-669. https://doi.org/10.1007/s00778-011-0236-8
[66] Dittrichter. (2011). MOVIES: Indexing moving objects by shooting index images. *Geoinformatics*, 15(4), 727-767. http://dx.doi.org/10.1007/s10707-011-0122-y
[67] Xie, M., Wang, H., Yin, J., & Meng, X. (2007). Integrity auditing of outsourced data. *In Proceedings of the International Conference on Very Large Databases*, 782-793.



http://dx.doi.org/10.36596/jcse.v3i2.548

[68] Vandana, D. K., Jayaraman, U., Amit, K., Aman, K. G., & Gupta, P. (2013). An efficient indexing scheme for face database using modified geometric hashing. *Neurocomputing, 116*, 208-221. https://doi.org/10.1016/j.neucom.2011.12.056
[69] Umarani, J., Surya, P., & Phalguni, G. (2013). Use of geimetric features of principal components for indexing a biometric database. *Mathematical and computing modelling, 58*, 147-164. https://doi.org/10.1016/j.mcm.2012.06.005
[70] Ling-Yin, Y.-T. H.-C.-C. (2013). Indexing spatial data in cloud data management. *Pervasive mobile computing*, 1-14. https://doi.org/10.1016/j.pmcj.2013.07.001
[71] Xiaohui, Yu, K. Q., & Nick, K. (2005). Monitoring K-nearest neighbour queries over moving objects. *In Proceedings of the 21st International Conference on Data Engineering*. https://doi.org/10.1109/ICDE.2005.92
[72] Wang, M., Viliam, H., John, M., & Patrick, O. (2013, 2). High volumes of event stream indexing and efficient multi-keyword searching for cloud monitoring. *Future generation computer, 29*, 1943-1962. https://doi.org/10.1016/j.future.2013.04.028
[73] Wang, J., Wu, S., Gao, H., Li, J., & Ooi, B. C. (2010). Indexing Multidimensional Data in a Cloud System. *ACM SIGMOD International conference on management of data*, 591-602. https://doi.org/10.1145/1807167.1807232
[74] Spek, P. V., & Steven, K. (2011). Applying a dynamic threshold to improve cluster detection LSI. *Science of computer programming, 76*, 1261-1274. https://doi.org/10.1016/j.scico.2010.12.004
[75] James, C., Yiping, K., Ada, W.-C. F., & Jeffrey, X. Y. (2011). Fast grapbh query processing with low-cost index. *The VLDB journal, 20*(4), 521-539. https://doi.org/10.1007/s00778-010-0212-8
[76] Giangreco, I., Kabary, I. A., & Schuldt, H. (2014, 06). Adam - a database and information retrieval system for big multimedia collections. *IEEE International Conference on*, 406-413. https://doi.org/10.1109/BigData.Congress.2014.66
[77] Collins, E. (2014). Intersection of the cloud and big data. *IEE Cloud Computing 1*, 84-85. http://dx.doi.org/10.1109/MCC.2014.12
[78] Cackett, D. (2013). Information Management and Big data: A reference Architecture. *Oracle corperation*.
[79] Wook-shin, H., Jinsoo, L., & Minh-Duc, P. (2010). iGraph: A framework for comparing a disk-based grapph indexing techniques. *Proceedings of the VCLD endowment, 3*(1).
[80] Jin Z, L. C., Lin, Y., & Cai, D. (2014, august). Density Sensitive Hashing. *IEEE transactions on Cybernetics, 44*(8), 1362-1371. https://doi.org/10.1109/TCYB.2013.2283497
[81] Ferragina, P., & Rossano, V. (2007, 7). The ompressed permuterm index. *In proceedings of SIGIR*, 535-542. https://doi.org/10.1145/1868237.1868248
[82] JJinbao, W., Wu, S., Gao, J., Li, J., & Ooi, C. B. (2010). Indexing multi-dimensional data in a cloud system. *SIGMOD*. https://doi.org/10.1145/1807167.1807232



http://dx.doi.org/10.36596/jcse.v3i2.548