AL-Bahir
Journal for Engineering and Pure Sciences
BJEPS

Volume 1 | Issue 1 Article 4

Evaluation of Geo-SPEBH algorithm based on Bandwidth for Big Data retrieval in Cloud Computing

Abubakar Usman Othman
Department of Mathematical Sciences, Faculty of Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

Moses Timothy
Department of Computer Science, Faculty of Computing, Federal University, Lafia, Nigeria

Aisha Yahaya Umar
Department of Computer Science, Faculty of Science, Gombe State University, Gombe, Nigeria

Abdullahi Salihu Audu
Department of Computer Science, Faculty of Natural and Applied Sciences, Nasarawa State University Keffi, Nigeria

Boukari Souley
Department of Mathematical Sciences, Faculty of Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

See next page for additional authors

Recommended Citation
Othman, Abubakar Usman; Timothy, Moses; Umar, Aisha Yahaya; Audu, Abdullahi Salihu; Souley, Boukari; and Gital, Abdulsalam Ya'u (2022) "Evaluation of Geo-SPEBH algorithm based on Bandwidth for Big Data retrieval in Cloud Computing," *Al-Bahir*: Vol. 1: Iss. 1, Article 4.
Available at: https://doi.org/10.55810/2313-0083.1005

ORIGINAL STUDY

Evaluation of Geo-SPEBH Algorithm Based on Bandwidth for Big Data Retrieval in Cloud Computing

Abubakar U. Othman [a], Moses Timothy [b,*], Aisha Y. Umar [c], Abdullahi S. Audu [d], Boukari Souley [a], Abdulsalam Y. Gital [a]

[a] Department of Mathematical Sciences, Faculty of Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria
[b] Department of Computer Science, Faculty of Computing, Federal University, Lafia, Nigeria
[c] Department of Computer Science, Faculty of Science, Gombe State University, Gombe, Nigeria
[d] Department of Computer Science, Faculty of Natural and Applied Sciences, Nasarawa State University Keffi, Nigeria

Abstract

The fast increase in volume and speed of information created by mobile devices, along with the availability of web-based applications, has considerably contributed to the massive collection of data. Approximate Nearest Neighbor (ANN) is essential in big size databases for comparison search to offer the nearest neighbor of a given query in the field of computer vision and pattern recognition. Many hashing algorithms have been developed to improve data management and retrieval accuracy in huge databases. However, none of these algorithms took bandwidth into consideration, which is a significant aspect in information retrieval and pattern recognition. As a result, our work created a Geo-SPEBH algorithm to solve this basic gap. The paper then assesses the performance of the Geo-SPEBH algorithm in terms of bandwidth in a distributed computing environment. Geo-performance SPEBH's was compared to existing state-of-the-art approaches using a network analyzer called Wire shark. The simulation results reveal that during retrieval, the same kb/sec of data is carried from source to destination and from destination to user. When the coding length is 8bit, the findings show that 0.091 kb/s of data is required to transport data from source to destination. Each algorithm with the same bit code requires the same amount of bandwidth to convey data.

Keywords: Balance partitioning, Bandwidth, Cloud computing, Indexing, Information retrieval, Geo-SPEBH

1. Introduction

Cloud computing is a new technology that offers a framework for dealing with complicated data applications that demand high-performance applications. As a consequence of advancements in mobile industry, devices such as smart phones and tablets are being utilized for a variety of applications [1−4]. The accessibility of the online, for example, broadband Internet connection [5], paired with mobile devices, has made digital information collecting simpler. Sufficient and legal management of this data is predicted to lead to new discoveries and understanding about markets, social orders, human behavior, and the environment [6,7]. Significant information has been extracted from raw data using information extraction algorithms [8,9]. Astronomers have lately used the Sloan digital sky survey conducted in the past as a database [10,11]. Many hashing algorithms have been developed to improve data management and retrieval accuracy in huge databases. However, none of these algorithms took bandwidth into consideration, which is a significant aspect in information retrieval and pattern recognition. The Geo-SPEBH, like many other hashing algorithms, is an improved indexing method designed for efficient data storage and management in distributed computing applications. This paper therefore, intend to evaluate Geo-SPEBH algorithm with other algorithms based

Received 29 July 2022; revised 23 August 2022; accepted 24 August 2022.
Available online 16 September 2022

* Corresponding author.
E-mail address: moses.timothy@science.fulafia.edu.ng (M. Timothy).

https://doi.org/10.55810/2313-0083.1005

on bandwidth for big data retrieval in cloud computing.

Given that bandwidth is one of the most valuable assets, particularly in multimedia and distributed computing applications, its absence may result in significant loss in Quality of Service. The amount of bandwidth required in a multimedia application is often influenced by the program demand, the user's workload, the user's location, and the type of device being used [12]. These factors vary fast and may cause bandwidth shortages at times. Having enough bandwidth to foresee these issues is consequently crucial. This paper therefore, compares the performance of hashing base algorithms to the geometric similarity embedding-based hashing algorithm (Geo-SPEBH) in terms of bandwidth requirements and how much data is required for information transmission from source to destination using wire shark network analyzer in kb/sec. The fundamental idea is to long-hash the codes of these improved algorithms.

Recent researches have attempted to provide solutions that will improve data management and retrieval accuracy in huge databases [13]. explored the virtualization overhead in relation to network performance in Virtual Machine (VM) based cloud platforms to identify the performance bottleneck and idiosyncrasies of these platforms to determine the cause. As the contextual investigation, the Xen hypervisor-based cloud Amazon EC2 was used, and wide estimations on the network performance of its massive cases were performed. Estimated results reveal that network performance bottlenecks and idiosyncrasies might be prevalent and significant within a comparable server farm [14]. evaluate the wide-band performance of multiple dipole antenna matching networks. The bandwidth of correlation and matching efficiency represents the performance of matching networks, which are extensions of the single-antenna transfer speed idea to multiple antenna systems. Similarly, the effect of propagation circumstances on matching and bandwidth was investigated [15]. developed Hy-bridORAM, a useful Oblivious RAM (ORAM) with constant transfer speed that can be used in a broad range of applications. HybridORAM combines the best features of layer and tree ORAMs by combining frequency-accessed tiny levels of the former to increase reaction time with minor shuffles of the latter to conserve store capacity [16]. developed a novel geometric design that eliminated edge user inter-cell interference (ICI) that occurs in overlapping zones between neighboring cells to improve edge user throughput based on the combined optimization of power and bandwidth [17]. performed an end-user research of cloud-based gaming services, polled players on subjective quality engagement, and assessed their in-game execution. They also ran an experiment to assess the network properties of cloud-based gaming services. The experiment findings reveal that when latency grows, so does the quality of experience and player execution, although latency has no influence on the frame rate or average throughput of cloud-based gaming services.

Approximate Nearest Neighbor (ANN) gives assistance in enhancing search velocity and is frequently required [18,19]. Data-independent [20,21] and data-dependent [22−26] are examples [27]. created a supervised FastHash method that consists of a two-step learning technique based on binary code derivation, followed by binary classification based on an ensemble of decision trees. However, the writers overlooked bandwidth as a tool accountable for the amount of data transmitted from source to destination.

[28] developed a hashing approach that uses two hash codes of varying lengths as searches for stored photos in the database. Compact hash code was employed in this strategy to save storage costs. Long hash code was also used for queries to improve search precision. To retrieve photographs from the database, the query is processed using the Hamming distance of the long hash code and the cyclical concatenation of the compact hash code of the stored images for a higher accuracy recall rate. Iterative Quantisation (ITQ), Iterative Quantisation with Random Fourier Feature (ITQRFF), and Shift-invariant Kernel based Locality Sensitive Hashing (SKLSH) are compared to the suggested technique. This technique outperforms ITQFF on the basis that the ACH's asymmetric hashing approach provides more exact location data for the request. The exploratory results demonstrate that ACH outperforms the existing techniques in terms of accuracy with a code length of 64 bits. The disadvantage of ACH is that it did not consider bandwidth as a tool for determining the quantity of data transported from source to destination.

[29] presented a method for preserving the underlying geo-metric information in data. To learn compact hash codes, the authors study the sparse reconstructive connection of data. It mostly gets past fitting in estimating the experimental exactness on named data since the information provided by each bit is used to obtain the necessary features of hash codes. To get the goal function, the information

theoretic constraint is fused into the relaxed empirical fitness as a regularizing term. Equations (1) and (2) provide the empirical fitness and objective function, respectively.

$$J(W)=\frac{1}{2}Tr\{W^T X_l T X_l^T W\} \tag{1}$$

$$J_l(W)=\frac{1}{2}Tr\{W^T X_l T X_l^T W\}+\frac{\lambda}{2}Tr\{W^T X X^T W\} \tag{2}$$

where $Tr\{W^T X_l T X_l^T W\}$ is the information theory term.

To learn the weighted matrix W to build the hash function as in Equation (2), the hashing technique employs sequential learning to maximize the goal function in Equation (1).

$$X(X_l)=sign(W^T X_l) \tag{3}$$

By minimizing the goal function in Equation (3), the weight matrix W and the sparse weight matrix optimally learn as in Equation (4). However, the author did not take bandwidth into account.

$$(W,S)=\underset{W,S}{\arg min} J_2(W,S) \tag{4}$$

[30] established a unique hypersphere-based hashing function to transfer more spatially coherent data points into a binary hash code using a new binary code distance function, the spherical Hamming distance applicable to the hypersphere-based coding scheme. The main disadvantage of this strategy is that a tremendous amount of bandwidth will be required to transmit information from source to destination.

Hypersphere-based hashing functions are used in the proposed Geo-SPEBH to encode proximity areas in high-dimensional domains. As the code length increases, the use of hypersphere enhances search precision performance. Geo-SPEBH will increase as the code length increases to provide better performance. Geo-SPEBH makes use of the mathematical qualities of the primary component of features, which have been proven to be extremely discriminative, to ensure that fewer features are entered into the hash table.

[31] presented a BGAN, which is a Binary Generative Adversarial Network with the sole intention to retrieve image. BGAN was developed to overcome the issues binary codes generation without relaxing and to equip the representation of binary with the capacity of accurate retrieval of image. They used an unsupervised BGAN to insert images into binary codes. The BGAN may construct an image that looks similar to the preserved one by limiting the input noise variable of the Generative Adversarial Network (GAN) to binary and adapting it to the highlights of each input image. Another sign-activation approach and loss function were also presented to drive the learning process, which includes neighbourhood structure loss, content loss, and the novel models for adversarial loss. To extract features for the encoder, the author proposes a structure of five maximum convolution pooling layers and five sets of convolution layers.

[32] created a hashing-based estimator for kernel density in high dimensions. They investigate the challenge of creating its data structure using a collection of data P and a kernel function that approximates the kernel density of an inquiry point in sub-linear time. They provide a class of unbiased estimators for bounding their variance. The resulting estimator generates competent data structures for estimating the kernel density in high dimensions for a variety of regularly used kernels. Their work is particularly notable in the development of data structures that enhance basic random sampling in high dimensions.

The difficulties with these approaches are that bandwidth, which is a critical necessity in determining the quantity of data transported from source to destination, is not taken into account. As a result, we compare the performance of hashing algorithms based on bandwidth requirements to estimate the amount of bandwidth required to transport data using the wire shark network analyzer in kb/sec. This paper is organized into five sections. Section one is the introduction that captures problem definition and the need to evaluate hashing algorithms based on bandwidth utilization in cloud. The second section explains methods adopted in evaluation of Geo-SPEBH algorithms with other algorithms to check bandwidth usage. Section three shows the results obtained from the evaluation while section four discussed the results and section five is the conclusion.

2. Materials and methods

The suggested framework is made up of four sections, each of which serves a distinct purpose in achieving the goals. The goal of learning hashing-based techniques is to use the mapping function $h(x)$ to project an m-dimensional real valued feature vector to an n-dimensional binary hash code while maintaining the feature vector's and dataset's similarity. The suggested technique can conserve the hidden discriminative geometric information among the data points. The framework investigates the magnitude structure of geometric data features. The quantised hashing results are used to index the image features. The Geo-SPEBH employs a hypersphere-based hashing function for processing binary hash codes with a joint method that simultaneously improves search precision and search speed. A dataset including sample data points will be indexed to decrease cost of storage, computing expenses, and to simplify query precision and speed. This work addresses the samples of the data points as $x_1, x_2, x_3, \ldots, x_N$, where X represents the database. $X = \{x_1, x_2, x_3, \ldots x_n, \ldots, x_N\} \in R^{d \times N}$ 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 (5)

$$H(x) = \{h_1(x), \ldots h_k(x)\} \in \{-1, 1\}^k \tag{5}$$

where length of the binary hash code is denoted by k..

2.1. Similarity preserving term Q(y)

To increase the precision of searches in a dataset, a similarity preserving term was applied. In Equation (11) of [34], the similarity preservation term comprises the similarity characteristics among the data points Q(y), with a restricted Hamming distance. This section of the suggested framework is responsible for retaining the commonalities of two sample data points in the created framework's training dataset. The training set contains two data samples X_i and X_j from a dataset X. The similarity between the two data samples is extracted as Q_{ij} from comparative geometric feature points of image data. Geometric coordinate qualities are required for similarity preservation in hashing techniques. Following that, similar data points are assured to have similar binary hash codes with minor hamming distance.

2.2. Balance partitioning for independence

To ensure that data points are distributed uniformly in the hash container, we make each hash function independent of the others. That is, the functionality of one hash function does not rely on the operation of the other. This is because each hash function relies on itself to distribute data points evenly among distinct hash codes. As a result, because binary digits are addressed by zeros (0's) and ones (1's), each hash function is given the opportunity to become a 0 or a 1. This means that in order for hash functions to be free, each hash function must have the option of being one or zero, and the various binary hash codes must be independent of one another, as shown in Equation (4) above. The freedom of hash functions is demonstrated in the following scenario: In a normal setting, the likelihood that an event B_i will be a hash function is one (1). B_i is the event that $h_i(x) = 1$. Then define two occurrences B_i and B_j as independent if and only if the probability of $B_i = 1$ and the probability of $B_j = 1$ are similar to the probability of $B_i = 1$ multiplied by the probability of $B_j = 1$ as in Equation (10). By establishing each bit's independence, related bits are mapped into the same bucket with a high likelihood of having an equal chance of becoming one (1). To balance the partitioning of data points for each bit, one of Equation (6) or (7) is employed.

$$p_r[h_i(x_i) = 1] = \frac{1}{2}, x \in X, 1 \le i \le t \tag{6}$$

$$N_i = \sum_{i=1}^{2^M} N_i \tag{7}$$

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 $1 \le i < j \le t$ 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 (8).

$$p_r\left[h_i(x) = 1, \; h_j(x) = 1\right] = p_r[h_i(x) = 1] \, . \, p_r\left[h_j(x) = 1\right] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \tag{8}$$

The intersection represents the equal chance of the code bit having a binary hash code 1.

The next step is to combine the similarity preserving term with the balancing partitioning sections to increase search accuracy and time at the same time. In Equation (6), we embed the data points into each container.

2.3. Joint optimization

We use the similarity preserving term $Q_{(y)}$ for search accuracy and the least information criteria in this section. The suggested framework's joint optimization component is constructed and is answerable for the simultaneous optimization of search accuracy and search time, allowing for high search precision with short search time. A linear function is parameterized and relaxed to facilitate optimization.

The joint optimization is in charge of processing the hash bit that will be used for the query and identifying the container with comparable hash bits to the query, as well as regulating the loading of data samples from the selected containers into memory. The hash function autonomous is designed in this case to be free to distribute data points evenly across multiple binary hash codes. To keep the temporal complexity to a minimum, each container will have an equal amount of samples in order to have a balanced bucket (container). This is done to cut down on search time. To have an equal amount of samples in each bucket in order to balance the buckets $N = \frac{N}{2^M}$ [36], use equation (7).

By minimizing the Hamming distance between comparable data points, the search accuracy is enhanced.

Q(y) = sum of samples of x from i, for i = 1 to N + sum of samples of x from j, taking values from 1 to N.

This may be stated mathematically as:

$$Q(y) \sum_{i=1,\ldots N} x + \sum_{j=1,\ldots N} x \tag{9}$$

The similarity preserving term and balanced partitioning are combined to increase search precision and time at the same time [35]

The joint optimization algorithm for search exactness and time is introduced in Algorithm 1.

Algorithm 1: Joint Optimization

```
1.  Start
2.  Input: the training dataset$X_i$, $i = 1,2,3,\ldots,N$, similarity matrix $W$ and $W = W_{ij}$; the
    number of required bits $K$ to map the full dataset as hash codes; BP; N; M;
3.  Initialise: Sum = 0; Sim = 0; SimM = 0; BP = 0; V = 2**M; yi = 0; JointO = 0//jointO is
    the memory location for joint optimisation
4.              $for\ i = 1\ to\ c$
5.                  $for\ j = 1\ to\ c$
6.                  Get y($i$), y($j$), x($i,j$)
7.  Sum = Sum + $(y(i) - y(j))$**2
8.                  j = j + 1
9.                 $if\ j \le c$ goto step 6
10.                 end if
11.                 $i = i + 1$
12.                  $if\ i \le c$ goto step 17
13.                      end if
14.                  end for
15.             end for
16. Sim = Sum
17.                break;
18.                 $for\ i = 1\ to\ V$
19.                  get $N(i)$
20.                  BP = N($i$) $**$ 2
21.                  $i = i + 1$
22.                  $if\ i \le V$ goto step 40
23.                    end if
24.                 end for
25. Print Sim, BP
26. //Incorporating similarity preserving term and balanced partitioning//
27. JointO = Sim + BP
28. //computing $u_i$//
29. $T(a,b)$ = 0, swap = 0
30. Get x
31. Get b
32.                 $for\ i = 1\ to\ a$
33.                  $for\ j = i + 1\ to\ b$
34.                    Get $T(i,j)$
35.                    j = j + 1
36.                   $if\ j \le b$ goto step 55
37.                   i = i + 1
38.                   $if\ i \le a$ goto step 55
39.                         end if
40.                       end if
41.                     end for
42.                 end for
43.        $for\ i = 1\ to\ a$
44.            $for\ i = 1\ to\ b$
45. Swap = $T(i,j)$
46. $T(i,j) = T(j,i)$
47. $T(j,i) = swap$
48. $h(i)$ = $sign(T(j,i) * x(i) - b$//T is the projection matrix of $d \times M$ and $b$ is a vector//
49.            $j = j + 1$
50.               $if\ j \le b$ goto step 45
51.           $i = i + 1$
52.               $if\ i \le a$ goto step 44
53.                         end if
54.                       end if
55.                     end for
56.                 end for
57. for i = 1
58. Print h(i)
59. $i = i + 1$
60.           $if\ i \le a$ goto step 78
61.            end if
62. end for
63. Stop
```

2.4. Metrics of performance

Bandwidth is the metric used to evaluate the suggested approach. Using the wire shark network analyzer, cutting-edge methodologies were compared with Geo-SPBH to determine the network bandwidth required to transport data from source to destination. This is done to evaluate network bandwidth performance using the SIFT 1B dataset as obtained from http://corpus-texmex.irisa.fr/. The Bandwidth metric quantifies the amount of network bandwidth required to transfer data from source to destination. It calculates how much network bandwidth is required to send data from source to destination for each code length of 8, 16, 32, 48, and 64 bits.

2.5. Comparison competitors

Simulation was used in the validation of the proposed algorithm. The wire shark is a network analysis device used to determine how much bandwidth is required to transfer data from source to destination. Robust Discrete Code Modeling [29], Robust Geometric Correction [30], Discrete Discriminant Hashing [26], Binary Generative Adversarial Networks [31], Large Graph Hashing [33], and Geo-SPEBH [22] are the cutting-edge techniques employed in the evaluation of the suggested framework.

2.6. System requirements and tools

The tests were all conducted and ran on a 3.40 GHz CPU with four cores and 16 GB RAM. For experimentation, simulation, and implementation, a Java programming tool built on CloudSim was employed. The CloudSim is built with one server farm on 100 cloud-lets, each having a capacity of 300 input and output sizes and a length of 5000. To integrate the suggested method with the cloud, the CloudSim is required. For the wire shark network analyzer tool, 1 GB of network bandwidth is required.

3. Results

The goal is to produce discriminative binary hash codes that use just a small number of bits to code a large amount of data in a dataset, yielding excellent search precision and a faster query time with minimal memory consumption.

SIFT 1B dataset was generated from simulation results using cutting-edge methodologies, and the results were compared to the Geo-SPEBH algorithm. As shown in Table 1 and subsequently Fig. 1, the results reveal that all strategies required the same amount of network bandwidth of 0.091 kb/s when the code length is 8.

4. Discussion

The SIFT B dataset obtained from ftp://ftp.irisa.fr/local/texmex/corpus/bigann_learn.bvecs.gz contains one billion SIFT features represented by 128 dimension vectors. The amount of basis vectors is 1,000, 000, 000, and the query vectors are 10,000, 100, 000, 000 for learning. This dataset was used to run all of the algorithms with varying bit counts of 8, 16, 34, 48, and 64 to determine the bandwidth required to transport information from source to destination. The approaches being compared include RDCM, RGC, DDH, BGAN, and Geo-SPEBH. According to Table 1, all of the examined approaches used the same

Table 1. Simulation results for the proposed Geo-SPEBH and existing methods.

Methods	Bandwidth (kb/sec)	Code length (bits)
RDCM	8	0.091
	16	0.930
	32	0.0908
	48	0.092
	64	0.0147
RGC	8	0.091
	16	0.930
	32	0.0908
	48	0.092
	64	0.0147
DDH	8	0.091
	16	0.930
	32	0.0908
	48	0.092
	64	0.0147
BGAN	8	0.091
	16	0.980
	32	0.0908
	48	0.092
	64	0.0147
LGH	8	0.091
	16	0.930
	32	0.0908
	48	0.092
	64	0.0147
Geo-SPEBH	8	0.091
	16	0.930
	32	0.0908
	48	0.092
	64	0.0147

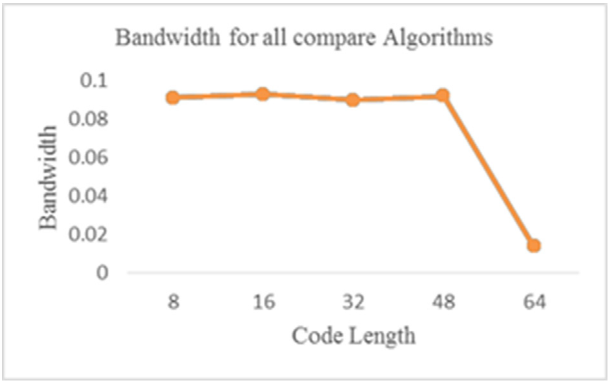

Fig. 1. Depicts a graphical representation of simulation results produced in a run alongside wire shark for all strategies based on bandwidth. The orange line represents the amount of bandwidth necessary for code bits of 8, 16, 32, 48, and 64.

amount of network bandwidth of 0.091, 0.930, 0.0908, 0.092, and 0.0147 kb/s for code lengths of 8, 16, 32, 48, and 64 respectively. This is because predictions are generated using lengthy code words and an even distribution of data points. Because of the high precision-recall rate attained by the algorithms using Hamming distance, the same amount of network bandwidth is required for each corresponding code length. The ability of RDCM [29] to learn good quality discrete codes and hash functions contributes to its performance. RGC [30] obtained the performance by removing geometric transformation and the composite rotation-scaling-translation.

The DDH [26] technique, which learns a robust similarity metric for maximizing similarity of same class discrete hash codes as well as similarity of different class discrete hash codes at the same time, facilitates information transmission. Furthermore, the use of Binary Generative Adversarial Networks (BGAN) [31] to incorporate images in binary codes influences data transport. The minimization of Euclidean distance in [33] to get binary codes for index creation increases the bandwidth needed for data delivery. The narrower area between the data points of the compared methods increased the bandwidth required for data transport from source to destination. Table 1 displays the bandwidth requirement values for the SIFT 1B dataset for all compared algorithms in kb/sec.

5. Conclusion

Given that bandwidth is one of the most valuable resources, especially in multimedia and distributed computing applications and its absence might cause a serious reduction in Quality of Service, this paper evaluate Geo-SPEBH algorithm and other state-of-the-art algorithms to check the level of bandwidth utilization for big data retrieval in cloud. Three steps were taken to achieve the goal of this work, each of the steps serving a distinct purpose. Similarity preserving term Q(y) was used to increase the precision of searches in the dataset. To ensure that data points are distributed uniformly in the hash container, each hash function was made independently of the others so as to balance partitioning for independence. Joint optimization component was constructed to answer for simultaneous optimization of search accuracy and search time, allowing for high search precision with short search time. Results obtained showed that, the developed method (Geo-SPEBH) and the compared algorithms recorded the same amount of network bandwidth for each code length on the SIFT 1B dataset for all code lengths. This is due to the fact that the predictions were created from the data points, which are uniformly distributed. Further research should look at data points not uniformly distributed.

Acknowledgment

No primary founder for this article and no grant whatsoever were received from any organization for the purpose of carrying out this research. All works in this article was carried out and funded by the authors.

References

[1] Danan T, Surya SC, Rafael CN, Leila A. A platform for monitoring and sharing of generic health data in the cloudvol. 35. Future Generation Computer System; 2014. p. 102–13.
[2] Abubakar UO, Aisha YU, Maryam M, Hauwa A, Boukari S, Gital AY. Minimum search-time algorithm for image retrieval in cloud computing. Int J Intell Eng Syst 2021;15:596–606.
[3] Fiona HM, Christina C, Annemarie C, Jane S, Hugh S, Mary U. Evaluating mobile phone application for health behaviour change: a systematic review. J Telemed Telecare 2018;2:22–40.
[4] Andre B, Shane G, Jeffrey P. The persistence of broadband user behaviour: implications for universal services and competition policyvol. 43. Elsevier Telecommunications Policy; 2018. p. 1–27.
[5] Tatcha S, Hitoshi M. The Internet of Things as an accelerator of advancement of broadband networks: a case of Thailandvol. 42. Telecommunications Policy; 2018. p. 293–303.
[6] York E, Conley SN, Henriksen AD, Caserta D, Etka N, Harrington N, et al. Co-imagining the futures of implementation precision medicine using scenario analysis and design fiction. OMICS A J Integr Biol 2019;7:340–9. https://doi.org/10.1089/omi.2019.0083.
[7] Wang Y, Shrivastava A, Ryu J. Flash : randomized algorithms accelerated over CPU-GPU for ultra-high dimensional similarity search. 2017. p. 1–4.
[8] Salih BA, Wongthongtham P, Beheshti SMR, Zajabbari B. Towards a methodology for social business intelligence in the era of big social data incorporating trust and semantic analysis520. Lecture Notes in Electrical Engineering; 2019. p. 519–27. https://doi.org/10.1007/978-981-13-1799-6_54.
[9] Liu S, Wang Y, Wen A, Wang L, Hong N, Shen F, et al. CREATE: cohort retrieval enhanced by analysis of text from electronic health records using OMOPvol. 5; 2019. Common Data Model [Internet], http://arxiv.org/abs/1901.07601.
[10] Merkys A, Vaitkus A, Chateigner D, Moeck P, Murray-rust P, Quiros M, et al. Crystallography open database : history, development, perspectives. 2019. p. 1–16 [Internet], https://www.researchgate.net/publication/335604699_Crystallography_Open_Database_History_Development_and_Perspectives/link/5dd66feca6fdcc2b1fa977b4/download.
[11] Tonello N, Tallada P, Serrano S, Carretero J, Eriksen M, Folger M, et al. The PAU Survey: operation and orchestration of multi-band survey datavol. 27. Astronomy and Computing; 2019. p. 171–88.
[12] Deepak P, Sahoo BPS, Sambit M, Satyabrata S. Cloud computing features, issues and challenges: a big picture. In: IEEE International Conference on Computational Intelligence and Networks; 2015. p. 116–23.
[13] Ryan S, Wang F, Wang H, Liu J. A deep investigation into network performance in virtual machine based cloud environments. In: IEEE international conference on Computer Communications; 2014. p. 1285–93.
[14] Buon K, Lau M, Jørgen BA, Gerhard K, Andreas FM. Impact of matching network on bandwidth of compact antenna arrays. In: IEEE Transactions on Antennas and Propagation. vol. 54; 2006. p. 3225–38.

[15] Li B, Yanyu H, Zheli L, Jin L, Zhihong T, Siu-Ming Y. HybridORAM: practical oblivious cloud storage with constant bandwidth. 2018. https://doi.org/10.1016/j.ins.2018.02.019 [Internet].

[16] Assabir A, Elmhamdi J, Hammouch A. Throughput enhancement of the edge user equipment based on the power- bandwidth tradeoff in the optical attocell networks. International Journal of Intelligent Engineering and Systems 2020;13:337–55.

[17] McManus JP, Day TG, Mailloux ZJ. The effects of latency, bandwidth, and packet loss on cloud-based gaming services. 2019. p. 1–58 [Internet], https://digitalcommons.wpi.edu/iqp-all/5327.

[18] Keane MT, Kenny EM. How case-based reasoning explains neural networks. A theoretical analysis of xai using post-hoc explanation-by-example from a survey of ANN-CBR twin-systems. Lect Notes Comput Sci 2019;11680:155–71. https://doi.org/10.1007/978-3-030-29249-2_11.

[19] Cai Q, Pan Y, Yao T, Yan C, Mei T. Memory matching networks for one-shot image recognition. In: IEEE Comput Soc Conf Comput Vis Pattern Recogn; 2018. p. 4080–8. https://doi.org/10.1109/CVPR.2018.00429.

[20] Lv Q, Josephson W, Wang Z, Charikar M, Li K. Multi-probeLSH: efficient indexing for high-dimensional similarity search. Proceedings of international conference on very large data bases. 2007. p. 950–61. https://www.cs.princeton.edu/cass/papers/mplsh_vldb07.pdf.

[21] Dong W, Wang Z, Josephson W, Charikar M, Li K. Modelling LSH for performance tuning. In: Proceedings of the ACM conference on information and knowledge management; 2008. p. 669–78. https://www.cs.princeton.edu/cass/papers/cikm08.pdf.

[22] Abubakar UO, Boukari S, Abdulsalam YG, Hauwa A. Performance evaluation of geometric similarity preserving embedding-based hashing for big data in cloud computing. J Theo Appl Info Technol 2020;98:378–90.

[23] Oladunjoye JA, Moses T, Okpor J, Bako AR. Performance study of the memory utilization of an improved pattern matching algorithm using bit-parallelism. J Comp Sci Eng 2022;3:49–59.

[24] Boukari S, Abubakar UO, Abdulsalam YG, Iliya MA. Performance evaluation of geometric similarity preserving embedding based hashing. Global Scient J 2019;7:642–57.

[25] Ram KK, Arunav S, Rabul HL. Image authentication based on robust image hashing with geometric correction. J Multimdia Tool Appl 2017;7:25409–29.

[26] Yan C, Jielin J, Zhihui L, Zuojin H, Waikeun W. Supervised discrete discriminant hashing for image retrieval. Int Conf Pattern Recogn 2018;78:79–90.

[27] Guosheng L, Chunhua S, Anton VD. Supervised hashing using graph cuts and boosted decision trees. IEEE transactions on pattern analysis and machine intelligence 2015;37:2317–37. https://doi.org/10.1109/TPAMI.2015.2404776.

[28] Yueming L, Wing WY, Ziqian Z, Daneil SY, Patrick PK. Asymetric cyclcial hashing for large-scale-image retrieval. In: IEEE Transaction on Multimedia. vol. 17; 2015. p. 1225–35.

[29] Ye R, Xuelong L. Compact structure hashing via sparse and similarity embedding. IEEE Transactions on Cybernetics 2016;46:718–28.

[30] Heo JP, Youngwoon L, Junfeng H, Shih-Fu C, Sung-Eui Y, Hashing Spherical. Binary code embedding with hypersphere. In: IEEE Transaction on Pattern Analysis and Machine Intelligence; 2015. p. 1–14.

[31] Jingkuan S, He T, Lianli G, Xing X, Alan H, Heng ST. Binary generative adversarial networks for image retrieval. Proc AAAI Conf Artif Intell 2018;32:394–401.

[32] Charikar M, Paris S. Hashing_based-Estimators for kernel density in high dimensions. 2018. ar.xiv:1808.10530v1[cs.DS].

[33] Li X, Hu D, Nie F. Large Graph hashing with spectral rotation. In: Proc AAAI Conf Artif Intell. 31; 2017. p. 1–7. https://dtaoo.github.io/papers/2017_LGHSR.pdf.

[34] Junfeng H, Regunathan R, Shih-Fu F, Claus B. Compact hashing with joint optimization of search accuracy an time. 2011. p. 753–60 [Internet], https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.362.6675&rep=rep1&type=pdf.