International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Decision Support Based Resource Allocation for Cost Reduction in Cloud Storage using Big Data Analytics

Krishnakumar L, Nithya A

ABSTRACT: ***Provision of highly efficient storage for dynamically growing data is considered problem to be solved in data mining. Few research works have been designed for big data storage analytics. However, the storage efficiency using conventional techniques was not sufficient as where data duplication and storage overhead problem was not addressed. In order to overcome such limitations, Tanimoto Regressive Decision Support Based Blake2 Hashing Space Efficient Quotient Data Structure (TRDS-BHSEQDS) Model is proposed. Initially, TRDS-BHSEQDS technique gets larger number of input data as input. Then, TRDS-BHSEQDS technique computes 512 bits Blake2 hash value for each data to be stored. Consequently, TRDS-BHSEQDS technique applies Tanimoto Regressive Decision Support Model (TRDSM) where it carried outs regression analysis with application of Tanimoto similarity coefficient. During this process, proposed TRDS-BHSEQDS technique finds relationship between hash values of data by determining Tanimoto similarity coefficient value. If similarity value is '+1', then TRDS-BHSEQDS technique considered that input data is already stored in BHSEQF memory. TRDS-BHSEQDS technique enhances the storage efficiency of big data when compared to state-of-the-art works. The performance of TRDS-BHSEQDS technique is measured in terms of storage efficiency, time complexity and space complexity and storage overhead with respect to different numbers of input big data.***

Keywords: Big data, Blake2 Hashed Space Efficient Quotient Filter, Hash Value, Regression Analysis, Storage overhead, Tanimoto Regressive Decision Support Model

I. INTRODUCTION

Big data represents rapidly increasing quantities of data collected from heterogeneous devices. Sensor networks, scientific experiments, websites, and many other applications generate data in different formats. Big data storage is a storage infrastructure. A distributed storage system named DCDedupe was designed in [1] that employs delta compression and deduplication to get better storage efficiency. However, space and time complexity taken for big data storage analytics was more.

Manuscript published on 30 September 2019
* Correspondence Author
Mr. Krishnakumar L[1], Research Scholar, School of Computer Studies, Rathnavel Subramaniam College of Arts and Science, Coimbatore, India
Dr. Nithya A, Research Supervisor, School of Computer Studies, Rathnavel Subramaniam College of Arts and Science, Coimbatore, India

Fuzzy cluster algorithm based on US-ELM (US-ELM-FC algorithm) was presented in [2] for efficient high-dimensional big data storage. But, data duplication during storage process was not considered in US-ELM-FC algorithm which increases the storage overhead. In order to solve the above mentioned existing issues, TRDS-BHSEQDS technique is introduced in this research work.

II. TANIMOTO REGRESSIVE DECISION SUPPORT BASED BLAKE2 HASHING SPACE EFFICIENT QUOTIENT DATA STRUCTURE MODEL

Tanimoto Regressive Decision Support Based Blake2 Hashing Space Efficient Quotient Data Structure (TRDS-BHSEQDS) technique is developed with the aim of increasing the efficiency of big data storage with a minimal time complexity.

In addition to that, Blake2 Hashed Space Efficient Quotient Filter (BHSEQF) is introduced in TRDS-BHSEQDS technique by using Blake2 Hashing Function in Quotient Filter in order to improve the storage efficiency while considering big data as input. The BHSEQF generates a hash value for each data and consequently determines the remainder and quotient of the hash value. Thus, TRDS-BHSEQDS technique lessens the amount of memory space taken for storing the larger amount of data when compared to conventional works.

The flow processes of TRDS-BHSEQDS technique to achieve higher big data storage efficiency. As presented in above architecture diagram, TRDS-BHSEQDS technique initially gets larger number of data as input. After that, TRDS-BHSEQDS technique employs Blake2 Hashed Space Efficient Quotient Filter (BHSEQF) in order to efficiently store the input data with minimal space and time complexity. During the data storage process, BHSEQDS technique utilizes Tanimoto Regressive Decision Support Model (TRDSM) in order to avoid data duplication and thereby reducing the overhead involved big data storage analytics. From that, TRDS-BHSEQDS technique enhances big data storage efficiency with a lower amount of time when compared to state-of-the-art works. The exhaustive processes of TRDS-BHSEQDS technique described in below.

2.1 TRDS-BHSEQDS Technique

BHSEQF designed in TRDS-BHSEQDS technique is 20% larger than conventional bloom filters. This supports for BHSEQF to store a more number of data on the contrary to state-of-the-art works.

As well, BHSEQF is very faster as compared to conventional data structures. Because, each data access in BHSEQF necessitates only single hash function which assists to attain better big data storage efficiency. On the contrary to state-of-the-art works, the BHSEQF creates a hash value for all the input data by using a Blake2 hash function. The processes of Blake2 hash function is depicted in below Figure 1.

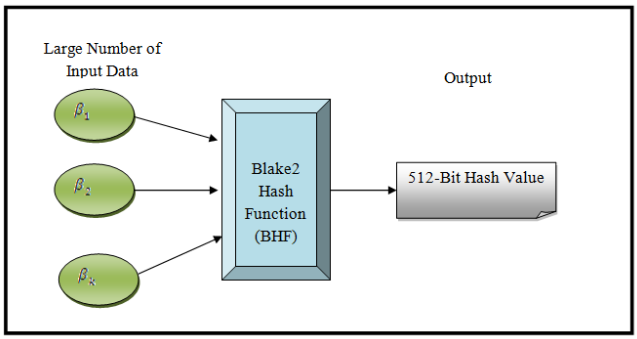

Figure 1: Blake2 Hash Function Process

As presented in above Figure 1, Blake2 hash function produces unique 512 bit hash value for all input data. The created hash value needs a minimal amount of memory space to store data when compared to conventional techniques. For this reason, TRDS-BHSEQDS technique obtains minimal space complexity for big data storage analytics.

2.2 TRDS-BHSEQDS Representation

The BHSEQF builds a hash value for each input data with help of below mathematical representation,

$$\$ = BHF(\beta_i) \quad (1)$$

From the above mathematical expression (1), 'BHF' designates a BLAKE2 Hash Function whereas '$\$$' indicates the generated hash value of data 'β_i'. The BHSEQF converts arbitrary size data into a bit string of a fixed size (i.e. hash value). On the contrary to existing storage techniques, BHSEQF produces 512 bits hash value for any size of input data and which is distinctive for each data. Further, BHSEQF is very fast because it includes of modern CPUs, instruction-level parallelism, SIMD instruction set extensions, and multiple cores which helps for TRDS-BHSEQDS technique to reduce time complexity of big data storage analytics. After creating a hash value, TRDS-BHSEQDS technique stored it in BHSEQF memory.

Figure 1, shows the overall process of TRDSM to efficiently solve the duplication problem on big data storage. As presented in above diagram, TRDSM initially get 512 bits hash value of data as input. Then, TRDSM verifies the input hash value of data is already stored in BHSEQF memory through regression analysis with the application of Tanimoto similarity coefficient. From that, TRDSM designed in proposed TRDS-BHSEQDS technique makes sure that only one distinctive instance of data is stored in BHSEQF memory. Thus, redundant data stored in BHSEQF memory are replaced with a pointer to the hash value of unique data. If the hash value is not stored in BHSEQF memory, then TRDSM stores hash value of data in BHSEQF memory. Whenever the hash value is already stored, TRDSM makes a pointer to existing one. Accordingly TRDSM efficiently resolves the data duplication problem on big data storage analytics. This supports for BHSEQF to diminish the overhead involved during process of big data storage when compared to state-of-the-art works.

While a new hash value is acquired for storage, the TRDSM compares it with existing hashes to ensure data is already stored in BHSEQF memory by measuring Tanimoto similarity coefficient using below formula,

$$S(\$_i, \$_j) = \frac{k * \sum \$_i \$_j}{\sum \$_i{}^2 + \sum \$_j{}^2 - \sum \$_i \$_j} \quad (2)$$

From the above mathematical expression (2), '$S(\$_i, \$_j)$' represent a tanimoto similarity coefficient value. Here 'k' denotes the total number of hash values is stored in BHSEQF memory, '$\$_i, \$_j$' point outs a hash values of two data. Here, '$\sum \$_i{}^2$' designates a sum of squared score of the data '$\$_i$' and '$\sum \$_2{}^2$' refers a sum of squared score of the data '$\$_j$' whereas '$\sum \$_i \$_j$' refers the sum of the product of the paired score of '$\$_i$' and '$\$_j$'. The tanimoto similarity coefficient returns the output results from 0 to +1. In TRDSM, '+1' point outs the input hash value of data is already stored in BHSEQF memory and '0' denotes the input hash value of data is not stored in BHSEQF memory.

If input data is not a duplicate of an existing hash value, then TRDSM stores it in BHSEQF memory. On the contrary to conventional storage techniques, BHSEQF doesn't store the data itself where it stores only an x-bit hash value of data. Therefore, the hash value is partitioned into a remainder and the quotient. From that, BHSEQF determines remainder (i.e. 'p' least significant bits) of hash value '$\$$' for each data 'β_i' with help of below mathematical expression,

$$\alpha = \$ \bmod 2^p \quad (3)$$

From the above mathematical equation (3), 'α' signifies the remainder of input hash value. Subsequently, BHSEQF finds quotient (i.e. 'q' most significant bits) of hash value for each data with help of below mathematical representation,

$$\gamma = \left\lfloor \frac{\$}{2^q} \right\rfloor \quad (4)$$

From the above mathematical formulation (4), 'γ' signify the quotient of input hash value. After finding the remainder 'α' and the quotient 'γ', BHSEQF stores remainder of input data in a bucket indexed by quotient.

The each bucket includes of three bits such as '$is\_occupied$', '$is\_continuation$', '$is\_shifted$'. Here, all the three bits are at the beginning initialized as '0'. In BHSEQF storage structure, '$s\_occupied$' is set as '1' when the bucket 'i' is the canonical bucket for some

hash value '$' stored in memory. Then, '$is_continuation$' in BHSEQF storage structure is defined as '1' when the bucket is full but not by the first remainder in a run. While the remainder is not in its canonical bucket, the '$is_shifted$' in BHSEQF storage structure is set as '1'. From the above figure, '$\beta_1, \beta_2, \dots \beta_k$' represent the input data effectively stored in BHSEQF memory. This helps for proposed TRDS-BHSEQDS technique to get better data storage efficiency with a minimal amount of time consumption as compared to existing methods. The algorithmic processes of TRDS-BHSEQDS technique is explained in below.

2.3 TRDS-BHSEQDS Algorithm

// Tanimoto Regressive Decision Support Based Blake2 Hashing Space Efficient Quotient Data Structure Algorithm

Input: Large Number Of Input Data '$\beta_i = \beta_1, \beta_2, \dots, \beta_k$'

Output:

Step 1: Begin

Step 2: **For** each input data 'β_i'

Step 3: Generate Blake2 hash value '$\$_i$' using (1)

Step 4: Apply Tanimoto Regressive Decision Support Model (TRDSM)

Step 5: Find relationship between the hash value of data through regression analysis using (2)

Step 6: **If** '$S(\$_i, \$_j) == +1$', **then**

Step 7: '$\$_i$' of data is already stored in BHSEQF memory

Step 8: **Else If** '$S(\$_i, \$_j) == +1$', **then**

Step 9: '$\$_i$' of data is not stored in BHSEQF memory

Step 10: **End If**

Step 11: **Else**

Step 12: Find remainder 'α' of created '$\$_i$' using (5)

Step 13: Determine quotient 'γ' of '$\$_i$' using (6)

Step 14: Stores remainder 'α' in bucket indexed by 'γ'

Step 15: **End If**

Step 16: **End for**

Step 17:End

Algorithm 1 Tanimoto Regressive Decision Support Based Blake2 Hashing Space Efficient Quotient Data Structure Model

Algorithm 1 presents the step by step processes of TRDS-BHSEQDS technique. By using the above algorithmic process, TRDS-BHSEQDS technique saves only a remainder of hash value of input data in a bucket indexed by quotient. Therefore, TRDS-BHSEQDS technique significantly reduced memory space while storing any size of input data on the contrary to conventional works. As a result, TRDS-BHSEQDS technique provides better big data storage analytics performance in terms of storage efficiency, time complexity and space complexity as compared to state-of-the-art techniques.

III. EXPERIMENTAL SETTINGS

In order to determine the performance of proposed, TRDS-BHSEQDS technique is implemented in Java Language, The dataset includes more than ten thousand entries with 15 columns. From that, TRDS-BHSEQDS technique considers different number of user data in the range of 1000-10000 as input to perform experimental process to analyze the performance of proposed and existing methods.

IV. CONCLUSION

The effectiveness of TRDS-BHSEQDS technique is estimated using metrics such as storage efficiency, time complexity and space complexity with respect to various numbers of input big data. The performance result of TRDS-BHSEQDS technique is compared against with two conventional methods namely distributed storage system called DCDedupe [1] and fuzzy cluster algorithm based on US-ELM (US-ELM-FC algorithm) [2].

REFERENCES

1. Binqi Zhang, Chen Wang, Bing Bing Zhou, Dong Yuan, Albert Y. Zomaya, "DCDedupe: Selective Deduplication and Delta Compression with Effective Routing for Distributed Storage", Journal of Grid Computing, Springer, Volume 16, Issue 2, Pages 195–209, June 2018
2. Linlin Ding, Yu Liu, Baishuo Han, Shiwen Zhang, Baoyan Song, "HB-File: An efficient and effective high-dimensional big data storage structure based on US-ELM", Neurocomputing, Elsevier, Volume 261, Pages 184-192, October 2017

AUTHOR PROFILE

Mr. Krishna Kumar L is currently a research scholar Research Scholar in School of Computer Studies, Rathnavel Subramaniam College of Arts and Science, Coimbatore, India.

Dr. Nithya A working as Associate Professor & Research Supervisor, School of Computer Studies, Rathnavel Subramaniam College of Arts and Science, Coimbatore, India.