Cuckoo Hashing: An efficient technique for data de-duplication in cloud storage

Aniket Kinage

Abstract: Cloud computing has provided a different perspective to users for storing and accessing their data by bringing a new era of a digital revolution. In the cloud, users might store the same data which results in more memory consumption. In order to solve this problem, one needs to restrict the duplicate entries of data in the cloud system. For this purpose, the cuckoo hashing algorithm is preferred because of its fast execution and retrieval time. It is also one of the best algorithms for avoiding collision of keys. The proposed system handles the data de-duplication process and thus saves bandwidth.

Keywords: Data de-duplication, cuckoo hashing, cloud computing, time complexity, chunk-level, bandwidth, hash function.

1. INTRODUCTION

In today's world cloud computing has gained a lot of attention due to its high-speed processing capacity and data storage. The accessibility of high capacity systems, minimum cost PCs and well-equipped storage devices have resulted in the advancement of cloud computing. A lot of users are using cloud environment for storage purpose. There are many concerns that need to be focused on like data security, increased performance and providing a convenient environment to the user.

Data de-duplication has become an important factor in storage backup systems because of its ability to improve storage utilization. It is a technique which excludes the duplicate chunks of repeating data. In this way only the unique copy of the information is stored, thus saving the memory space in the cloud environment.

In order to avoid data redundancy of data, a cuckoo hash-based method is proposed in this paper. Cuckoo hashing is the fastest technique which has worst case constant lookup time which helps to resolve collision of keys efficiently.

2. Literature Survey

In cuckoo hashing, the expected retrieval time is O(1) with constant time for insertion and deletion [2]. But, during insertion cuckoo hashing usually suffers from infinite loop problem. In order to address the infinite loop problem, unbusy kicking out routes technique was proposed [1]. In this technique, the frequency of data units is determined by the counter associated with all the values in the hash table.

Cuckoo hashing is currently being utilized in modern switches where the path to route can be found out in constant time [3].

To address the problems like data security, availability and vendor lock-in De-duplication-assisted cloud-of-clouds is used [3]. In these clouds the identification of duplicate data is done using MD5 or SHA1 hash values to improve cost and performance efficiently.

Data de-duplication is a distinctive form of compression [4]. A single pointer is used to identify identical chunks of data but for identifying similar chunks various techniques were followed such as dynamic block level de-duplication in which new data along with its pointer length were decided at runtime.

3. Key Concepts

3.1 Cuckoo Hashing

Cuckoo hashing is the hash-based data de-duplication technique. Cuckoo hashing generates a simple hash table where insertions and deletions have worst case O(1) time complexity. Insertions are reduced to constant time with high probability and thus helps to save bandwidth.

In cuckoo hashing we maintain two hash tables each consisting of n elements each, we then compute two hash functions h1 and h2. Every incoming element will be either placed at position h1(x) in the first table or h2(x) in the second table. During the process of insertions two possibilities may arise:

Case 1: While inserting an element if we can make arrangements to move the preoccupied element to a new empty position so that we can easily insert another element into its position.

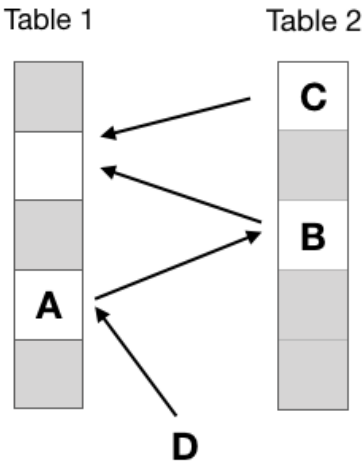


Fig.1 Cuckoo Hashing-Without cycle

Here we can move element A to the position where element B is residing and then element B to a vacant position. Hence by making such shifts, we can make a vacancy for the new incoming element D.

Case:2 There might arise a situation where no free space is available. In such a case the infinite loop problem comes into the picture, as the cycle is formed. Cycles are formed if we tend to revisit the same position with the same element for insertion. We have to rehash again and generate new values for the hash functions. Multiple rehashes might be required before the cuckoo hashing technique succeeds.

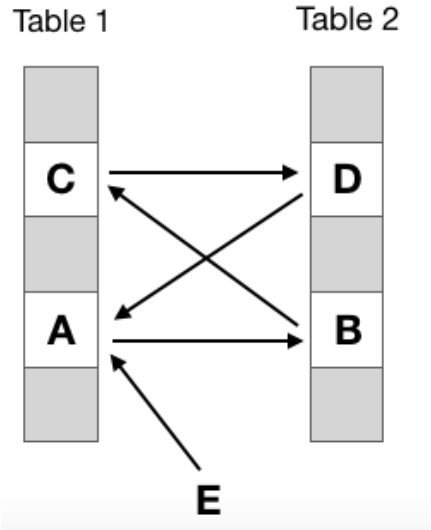
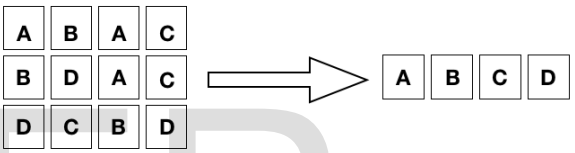


Fig.2 Cuckoo Hashing-With cycle

Here we are trying to insert new element E in the position of element A. But we are not able to shift the elements as there is no vacant position. So here we end up forming a cycle. We need to rehash again and generate new hash functions with new positions.

3.2 Data De-duplication

Data de-duplication is a technique through which duplicate copies of data can be removed from the data set. This technique stores a unique copy of data on the server whereas the duplicate data is referenced to that unique copy. For example, if 100 users have the same identical file on their computers and if each of them stores it on a single server, then storage space will be wasted. Instead of this only one copy of that data is stored while other copies of data are referenced using a pointer. Following this technique duplicate data entries will be restricted leading to optimal storage utilization.



Without De-duplication With De-duplication

Fig.3 De-duplication technique

4. Proposed Work

In cloud storages, the problem of data duplication arises when users try to store the same data to their storage. To assist this problem we use data de-duplication technique which ensures that only unique copy of data is stored. The proposed system will help us to enhance the performance and save storage space by adopting a proper approach to the de-duplicate cloud storage system.

In the cloud storage, users can provide duplicate data so there emerges a need to store all the data in a single folder together. Let XYZ be the folder where data of all the users will be stored centrally and each user will act as a subdirectory of that folder. Users can upload files of any types for carrying out de-duplication of data. Once all the users finish uploading their files, the de-duplication process starts. It runs once through all the files and removes the duplicated data. This process is carried out on chunk level and it supports all file types as data can be considered in binary form. The chuck-level approach provides us with a better probability of restricting duplicate data.

Chunks of fixed size are used for data de-duplication. Suppose a user uploads 20 MB of data and the chunk size

is 5MB, then the entire file will be divided into four parts and the de-duplication process will be carried out. This process will ensure that only a unique instance of data is stored in the system. But comparing a chunk of data that is 5MB every time with incoming chunks is going to be an exhausting job and may reduce the overall performance of the system.

Therefore the chunks need to be hashed and a 48-bit hash value for each chunk having specific size id generated. Now ignorer to avoid collision of keys, a good key generating algorithm must be selected. The algorithm must use optimal space and should provide better efficiency. Thus cuckoo hashing is used to generate keys as it has constant time complexity. Now, each hash value is compared with the previously generated hash values. To provide greater de-duplication efficiency, the chunk size must be small as more data can be found duplicated. Remember this process is adopted only for non-duplicated data storage.

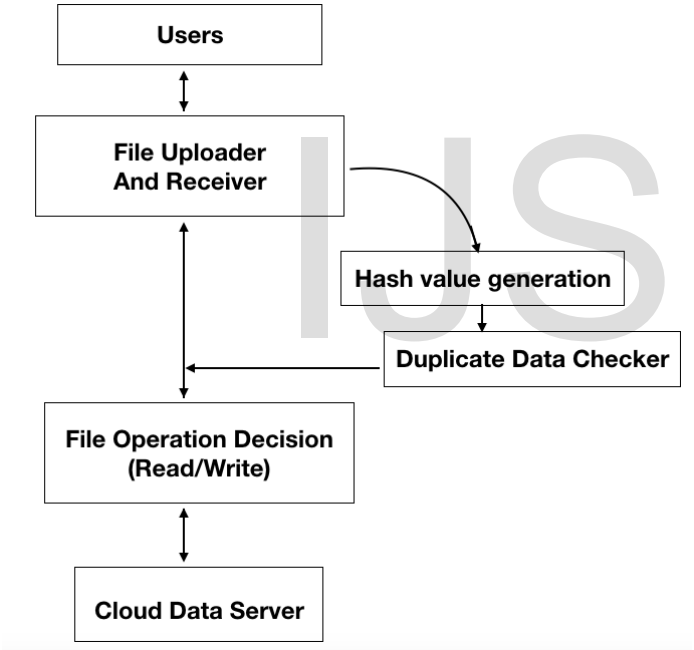


Fig.4 Proposed System Flow

5. Conclusion and future work

Cuckoo hashing technique thus helps us to prevent the duplicate data in minimum time while making efficient use of storage space. This proposed system can be used to carry out de-duplication in real-time systems. This paper focuses on cuckoo hashing with two hash functions and one element residing in one bucket, but we can try to implement it with more than two hash functions and more items residing per bucket which can result in high memory utilization.

References

[1] Sun, Y., Hua, Y., Feng, D., Yang, L., Zuo, P., Cao, S., Guo, Y.: A Collision-mitigation cuckoo hashing scheme for large-scale storage systems. IEEE Trans. Parallel Distrib. Syst. 28, 619–632 (2017)

[2] Pagh, R., Rodler, F.: Cuckoo Hashing. Springer, Berlin, Heidelberg(2001).

[3] Wu, S., Li, K., Mao, B., Liao, M.: DAC: Improving storage availability with deduplication-assisted cloud-OF-clouds. Future Gener. Comput. Syst. 74, 190–198 (2017)

[4] Hirsch, M., Ish-Shalom, A., Klein, S.T.: Optimal partitioning of data chunks in deduplication systems. Discrete Appl. Math. 212, 104–114 (2016)

[5] Dietzfelbinger, M., Weidling, C.: Balanced allocation and dictionaries with tightly packed constant size bins. Theoretical Computer Science 380(1-2), 47–68 (2007)