Dynamic Perfect Hashing

Md Hasib Hasan [1] Md. Abdun Fattah Lam [2] Mumtahina Arbi [3] Md Shahduzzaman [4]
Nahian Najah [5] Anika Islam [6]

Department of Computer Science and Engineering
Independent University, Bangladesh
Dhaka, Bangladesh.
{[1]2220166, [2]2220014, [3]2130152, [4]2131297, [5]2130642, [6]2220151}@iub.edu.bd

Conceptual

In Computer Science hashing is use for storing data retrieving data in constant time.On the other hand, with a huge use and collection of data, cloud storage has gained popularity among the users. In cloud storage most of the data is redundant. Cloud storage providers are using deduplication as the mechanism to keep only one copy of the file, by which cloud storage providers are minimizing the storage. But for deduplication counting of duplicate is required. We are showing a mechanism for data deduplication counting using dynamic perfect hash techniques.

Introduction

The idea of dynamic perfect hashing was first invented by Edward A. Fox, Qi Fan Chen, and Amjad M. Daoud in 1984, they proposed Extendible Hashing with Dynamic Overflow which was one of the earliest dynamic perfect hashing algorithms. Dietzfelbinger et al. implemented a dynamic dictionary algorithm on the basis of dynamic perfect hashing in 1994. Dynamic perfect hashing is a method of programming in a hash table data structure for resolving collisions by constructing a perfect hash function for a set of keys that can be updated dynamically.
Processing of data is extremely quick because to a method called dynamic perfect hashing that ensures there are no collisions. There are two stages of hashing used in this method. The second-level hash table uses the keys from the first-level hash function, which converts keys to a small selection of indices. The keys are mapped to their final indexes in the second-level hash table.
The dynamic perfect hashing is a technique for eliminating overflow chains by splitting a bucket when it overflows in a hash table data structure. It is useful for fast queries, insertions, and deletions must be made on a large set of elements.

Abstract program representation

In this algorithm the main table is Tbl, the size of the table is Sz. The two level hash function used here is hash1, hash2. In the main hash table (Tbl) there can be None or Set (clash, n, pop, T). Count is a global variable which counts the elements of the table.(hash1 and hash2 is generated by a perfect hash function).
Here, clash = Collision handler ← 0 ; n = Number of data in the array;
T = Secondary hash table pop = array count of Popular data [If any data Z times in the hash table]

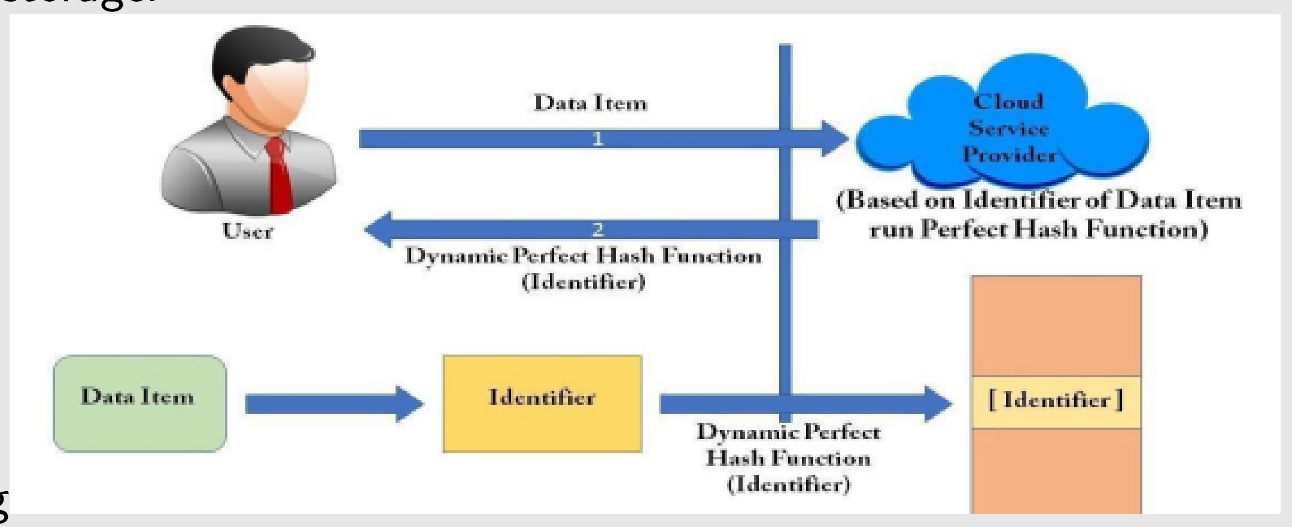
```
procedure INSERT(Data)
    // Empty slots in the main table and pop is marked as 0
    Count ← Count + 1
    if (Count is greater than Sz) then
        Rehashall(data)
    else
        j ← hash1(data)
        if Tbl[j] == 0 then
            k ← hash2(data)
            Create a Set (clash, n, pop, T) in Tbl[j]
            insert data in the subarray T[k]
            pop[k] ← pop[k] + 1
        else
            k ← hash2(data)
            if index k is empty in Secondary hash table T then
                insert data in the subarray T[k]
                pop[k] ← pop[k]+1
            else
                if data in the subarray T[k] == data then
                    pop[k] ← pop[k] + 1
                else
                    clash ← clash + 1
                    if clash $\geq$ 1 then
                        Double the size of array T
                        Put all the elements of T in a new temporary array L
                        x ← hashRphf(data)   //Uses randomized perfect hash function
                        for i = 0 to T.length() do
                            T[x] = data
                            Delete data from the array L
                        if L.length() $\neq$ 0 then
                            Rehashall(data)
```

Algorithmic complexity

The time complexity of the dynamic perfect hashing algorithm (Data deduplication in cloud storage using dynamic perfect hash functions) depends on several factors, such as the size of the input, the number of keys in the hash table, and the distribution of the keys.
In general, the time complexity of insertion, deletion, and lookup operations in a dynamic perfect hash table is O(1) on best case, since the hash function and secondary hash function allow for constant-time access to the correct bucket and index within the bucket. However, in the worst case, the time complexity can be O(n), where n is the number of keys in the hash table, if all keys happen to hash to the same bucket and the secondary hash table needs to be rebuilt.
The time complexity of building a new secondary hash table using the build secondary table method is also O(n) in the worst case, since all keys in the bucket may need to be rehashed using the new hash function.
*** In best case the Time complexity for insertion, deletion and lookup of that algorithm is O(1) for all.**
*** In average case the Time complexity for insertion, deletion and lookup of that algorithm is O(n), O(1) & O(1).**
*** In worst case the Time complexity for insertion, deletion and lookup of that algorithm is O(n), O(1) & O(1).**
*** The Space complexity of that algorithm is O(n).**

Data deduplication using dynamic perfect hash techniques

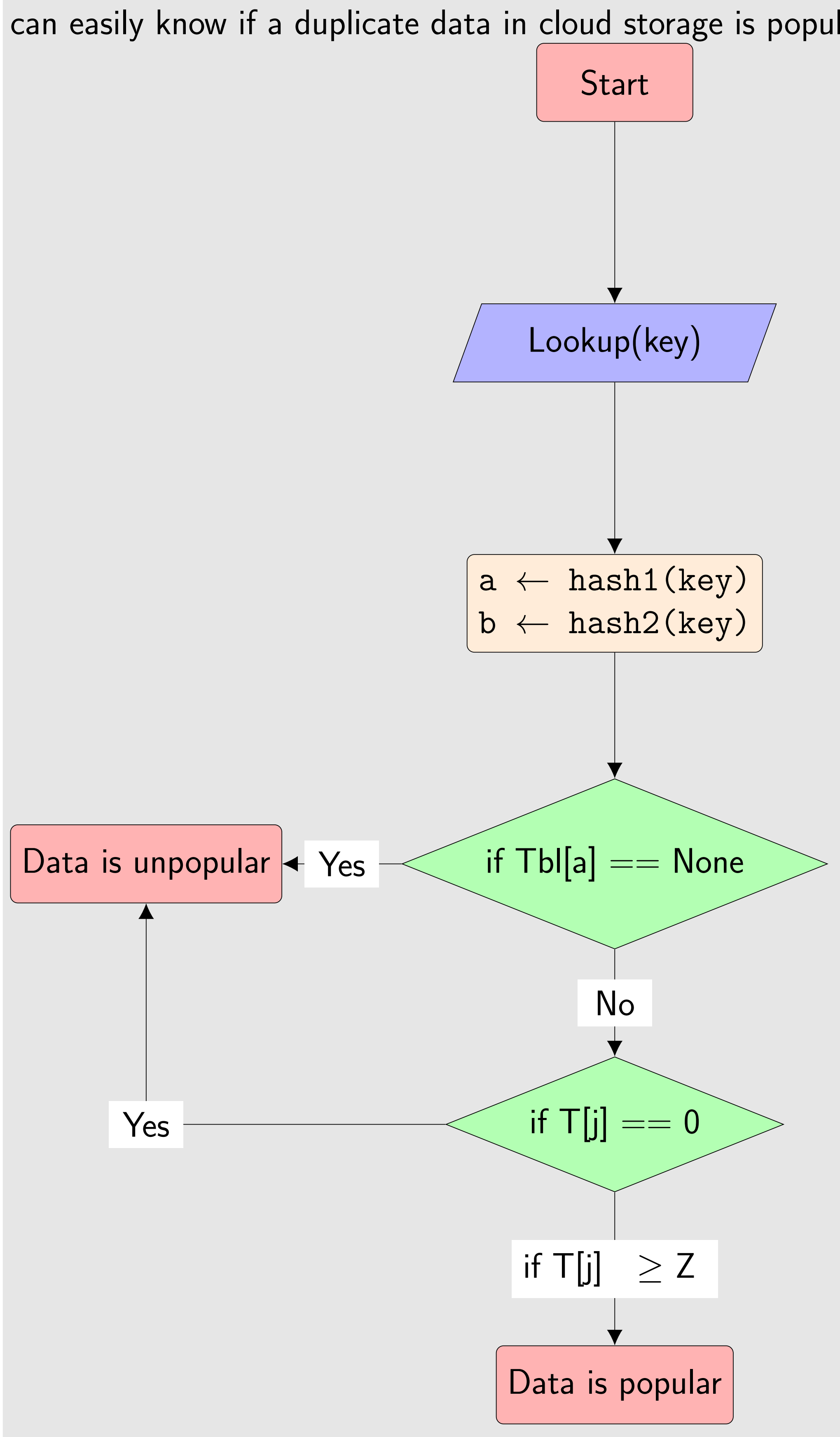
As the increasing demand of data rates, cloud storage system has become a necessity for computer users. But in cloud storage many users upload same type of data like a popular movie, song, research paper etc. This causes the use of cloud storage inefficient. So, the cloud storage providers use data deduplication. Data deduplication is the process of removing duplicate data from the cloud storage.


User
Data Item
1
Cloud Service Provider
(Based on Identifier of Data Item run Perfect Hash Function)
2
Dynamic Perfect Hash Function (Identifier)
Data Item
Identifier
Dynamic Perfect Hash Function (Identifier)
[Identifier]
Data processing

***Using dynamic perfect hashing for data deduplication makes the counting of duplicate data easier.**
***It helps to keep track of the popular unpopular data.**
*** It is useful for the retrieval of data in a constant time.**
*** It makes the data layer more secured in cloud storage because of the two-level hashing.**

Process diagram

Here, Tbl is the main hash table and T is the secondary hash table. Using dynamic hash function we can easily know if a duplicate data in cloud storage is popular or not.



Conclusion

Data deduplication using dynamic perfect hashing plays an important role in cloud storage systems by eliminating redundant data and reducing storage costs. It helps to optimize storage capacity, backup, recovery times and security. Identifying popular data in the cloud using dynamic perfect hashing is a crucial step towards optimizing the performance and efficiency of cloud storage systems. By understanding which data is popular and in high demand, cloud providers can allocate their resources more effectively and reduce costs of the storage and retrieval. Additionally, users can benefit from faster access to popular used data, leading to improved productivity and overall user satisfaction. It also provides a wide level security to data by two level hashing (which work as encryption). There is no unmixed blessing on earth this algorithm has some flaws. For rehashing it needs a space on O(n). Which is costly. It also uses a bucket for counting the popularities. Which is also storage inefficient. Overall, dynamic perfect hashing provides a reliable way to handle duplicate data in cloud storage system and can significantly improve the performance of cloud storage for data deduplication.