A Collision-Mitigation Cuckoo Hashing Scheme for Large-scale Storage Systems

Objective:

Main objective of the system is to find the malicious IP cluster using simple data analysis techniques.

Abstract:

With the rapid growth of the amount of information, cloud computing servers need to process and analyze large amounts of high-dimensional and unstructured data timely and accurately. This usually requires many query operations. Due to simplicity and ease of use, cuckoo hashing schemes have been widely used in real-world cloud-related applications. However, due to the potential hash collisions, the cuckoo hashing suffers from endless loops and high insertion latency, even high risks of re-construction of entire hash table. In order to address these problems, we propose a cost-efficient cuckoo hashing scheme, called MinCounter. The idea behind MinCounter is to alleviate the occurrence of endless loops in the data insertion by selecting unbusy kicking-out routes. MinCounter selects the "cold" (infrequently accessed), rather than random, buckets to handle hash collisions. We further improve the concurrency of the MinCounter scheme to pursue higher performance and adapt to concurrent applications. MinCounter has the salient features of offering efficient insertion and query services and delivering high performance of cloud servers.

As well as enhancing the experiences for cloud users. We have implemented MinCounter in a large-scale cloud testbed and examined the performance by using three real-world traces. Extensive experimental results demonstrate the efficacy and efficiency of MinCounter.

Introduction:

analyze large amounts of data timely and accurately.According to the report of International Data Corporation (IDC) in 2014, the digital universe is doubling in size every two years from now until 2020, and the data we create and copy annually will reach 44 ZettaBytes in 2020. The digital bits in data universe will be as many as stars in the physical universe . Although cloud computing systems consume a large amount of system resources, it is still challenging to obtain accurate results for query requests in a real-time manner.

In order to improve entire system performance and storage efficiency, existing schemes have been proposed, such as hierarchical Bloom filter index to speed up the searching process , continuously monitoring query execution to optimize the cloud-scale query , query optimization for parallel data processing , approximate membership query over cloud data , multi-keyword search over encrypted cloud data similarity search in file systems ,minimizing retrieval latency for content cloud information and retrieval for ranked queries over cloud data . Due to space inefficiency and high-complexity hierarchical addressing, these schemes fail to meet the needs of real-time queries.

In order to support real-time queries, hashing-based data structures have been widely used in constructing the index due to constant-scale addressing complexity and fast query response. Unfortunately, hashing-based data structures cause low

space utilization, as well as high-latency risk of handling hashing collisions. Traditional techniques used in hash tables to deal with hashing collisions include open addressing ,chaining and coalesced hashing .Unlike conventional hash tables, cuckoo hashing addresses hashing collisions via simple “kicking-out” operations (i.e., flat addressing), which moves items among hash tables during insertions, rather than searching the linked lists (i.e., hierarchical addressing).