Journal of Engineering and Applied Sciences 12 (23): 7189-7193, 2017
ISSN: 1816-949X


# Quaternary Defense Mechanism Handling Predicaments in Cuckoo Hashing

[1]D. Seethalakshmi and [2]G.M. Nasira
[1]Research and Development Centre, Bharathiar University, Coimbatore, Tamil Nadu, India
[2]Department of Computer Applications, Chikkanna Government Arts College,
Tirupur, Tamil Nadu, India

**Abstract:** The rapid boom of amount of records, cloud computing servers required to store and analyzes huge amounts of excessive-dimensional and unstructured data. Cuckoo hashing schemes were broadly used in actual-world cloud-related programs. However, because of the capable hash collisions, the Cuckoo hashing suffers from limitless loops and high insertion latency. According to the potentiality the collision in Cuckoo hashing has recursive loops with entire latency insertion. The concurrency problem can have effective insertion of services through query based on its high performance of cloud commodity servers. In the proposed study, we divide hashing into four threads invokes the quaternary thread with separate threads to handle those hot and cold buckets and the other two threads monitors and handles predicaments occurred dynamically. Thus, in the proposed by segregating into four threads and each thread will handle hot, cold buckets and one thread for complete monitoring of transmissions and the other thread will help in overcoming attack. If any attack found then the data from that particular commodity server transfers to secured server and the connection will be disconnected thus to provide strong security.

**Key words:** Cloud computing data, Cuckoo hashing, cloud commodity server, quaternary thread, hot and cold buckets, predicaments

## INTRODUCTION

The big data cloud computing services processing and analysis of huge amount of data with efficient datasets (Turner *et al*., 2014). As discussed in international data corporation with digital doubling of size throughout the years. The digital doubling of universal size of copies reached every year. The physical data can handle the terabyte scaling and megabytes scaling of everyday data (Armbrust *et al*., 2010). Huge and massive fractions of data used popularly by the mobile devices. According to the energy constraints and limited storage capacity with actual time for processing the analysis of nontrivial content of cloud based services.

The cloud computing system stores huge amount of system resources by attaining their accuracy in results (Bykov *et al*., 2011). The query request approaches at actual time in the cloud computing resources. Thus, to improvise the entire system performing has their efficient storage of existing systems. The speeding up process while searching the hierarchical filters for indexing the fast process of searching. This continuous process can monitor the query execution thus to optimize the cloud scale queries. The query optimization has parallel data processing along with approximate query for cloud data with encrypted data in keyword search.

The multiple keyword searches have minimized file systems by retrieving the latency of context in cloud information (Wu *et al*., 2011). The ranks will be minimized by the queries for retrieving the latency containing the cloud details. As the inefficiency in space with large and complex addressing of hierarchical system contain actual query instance. By supporting the real time query with hash based data structures develops indexing in accordance with them. The hashing based data structures leads to utilization of less space accommodation. The huge latency risk for managing the hashing collisions addresses the complexity (Hua *et al*., 2009). Conventional techniques used for hash table deals with hashing collisions that may address them openly by having chain of coalesced hashing.

The traditional hash tables used in Cuckoo hashing address the hashing in collisions through sample out of operations by moving items among hash table inclusion by searching the linked lists with hierarchical addressing. The parallel hashing demonstration of Cuckoo hashing involved in chain of hash with increased loads of

**Corresponding Author:** D. Seethalakshmi, Research and Development Centre, Bharathiar University, Coimbatore, Tamil Nadu, India

Cuckoo hashing. To use hash tables with every instance of storage buckets (Wang *et al*., 2012). The Cuckoo hashing is a kind of bucket suitable for including the dynamic hashing collision. The items moved by the positions irrespective of their positioning table and inclusion factor. The system that ensures equally distributed data items in the hash tables.

**Literature review:** Cuckoo hashing can be efficiently varies by multiple options with hashing systems with replaced items (Liu *et al*., 2012). It can have multiple processes of buckets with hash tables. There are no vacant buckets with items in the process. They make it out of the existing buckets as an alternative of inclusion cost overflow. The linked list has items for operating the recursive items that can occupy the items within the buckets in the inclusion operation.

M. Mitzenmacher explains the hyper graph selection of various options can be operated by the Cuckoo hashing (Crainiceanu, 2013). Existing Cuckoo hashing develops the theoretical results by similar properties with hierarchy of bipartite graph. Cuckoo hashing has variables referred by managing the hash collisions without any breadth first search methodologies.

The selection of items in the buckets with inclusion of item has empty possibility of locations is detailed by Bruno *et al*. (2013) and Hua *et al*. (2012). The decrease in hash table techniques based on its data structures handles collision using inclusions, deletions more than searching of index in linked lists. The concept of constant query complexity that guarantees amortization time was used for inclusion and deletion of process. Thus, it improvises the utilization of space by increasing the latency of query based results.

The implementation of data structures in the hash table by acquiring the sequential hardware using multiprocessor machines that synchronizes their access (Cao *et al*., 2014). Such concurrent access of information in data structures will be more important and their significance will be focused on concurrency of hash tables. By handling such collision leverages the randomness in strategies that selects recursive loops holds metadata for their mitigated Cuckoo hashing.

The data mitigation along with key relocations has extra spatial cost as discussed by Wang and Yu, the performance will be high (Li *et al*., 2010). Intensive data items inserted by the real time queries based on storing the Cuckoo hashing. It kicks out the functionalities that incur the intensive data migration within the servers. The candidate buckets of other items within the existing operations by migration of the hash tables between multiple buckets.

The Cuckoo hashing considering various differentiations can be handled and as defined by Frieze *et al*., the hash collisions without any analysis over the breadth first search (Hua *et al*., 2008). It has random walk efficiency by random methodologies. The buckets of different candidates will be selected from the item included and there is no vacancy in their probable locations. The decreased probability of failures when including an item the robust hashing or Cuckoo hashing has low amount of constants in stashing (Bjorkqvist *et al*., 2011). The equivalent improvisations in analysis of items have simulations.

The various schemes in Cuckoo hashing have variations as described by Necklace *et al*. with identical min counter methodology (Peterson, 1957). The various methodologies and phases of Cuckoo hashing focused storage performance. Inclusion functionalities in Cuckoo hashing have items that search estimated locations among the people. A pair of recursive items that can send out the buckets from each other will be common. The min counter variations can be handled by such methodologies.

## MATERIALS AND METHODS

**Cuckoo hashing scheme:** Cuckoo hashing scheme is the dynamic static dictionary that has leverage over hash collisions by mitigating the estimation complexity by using linked lists in traditional hash tables. The number of items having conventional hash tables placed over the items with increased hash collisions. In all the candidate buckets of hash collision that included in the execution of operations dynamically operates the hashed buckets. They choose the suitable buckets for the new item to be inserted. As it is identical to the character of Cuckoo bird that kicks out its younger one from their nest this proposal is named as Cuckoo hashing. In such manner hashing recursively kicks out the items from the buckets by leveraging various hash functions by alleviating the hash collision functions. It does not avoid hash collisions completely. By inserting them the endless loop will be formed and managed by the position of collision until the loop breaks out. The cost effective overheads takes advantage of less spaced addition and constants it offers variations in hash positions.

The broadly used applications in the actual world the implementation of router should avoid huge number of items included in the operations. The cost effective overheads implementation the hardware will be acceptable. The chunk stash is one of the variants of Cuckoo hashing that resolves the hash collisions to be indexed. The hash collision and data migration based on

the Cuckoo based hashing scheme in cloud storage servers. Allocating the min counter in hash tables in buckets to record the kick out instances occurred in the buckets. The position of candidates in the new item included by the other items chooses the minimum counter more than the random execution of replacement functionalities. The importance of executing the hash collisions decreases the data migration by items balanced by the hash tables. Improvising the space efficiency by decreasing the latency illustrates the salient features of load balance by infrequent access of items. The higher risks can again build the complete hash table thus, to reflect the problems in using cloud commodity server. Cuckoo hashing has cost effective counters to lessen the existence of recursive loops by choosing the cloud commodity server for rent.

The prototype implementation by the comparison of approach towards the Cuckoo hashing helps the stash chunk that tracks the random dataset. The illustration of performance improvisations utilizes the hash tables and inclusion latency. Cuckoo hashing has multiple hash positions of an item that helps the items to move in between the hash positions. The extra space overheads precisely have space units identical to the space overhead by the binary search tree.

Cuckoo hashing traditionally uses two hash tables by providing length with two hashing functions with each items length. The two buckets will be involved but not both will be involved at once. The hash methodologies required by the independence of the rules with random distribution of hash tables in the operations involving standard Cuckoo hashing. The arbitrarily chosen empty buckets will be inserted with the item in table and the process will be recursive till all the items find appropriate buckets of their own storage space.

The commodity server used for rent by the cloud storage system stores the data when the cloud storage borrowed is not enough. The neighboring cloud will be used for elasticity as commodity server. The Cuckoo hashing has light weight process with other server transmissions that verifies its data integrity. The common crisis occurs in elasticity of commodity server has endless recursive loops which takes more time to restore data. The hard bucket handles more endless loops than the cold bucket which handles easy hashing loops.

The recursive process of identifying the buckets will be illustrated by the dynamic process successfully included by moving items from one table to the other. The threshold will be attained when the endless loops stored in their suitable buckets and new items to be inserted and concurrently the unused items will be kicked out. The number of recursions specifying the defined thresholds will happen to the random options. As the hot buckets are kicked out and cold buckets only are concentrated with low frequency it adds more disadvantage to the existing hashing algorithms. It mainly improvises performance and avoids unnecessary waiting time providing high accuracy and mitigating the efficient collision mechanism.

**Quaternary defense mechanism in Cuckoo algorithm implementation:** The dynamic Cuckoo hashing with static meaning used to speed up queries with worst case and constant scaling of instances according to the flat item response between multiple choices. The endless loops followed by the high performance in concurrency of applications. The feature that offers inclusion of items efficiently it enhances the experience of cloud users. The threaded selection by quaternary algorithms handles the hash collision of Cuckoo hashing. The segregation of hot bucket and cold bucket first in the quaternary thread operation will be invoked. In this each thread handles the hot and cold buckets and improvises their performance and it avoids unwanted waiting time. The other two threads one monitor the hashing for recursive loops. The other thread will be dedicated for overcoming attacks. As most of the time attack does not occurs the thread dedicated for attack defense assigned for handling hard and cold bucket problems. It acts as elasticity defender for occurring problems. The threads handling similar instance, for providing the significance of hot bucket and it leads to more accurate and efficient mitigation in collision operation.

The quaternary threaded selection algorithms can manage the hash collision effectively by the Cuckoo hashing. The segregation of hot bucket and cold bucket of two different threads makes the implementation of algorithm more efficient. Improvisation of the efficiency done by avoiding unwanted waiting time and manages both threads at same instance of time. It signifies both hot bucket and cold bucket with more accuracy and mitigation over hash collision.

As both hot and cold buckets were concentrated concurrently they achieve recursive and endless loop of hash table organization which will be illustrated in Fig. 1. The performance improvisation helps in speed, execution quality and increase the quality of service of cloud servers by giving them equal priority.

The other two threads monitors and overcomes attacks. Monitoring of attacks helps identifying attacks before attack and defense attacks before losing any data. The other thread overcomes attacks if exist. It also participates in monitoring attacks when no attack takes place. If it finds any attacks then the data from that commodity server will be transferred to the other alternative commodity server. Then it disconnects the commodity server and blacklists it for other user's

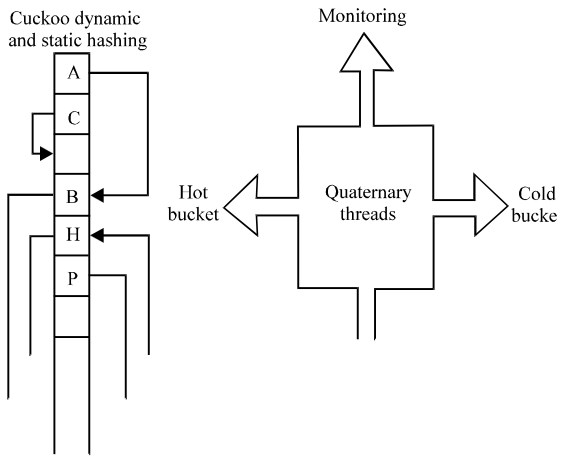


Fig. 1: Architecture of quaternary defense mechanism

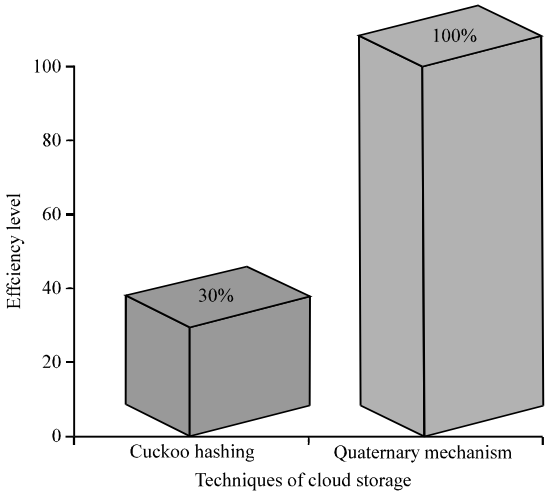


Fig. 2: Efficiency level representation for quaternary defense mechanism

reference. There is a chance of recovery of data from that server by the attacker thus, the quaternary defense mechanism stores the fake values in that place of data thus attacker will never get the appropriate data.

The hard bucket hashing will be handling attacks like denial of service increase in traffic efficiently. The commodity server prone to hacking attacks such as intrusion, leeching thus, monitors and defense over the attacks. By attacking the commodity server it will directly attacks the main server creating defense by attacking the immediate other commodity servers also.

Cloud servers will be depicted by creating and managing the analysis of huge amount of high-dimensional data. Its unstructured data provides process punctually and accurately by processing their queries in the cloud servers. The frequently accesses hot bucket will be taken for one process and infrequently accessed cold bucket will be taken for consideration by another process. Both buckets will be segregated in accordance with their duration and accessing instance of time.

Figure 2 shows the proposed quaternary defense mechanism will overcome the commodity server attack by sequential monitoring. If any attack is suspected then it disconnects the connection of commodity server from the main server. Before disconnection for safety purpose quaternary defense system will:

- Black lists that commodity server so that, other cloud servers will not be snared by this attacked server
- It stores the fake values in the address space where the actual data stored thus there is no leakage of data if any recovery applied also
- Then that cloud commodity server will be freed for its database storage and have no interactions between each other

Thus, we handle hashing and cloud storage problems efficiently. The monitoring system acts as firewall for attacks such as Denial of Service (DOS), network traffic, hacking intrusion attacks and leeching. Quaternary defense system can handle endless loops through hard bucket, cold bucket, monitoring attacks and overcomes other attacks efficiently.

## RESULTS AND DISCUSSION

**Experimental and performance analysis:** The quaternary threaded selection algorithm used for concurrent handling of both hot bucket and cold bucket that will be executed by invoking both types of threads. The management of hot and cold buckets will be depicted and the min counter will be plotted according. Thus these threads provide identical significance to complete the task much efficiently. Although, the hot bucket blocked by infinite number of loops the complete reconstruction helps the cold bucket thus it helps to recover it.

When four threads are executing then there exists deadlock situation which might be avoided by the synchronization technique throughout the execution time. It greatly helps out to avoid deadlock situation thus it can be a fault tolerant. The performance level and efficiency for handling attacks will be higher in quaternary defense mechanism than any other alleviated Cuckoo hashing technique.

## CONCLUSION

In the study, we propose the quaternary threaded algorithm which handles both hot and cold bucket

hashing and monitoring and handles attack efficiently. The optimized Cuckoo hashing in the large scale cloud computing system based on the data structures used will have intensive data mitigation by utilizing the latency inclusion and low space utilization. As the defense mechanism in the proposed approach is strong and determines the efficient Cuckoo implementation in commodity servers. The extensive usage of Cuckoo hashing in such quaternary threaded platform helps improvising the utilization ability and efficiency throughout the algorithm implementation.

## REFERENCES


Armbrust, M., A. Fox, R. Griffith, A.D. Joseph and R. Katz *et al*., 2010. A view of cloud computing. Commun. ACM, 53: 50-58.

Bjorkqvist, M., L.Y. Chen, M. Vukolic and X. Zhang, 2011. Minimizing retrieval latency for content cloud. Proceedings of the IEEE International Conference on INFOCOM, April 10-15, 2011, IEEE, Shanghai, China, ISBN: 978-1-4244-9919-9, pp: 1080-1088.

Bruno, N., S. Jain and J. Zhou, 2013. Continuous cloud-scale query optimization and processing. Proc.VLDB. Endowment, 6: 961-972.

Bykov, S., A. Geller, G. Kliot, J.R. Larus and R. Pandya *et al*., 2011. Orleans: Cloud computing for everyone. Proceedings of the 2nd ACM Symposium on Cloud Computing, October 26-28, 2011, ACM, Cascais, Portugal, ISBN: 978-1-4503-0976-9, pp: 1-16.

Cao, N., C. Wang, M. Li, K. Ren and W. Lou, 2014. Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE. Transac. parallel Distrib. Syst., 25: 222-233.

Crainiceanu, A., 2013. Bloofi: A hierarchical bloom filter index with applications to distributed data provenance. Proceedings of the 2nd International Workshop on Cloud Intelligence, August 26-26, 2013, ACM, Trento, Italy, ISBN:978-1-4503-2108-2, pp: 1-4.

Hua, Y., B. Xiao and J. Wang, 2009. BR-Tree: A scalable prototype for supporting multiple queries of multidimensional data. IEEE. Trans. Comput., 58: 1585-1598.

Hua, Y., B. Xiao, B. Veeravalli and D. Feng, 2012. Locality-sensitive bloom filter for approximate membership query. IEEE. Trans. Comput., 61: 817-830.

Hua, Y., B. Xiao, D. Feng and B. Yu, 2008. Bounded LSH for similarity search in peer-to-peer file systems. Proceedings of the 37th International Conference on Parallel Processing, September 9-12, IEEE, Portland, Oregon, USA., ISBN:978-0-7695-3374-2, pp: 644-651.

Li, J., Q. Wang, C. Wang, N. Cao, K. Ren and W. Lou, 2010. Fuzzy keyword search over encrypted data in cloud computing. Proceedings of the 9th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, March 15-19, 2010, San Diego, CA., USA., pp: 1-5.

Liu, Q., C.C. Tan, J. Wu and G. Wang, 2012. Efficient information retrieval for ranked queries in cost-effective cloud environments. Proceedings of the IEEE International Conference on INFOCOM, March 25-30, 2012, IEEE, Orlando, Florida, ISBN:978-1-4673-0773-4, pp: 2581-2585.

Peterson, W.W., 1957. Addressing for random-access storage. IBM. J. Res. Dev., 1: 130-146.

Turner, V., J.F. Gantz, D. Reinsel and S. Minton, 2014. The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. International Data Corporation, Framingham, Massachusetts, USA.,.

Wang, C., K. Ren, S. Yu and K.M.R. Urs, 2012. Achieving usable and privacy-assured similarity search over outsourced cloud data. Proceedings of the IEEE INFOCOM, March 25-30, 2012, Orlando, FL., pp: 451-459.

Wu, S., F. Li, S. Mehrotra and B.C. Ooi, 2011. Query optimization for massively parallel data processing. Proceedings of the 2nd ACM Symposium on Cloud Computing, October 26-28, 2011, ACM, Cascais, Portugal, ISBN:978-1-4503-0976-9, pp: 1-12.