# Improving Read/Write Performance for Key-value Storage System by Automatic Adjustment of Consistency Level

ThazinNwe, Tin Tin Yee, Ei Chaw Htoon
*University of Information Technology, Myanmar*
*thazin.nwe@uit.edu.mm, tintinyee@uit.edu.mm, eichawhtoon@uit.edu.mm*

## Abstract

*Distributed Key-value database is designed for storing, retrieving, managing associative arrays and data is replicated across different nodes for high availability, and no single point of failure. In such systems, Apache Cassandra is a peer-to-peer architecture which any user can connect to any node in any data center and can read and write data anywhere. Most of the systems usually select a fixed number of replicas of read/write requests in key-value storage. When the more replicas a read request chooses, it may increase the response time and reduce the system performance. Therefore, the encoded data is written to a Distributed Hash Table (DHT) and the decoded data are retrieved by automatically adjusting the read and write consistency level. The proposed approach tends to achieve the read/write performance of client requests for key-value storage system by automatically select the minimum number of consistent replicas of defining the consistency level.*

**Keywords:** Consistent Replicas, Consistency Level, Key-value database, DHT

## 1. Introduction

Replication is a widely used technology in distributed key-value storage systems to achieve data availability, durability, fault tolerance and recovery. In these systems, maintaining data consistency of replication becomes a significant challenge. Although many applications benefit from strong consistency, latency sensitive applications such as shopping carts on e-commerce websites choose eventual consistency [6]. Eventual consistency is a weak consistency that does not guarantee to return the last updated value [8]. Eventually consistent systems are high operation latencies and thus in bad performance. Achieving high throughput and low latency of responses to client requests is a difficult problem for cloud services. To fix these issues, a consistent replica selection process needs to include mechanisms for filtering and estimating the latency when processing requests. The replica selection process is inherently complicated.

Therefore, this system proposes a consistent replica selection approach to read/write access to distributed key-value storage systems by encoding and decoding data on Distriuted Hash Table (DHT).

This approach can determine the minimal number of replicas for reading request needs to contact in real time by defining the consistency level (one, two, quorum, local quorum, etc.). Depending on these consistency levels, the system can choose the nearest consistent replicas using replica selection algorithms. By using these algorithms, the system will improve the read/write execution time of defining the consistency levels and reduce the read/write latency cost of choosing the nearest consistent replicas.

## 2. Related Works

Performance and reliability of quorum based distributed key-value storage systems [1, 5] are proposed in the literature. H. Chihoub et al. [3] proposed an estimation model to predict the stale read and the system adjusted replica consistency according to the application requirements. Harmony uses a White box model uses mathematical formula derivation to choose the replicas numbers of each request. To select the number of replicas to be involved in a read operation necessary, this model finds the stale read rate smaller or equal to the defined threshold value. However, since there are so many factors that can impact the result and lots of those factors change in real time, such white box analysis may not get precise result and the system did not consider the performance of read/write operations. Harmony assumes the request access pattern meets Poisson process. However, Harmony has usage limitation because different application access patterns are different.

In most systems, it defines the rate of stale read that can be tolerated, and then try to improve the system performance as much as possible while still not exceed such stale read rate. However ZHU Y et al. [9] takes another mechanism, the longest response time is defined that it can tolerate and try to enhance the consistency level as much as possible within this time. The read/write access is broken into 6 steps: reception, transmission, coordination, execution, compaction and acquisition, and each of which can further break into smaller steps. Then a linear regression is used to predict the execution time and latency of the next request for each step. When a request comes, it maximizes the number of steps this request cover within the tolerated time, thus achieves the maximize consistency. However, the stale read rate of this system is unpredictable. Tlili et al. [7] proposed that a master peer is assigned by the lookup service of the DHT. The master node holds the last update timestamp to avoid reading from a stale replica. However, this system cannot get the precise result of the consistent data and read/write execution time will reduce compared with encoded data on DHT.

There are two parts in the system. These are read and write operations for automatic adjustment of the Consistency Level of Distributed Key-value Storage by a Replica Selection Approach. For the write execution time, data is distributed among the nodes in the cluster using Distributed Hash Tables (DHT) that the atomic ring maintenance mechanism over lookup consistency. DHT is mentioned in Section 5. To get the consistent data into the read performance, two algorithms are proposed to Section 4.

## 3. Proposed Architecture

Read and Write operations of the distributed key_vlaue storage system to dynamically adjust the consistency level are mentioned in this section. In the write part of the system, Data is distributed among the nodes in the cluster using Consistent Hashing based Function. Consistent Hashing is a widely used technology in distributed key-value storage system. It is a good solution when the number of nodes changes dynamically. And when the virtual node is combined, the load balancing problem will also be solved. Consistent hashing is the algorithm the helps to figure out which node has the key. The algorithm guarantees that a minimal number of keys need to be remapped in case of a cluster sizes change.

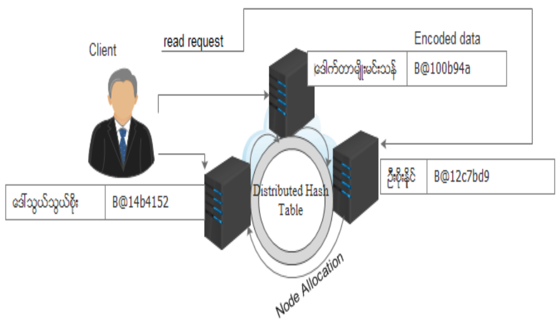

**Figure1. System Architecture**

In figure 1, when the write requests are incoming to the coordinator node. The coordinator node performs the encoding input data is saved for the Cassandra cluster by consistent hash on different quorum nodes.

When the client reads a file, it sends a read request for the coordinator Node. The Coordinator Node collects the list of DataNodes that it can retrieve data by using the replica selection algorithm described in the next section. When sufficient fragments have been obtained, the Coordinator Node decodes the data and supplies it to the read application request.

## 4. Algorithm Definition

The replica selection algorithm has two parts. It includes (i) searching the nearest replica and (ii) selecting consistent replicas. In algorithm_1, the coordinator node sends the request message to each replica and latencies of different replicas are listed in the read latency map. And it chooses the lowest latency of replicas of this map.

In algorithm_2, the replica selection algorithm in the coordinator node chooses the consistent replica of the nearest replicas.

```
1.Input: Replicas in DHT //Distributed Hash
      Table
2.Output: Nearest Replica NR
3.Set latencyCost= MAX_VALUE;
4.Set lowestLC [] =null; //Initialize return lowest
      latency replica
5.For each r in DHT
6.Begin
7.Set latencyCost=getLatencyCost (RF_r, job);
8.If (latencyCost<=MAX_VALUE)
      Then//MAX_VALUE =threshold values
9.MAX_VALUE =latencyCost;
10.LowestLC. add (RF_r);
11.End
12.End for
13.Return lowest LC //nearest replica NR
```

**Algorithm 1: Search the nearest Replica**

Search the nearest Replica part executes in two stages. First, all replicas are sorted based on their physical location, so that all replicas of the same rack and then the same data center as the source are at the top of the list. Second, the latencies are computed from the local node (originator of the query) to all other nodes. If the latency cost is greater than a threshold of the closest node, then all replicas are sorted based on their latency costs. Finally, the top replicas of the list are chosen.

Firstly, total numbers of replicas are listed as input (line1). The threshold value is set at the latency cost of line3. In line8, the coordinate node contacts every other replica with request messages. The round tripped to the time it takes from the request until the reply is passed through the following equation.

$T_{total} = RTT_{requsest}/_2 T_{processing} + RTT_{reply}/_2$

$T_{total}$ is used to get the latency cost of computing, data nodes in algorithm1. These latency costs are used when the local node needs to forward the client request to other replicas.

And then total times taken from different replicas are listed in latencyCost (line8). Finally algorithm1 returns the list of lowest latency cost of the replicas of lowestLC as output to client. (line14).

```
1.Input: Nearest Replica NR= {NR_1, NR_2,… NR_n}
2.Output: Consistent Replicas
3.For each Nearest Replica NR_i
4.Begin
5.Set RCL=2//ConsistencyLevel.QUORUM
6.Set noOfConsistentRead = 0
7.While (noOfConsistentRead <= RCL)
8.If(stalerate<=maxStalteRate) Then
9.consistentRead.add(NR_i)
10.noOfConsistentRead++;
11.Return consistentRead;
12.End for
13.End
```

**Algorithm 2: A Consistent Replica Selection**

In algorithm_2, the set of the nearest replicas is collected as input that comes from output of algorithm1 by computing latency costs. And then algorithm2 sets the read consistency level (RC) that the client will need the most up-to-date information. Read/Write latencies of different replicas are listed in history file on the coordinator node.

This algorithm determines the number of consistent replica nodes, one read request should select in real-time, according to calculate arrival times of nearest update request and the processing order of read request and write request in different replicas.

For computing stale rate of algorithm_2, In this section, we will use PBS to simulate more cases of replica number n, read consistency level r, and write consistency level w to prove the conclusion. Bailis et al proposed PBS to predict the consistency [2]. They believed the quorum size impacts the consistency significantly and given a formula to calculate the probability of k-staleness [2].

## 5. Analysis of Read/Write execution time

The read/write execution time of consistency level is tested by using a Cassandra cluster of VMware Ubuntu 14.04 LTS i386. The processor is Intel (R) Core (TM) i7-4770 CPU @ 3.40 GHz. Installed memory (RAM) is 4.00GB.

The staff data onto Ministry of Higher Education is used on Cassandra cluster. Staff information is described by Unicode in “staff.csv”.

When importing data onto the csv to Cassandra, Java hector code truncates the input csv data with a comma (",") lined by line. And then the output csv data are exported on Cassandra.

Cassandra supports Unicode, but Hbase does not support it. Therefore, staff data can be tested for Cassandra cluster. Unicode is the international accepted standard of the World Wide Web Consortium, the main international standards organization for the World Wide Web. And it also makes that it is extremely easy to translate the Wikipedia's interface. And Unicode fonts support 11 languages that use the Myanmar script: Burmese, 2 liturgical languages: Pali and Sanskrit, 8 minority languages: Mon, Shan, Kayah, four Karen languages and Rumai Palaung [10]. It was officially released by Myanmar Natural Language Processing (NLP) Research Center joining existing Myanmar Unicode 5.1.

Ubuntu14.04 LTS is installed on three servers and one client by Cassandra clusters. There are 203 rows and 40 columns from csv file are inserted into Cassandra clusters with Consistent Hashing DHT (Distributed Hash Table).

DHT is one of the fundamental algorithms used in distributed scalable systems. DHT deals with the problem of locating data that are distributed among a collection of machines. In the general case, a lookup service may involve full-content searching or a directory-services or structured database query approach of finding data record that matches multiple attributes. Lookup service similar to a hash table: (key, value) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes [4]. When the data is written to other nodes, data is encoded and queries can be parallel distributed to connected nodes by using DHT table. And Replication Factor (RF), Read Consistency Level (RCL) and Write Consistency Level (WCL) are defined by changing the consistency level (one, two and quorum: Replication Factor/2+1) and tested by Java hector code. Write execution time of servers and read time of the client according to consistency level are shown in figure2 and figure 3. In figure 2, the encoded data by writing execution time for distributed DHT nodes are better than simple DHT. In figure 3, DHT is a widely solution to search the nearest neighbor node that transforms the data to obtain short code consisting of a sequence bits.

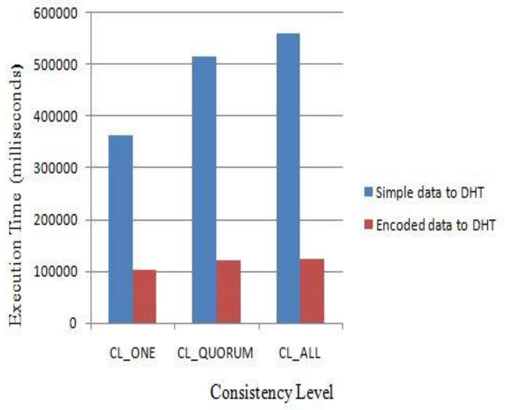

**Figure2. Write execution time**

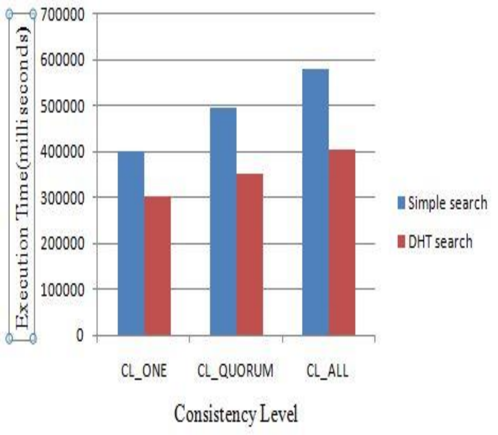

**Figure3. Read execution time**

## 6. Conclusion

The proposed system presents the performance of the key-value data storage of staff data onto DHT by adjusting the consistency level. In defining these consistency levels, two algorithms are proposed to choosing the consistent replicas of different clusters by searching the nearest replica and selecting the consistent replica. The proposed algorithms can determine the minimal number of consistent replicas of reading request needs to contact in real time and thus improve the system performance as a result of reduced read/write execution time.

## 7. Future Work

In the future, the proposed algorithms will be validated on Cassandra clusters of OpenStack Cloud Storage. And latency,bandwidth and throughput will be compared with measured values and will compute the stale read rate for consistent replicas by adding more nodes and more dataset size on Cassandra distributed key-value data storage. And the probability of stale rate, read/write execution time, latency cost and storage cost of the system will be compared with existing systems.

## References

[1] Agrawal, D., El Abbadi, A.: The generalized tree quorum protocol: An efficient approach for managing replicated data. ACM Trans. Database Syst. 17(4), 689.

[2] Bailis P, Venkataraman S, Franklin MJ, Hellerstein JM, Stoica I. Probabilistically bounded staleness for practical partial quorums. Proc VLDB Endowment. 2012;5(8):776–787.

[3] H. Chihoub, S. Ibrahim, G. Antoniu and M. S. Perez, "Harmony: Towards Automated Self-Adaptive Consistency in Cloud Storage", IEEE International Conference on Cluster Computing, September 24-28; Beijing, China , 2012.

[4] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen, "A Survey on Learning to Hash", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 13, NO. 9, APRIL 2017.

[5] Malkhi, D., Reiter, M.: Byzantine quorum systems. pp. 569_578. STOC '97, ACM (1997), http://doi.acm.org/10.1145/258533.258650.

[6] P. Garefalakis , P. Papadopoulos, I. Manousakis, and K.Magoutis, "Strengthening Consistency in the Cassandra Distributed Key-Value Store", International Federation for Information Processing 2013.

[7] Tlili, M., Akbarinia, R., Pacitti, E., Valduriez, P.:Scalllable P2P reconciliation infrastructure for collaborative text editing, Second International Conference on Advances in Database Knowledge and Data Applications (DBKDA), pp. 155_164, April 2010.

[8] W. Vogels, "Eventually consistent", CACM, 52:40–44, 2009.

[9] Y. Zhu and J. Wang. Malleable, "Flow for Time-Bounded Replica Consistency Control", OSDI Poster, October 8-10; Hollywood, USA , 2012.

[10] https://wikivisually.com/wiki/Burmese_ (language)