MAP REDUCE BASED ON CLOAK DHT DATA REPLICATION EVALUATION

MamadouDiarra and TelesphoreTiendrebeogo

Department of Mathematics and Computer Science, Nazi Boni University, Bobo-Dioulasso, Burkina Faso

ABSTRACT

Distributed databases and data replication are effective ways to increase the accessibility and reliability of un-structured, semi-structured and structured data to extract new knowledge. Replications offer better performance and greater availability of data. With the advent of Big Data, new storage and processing challenges are emerging.

To meet these challenges, Hadoop and DHTs compete in the storage domain and MapReduce and others in distributed processing, with their strengths and weaknesses.

We propose an analysis of the circular and radial replication mechanisms of the CLOAK DHT. We evaluate their performance through a comparative study of data from simulations. The results show that radial replication is better in storage, unlike circular replication, which gives better search results.

KEYWORDS

Replication, Big Data, MapReduce, CLOAK DHT, Load balancing

1. INTRODUCTION

As the volume of information has exploded, new areas of technology have emerged that offer an alternative to traditional database and analysis solutions. A few servers with very large storage capacities and processing power are no longer feasible at reasonable costs.

This study focuses on a distributed processing approach to overcome the limitations of Hadoop MapReduce, and to free ourselves from a cost equation in relation to the innovationneeds driven by Big Data.

Our objective is to design a distributed data processing model inspired by the CLOAK DHT [1] [2] [3] and MapReduce [4] [5] [6].

We will evaluate its performance through simulations by comparing it to existing solutions.

Our study focuses on the following aspects:

1. Our first contribution is a methodology to evaluate the replication mechanisms of the CLOAK DHT. Our approach is based on a study of some performance indicators, a simulator and an analysis of the collected measurements.
2. Our second contribution consists in comparing the circular and radial replication mechanisms offered by the CLOAK DHT.

3. The third contribution consists in studying the quality of the results produced by the replication analysis and proposing an adequate mechanism for our model.

Our article will be organized as follows:

1. A first step, we discuss the relative works;
2. A second step, we describe our model of the distributed processing platform;
3. The third part, we analyse the results of the different simulations.

2. RELATED WORKS

2.1. MapreducePlatform

Popularised by Jeffrey Dean and Sanjay Ghemawat, MapReduce is a massively parallel programming model adapted to the processing of very large quantities of data which is based on two main stages Map and Reduce. Its working principle consists of splitting the initial volume of data V into smaller volumes v_i, of pairs (key, value), which will be handled separately on several machines [4] [5] [6].

- Map (InputKey k, InputValue v) → (OutputKey * IntermediateValue list)
- Reduce (OutputKey, IntermediateValue list) → OutputValue list

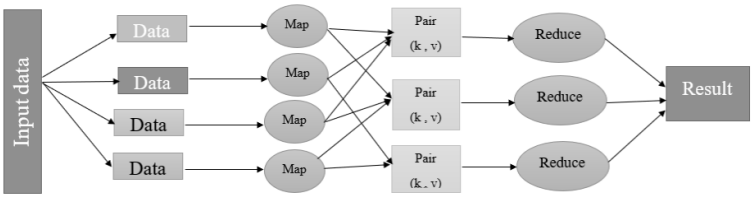

Figure 1. MapReduce Execution Flow [7]

2.2. Typology of a Hadoop cluster

Apache Hadoop is an ecosystem of the Apache Software Foundation [4]. Hadoop Data File System (HDFS) is the distributed file storage system of Hadoop. It has a master/slave architecture, allowing native file storage (image, video, text files etc...) by spreading the load over several servers. An HDFS cluster architecture is based on two essentials functions [4], [5]:

1. Masters: NameNode, Job Tracker and SecondaryNode:

   - The NameNode: Manages distributed blocks of data.
   - The Job Tracker: Assigns MapReduce functions to the various “Task Trackers”.
   - The SecondaryNode: Periodically makes a copy of the data of the “NameNode” and replaces it in case of failure.

2. Slaves: DataNode and Task Trackers:

   - The Task tracker: the execution of MapReduce orders.
   - The DataNode: the storage of data blocks.

2.3. DHT –MapReduce

2.4. Distributed Hash Tables

DHT is a technology for constructing a hash table in a distributed system where each piece of data is associated with a key and is distributed over the network. DHTs provide a consistent hash function and efficient algorithms for storing STORE (key, value) and locating LOOKUP (key) of the node responsible for a given (key, value) pair.

It is therefore important to specify that only a reference of the declared object (Object IDentifier (OID)) is stored in the DHT. Concerning the search for an object, only by routing the lookup key can the set of associated values be found [1], [2], [7].

In practice, DHTs are substituted by "overlay" networks on top of the physical networks, thus reducing the size of the table at each node while considerably increasing the efficiency of the lookup algorithm.

DHTs have been widely studied because of their attractive properties: efficiency and simplicity with Chord [8], controlled data placement with SkipNet [9], Pastry [10] routing and localization, and better consistency and reliable performance with Kademlia [11].

2.5. Architecture and Components of DHT-Mapreduce:

Although the SecondaryNode is a solution in the second version of Hadoop to take over in case of NameNode failure, other solutions were proposed such as:

- Distributed MapReduce using DHT: [12] It is a framework inspired by P2P-based MapReduce architecture built on JTXA [13]. The basic idea is to provide another master node as the backup node. When the master mode is crashing, another master node (backup node) could take over the remaining tasks of the job. In that distributed framework approach the multiple master nodes not only are backup nodes, but also act as individual master nodes and work collaboratively.
- ChordMR: [14] is a new MapReduce system, which is designed to use a Chord DHT to manage master node churn and failure in a decentralized way. The architecture of ChordMR includes three basic roles: User, Master and Slave. The User nodes are responsible for submitting jobs. The master nodes, organised in Chord DHT, are responsible for the allocation and execution of tasks. However, the slave nodes, which are responsible for executing the Map/Reduce jobs, are still maintained in their traditional function as on the Hadoop platform. ChordMR has no scheduler or central management node, unlike HDFS and Apache Cassandra [15].
- ChordReduce: [16] ChordReduce is a simple, distributed and very robust architecture for MapReduce. It uses Chord DHT to act as a fully distributed topology for MapReduce, eliminating the need to assign explicit roles to nodes or to have a scheduler or coordinator. ChordReduce does not need to assign specific nodes to the backup job; nodes backup their jobs using the same process as for all other data sent around the ring. Finally, intermediate data returns to a specified hash address, rather than to a specific hash node, eliminating any single point of failure in the network.
- Chord-based MapReduce: [17] The present invention relates to a distributed file system and more particularly to a Chord DHT based MapReduce system and method capable of achieving load balancing and increasing a cache hit rate by managing data in a double-layered ring structure having a file system layer and an in-memory cache layer based on a Chord DHT.

2.6. Data replication

The objective of data replication is to increase data availability, reduce data access costs and provide greater fault tolerance [18].

By replicating datasets to an off-site location, organisations can more easily recover data after catastrophic events, such as outages, natural disasters, human errors or cyberattacks.

2.7. CLOAK DHT

CLOAK DHT, the basis of our work, is based on a hyperbolic tree constructed in hyperbolic space (hyperboloid) and projected stereographically into a Poincaré disk (Euclidean plane) of radius 1 centred at the origin where the tree node uses a virtual coordinate system [1].

2.7.1. Anatomy of Writes and Reads

CLOAK DHT uses local addressing and gluttonous routing, which is carried out by using hyperbolic virtual distances. When a node η seeks to join the network, it calculates the distancebetween its correspondent μ and each of its own neighbours, and selects, *in fine*, the neighbour *v* with the shortest hyperbolic distance.

The distance between any two points*u* and *v*, in the hyperbolic space H, is determined by equation 1:

$$d_{\mathbb{H}}(u,v) = arccosh(1+2\lambda) \;\; with \;\; \lambda = \frac{|v-u|^2}{(1-|u|^2)*(1-|v|^2)}$$

Equation 1. Distance between any two points u and v

DHTs store information in pairs (key, value) in a balanced way on all nodes of the overlay network.
However, in the CLOAK DHT, not all nodes in the overlay network necessarily store pairs. Only the elected nodes store pairs.

In a CLOAK DHT, stored queries are named STORAGE and search queries are named LOOKUP. The figure 3 shows how and where a given pair is stored in the overlay.

A node depth in the addressing tree is defined as the number of ascending nodes to be traversed before reaching the root of the tree (including the root itself). When creating the overlay network, a maximum depth $p_{max}$is chosen. This value is defined as the maximum depth that any store in the tree can have. All nodes that have a depth less than or equal to the p_{max} can store (key, value) pairs and thus be starters. All nodes with a depth greater than p_{max} do not store pairs.

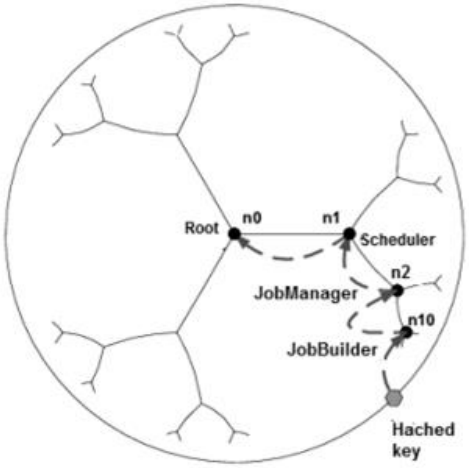

Figure 1. DHT Based on Hyperbolic Geometry

2.7.2. Storage Mechanism

When a node wants to send a storage request, the first server, it is connected to considers a request as a Big Data object and therefore generates an OID using its name with the SHA-512 algorithm. It divides the 512-bit key into 16 equal-sized 32-bit subkeys for redundant storage.

Similarly, the division of the key into 16 subkeys is arbitrary and could be increased or decreased depending on the redundancy required. The node, then selects the first subkey and calculates an angle by a linear transformation is given by equation 3. From this angle, the node determines a virtual point v on the Poincaré disk according to equation 4. Figure 3 shows how and where a node is stored in the overlay network [2].

2.7.3. Replication Mechanisms

CLOAK DHT has two data replication mechanisms: circular and radial:

- **Circular Replication:** To have replicas, the OID is split into keys by the SHA-512 algorithm, so the 512-bit key is split into 16 subkeys of 32 bits. Thus, 16 different storage angles are used and this improves the search success rate as well as the homogeneity of the distribution of peers on the elected nodes. The division of the key into $r_c = 16$ subkeys is arbitrary and can be increased or decreased depending on the need for redundancy. We refer to this redundancy mechanism as circular replication. In general, from a pair named P and the hash of its key with the SHA 512 algorithm, we obtain a 512-bit key which we split into an arbitrary number r_c of subkeys.
- **Radial replication:** For redundancy reasons, a pair can be stored in more than one storage device within the storage radius. A storage device can store a pair and then redirect its storage request to its ascendant to store it in turn. The number of copies of a pair along the storage spoke can be an arbitrary value, r_r defined at the creation of the overlay network. We use the term radiale replication to designate the replication of the pair P along the storage radius. In effect, starting from the first existing storage closer to the circle, we replicate the pair r_r times on the ascending nodes towards the root of the addressing tree. We stop, either when we reach a number of radial replications equal to $r_r = \lfloor \log(n)/\log(q) \rfloor$ (where q corresponds to the degree of the hyperbolic addressing tree and n the size of the overlapping networks), or when we reach the root of the addressing tree.

3. OUR CONTRIBUTIONS

3.1. Summary of the Invention

CLOAK-Reduce is built by implementing, on the CLOAK DHT, a module to create and execute MapReduce operations. It is a distributed model that exploits the advantages of the CLOAK DHT and MapReduce. CLOAK DHT to submit Map () and Reduce () jobs in a balanced way, thanks to the replication mechanisms it offers in addition to the task scheduling strategy we provide.

The tree structure of the CLOAK DHT allowed us to define a hierarchical and distributed load balancing strategy at two levels: Intra-scheduler or Inter- JobManagers and Inter-schedulers

Our model consists of a root node, three schedulers, several candidate nodes or JobManagers candidates and builder nodes or JobBuilders:

- **Root node:** The root is a node of coordinate (0,0), the minimal depth (D_0) tree, of the Poincare´ disk. It allows to:

  - ✓ Keep tasks that cannot be executed immediately in the root task spool;
  - ✓ Maintain schedulers load information;
  - ✓ Decide to submit a task to a scheduler.

- **Schedulers:** The schedulers of depth (D_1), have the function of:

  - ✓ Maintain the load information of all their JobManagers;
  - ✓ Decide global balancing of their JobManagers;
  - ✓ Synchronise their load information with their replicas to manage their failure and the others schedulers for load balancing.

- **JobManagers:**JobManagers candidates have a minimum depth ($\geq D_2$). They are elected for:

  - ✓ Supervise MapReduce jobs running;
  - ✓ Manage the load information related to JobBuilders;
  - ✓ Maintain their load state;
  - ✓ Decide local load balancing with their circulars replicas JobManager candidate of the same scheduler;
  - ✓ Inform JobBuilders of the load balancing decided for the backup intermediate jobs.

- **JobBuilders:**JobBuilders are of minimum depth ($\geq D_3$). Their function is to:

  - ✓ Execute MapReduce jobs;
  - ✓ Synchronize the images of their different works as they go on their radial replicas of the depth > D_3.
  - ✓ Keep the information on the state of their charge up to date;
  - ✓ Update this workload information at the JobManager;
  - ✓ Perform the balancing necessities ordered by their JobManagers.

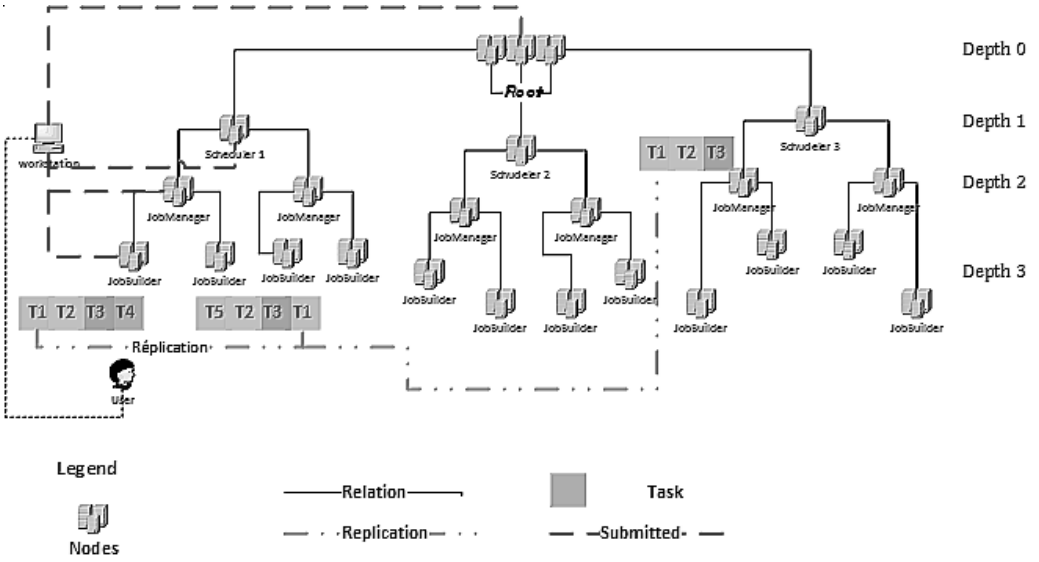

Figure 4. CLOAK-Reduce Architecture

3.2. Performance Analysis

3.2.1. Experimental setup

For the data collection, circular and radial replication mechanisms, in a dynamic network with 10% of churns, where necessary. In order to study the scalability of the system, we opted to consider a random number of nodes as follows: 100, 300, 500 and 1000.

Each simulation phase lasted 2 hours with an observation of the results every 10 minutes, i.e. a total of 12 observations per phase.

The data processed in this paper is the result of ten (10) independent runs of each simulation phase in order to extract an arithmetic mean.

This collection allowed us to conduct a comparative study on the storage and lookup performance of radial replication, circular replication and both replication mechanisms. Next, we compared the mean number of storage and lookup hops as a function of scalability.

This limitation is due to the limited resources to executing the simulation.

- The first phase: We set the circular replication to (01) and varied the radial replication from (02) to (05).
- The second phase: We set the radial replication to one (01) and varied the circular from two (02) to five (05).

To achieve our simulation objectives, we opted to use the PeerSim simulator.

3.2.2. Simulator

- PeerSim [13] is a simulator whose main objective is to provide high scalability, with network sizes up to 1000000 nodes, which characterises its dynamics and extensibility. Its

modularity facilitates the coding of new applications. The PeerSim configuration file has three types of components: protocols, dynamics and observers.

- Simulation is a widely used method for performance evaluation. It consists of observing the behaviour of a simplified model of the real system with the help of an appropriate simulation program which will result in graphs that are easy to analyse and interpret. This method is closer to the real model than analytical methods, and is used when evaluation of direct measure becomes very expensive [17].

3.2.3. Some performance indicators

We are interested by the routing efficiency through the study success latency of storage, lookup and their hops count of Big Data objects. The following performance metrics have been analysed in order to measure the performance.

- **Routing Efficiency** (RE): Also known as lookup success rate is calculated by taking the ratio of total number of lookup messages managed to succeed before timeout and the total number of lookup messages broadcasted by the system. This metric is calculated using. RE = (Successful storage or lookup / Total messages) × 100
- **Hop count** (HC)**:** Or workload per nodes is calculated by taking the ratio of total broadcasted complex query messages and total number of nodes required to successfully forward the messages to the destination node. This is defined in:
  HC = Total broadcast MSG / Total nodes required to forward MSG
- **Success Latency**: The success latency is defined as the time taken for a packet of data to get from one designated point to another and sometimes measured as the time required for a packet to be returned to its sender.
- **Variance**: The variance of a series, noted V, is the average of the squares of the deviations of each value from the mean m of the series.
  $V= (n_1 \times (x_1 - m)^2 + n_2 \times (x_2 - m)^2 + ... + n_p \times (x_p - m)^2)/ N$
- **Standard deviation:** The standard deviation of a series is the number, noted σ,
  with $\sigma = \sqrt{V}$, where V is the variance of the series.
- **Max / Min**: Represented respectively the maximum and minimum mean of the results of the ten (10) independent runs of each simulation phase.

3.2.4. Storage performance evaluation

3.2.4.1. Scalability analysis

Figure 5 shows the random generation of 10,000 nodes in the hyperbolic plane. It shows an almost uniform distribution of nodes around the root. This scalability of the nodes implies that our system builds a well-balanced tree which will make it easier to manage the MapReduce jobs.

Figure 5. Nodes distribution on Poincaré disk

3.2.4.2. Circular storage analysis

Table 1, Circular replication 5 shows a better result with a storage success mean of 67.38% with a standard deviation of 11.66%.

Table 1. Circular storage success ratio (%)

Replicas	Storage Success		Mean	σ
	Max	Min		
2	82.04	32.76	54.37	14.73
3	84.74	40.12	58.04	14.26
4	89.75	49.25	62.48	12.34
5	90.27	54.78	67.38	11.66

Figure 6 shows the evolution of the successful storage ratio during a circular replication as a function of time. These different curves are the arithmetic mean of the (10) independent replications in each simulation phase. The curves are globally decreasing.

We can conclude that the storage success ratio increases with the number of replications.

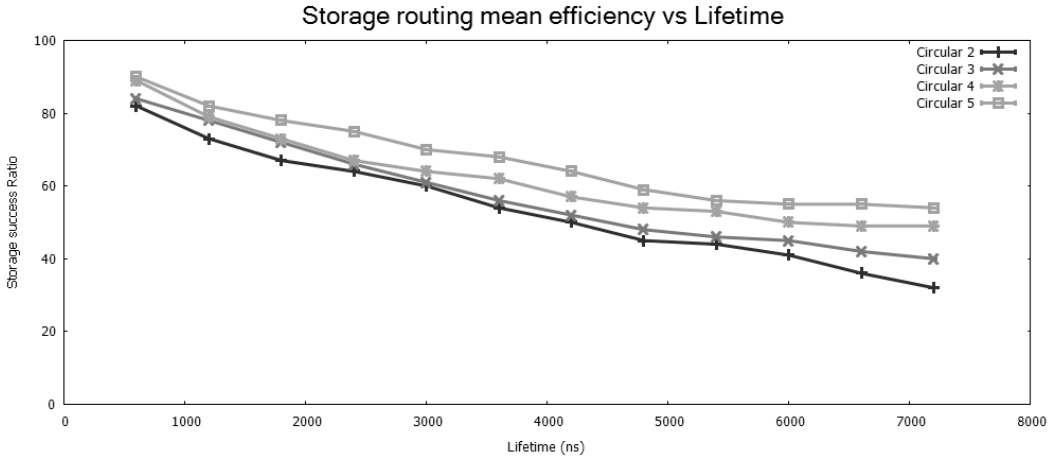

Figure 6. Circular storage efficiency

3.2.4.3. Radial storage analysis

Table 2, Radial replication 5 presents a better result with a mean storage success ratio of 69.35% with a standard deviation of 10.86%. We can conclude that the higher the number of replications, the higher the storage success ratio.

Table 2. Radial storage success ratio (%)

Replicas	Storage Success		Mean	σ
	Max	Min		
2	82.91	40.05	57.41	13.42
3	84.41	42.64	59.67	13.37
4	87.83	50.94	64.75	11.69
5	91.27	56.07	69.35	10.86

Figure 7 shows the variation of the storage success ratio during radial replication as a function of time. The curves are globally decreasing.

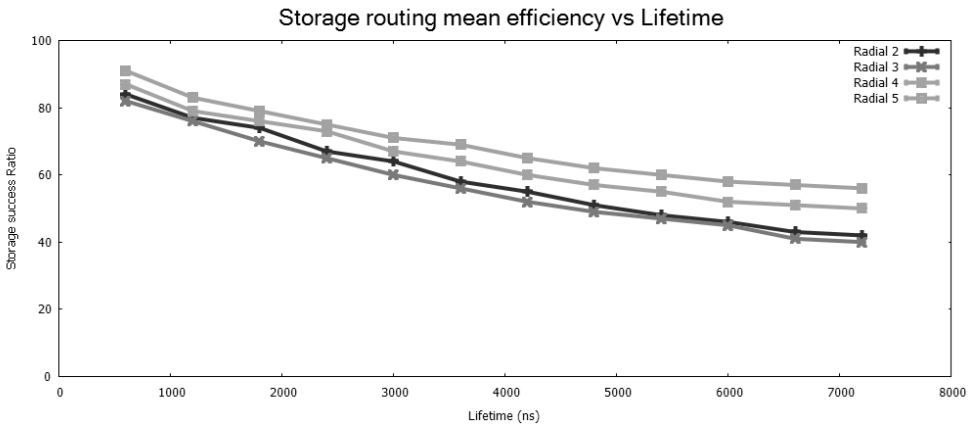

Figure 7. Radial storage efficiency

3.2.4.4. Circular and Radial storage analysis

Figure 8, the curves show broadly the same trend. The two replication mechanisms have a standard deviation difference of 0.8%.

However, radial replication 5 is slightly better than circular replication 5 with an average of 1.97% more storage success.

Table 3. Storage success ratio (%)

Replicas	Storage Success		Mean	σ
	Max	Min		
C5R1	99.25	56.43	74.68	13.63
C1R5	91.27	56.07	69.35	10.86

Furthermore, increasing the number of replications generally increases the storage success ratio. From the above analysis, we can say that radial replication 5 has a better storage ratio. Also, from the previous comparisons we can confirm that as the number of replication increases, the success ratio improves.

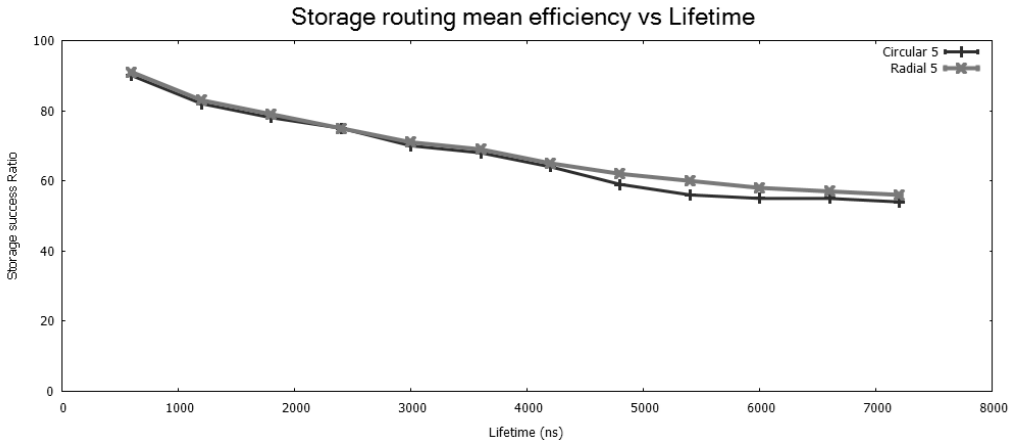

Figure 8. Storage efficiency

3.2.5. Lookup performance evaluation

3.2.5.1. Circular replication

Table 4, Circular replication 5 in Figure 9 shows a better result with an average search success of 77.73% with a standard deviation of 12.68%. We see that the higher the number of replications, the higher the lookup success ratio.

Table 4. Circular lookup success ratio (%)

Replicas	Storage Success		Mean	σ
	Max	Min		
2	97.58	42.85	67.76	18.96
3	97.62	48.68	67.79	15.02
4	98.31	53.91	70.49	13.29
5	100	60.15	77.73	12.68

Figure 9 shows the evolution of the lookup success ratio during a circular replication as a function of time. The curves are globally decreasing.

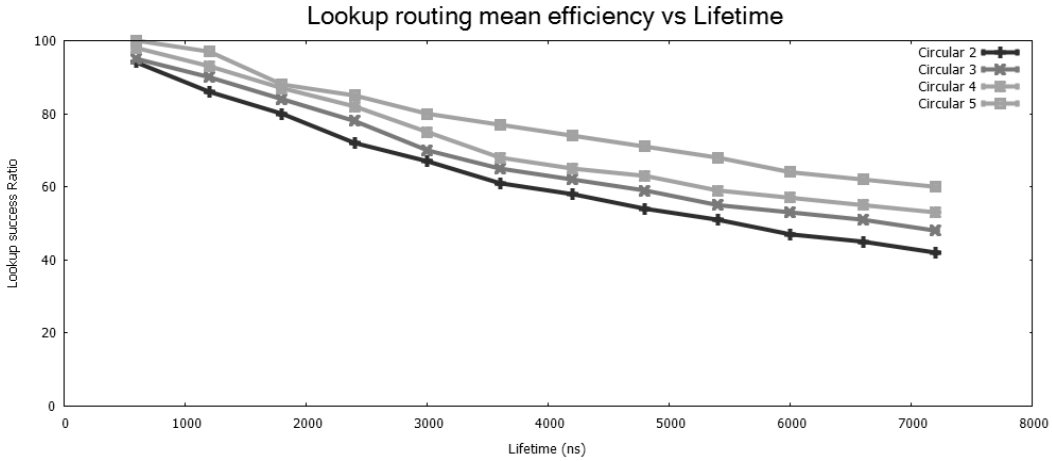

Figure 9. Circular lookup efficiency

3.2.5.2. Radial replication

Table 5, Circular replication 5 shows a better result with lookup success mean ratio of 74.68% and a standard deviation of 13.63%.

Table 5. Radial lookup success ratio (%)

Replicas	Storage Success		Mean	σ
	Max	Min		
2	94.96	44.26	64.25	16.08
3	97.36	48.28	68.33	15.23
4	98.24	51.21	70.14	15.18
5	99.25	56.43	74.68	13.63

Figure 10, We see that the higher the number of replications, the higher the Lookup success ratio.

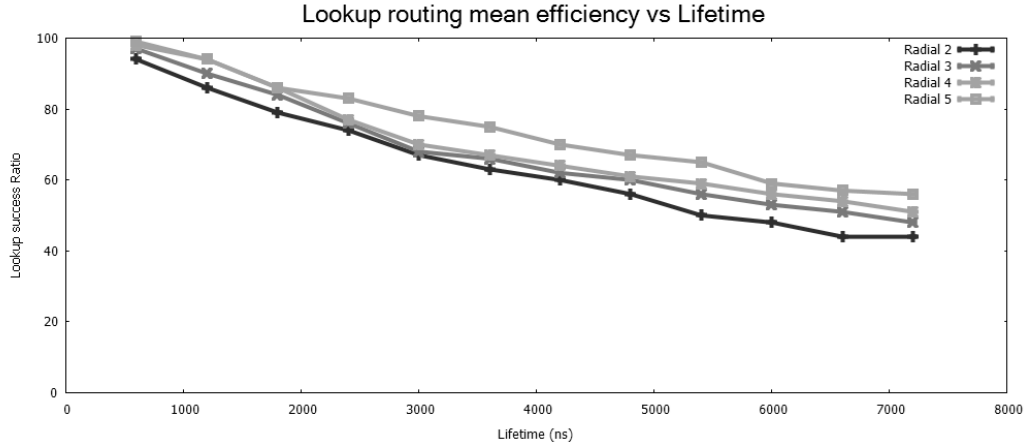

Figure 10. Radial lookup efficiency

3.2.5.3. Lookup Analysis

The comparative study in Figure 11 shows that circular replication 5 Lookup success ratio 77.73%, which is the highest of the results obtained, and a standard deviation of 12.68%, which is the lowest of all replications. It is followed by radial replication 5 with Lookup success ratio 74.68% and a standard deviation 13.63%. Both replication mechanisms have curves that show broadly the same trend with a slight advantage of circular replication 5.

Furthermore, increasing the number of replications generally increases the Lookup success ratio. In the Figure 12, the radial replication hops count ratio increases from 3.31 to 5.70 with a standard deviation of 1.10 % when the nodes increase from 100 to 1000, while the circular replication increases from 3.54 to 5.73 with a standard deviation of 1.20% with the same variation of nodes. Radial replication has a slightly better storage performance than circular replication.

Table 6. Lookup success ratio (%)

Replicas	Storage Success		Mean	σ
	Max	Min		
C1R5	99.25	56.43	74.68	13.63
C5R1	100	60.15	77.73	12.68

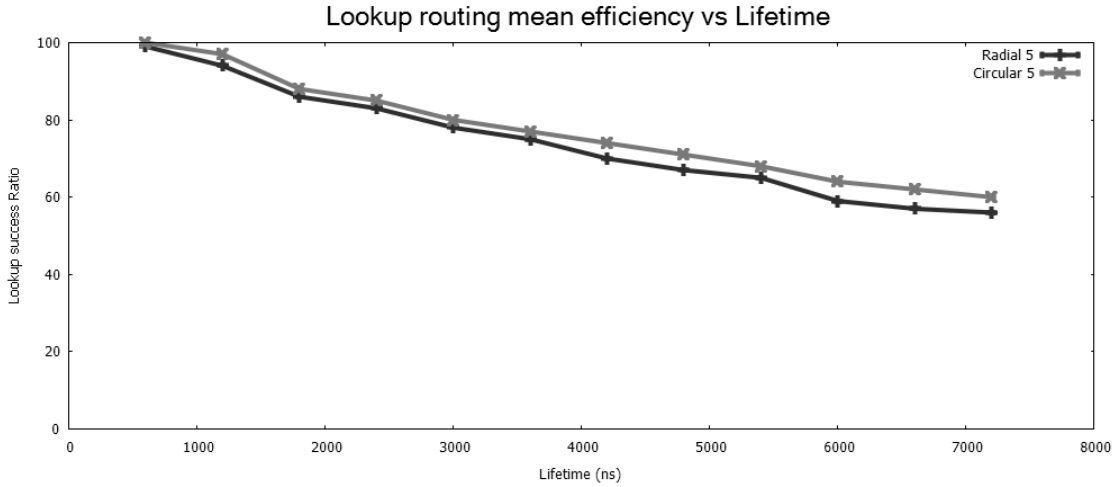

Figure 11. Lookup efficiency

3.2.6. Hops mean by system size performance

This part of our paper highlights a comparative study of the storage and lookup hops mean, according to the size of the network. The network size limitation of 1000 nodes is due to the resources we have at our disposal.

3.2.6.1. Storage hops success

- Circular replication: The hops count mean, standard deviation is 1.40 between 100 and 300 nodes. It decreases to 0.62 between 300 and 500 nodes and is 0.37 between 500 and 1000 nodes.

- Radial replication: The hops count mean, standard deviation is 1.04 between 100 and 300 nodes. It drops to 0.89 between 300 and 500 nodes and 0.26 between 500 and 1000 nodes.

Table 7, Radial replication hops count ratio increases from 3.31 to 5.70 with a standard deviation of 1.10 % when the nodes increase from 100 to 1000, while the circular replication increases from 3.54 to 5.73 with a standard deviation of 1.20% with the same variation of nodes.

Table 7. Storage hops success ratio (%)

Replicas	Hops Mean		Mean	σ
	C5R1	C1R5		
100	3.54	3.31	3.43	0.12
300	4.57	4.71	4.64	0.07
500	5.47	5.33	5.40	0.07
1000	5.73	5.70	5.72	0.02

Figure 12, Radial replication has a slightly better storage performance than circular replication.

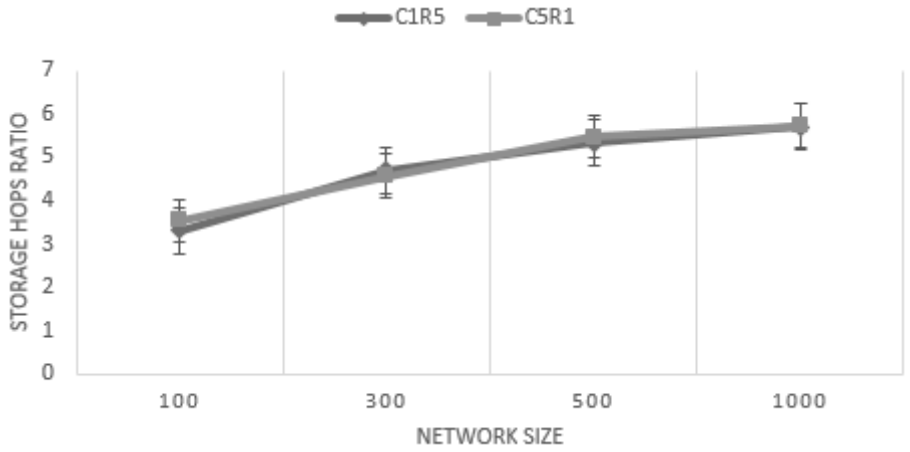

Figure 12. Storage hops count mean

3.2.7. Lookup hops success

- Radial replication increases from 3.90 to 5.84 with a standard deviation of 0.97% when the number of nodes increases from 100 to 1000. We see a small increase of hops number means, which is 1.10 between 100 and 300 nodes, 0.58 between 300 and 500 nodes and 0.27 between 500 and 1000 nodes.
- Circular replication increases from 3.47 to 5.35 with a standard deviation of 0.94% when the number of nodes increases from 100 to 1000. We find a small increase of hops number means, which is 1.05 between 100 and 300 nodes, 0.59 between 300 and 500 nodes and 0.25 between 500 and 1000 nodes.

Table 8, the circular replication hops mean changes from 3.47 to 5.35 with a standard deviation of 0.94% when the number of nodes increases from 100 to 1000, while that radial replication changes from 3.90 to 5.84 with a standard deviation of 0.97% when the number of nodes increases from 100 to 1000.

Table 8. Lookup hops success ratio (%)

Replicas	Hops Mean		Mean	σ
	C5R1	C1R5		
100	3.47	3.90	3.60	0.22
300	4.51	4.99	4.75	0.24
500	5.10	5.57	5.37	0.24
1000	5.33	5.84	5.60	0.25

Figure 13, we conclude that radial replication has a slightly better storage performance than circular one

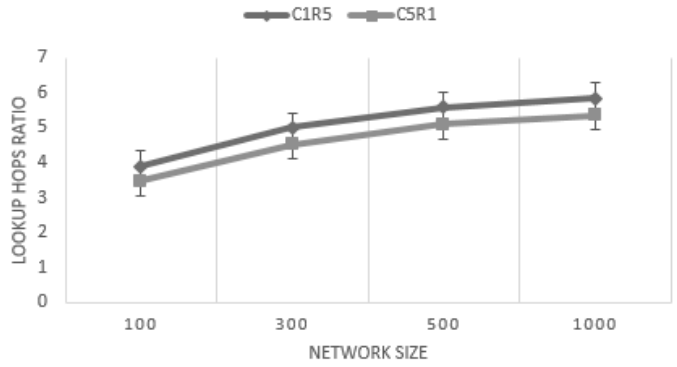

Figure 13. Lookup hops count mean

3.2.8. Hops Success Ratio According to the Type of Replication

Figure 14, the circular replication lookup hops mean has ranged from 3.47 to 5.35 with a standard deviation of 0.94% when the number of nodes increases from 100 to 1000.

It is the lower of the two replication mechanisms.

In the mean number of storage hops, the radial replication mechanism, grows from 3.31 to 5.70 with a standard deviation of 1.2% when the number of nodes increases from 100 to 1000. It has a slightly better mean number of storage hops than circular replication.

Also, from the above, we can say that the standard deviation, mean hop count decreases as the number of nodes increases.

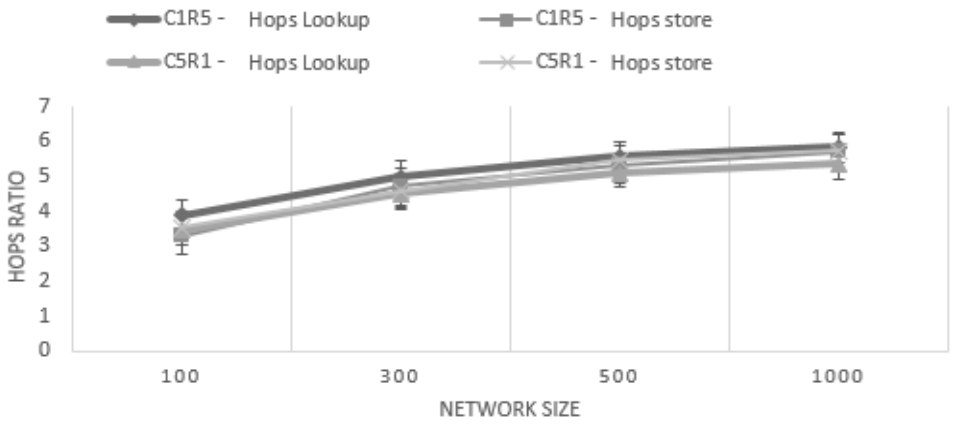

Figure 14. Hops efficiency

4. CONCLUSIONS AND PERSPECTIVES

Our simulations have shown that replications allow a mean storage and lookup success ratio of over 60%. The low mean number of lookups hops and the high lookup success ratio of circular replication is in line. The high storage success ratio of radial replication and its low mean number of storage hops are supported our simulation results.
The main contribution of our work was to find an ideal storage and lookup mechanism for CLOAK-Reduce. Also, this comparative study of the replication mechanisms of the CLOAK DHT allowed us to highlight the efficiency of radial replication storage and circular replication in data lookup.

For future work, we will make a comparative study of the results of this paper with a replication mechanism of five circular replications and five radial circular replications. After this comparison, we will evaluate the dynamic load balancing mechanisms of CLOAK-Reduce.

Then we will implement a fully functional version of CLOAK-Reduce. Finally, we will perform simulations to demonstrate its performance against the architectures mentioned in the related works.

REFERENCES

[1] TelesphoreTiendrebeogo, DaoudaAhmat, and Damien Magoni, (2014) "Évaluation de la fiabilitéd'une table de hachagedistribuéeconstruitedans un plan hyperbolique", Technique et Science Informatique, TSI, Volume 33 - n◦ 4/2014, Lavoisier, pages 311–341

[2] TelesphoreTiendrebeogo and Damien Magoni, (2015) "Virtual and consistent hyperbolic tree: A new structure for distributed database management". In International Conference on Networked Systems, pages 411–425. Springer, 2015.

[3] Tiendrebeogo, Telesphore, (2015) "A New Spatial Database Management System Using an Hyperbolic Tree", DBKDA 2015: 53.

[4] Jeffrey Dean and Sanjay Ghemawat, (2008) "MapReduce simplified data processing on large clusters", Communications of the ACM, 51(1) :107–113, 2008.

[5] Neha Verma, Dheeraj Malhotra, etJatinder Singh (2020) "Big data analytics for retail industry using MapReduce-Apriori framework", Journal of Management Analytics, p. 1-19.

[6] Than ThanHtay and SabaiPhyu, (2020) "Improving the performance of Hadoop MapReduce Applications via Optimization of concurrent containers per Node", In: 2020 IEEE Conference on Computer Applications (ICCA). IEEE, p. 1-5.

[7] TelesphoreTiendrebeogo, MamadouDiarra, (2020) “Big Data Storage System Based on a Distributed Hash Tables system” International Journal of Database Management Systems (IJDMS) Vol.12, No.4/5
[8] Ion Stoica, Robert Morris, David Liben-Nowell, David R Karger, M FransKaashoek, Frank Dabek, and Hari Balakrishnan, (2003) “Chord: a scalable peer-to-peer lookup protocol for internet applications”, IEEE/ACM Transactions on Networking (TON), 11(1) :17–32
[9] Nicholas JA Harvey, John Dunagan, Mike Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman, (2002) “Skipnet: A scalable overlay network with practical locality properties”
[10] Antony Rowstron and Peter Druschel. Pastry, (2001) “Scalable, decentralized object location, and routing for large-scale peer-to-peer systems”, In IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, pages 329–350. Springer.
[11] PetarMaymounkov and David Mazieres, (2002) “Kademlia : A peer-to-peer information system based on the xor metric”, In International Workshop on Peer-to-Peer Systems, pages 53–65. Springer
[12] Chiu, Chuan-Feng, Steen J. Hsu, and Sen-Ren Jan. (2013) "Distributed mapreduce framework using distributed hash table." International Joint Conference on Awareness Science and Technology, Ubi-Media Computing (iCAST 2013, UMEDIA 2013). IEEE.
[13] FabrizioMarozzo, Domenico Talia, and Paolo Trunfio. P2p-mapreduce: Parallel data processing in dynamic cloud environments. Journal of Computer and System Sciences, 78(5) :1382–1402, 2012.
[14] Jiagao Wu, Hang Yuan, Ying He, and Zhiqiang Zou, (2014) “Chordmr : A p2p-based job management scheme in cloud”. Journal of Networks, 9(3) :541.
[15] Carpenter, Jeff, and Eben Hewitt, (2020) Cassandra: the definitive guide: distributed data at web scale. O'Reilly Media.
[16] Andrew Rosen, Brendan Benshoof, Robert W Harrison, and Anu G Bourgeois, (2016) “Mapreduce on a chord distributed hash table”. In 2nd International IBM Cloud Academy Conference, volume 1, page 1.
[17] Nam, Beomseok, (2019) “Chord distributed hash table-based map-reduce system and method” U.S. Patent No. 10,394,782. 27 Aug.
[18] YahyaHassanzadeh-Nazarabadi, AlptekinKüpçü, etÖznurÖzkasap, (2018) “Decentralized and locality aware replication method for DHT-based P2P storage systems”, Future Generation Computer Systems, vol. 84, p. 32-46.
[19] Harold Scott Macdonald Coxeter and GJ Whitrow. (1950) “World-structure and non-euclidean honeycombs”. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 201(1066) :417–437.
[20] Harold Stephen Macdonald, (1954) Coxeter. Regular honeycombs in hyperbolic space.In Proceedings of the International Congress of Mathematicians, volume 3, pages 155–169. Citeseer.

AUTHOR

DiarraMamadou, master degree in information systemsand decision support system in nazi BONI universityBurkina Faso. My interest research topic is imagewatermarking for decision support and multimedia system.

TelesphoreTiendrebeogo PhD and overlay network and assistant professor at Nazi Boni University. I have a master's degree in multimedia and real time system. My current research is on big data and image watermarking