PERFORMANCE EVALUATION OF BIG DATA PROCESSING OF CLOAK-REDUCE

Mamadou Diarra and Telesphore B. Tiendrebeogo

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

ABSTRACT

Big Data has introduced the challenge of storing and processing large volumes of data (text, images, and videos). The success of centralised exploitation of massive data on a node is outdated, leading to the emergence of distributed storage, parallel processing and hybrid distributed storage and parallel processing frameworks.

The main objective of this paper is to evaluate the load balancing and task allocation strategy of our hybrid distributed storage and parallel processing framework CLOAK-Reduce. To achieve this goal, we first performed a theoretical approach of the architecture and operation of some DHT-MapReduce. Then, we compared the data collected from their load balancing and task allocation strategy by simulation.

Finally, the simulation results show that CLOAK-Reduce C5R5 replication provides better load balancing efficiency, MapReduce job submission with 10% churn or no churn.

KEYWORDS

Big Data, DHT, MapReduce, CLOAK-Reduce.

1. INTRODUCTION

Big Data [1] showed the limits of traditional storage and processing frameworks. To solve its problems, many projects have been developed. Hadoop [2] and MapReduce [3] are pioneering software frameworks dedicated to this field. The parallel programming model of Hadoop MapReduce designed for scalability and fault tolerance assumes that calculations occur in a centralized environment with a master/slave architecture.

Also the lack of storage for incoming queries that can be processed at a later time and the problems associated with the churn in early versions of Hadoop, have led Hadoop designers to move towards new resource management policies [4]. But with the heterogeneity of computing frameworks, Apache Hadoop still faces problems of load balancing and data replication [2, 5].
Thus to overcome the storage problem, Distributed Hash Tables (DHTs) were used. However, with the proliferation of data and the heterogeneity of storage and processing nodes, this approach was faced with the same problem of load balancing and replication [6, 7].

While many studies have been conducted on the dynamic load balancing strategy [8, 9] and data replication mechanisms [10, 11] of Hadoop and DHTs, new and innovative hybrid storage and processing frameworks have emerged. Our objective is to show the efficiency of our Big Data framework: CLOAK Reduce. We will evaluate its performance by comparing its load balancing strategy and replication mechanism to some existing solutions. This paper is structured as

follows: in section 2, an overview of different Hadoop approaches for efficient management of massive data. In section 3, some approaches of DHT-MapReduce frameworks used in Big Data management are discussed. Experimental results are presented in section 4. Finally, conclusions are drawn and future work is described in section 5.

2. RELATED WORKS

2.1. Hadoop's Different Approaches

Its scheduler influences the performance of Hadoop greatly. Originally, the main purpose of Hadoop was to perform large batch jobs such as web indexing and log mining. Since its first deployment in 2006, four versions of Hadoop frameworks have been released, namely Hadoop 0.x, Hadoop 1.x, Hadoop 2.x and in 2017 Hadoop 3.x. In Hadoop 0.x and Hadoop 1.x, slot-based resource management was used, while Hadoop 2.x and Hadoop 3.x use a resource management system known as Yet Another Resource Negotiator (YARN) [12].

How does Hadoop 3.x add value over Apache Hadoop? First, let us introduce MapReduce, our dedicated processing framework.

2.2. MapReduce

MapReduce is a framework that enables automatic parallelization and distribution of large-scale computations, while abstracting the "disjointed details of parallelization, fault tolerance, data distribution and load balancing." Its principle consists in splitting the initial volume of data V into smaller volumes v_i, of couples (key, value), which will be handled separately on several machines [3]. To simplify, a user can take a large problem, divide it into smaller equivalent tasks and send these tasks to other processors for computation. The results are returned to the user and combined into a single answer [2].

2.3. Hadoop 1

Hadoop 1.x has a limited scalability of 4,000 nodes per cluster and only supports the MapReduce processing model [5]. Its "master/slave" architecture, consisting of the NameNode and JobTracker masters, is its point of failure. The failure of the NameNode daemon eliminates all accessibility to the Hadoop Distributed File System (HDFS) file manager. JobTracker failure also will cancel MapReduce jobs running and all new jobs to be submitted [12, 13]. In short, the Single Point Of Failure (SPOF) in a Hadoop 1 cluster is the NameNode. Any other churn does not cause data loss or the results of the cluster's NameNode. Hence the need to go to another step in this configuration to save the NameNode metadata.

2.4. Hadoop 2

Hadoop 2.x (Figure 1) has resolved most of the restrictions of Hadoop 1.x. Its resource and node management system, Yet Another Resource Negotiator (YARN) serves as a framework for a wide range of data analytics such as event management, streaming and realtime operations [5]. Hadoop 2.x has therefore combined Hadoop 1.x with other distributed computing models such as Spark, Hama, Giraph, messaging interface, MPI coprocessors and HBase [14]. Hadoop 2.x has scalability of up to 10,000 nodes per cluster and Secondary NameNode with functionality to overcome SPOF [15].

However, Hadoop 2.x has some limits. Its tolerance for failures is managed by replication. HDFS replicates each block three times for multiple purposes by default. A single data node manages many disks. These disks fill up during a normal write operation. Adding or replacing disks can cause significant problems within a data node. For data management, Hadoop 2.x has an HDFS balancer that distributes data across the disks of a data node. Replication is the only way to manage fault tolerance that is not space-optimised. Because if HDFS has six blocks to store, there will be 18 occupied blocks, which is 200% overhead in the storage space. In addition, Hadoop 2.x only supports one active and one Secondary NameNode for the entire namespace.

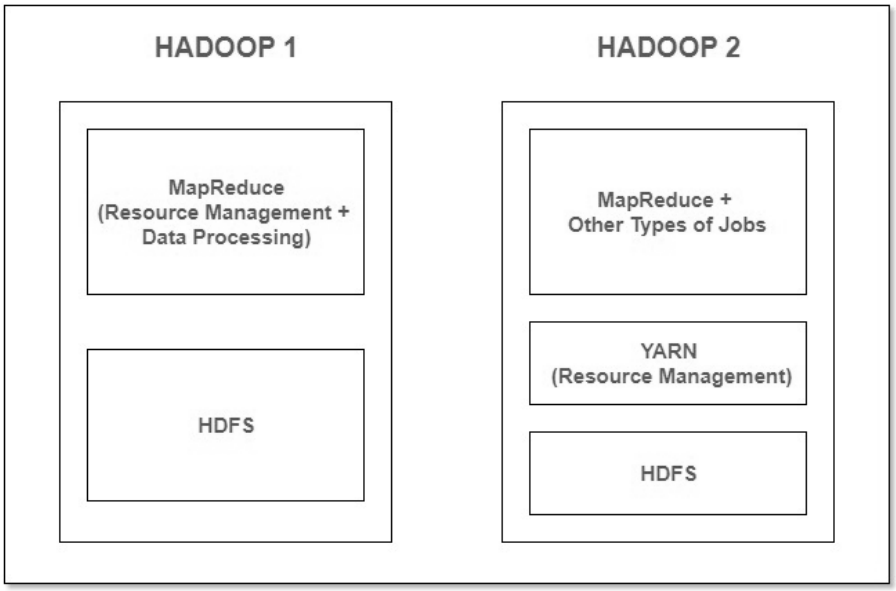

Figure 1. Hadoop 1 vs Hadoop 2

2.5. Hadoop 3

Hadoop version 3.x (Figure 2) was released in December 2017 [15] incorporating a number of significant improvements over the previous major release, Hadoop 2.x. Apache Hadoop 3.x with its Erasure Coding (EC) in place of data replication provides the same level of fault tolerance with much less storage space. In typical erasure coding configurations, the storage overhead is no more than 50%. Integrating EC with HDFS improves storage efficiency while providing data durability similar to that of traditional replication based HDFS deployments. As an illustration, a three time replicated file with six blocks will consume 6*3 = 18 blocks of disk space. But with EC deployment (six data, three parities) it will only consume nine blocks of disk space [5]. Hadoop 3.x improved scalability, usability by introducing streams and aggregation and reliability of Timeline service [10].

With Hadoop 3 we have some disadvantages. Node heterogeneity factors have a negative impact on its performance and limit the overall throughput of the system raising the issue of load balancing and data replication. To address these challenges, Hadoop also introduces the use of intra-node data balancing.

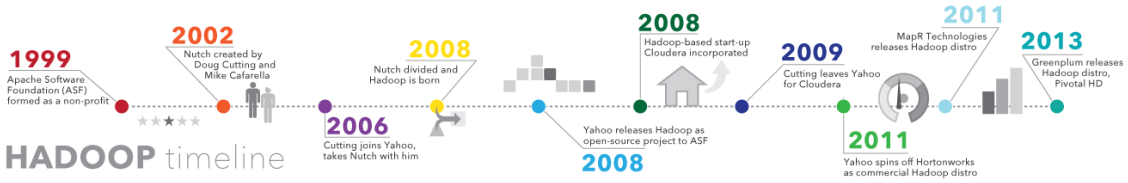

Figure 2. Some Hadoop History [25]

2.6. DHT - MapReduce

Although Hadoop 3.x supports more than two (02) NameNodes, the initial HDFS high availability implementation provided for a single active NameNode and a single SecondaryNode. By replicating edits across a quorum of three JournalNodes, this architecture is able to tolerate the failure of any node in the system. For the deployment of infrastructures that require higher levels of fault tolerance, JournalNodes can be used to allow users to run multiple backup NameNodes. For example, by configuring three NameNodes and five JournalNodes, the cluster is able to tolerate the failure of two nodes rather than just one. In parallel with Hadoop, other projects to design distributed storage and parallelized processing frameworks to develop high-performance computing using heterogeneous resources have emerged. We present some frameworks dedicated to this type of computation exploiting DHTs as organisational mechanisms. ChordMR[16] and ChordReduce[17] have focused their work on the failure of the master. The use of Chord [18] allows a set of backup nodes to be available for the resumption of outstanding work. When the active node fails, the backup nodes run an algorithm to elect a new node. Most DHTs focus their efforts on preventing record loss due to churn with scalability of more than 10^6 nodes and strategies to increase their storage performance [19].

2.7. EclipseMR

EclipseMR (Figure 3) consists of double-layered consistent hash rings - a decentralized DHT-based file system and an in-memory key-value store that employs consistent hashing. The in-memory key-value store in EclipseMR is designed to not only cache local data but also remote data as well so that globally popular data can be distributed across cluster servers and found by consistent hashing [20]. EclipseMR framework's signature is the usage of a double-ring to store data and metadata. In the upper layer of the ring, it utilizes a distributed in-memory key-value cache implemented using consistent hashing, and at its lower layer it uses a DHT type of distributed file system.

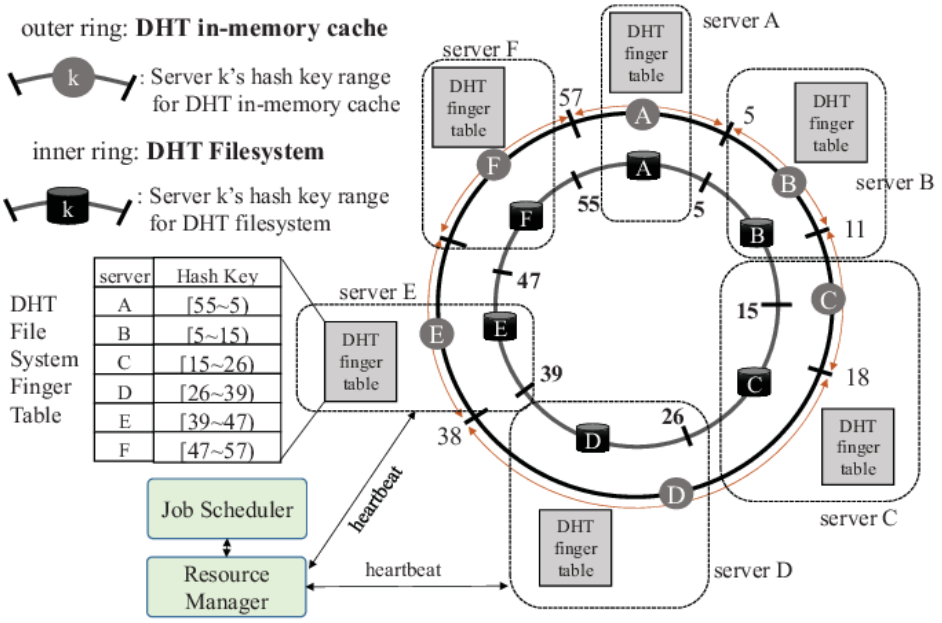

Figure 3. EclipseMR Architecture [19]

In order to leverage large distributed memories and increase the cache-hit ratio, EclipseMR proposes a Locality-Aware Fair (LAF) job scheduler that works as the load balancer for the distributed in-memory caches [15]. An experimental study shows that each design and implementation of EclipseMR components contributes to improving the performance of distributed MapReduce processing. Simulation shows EclipseMR outperforms Hadoop and Spark for several representative benchmark applications including iterative applications [20].

2.8. P2P-Cloud

P2P-Cloud offers innovative solutions to some of these concerns. Its architecture has four layers (Figure 4). A physical layer consisting of volunteer nodes connected via the Internet, above which is an overlay layer for logically connecting the nodes. The third layer performs the Map and Reduce functions on the data set. The outermost layer serves as an interface to receive user jobs.

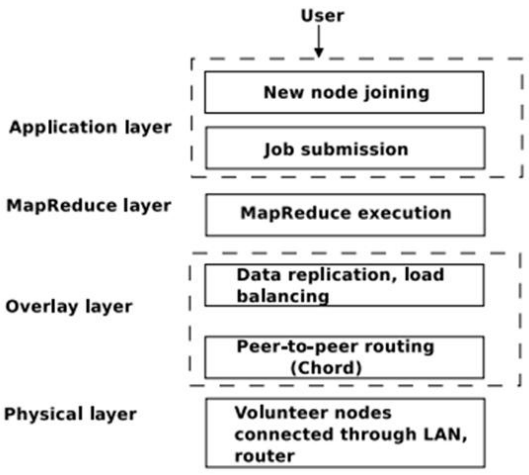

Figure 4. P2P-Cloud Layered Architecture [21]

P2P-Cloud uses Chord as an overlay protocol to organise nodes. The bootstrapper acts as an interface for joining nodes and submitting tasks. A node that wishes to join the network, contacts the bootstrapper. Each new node announces its success rate from previous runs. The success rate values correspond to the same domain as the nodeId. For each new task, the bootstrapper assigns a new identifier that is equal to the current number of tasks + 1.The new task is composed of input data and the Map and Reduce functions. Figure 4 illustrates the P2P-Cloud model. The slave nodes download the input chunks and run the Map task on them (step 2 of Figure 5). The intermediate results are replicated on the successor node (step 3 of the Figure 5) and transferred to the reduction nodes (step 4 of the Figure 5).The reducer performs reduce function on the intermediate data and transfers the final output to the primary master (step 5 in Figure 5). The primary master then copies the result to the bootstrap program so that the user can download the results.

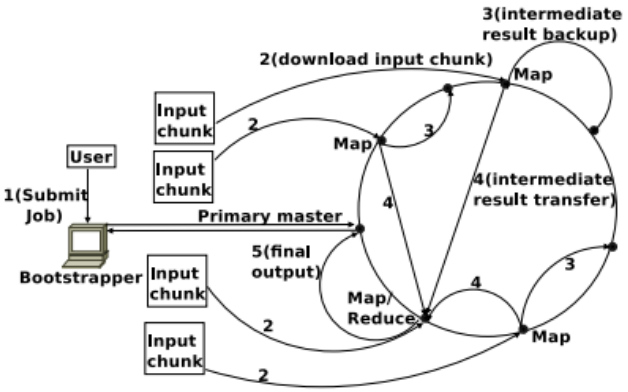

Figure 5. P2P-Cloud model [21]

3. PERFORMANCE ANALYSIS

In this section, we prove that CLOAK-Reduce [19, 22, 23] can play its part in a world where the main characteristics of computing frameworks are the efficient exploitation of storage and computing resources. We also take into account MapReduce job management using a load balancing strategy and peer failure management support to improve fault tolerance in the Internet environment.

3.1. Theoretical Review

Although the primary concern of distributed storage and parallelized computing is addressed in EclipseMR, P2P-Cloud and Cloak-Reduce, there are significant differences in their implementation. Challenges include managing node heterogeneity, task governance, churn and load balancing.

In our analysis, the first approach is a comparative study of mathematical models based on the search performance of P2P-Cloud (Equation 1) and EclipseMR (Equation 2):

Assuming that the probability of an arbitrary node at any location in P2P-Cloud at scale $N = 2^m$ or EclipseMR is equal, and that it satisfies a uniform distribution, then the probability is: $p = 1/2^m$

Assuming hop(n,key), the hop it takes for node n to find the identifier key, n(key) is the node where the resource key is located, so hop(n,key) is the number of hops node n takes to reach n(key).

Assuming Ehop, the expectation of jumps that a certain node takes to discover a random key, knowing that the number of nodes at level k is, C_m^k with a node discovery mechanism similar to an m − ary tree. So the node on level number *i* means that it takes *i* steps to reach this node, the number of nodes in number i is signed Level(i) here, so the Level(i) = m.

The probability that a random node can be found with i steps would be phi = Level(i) = C_m^k

$$Ehop_{P2P-Cloud} = \sum_{i=0}^{m} ph_i \times p \times i = \sum_{i=0}^{m} C_m \times \frac{1}{2^m} \times i \quad (1)$$

$$\frac{Ehop_{P2P-Cloud}}{Ehop_{EclipseMR}} = \frac{\sum_{i=0}^{m} C_m \times \frac{1}{2^m} \times i}{\sum_{i=0}^{m-1} C_{m-1} \times \frac{1}{2^m} \times i} \approx \frac{m}{m-1} \quad (3)$$

This quotient (Equation 3) means that the exception of hops in the lookup process in EclipseMR could be decreased by about *1/(m-1)* or that the efficiency could be improved by about *1/(m-1)*. In sum, the dual-ring architecture of EclipseMR reduces the average number of hops between peers in a search query and reduces query latency when peers in the same group are topologically close. Despite the fact that the reliability of P2P-Cloud nodes is estimated using the Exponential Weighted Moving Average (EWMA) algorithm [21].

However, we focused our comparative study on P2P-Cloud and CLOAK-Reduce, because despite the very promising future of EclipseMR, adding trivial features had become almost impossible [24].

3.2. Simulation

In this second approach, we compare the performance of CLOAK-Reduce with P2P-Cloud in terms of fault tolerance to justify the robustness of CLOAK-Reduce. For hardware constraints, we will set the churn rate to 10%, during the execution period of the Map and Reduce tasks and choose file sizes of 128, 256, 512 and 1024 MB respectively. All experiments were performed on a 2.60 GHz Intel Core i5 CPU PC with 8 GB of memory running Windows 10 Professional. The churns are independent and occur randomly. The comparison of the two frameworks will focus on the following performance indicators.

1. The total elapsed time with and without churn during the map and reduce phases.
2. Message exchanges between the Resource Manager (RM) and the slave executing nodes for P2P-Cloud and the Schedulers, JobManagers and JobBuilders for CLOAK-Reduce with and without churn.

3.3. Results of the Map and Reduce Submissions

3.3.1. Total Elapsed Time Without Churn

The CLOAK-Reduce C5R5 replication framework and the P2P-Cloud have almost the same trend when there are no churns. However, the simulation results give a slight performance to CLOAK-Reduce replication C5R5 which has an average time of about 24.805 seconds compared to 27.540 seconds for the P2P-Cloud framework when the data volume increases from 128 MB to 1024 MB. The mean processing time for the same data volumes of the C1R5 and C5R1

replications of CLOAK-Reduce, under the simulation conditions are 31.513 seconds and 30.251 seconds respectively. Figure 6 illustrates this behaviour.

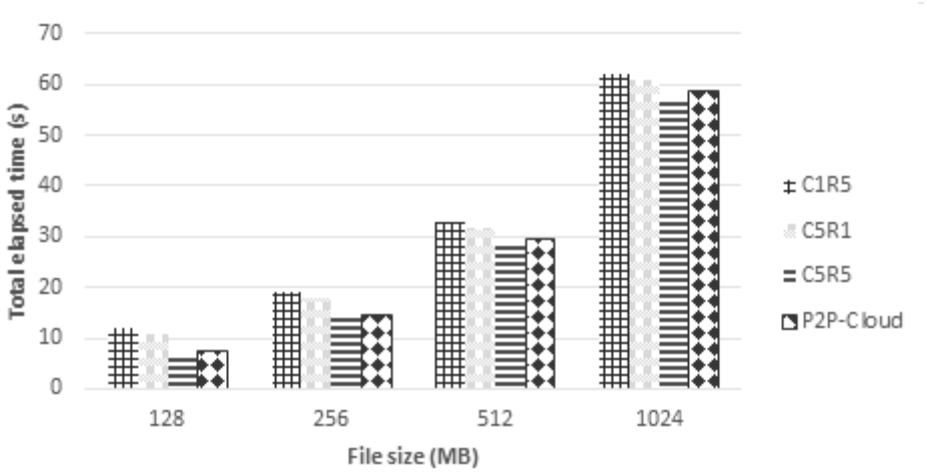

Figure 6. Total elapsed time in case of no churn

3.3.2. Map Phase with 10% Churn

Figure 7 compares the total elapsed time during a Map submission phase with a 10% churn rate. The results of the different simulations give the best performance to CLOAK-Reduce with C5R5 replication, with an average job submission time of 45.114 seconds, when we vary the data volume from 128 MB to 1024 MB. Then we have 55.687 seconds for P2P-Cloud and 57.801 and 66.988 for CLOAK-Reduce with C1R5 and C5R1 replications respectively. However, the time difference between P2P-Cloud and CLOAK-Reduce with C1R5 replication is 2.115.

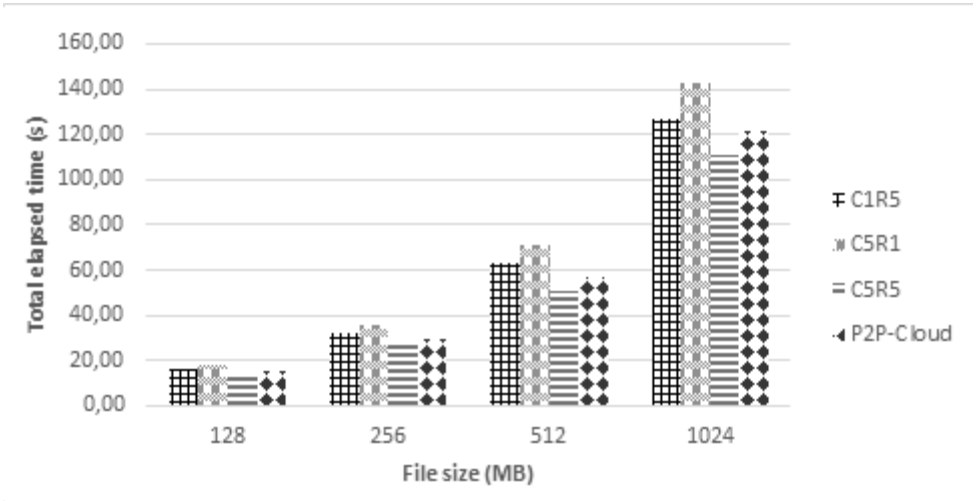

Figure 7. Map phase with 10% churn

3.3.3. Reduce Phase with 10% Churn

Figure 8 compares the total elapsed time during a Reduce submission phase with a 10% churn rate. The results of the different simulations give the best performance to CLOAK-Reduce with C5R5 replication, with an average job submission time of 50.944 seconds, when we vary the data

volume from 128 MB to 1024 MB. Then we have 67.200 seconds for P2P-Cloud and 116.086 and 68.522 seconds for CLOAK-Reduce with C1R5 and C5R1 replications respectively.However, we find a small variation in submission time between the Map and Reduce phases of CLOAK-Reduce with C5R1 replication, which is 66.988 for the Map phase and 68.522 for the Reduce phase. In contrast to CLOAK-Reduce with C5R1 replication, C1R5 replication often results in a much higher time with almost 50% failure in Reduce submissions.

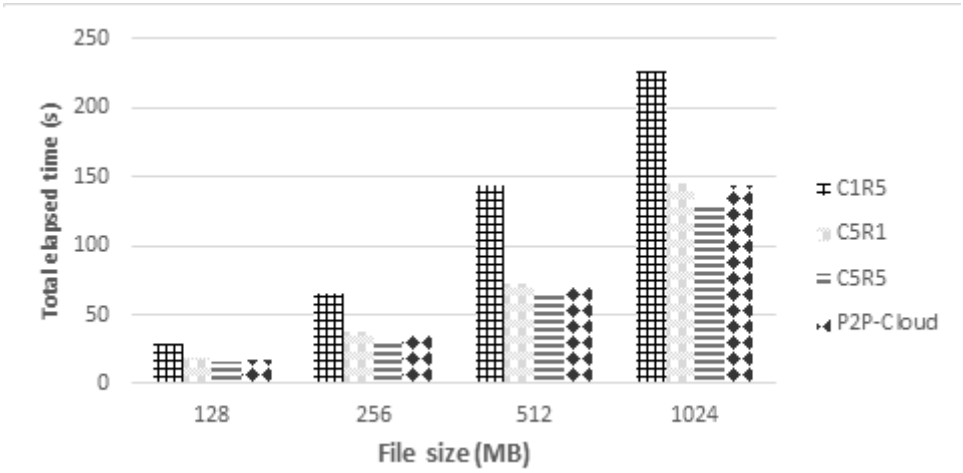

Figure 8. Reduce phase with 10% churn

4. CONCLUSIONS AND PERSPECTIVES

This study shows the flexibility of the address space and the efficiency of the CLOAK-Reduce load balancing strategy. The proposed hierarchical architecture uses multiple management nodes to achieve scalability and availability without significant downtime.

The performance of the proposed approach is verified using two phases of MapReduce. For each phase, we performed experiments to observe the performance of our framework. Compared to existing approaches that only consider load-based block placement or block redistribution. Our proposed global approach achieves good performance in terms of global load submission, load balancing and distributed task restitution. Its approaches have been solved by our mechanism of radial replication for intermediate jobs, circular replication for job submission and storage of unsubmitted jobs at the root level.

In the future, we will focus on optimising the cost of communication between idle and active nodes. That is, reduce the number of exchanges between candidate JobManagers, inactive JobBuilders and their elected JobManager.

REFERENCES

[1] Thomas Bourany, 2018, "Les 5v du big data. Regards croises sur l'economie", (2):27–31. read 05 sept 2019.

[2] Than Than Htay et Sabai Phyu, 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.

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

[4] Awaysheh, F. M., Alazab, M., Garg, S., Niyato, D., & Verikoukis, C., 2021, "Big data resource management & networks: Taxonomy, survey, and future directions", *IEEE Communications Surveys & Tutorials*.

[5] Xue, R., Guan, Z., Dong, Z., & Su, W., 2018, "Dual-Scheme Block Management to Trade Off Storage Overhead, Performance and Reliability", In *International Conference of Pioneering Computer Scientists, Engineers and Educators* (pp. 462-476). Springer, Singapore, September

[6] Ping, Y., 2020, "Load balancing algorithms for big data flow classification based on heterogeneous computing in software definition networks". Journal of Grid Computing, 1-17.

[7] Mansouri, N., Javidi, M. M., 2020, "A review of data replication based on metaheuristics approach in cloud computing and data grid". Soft computing, 1-28.

[8] Bhushan, K., 2020, "Load Balancing in Cloud Through Task Scheduling", In Recent Trends in Communication and Intelligent Systems (pp. 195-204). Springer, Singapore..

[9] Gao, X., Liu, R., Kaushik, A., 2020, "Hierarchical multi-agent optimization for resource allocation in cloud computing". IEEE Transactions on Parallel and Distributed Systems, 32(3), 692-707.

[10] Yahya Hassanzadeh-Nazarabadi, Alptekin Küpçü, et Öznur Özkasap, 2018, "Decentralized and locality aware replication method for DHT-based P2P storage systems", Future Generation Computer Systems, vol. 84, p. 32-46.

[11] Monnet, S., 2015, Contributions à la réplication de données dans les systèmes distribués à grande échelle (Doctoral dissertation.

[12] Pandey, Vaibhav, and Poonam Saini, 2018, "How heterogeneity affects the design of Hadoop MapReduce schedulers : a state-of-the-art survey and challenges." Big data 6.2 : 72-95.

[13] Prabhu, Yogesh, and Sachin Deshpande, 2018, "Transformation of Hadoop : A Survey.", IJSTE - International Journal of Science Technology & Engineering | Volume 4 | Issue 8 | February ISSN (online) : 2349-784X.

[14] Pathak, A. R., Pandey, M., & Rautaray, S. S., 2020, "Approaches of enhancing interoperations among high performance computing and big data analytics via augmentation". *Cluster Computing*, *23*(2), 953-988.

[15] Sewal, P., & Singh, H., 2021, "A Critical Analysis of Apache Hadoop and Spark for Big Data Processing; In *2021 6th International Conference on Signal Processing, Computing and Control (ISPCC)* (pp. 308-313). IEEE, October

[16] 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.

[17] 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.

[18] Ion Stoica, Robert Morris, David Liben-Nowell, David R Karger, M Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. "Chord : a scalable peer-to-peer lookup protocol for internet applications". IEEE/ACM Transactions on Networking (TON), 11(1) :17–32, 2003.

[19] Tiendrebeogo, T., Diarra, M., 2020, "Big Data Storage System based on a Distributed Hash Tables System",International Journal of Database Management Systems (IJDMS) Vol.12, No.4/5, October.

[20] Sanchez, Vicente AB, et al., 2017, "EclipseMR: distributed and parallel task processing with consistent hashing", In 2017 IEEE International Conference on Cluster Computing (CLUSTER) (pp. 322-332). IEEE

[21] Banerjea, Shashwati, Mayank Pandey et Manoj Madhava Gore., 2016, "Vers une solution peer-to-peer pour l'utilisation des ressources bénévoles afin de fournir un calcul en tant que service",. En 2016, IEEE 30th International Conference on Advanced Information Networking and Applications (AINA) (pp. 1146-1153). IEEE, mars.

[22] Mamadou Diarra and Telesphore Tiendrebeogo, 2021, "MapReduce based on CLOAK DHT Data replication evaluation", International Journal of Database Management Systems (IJDMS) Vol.13, No.4, August [23]

[23] Mamadou Diarra and Telesphore Tiendrebeogo, 2021, "CLOAK-Reduce Laod balancing strategy for MapReduce", International Journal of Computer Science and Information Technology (IJCSIT) Vol 13, No 4, August.

[24] Bolea Sanchez, V. A, 2019, "VELOXDFS : Elastic blocks in distributed file systems for big data frameworks".

[25] Shachi Marathe (April 29, 2017) AN INTRODUCTION TO HADOOP https://www.mindtory.com/an-introduction-to-hadoop/ rea