LOAD BALANCING LARGE DATA SETS IN A HADOOP CLUSTER

Andriavelonera Anselme A.[1], Rivosoaniaina Alain N.[1],Rakotomalala Francis[1],Mahatody Thomas[2] and Manantsoa Victor[2]

[1]Laboratory for Mathematical and Computer Applied to the Development Systems, University of Fianarantsoa, Madagascar
[2]Professor on the University of Fianarantsoa, Madagascar

ABSTRACT

With the interconnection of one to many computers online, the data shared by users is multiplying daily. As a result, the amount of data to be processed by dedicated servers rises very quickly. However, the instantaneous increase in the volume of data to be processed by the server comes up against latency during processing. This requires a model to manage the distribution of tasks across several machines. This article presents a study of load balancing for large data sets on a cluster of Hadoop nodes. In this paper, we use Mapreduce to implement parallel programming and Yarn to monitor task execution and submission in a node cluster.

KEYWORDS

Load balancing, Server, Cluster, Hadoop, Mapreduce & Yarn.

1. INTRODUCTION

Data size is considered large when it exceeds human analysis capabilities and the usual IT tools for database management.

Currently, most research is focused on load balancing independent tasks [4] [5]. As a result, a potential constraint is the management of resources across multiple computers to optimize processing. The existence of a computer network means that tasks can be distributed from one overloaded server to another underloaded machine. Load balancing is the result of a set of techniques admitting a balanced distribution of the load on a system's available resources [3].

1.1. Objective and problematic of the research

In terms of performance, it's still difficult to achieve the best response time when large amounts of data arrive instantaneously on the server. It is therefore necessary to improve database performance in order to process large quantities of data efficiently. Indeed, the aim of this research is to ensure a fair distribution of loads across a cluster of Hadoop nodes. To prevent server saturation, the state of the nodes needs to be monitored in real time during as the loadlarge amounts of data. This requires a performance study of the Hadoop database to determine its processing capacity. However, the implementation of parallel programming with Mapreduce offers the possibility of measuring the performance of the NoSQL database. For security reasons, it is essential to monitor server status in real time. This is provided by cluster node administration tools such as Ambari and Yarn. These are tools designed to manage and monitor resource

utilization during job execution. Some research has developed MapReduce-based model to predict task execution times in a heterogeneous cluster [2].

With the various potential threats to today's IT infrastructure, it is essential that the system administrator implements a server load balancing strategy to prevent overloading. This begs the question: how can loads be effectively managed within a cluster of nodes when processing massive data?

1.2. Contribution

As part of the improvement, the first aspect is to check the time it takes to load data to the cluster, then the settings required for block size before this data replicates across the nodes. The latter should be mastered by the system and network administrator to predict a server's load limit. However, the study on load balancing will focus on the implementation of Mapreduce parallel programming to test and simulate the load limit of a cluster of nodes according to the number of connected users and shared data volumes. However, we propose a new approach to improve load balancing of a Hadoop node cluster during instantaneous loading of large data by implementing Mapreduce parallel programming with the necessary parameter settings.

2. STATE OF ART

In this section, we'll cover the key elements, such as the overall architecture of MapReduce, file read and write operations in HDFS, as well as previous work related to load balancing in a Hadoop cluster.

2.1. MapReduce paradigm

MapReduce is a programming model proposed by Google. It enables parallel processing of large quantities of data within a cluster of nodes, and automates parallelism and distribution of computation. Since the system manages parallel execution, the programmer's role is to implement the two functions Map and Reduce. The Map function transforms input data into a series of key/value pairs. It assembles the data into key/value pairs according to context. Each node executing the Map function works on one or more pieces of the initial data. This data must be split into several fragments for parallel processing and to perform the Map operation at each cluster. In his article, the author has implemented the MapReduceApriori (MRA) algorithm on the Apache Hadoop cluster. This algorithm uses two distinct functions, namely Map and Reduce, with the aim of detecting repeated sets of k-elements [6].

$$\textit{Map function} = \textit{Map (key1, value1)} \rightarrow \textit{List (key2, value2)}$$

The Reduce function processes the values of each of the distinct keys generated by the Map operation. Next, the key/value pairs are forwarded to the different nodes, provided that they have the same key and are in the same Reduce node. Finally, each machine performs the Reduce operation for that key. The Reduce function is written as:

$$\textit{Reduce (key2, List (value2))} \rightarrow \textit{List (value2)}$$

The operation of MapReduce is summarized in Figure 1 below:

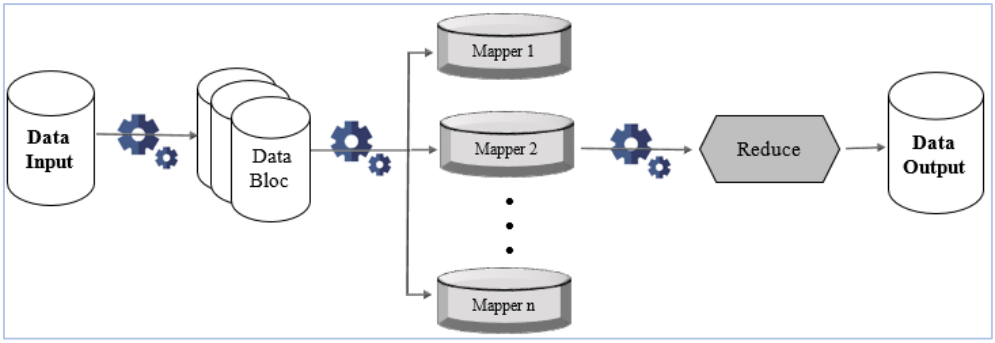

Figure 1: Overall MapReduce architecture

The use of MapReduce ensures efficient use of machine resources and is suitable for solving a wide range of computational problems. A number of optimizations aim to reduce the amount of data sent over the network. Big data is categorized according to its volume, variety and velocity. Most of this data is unstructured, quasi-structured or semi-structured, and heterogeneous in nature.

The sheer volume and heterogeneity of data, coupled with the speed at which it is generated, makes it difficult for today's IT infrastructure to manage it. The classification of algorithms according to various quality measures that affect MapReduce performance [1].

Traditional systems for managing, storing and analyzing data lack the tools to analyze it. They need to be stored in a distributed file system architecture such as Hadoop, which is most widely used to store and manage high-volume data. In his article, Gaykar proposes a method for identifying faulty nodes within a large distributed Hadoop environment using machine learning techniques. The method involves collecting the history of each data node, then applying several statistical and machine-learning algorithms to determine the state of the nodes [9].

2.2. Mapreduce Technique

MapReduce is a powerful large-scale data processing platform for sorting and merging huge datasets in parallel, with research underway since the 1980s [22]. Several new large-scale data processing platforms have emerged, inspired by MapReduce, to improve their scalability, including Dryad, Hyracks and Spark. These platforms all integrate MapReduce components, while expanding the options available for query execution. Some have decided to improve MapReduce, as described in the article by Chen He, who has developed a new scheduling method aimed at improving the geographical proximity of data for mapping tasks. This technique also incorporates a FIFO scheduler developed for Hadoop. Experimental results show that this approach significantly improves the geographical proximity of data, thus reducing the response time of mapping tasks. What's more, unlike the delay algorithm, this approach does not involve the need for tedious parameter adjustment [23]. The study conducted by Vishal examined various existing models for optimizing Map-Reduce job scheduling with regard to their use in the context of cloud computing [24].

Mohammad Reza Ahmadi's method is based on the use of a genetic algorithm with a parallel executive structure to improve the processing of relational data in MapReduce clusters. This approach proves particularly advantageous as data volume increases [25]. The MapReduce paradigm has generated a great deal of interest in both academic and industrial circles [26].Various MapReduce-related tools are available, three of which stand out as particularly significant: Hadoop, Apache HIVE and Sqoop. Each of these platforms has its own MapReduce

mechanism [27]. Research into techniques for evolving MapReduce remains active, adjusting to the various data. An in-depth analysis of MapReduce and its application in optimization algorithms can be found in Khezr's article [28].

2.3. Reading and writing a file in HDFS

The results of experiments carried out by NathierMilhem et al. show that HDFS with a distributed cache mechanism offers superior learning performance compared with conventional HDFS, which has no cache, and with HDFS with a centralized cache mechanism [6].

To read a block in HDFS, the HDFS client queries the NameNode to retrieve the list of DataNodes hosting its various replicas, the client then reads the block from the DataNode, and if there's a problem with the block, it reads the block from another DataNode. The client reads the nearest replica.

The figure 2 shows the read/write process for a file in the HDFS System:

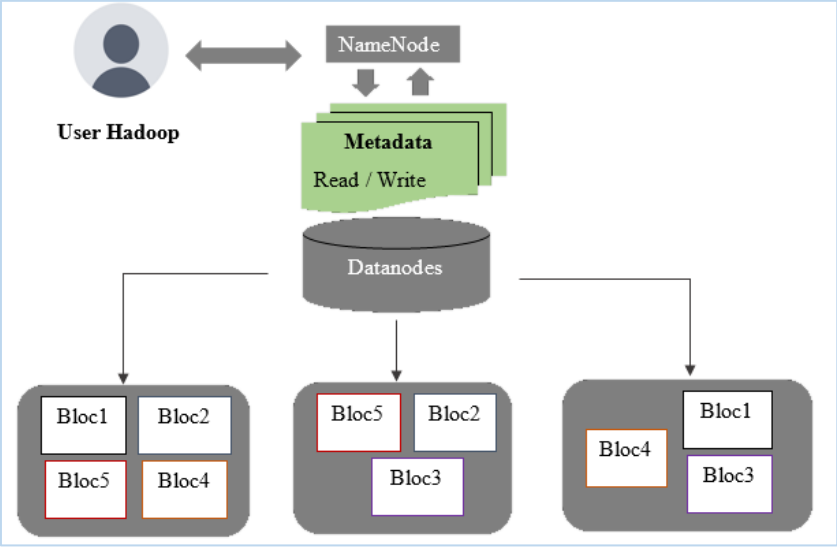

Figure 2: Read and write a file in HDFS.

Recently, the article by Vijay offers an identification of the factors that contribute to solving performance problems, while exploring other strategies for improving the efficiency of MapReduce queries within a cloud-based environment [7].

To write a file to an HDFS system, follow these steps: (a) use the main Hadoop management command: hadoop with the fs option. If you want to store a file on the HDFS system. (b) The program will divide the file into 64kb blocks. (c) The NameNode tells it which DataNodes to contact. (d) The client directly contacts the DataNode concerned and asks it to store the block. (e) The DataNodes will inform the NameNode to replicate the data between them to avoid any data loss. (f) The cycle is repeated for the next block.The inherent constraint of HDFS is that it is only extensible at the level of data nodes, without the possibility of extending name nodes or metadata management.

The proposed Dynamic Federated Metadata Management (DFMM) approach implements metadata management at the name node level. This DFMM method configures multiple name

nodes using the HDFS federation technique, thus distributing metadata across multiple name nodes thanks to the dispatching technique. The results obtained highlight the superiority of the proposed DFMM architecture, combined with the sharding technique, in terms of high scalability and metadata management efficiency. A comparison between the existing HDFS architecture and the DFMM architecture suggests that the latter, combined with sharding, improves performance and ensures more efficient metadata management [8].

2.4. Related work on load balancing

To optimize the processing of massive data in a cluster of Hadoop nodes, researchers have developed an improved approach to job scheduling. This method takes into account both task size and expected execution time. In addition, they have improved Hadoop performance by integrating an aggregation node into the default architecture of the HDFS distributed file system [11]. Other research has exploited AEML, an acceleration engine specially designed to balance the workload across multiple GPUs. They developed an execution model dedicated to heterogeneous tasks [12]. In order to overcome the limitations of the fair scheduling algorithm, Bawankule proposes two strategies to improve the resource utilization efficiency and performance of Hadoop [13].

In his article, Weyu Fu has improved Hadoop's resource utilization and performance by optimizing the allocation of reduction tasks. This optimization is based on algorithms inspired by the behavior of colonies of ants and hives, to ensure more efficient load balancing management [14].In another paper, load balancing between available fog nodes was investigated to reduce task processing response time, with results indicating a significant improvement in response time [15].Some authors have improved a load balancing architecture using the hybrid load balancing algorithm.

The proposed model was designed to improve resource utilization by implementing load balancing at the fog layer [16].However, improving load balancing can have a significant impact on service quality, which translates positively into the user experienceIn his work, the author explored different task scheduling and load balancing algorithms, also introducing a new seven-category classification for the HadoopMapReduce case, and providing a detailed analysis of each of these categories [17]. Mostafa's study methodically analyzes load balancing algorithms in fog computing, classifying them into four categories, and discusses the main outstanding challenges and future trends for these algorithms [18].

Load balancing ensures the balanced distribution of workloads generated by a number of data-connected devices. In-depth investigations have been carried out to integrate this practice into cloud computing. Ashish presents a study of load balancing algorithms in fog computing, including an analysis based on various computational metrics as well as a simulation tool for its deployment [19]. Another paper presents a load balancing method that takes energy efficiency into account in the context of fog computing, by establishing an energy consumption model related to the workload on the nodes. In addition, it prioritizes tasks according to the manufacturing cluster and introduces a multi-agent system to perform distributed scheduling within the manufacturing cluster [20].Pradeep's paper, aims to explain the importance of critical data analysis in big data and to show how load balancing in cloud computing allows different offload factors to be compared using parameters such as task execution time, distribution of tasks across virtual machines and energy consumption [21].

2.5. Synthesis

After reviewing the most relevant work on load balancing using the Hadoop cluster and MapReduce, we can see that most research has focused on two main areas:

- Proposed optimization algorithms: researchers have focused their work on developing specific algorithms to improve load balancing in a Hadoop environment. These algorithms are designed to efficiently distribute computing tasks between cluster nodes, reducing load variations and optimizing system performance.
- Load balancing in the context of big data: other areas of research have focused on load balancing in big data. In this environment, they ensure that data is distributed evenly between cluster nodes.

Both approaches aim to improve the efficiency and performance of Hadoop clusters by ensuring a balanced distribution of resources, whether at processing or data management level. Researchers continue to focus on these areas to solve load balancing problems in distributed data processing environments.In fact, there are many different methods and techniques for exploiting Hadoop and MapReduce in data processing. From a technical and operational point of view, we can see that the implementation of these tools offers various opportunities for companies. Indeed, to continue exploiting the strengths of Hadoop and Mapreduce, it is essential to supervise loads at the level of a cluster of nodes during data processing.

As a result, we propose a new architecture illustrated in Figure 3, leveraging the power of Logstash to improve data loading to the Hadoop cluster. We have used this tool to manage log files efficiently. However, during processing, the performance metric at node cluster level needs to be evaluated via the MapReduce program implementation, which simultaneously generates the load limit. However, we are also keen to evaluate the performance of the Hadoop system in combination with other big data management tools.

3. EXPERIMENTATION

3.1. Working Environment

The hardware architecture during the experiment is characterized by:

- **Master node**: Core De Duo processor (2 x 2.5 GHz), RAM: 2GB, Operating System: Centos 7 and 80GB hard disk.
- **Data node 1:**Single-core processor2.5 GHz, RAM: 1GB, Operating System: Centos 7 and 80GB hard disk.
- **Data node 2:**Single-core processor2.5 GHz, RAM: 1GB, Operating System: Centos 7 and 80GB hard disk.

3.2. Performance Metrics

Table 2 below shows the execution time for processing Mapreduce tasks as a function of the number of nodes and file size:

Table 1: Experimental data

3 Map et 1 Reduce		
File size in megabytes	Number of Nodes	Execution time in seconds
121,30	2	54
	3	37
216,35	2	93
	3	64

During the execution of a job, the NodeMaster (*NodeM*) automatically distributes data to client machines or data nodes. During processing, system log files generate a performance report. This factor reflects the query execution time in relation to the number of nodes in the Hadoop cluster.

NodeM is calculated according to the following formula:

$$NodeM = \frac{T}{N}$$

- *T:represents the test execution time.*
- *N:the number of nodes on which our query is executed.*

a) Variance observation :

$$V(x) = \frac{1}{p}\sum(xi - \bar{x})^2$$

b) Covariances:

$$cov(x,y) = \sigma = \frac{1}{n}\sum (xi - \bar{x})(yi - \bar{y})$$

c) Standard error:

$$\sigma(x) = \sqrt{V(x)}$$

d) Correlation coefficient: $[x - \sigma, x + \sigma]$

e) Change of variable :

$$N(m, \sigma) \qquad T = \frac{x-m}{\sigma}$$

3.3. Learning

Let X be a random variable with normal distribution **N(m,σ)**then$\mathbf{T} = \frac{x-m}{\sigma}$ follows a reduced entry normal distribution **N (0, 1)**; hence

- $P(m - \sigma < X < m + \sigma) = P(59,7-19,7 < X < 59,7+19,7) = 0,64 = 64\%$, Cluster at normal load.
- $P(m - 2\sigma < X < m + 2\sigma) = P(59,7-2*19,7 < X < 59,7+2*19,7) = 0,96 = 96\%$, Overloaded cluster.
- $P(m - 3\sigma < X < m + 3\sigma) = P(59,7-3*19,7 < X < 59,7+3*19,7) = 0,998 = 99,8\%$, Saturated cluster.

To improve data processing at the level of a cluster of Hadoop nodes, we propose the architecture described in Figure 3 below as a new approach:

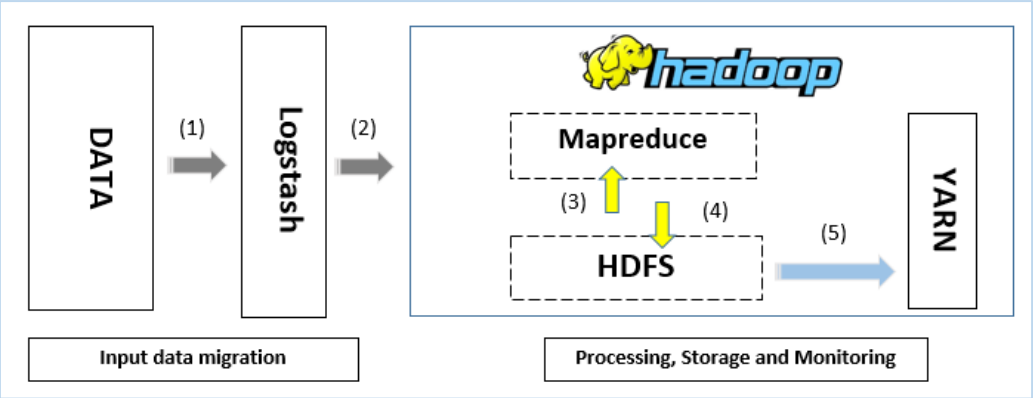

Figure3: Architecture for a new approach

(1): Transfer input data to Logstash.
(2): Receipt of data in the Hadoop database.
(3): Computing and data processing using Mapreduce parallel programming.
(4): Data storage in Hadoop Distributed File System or HDFS.
(5):YARN, which manages and monitors a cluster of nodes.

4. RESULTS

During the experimentation, we were able to draw the following several points:

- Cluster stability varies according to the number of nodes.
- The block size configuration is essential in relation to input data size, and should be automatic.
- Data replication ensures the reliability of an HDFS system in the event of failure of one of its nodes. This ensures system availability.
- Input data migration through the Logstash pipeline ensures automatic logging of all incoming data.
- Cluster monitoring is carried out via the AMBARI server administration interface, which also alerts the system administrator when the cluster is overloaded.

After MapReduce job execution, we can summarize the following results:

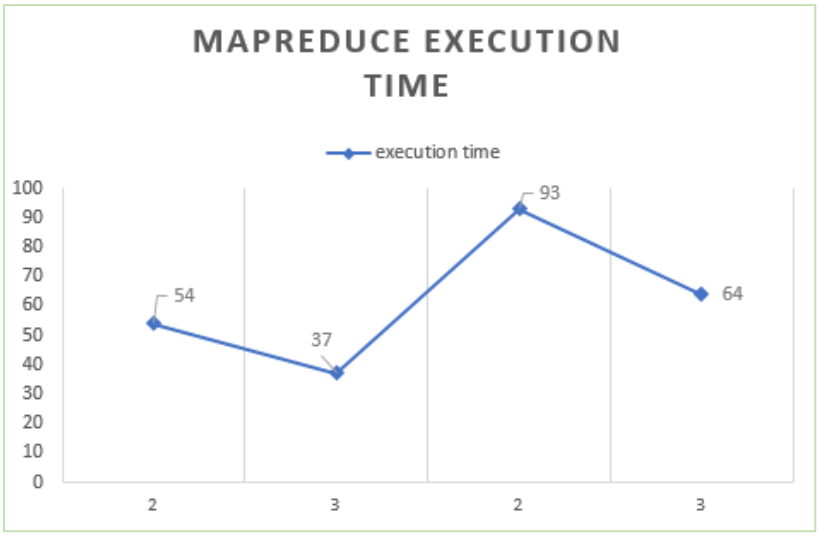

Figure 4: Mapreduce algorithm execution times

The graph shown above illustrates a decrease in execution time as the number of nodes in the cluster is increased. During the experiment, we evaluated the execution of MapReduce jobs using both 2 and 3 nodes, on two different file sizes, one of 121.30 MB and the other of 216.35 MB. The results show that the performance criterion of response time favors the multi-node architecture.

5. CONCLUSIONS

In this paper, we studied the performance of a cluster of nodes during the loading of large data. Many experiments were carried out to understand cluster management and the steps involved in setting up a data pipeline. Using Logstash simplifies the process of transferring data to the Hadoop database. We carried out our experiments on virtual machines. Since managing and simulating nodes at cluster level consumes a lot of resources, the choice of tool and block size settings are very important before controlling the flow of data through the nodes.

This work is limited by the need to reconfigure and rewrite the MapReduce program each time we want to achieve greater precision in measuring the performance of a cluster of nodes. The system administrator will need to make these adjustments for the node cluster based on the MapReduce program in order to achieve better, improved results. In terms of prospects, the idea will be to model an intelligent system for systematically predicting the real-time load of a cluster of nodes; with this in mind, experiments should be carried out in a physical architecture specifically adapted to the hardware requirements.

REFERENCES

[1] KhushbooKalia, Neeraj Gupta, 2020, “Analysis of hadoopMapReduce scheduling in heterogeneous environment”, Ain Shams Engineering Journal.

[2] AbolfazlGandomi, Ali Movaghar, MidiaReshadi& Ahmad Khademzadeh, 2020, “Designing a MapReduce performance model in distributed heterogeneous platforms based on benchmarking approach”, The Journal of Supercomputing.

[3] F. Dong and G. Akl, 2006, “Scheduling Algorithms for Grid Computing: State of the Art and Open Problems”, Technical Report No. 2006-504.

[4] H. Johansson, J. Steensland, 2006, “A performance characterization of load balancing algorithms for parallel SAMR applications”, Technical Report 2006-047, Uppsala University.

[5] H. Shan, L. Oliker, R. Biswas, W. Smith, 2004, "Scheduling in heterogeneous grid environments: The effects of data migration" In Proc. of ADCOM2004,International Conference on Advanced Computing and Communication.

[6] NathierMilhem, LaithAbualigah, Mohammad H. Nadimi-Shahraki, Heming Jia, Absalom E. Ezugwu&Abdelazim G. Hussien, 2022, "Enhanced MapReduce Performance for the Distributed Parallel Computing: Application of the Big Data", Studies in Computational Intelligence book series SCI, volume 1071.

[7] Vijay, V., Nanda, R., 2023,"A Priori Study on Factors Affecting MapReduce Performance in Cloud-Based Environment", Proceedings of Seventh International Congress on Information and Communication Technology.

[8] Praveen M Dhulavvagol, S G Totad, 2023, "Performance Enhancement of Distributed System Using HDFS Federation and Sharding", International Conference on Machine Learning and Data Engineering.

[9] Gaykar, Reshma S.; Khanaa, Velu; Joshi, Shashank D, 2022, "Faulty Node Detection in HDFS Using Machine Learning Techniques", Revue intelligenceArtificielle.

[10] Y. Gao, S. Feng and Z. Li, 2020, "A Distributed Cache Mechanism of HDFS to Improve Learning Performance for Deep Reinforcement Learning," IEEE Intl Conf on Parallel & Distributed Processing with Applications.

[11] R. Alanazi, F. Alhazmi, H. Chung & Y. Nah, 2020, "A multi-optimization technique for improvement of Hadoop performance with a dynamic job execution method based on artificial neural network," SN Computer Science, vol. 1, no. 3, pp. 184–211.

[12] Zhuo Tang, Lifan. Du, Xuedong Zhang, Li Yang & Ken Li, 2021, "AEML: an acceleration engine for multi-GPU load-balancing in distributed heterogeneous environment," IEEE Transactions on Computers, vol. 71, no. 6, pp. 1–1357.

[13] K. L. Bawankule, R. K. Dewang, and A. K. Singh, 2021, "Load Balancing Approach for a MapReduce Job Running on a Heterogeneous Hadoop Cluster", International Conference on Distributed Computing and Internet Technology, pp. 289–298, Springer, Cham.

[14] Weiyu Fu, Lixia Wang, 2022, "Load Balancing Algorithms for Hadoop Cluster in Unbalanced Environment, Hindawi Computational Intelligence and Neuroscience.

[15] A.B. Manju, S. Sumathy, 2018, "Efficient Load Balancing Algorithm for Task Preprocessing in Fog Computing Environment", Smart Intelligent Computing and Applications pp 291–298.

[16] Mandeep Kaur &Rajni Aron, 2021, "FOCALB: Fog Computing Architecture of Load Balancing for Scientific Workflow Applications", Journal of Grid Computing volume 19, Article number: 40.

[17] EinollahJafarnejadGhomi& Amir, 2017, "Load-balancing algorithms in cloud computing", Journal of Network and Computer Applications, Volume 88, Pages 50-71.

[18] MostafaHaghiKashani&EbrahimMahdipour, 2022 "Load Balancing Algorithms in Fog Computing", IEEE Xplore logo.

[19] Ashish Chandak&Niranjan Kumar Ray, 2019, "A Review of Load Balancing in Fog Computing", International Conference on Information Technology (ICIT).

[20] Jiafu Wan; Baotong Chen; Shiyong Wang; Min Xia; Di Li &Chengliang Liu, 2018, "Fog Computing for Energy-Aware Load Balancing and Scheduling in Smart Factory", IEEE Transactions on Industrial Informatics.

[21] Pradeep Kumar Shriwas, 2022, "OPJU International Technology Conference on Emerging Technologies for Sustainable Development", IEEE Xplore.

AUTHOR

ANDRIAVELONERA Anselme Alexandre, PhD student at EDMI (Ecole Doctorale de Modélisation Informatique), University of Fianarantsoa, Madagascar
10 years of experience in System and Network Administration IT Instructor on the studies direction, INSCAE Madagascar.