Efficient Auto Scaling and Cost-Effective Architecture in Apache Hadoop

Warda Ismahene NEMOUCHI[1], Souheila BOUDOUDA[2] and Nacer eddine ZAROUR[3]

[1] A. Mehri - Constantine 2 University, Algeria, warda.ismahene@univ-constantine2.dz
[2] A. Mehri - Constantine 2 University, Algeria, souheila.boudouda@ univ-constantine2.dz
[3] A. Mehri - Constantine 2 University, Algeria, nasro.zarour@ univ-constantine2.dz

Abstract. In the age of Big Data Analytics, Cloud Computing has been regarded as a feasible and applicable technology to address Big Data Challenges, from storage capacities to distributed processing computations. One of the keys of its success is its high scalability which refers to the ability of the system to increase its performance, resources and functionalities according to the workload. This flexibility has been seen as an appropriate way to decrease datacenters' energy consumption and thus assures cost-saving and efficiency without effecting performance of the system. In order to handle Big Data operations, Cloud Computing has implemented various platforms and tools such as Apache Hadoop and provides distributed processing of very large data sets across multiple clusters. This paper proposes an auto scaling architecture based on the framework of Hadoop; it adjusts automatically the computation resources depending on the workload. In order to validate the effectiveness of the proposed architecture, a case study about Twitter data analysis in a cloud simulated environment has been implemented to improve the cost-effectiveness and the efficiency of the system.

Keywords: Big Data, Cloud Computing, Apache Hadoop, Auto Scaling.

1 Introduction

Big Data refers to the increasing amount of data generated every second from e-business applications, smartphones and more and more connected objects. This explosion in the digital world has led to the evolution in different technologies in order to store, process, and analyze this important volume of data [1]. For that, multiple solutions have been proposed namely Apache Hadoop framework which allows distributed processing for large scale data sets across multiple computers in a cluster. Hadoop offers both massive storage and huge processing capacities in master slave architecture [2]. Hadoop Distributed file system (HDFS) is responsible for storing data in form of blocs of 64Mb or 120Mb. Each bloc is stored on a slave node, the information related to all data blocs are stored in the master node. To assure data availability, HDFS follows a replication policy where each bloc is replicated n times (n=3).

Despite the important value that Big Data has added to big companies through its delivered insights and knowledge, its requirements of having the appropriate infrastructure to deal with such growing volume and variety of data seem highly expensive. Hence the need for another technology capable of meeting both the needs of Big Data at a reduced cost and a fully managed infrastructure [3]. It's Cloud Computing (CC), a paradigm for managing and delivering services over the internet with its characteristics of elasticity and scalability to millions of instances in a completely transparent way to the final user [4].
Big Data and CC present the perfect combination to process huge amounts of data on a platform that is scalable and has the resources to analyze massive data [5]. However, there exist some challenges to overcome in particular the energy waste of data centers that are kept running without actually being used [6].
For this, the ability to automate the scaling process so it can provision or reduce resources based on workload automatically is needed in order to reduce the energy consumption of the cluster and thus reduce related costs. In a Big Data context, controlling Hadoop' resources automatically are more challenging due to its replication policy as powering off nodes is a time-consuming operation, data blocs need to be transferred at first before shutting down the node [7]. In this context, an auto-scaling approach has been investigated to overcome the energy waste and related costs problem.
This paper is organized as follows: section 2 discusses some research works related to Big Date auto-scaling applications, section 3 presents the proposed approach and details its different components whereas section 4 shows the implementation of the approach. Finally, some conclusions and research lines are presented in section 5.

2 Related Works

Multiple methods and models have been proposed in the context of implementing a dynamic scaling architecture in a cloud based Big Data applications. Some of these works are presented in this section.
To manage resources efficiently, authors in [8] have proposed a Distributed Dynamic and Customized Load Balancing (ddclb) algorithm in Amazon Elastic Compute Cloud (EC2). The proposed algorithm takes in consideration the CPU, RAM utilization of the cluster along with response time of each instance and assign requests to instances with the lowest metric. Although, the work proposes a dynamic scaling approach, it does not support Hadoop for processing large volume of data [10]. An efficient processing framework of large geospatial datasets has been proposed in [9]. The proposed framework is applied to Hadoop where a node separation to data/compute has been adopted to make unused nodes removing easier and faster. Also, a predictive algorithm has been implemented to calculate the number of resources needed for the workload. The experiments of the framework showed an 80% reduction of the resource's utilization. [10] proposed an auto-scaling framework for analyzing Big Data in the Cloud, it uses Amazon's Cloud Watch to control the environment (CPU, Map/Reduce Tasks, job's state) and add or remove instances upon it, the framework uses a covering HDFS along with Amazon S3 avoid data transfer when

scaling down. To prove its efficiency, a study case of analyzing Twitter Data of multiple universities has been implemented. Authors in [11] have identified container allocation as a key factor that affects Hadoop performance. To ease the overhead occurring they have come up with three methods of data redistribution. The experiments of this work show that adding resources to the cluster dynamically without redistributing data has improvements in term of time response. In [12], researchers have proposed a dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework. The algorithms turn off and on nodes running MapReduce jobs based on current workload, so when the cluster utilization rises above or fall under thresholds predefined by the administrator, scaling up or down action are performed. The results show energy reduction of 33% under average workloads and up to 54% under low workloads.

Some other important research works have gone for proposing a covering subset of nodes in order to perform a dynamic scaling. Leverich and Kozyrakis [13] have proposed an early work on modifying Hadoop to allow scaling down of operational clusters. At least one replica of data blocks must be stored in the covering subset to assure data availability and the scaling down operation are performed only on nodes outside the subset. This work improves cluster utilization, CPU efficiency and provides energy efficiency except that the operation related to scaling down was hardly considered. Also, in case of excess of the workload, the cluster performance degrades. Chen et al. [14] have introduced BEEMR system (Berkeley Energy Efficient MapReduce) where they focused on raising the cluster utilization through separating the cluster into classes according to the job type (Batch, Interactive). They validated their approach by empirical analysis of real-life MIA traces at Facebook which achieved 40% of energy savings. However, long jobs with low level of parallelism cases stay a challenge and result in energy waste [15]. In intention to achieve green computing, Kaushik et al. [16] have proposed green HDFS where they divided the cluster logically into Hot and Cold zones and based on some data classification policies, the files frequently accessed and the new created ones are stored in the Hot zone, which is always powered on. The work was able to meet all the scaling down mandates and has achieved up to 24% energy reduction.

It exists other research works [17, 18] that have focused on balancing the workload for better management of resources. While authors in [19] saw that auto scaling and scheduling background analytics tasks is a proper way for improving resources utilization to maintain required quality of service.

In order to analyze and evaluate the research works presented above in a meaningful way, Table 1 presents a comparison between the approaches which are similar to our research work. We have specified the following criteria: Cloud platform, Controlled metrics and datasets used in each work.

Table 1. Comparative study between the related research works.

Related work	Platform used	Controlled metrics	Datasets	Objective
[8]	Amazon Web Services (AWS).	CPU, RAM	Amazon EC2' users requests	-manage dynamically requests of Amazon EC2 users. -Scale EC2 instances up or down.
[9]	Public or Private Cloud	CPU, RAM	Geospatial applications	-Scaling dynamically Hadoop cluster by provisioning the right number of resources based on workload.
[10]	Amazon Web Services (AWS).	Workload	Social network Twitter	-reduce resources utilization costs. -Scaling Hadoop cluster resources and improving data processing and analyzing in real time.
[11]	Cloud (SAAS)	CPU, MapReduce jobs	Social network Twitter	-Predicting when adding dynamically MapReduce nodes.
[12]	/	Workload, MapReduce jobs	/	-Reduce datacenters' energy consumption by reconfiguring the cluster

Commenté [p1]: Faire attention à la colonne Objective: les objectifs des papiers sont un peu mélangés

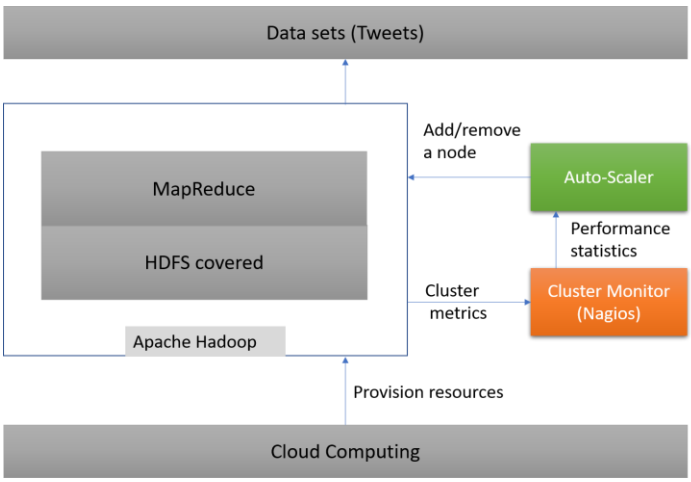
In spite of the increasing research works related to the Cloud based Big Data applications and services, little were actually beneficial in terms of scalability and improving energy efficiency as well as performance. For that, Hadoop Autoscaling still reviews a lot of efforts. In the same context, the goal of this work is to develop a dynamic scaling approach based on multiple metrics and according to various scaling policies. The next section presents the main focus of this paper, the proposed approach, the motivations and challenges of this work.

3 Auto Scaling Approach

The aim of this paper is to propose a dynamic scaling approach in Apache Hadoop to process efficiently large-scale data on a cloud environment. From what's have been established before, we notice that controlled metrics are the key value to decide exactly when adding or removing instances is required.
In this contribution, we have chosen to monitor the CPU, RAM along with pending tasks, the later refers to the number of CPU cores needed for their execution. Each

node has a maximum number of tasks that it can execute simultaneously according to the number of its CPU cores. We also have used Nagios as a cluster monitor, it collects data about the system, the network and the infrastructure. Figure 1 presents the architecture of the proposed approach with its different modules.

Fig. 1. Architecture of the proposed approach



The approach is based on CC as a working environment that englobes the architecture, each component plays its role in the system and assures the good proceeding of each phase (storage, processing, auto-scaling). Two components have been added to the Hadoop framework in order to control its resources, Nagios, the cluster monitor, to control all the system's components and the Auto-Scaler to manage and scale the cluster's resources based on Nagios' information.

3.1 Cluster Monitor Nagios

In order to guarantee the proper functioning of the system, it is fundamental to supervise the cluster. For that, we have chosen Nagios as a monitoring tool to control resources, application status and system operation. Nagios has a built-in alert-based notification system. It is a supervision component; it alerts us in the event of failures encountered on the various monitored components. It can be used to approach different perspectives of surveillance:

- Obtain instant information about the Hadoop infrastructure.
- Generate and receive alerts in case of system failures.
- Analyze, produce reports and graphs on usage and make decisions on future material acquisition.

- Monitor queue depletion and search for available nodes to run jobs.

For HDFS, Nagios checks space usage, files replication and slave nodes balancing. It also allows to monitor the master node (Name Node) which is considered as a sensitive point of the system (its failure leads to a complete shutdown of the system).
For MapReduce, Nagios checks the status of JobTracker and TaskTracker. It provides information related to the jobs being executed and the number of jobs in the queue, this information is almost important in deciding whether we need to add a new worker node or not.
In an autoscaling context, we use metrics provided by Nagios to trigger the resize operation, we can even set up an alarm that notifies us when a threshold condition is trigged.

3.2 Auto-Scaler

Auto-Scaling is a mean of automatically increasing or reducing the cluster, it adds compute nodes when the workload exceeds the capacity of the cluster (Scale-up), or it frees inactive ones when the load is low (Scale-down).
Depending on workload of the cluster; we can transparently increase the number of compute nodes without affecting the general functioning of the system. In case of unused capacity for a period of time, a decrease in node number is realized. However, the Scaling-Down operation causes some services to restart and therefore all running or pending tasks to fail. To avoid such a problem, nodes with running map or reduce tasks will not be considered (will not be released).
To have the auto-scaler perform resizing operations automatically, we define two scaling rules based on CPU usage. These rules tell the Auto-Scaler to add or remove nodes based on average CPU usage and the number of queues.
Once the metrics to be considered are trigged, we can proceed to an auto-scaling according to the defined rules. We specify the minimum and maximum threshold that the auto-scaler must reach to trigger an automatic resizing event.
Two scaling operations are defined, adding a node or Scaling-Up and removing a node or Scaling-Down. If average usage reaches target usage, auto-scale adjusts nodes in a Hadoop cluster either by increasing or decreasing as needed (Figure 3.8). A cool-down time between auto-resize events allows the system to stabilize.

Scaling-Up
We have chosen, in our contribution, to monitor the CPU component and therefore take into consideration all the related information. We also set the threshold for maximum CPU usage (70%). As each Map / Reduce task is executed by a CPU core, we will refer to the number of queued tasks by the number of CPU cores needed for their executions.
Figure 2 shows an algorithm that describes the flow of the Automatic Scaling-Up operation. When average CPU usage hits a threshold, Auto Resize dynamically and conveniently allocates resources in real time. The maximum number of nodes in the cluster must be determined by the user through a configuration interface.

Fig. 2. Scaling Up Algorithm

```
1   Node_max = maximum number of worker node
2   CPU_max = maximum CPU use
3   Begin
4       nodes = number_of_actif_nodes;
5       While (nodes < Node_max) then
6             CPU = current_CPU_use;
7             If (CPU < CPU_max) then
8                CPU_queue = number_of_pending_jobs;
9                CPU_free =  number_of_free_CPU
10               if (CPU_queue < CPU_free )then
11                  Execute_Tasks();
12               Else
13                  Provision_node();
14               End if
15            Else
16                  Provision_node();
17             End if
18      End While
19  End
```

Scaling Down

During a scaling-down operation, it is necessary to define a minimum number of nodes that must remain active to maintain the performance of the cluster and manage the tasks that have occurred in a given time. It is also essential not to remove nodes with unfinished spots or whose reduction spots are in progress. Since sizing operations are applied just on Worker type nodes, removing a node is applied instantly and does not affect cluster operation or data availability.

Figure 3.10 describes the flow of the Automatic Scaling-Down operation, when the average CPU usage reaches the minimum threshold, the auto-scaler decreases the number of active compute nodes which reduces costs and minimizes consumption. energy of the cluster.

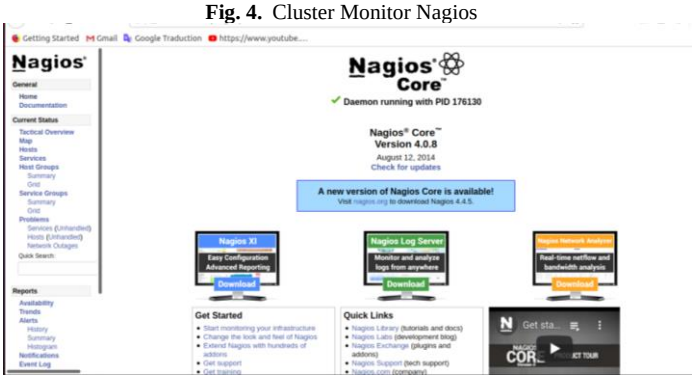
Fig. 3. Scaling Down algorithm

```
1   Node_min = minimum_number_of_worker_node;
2   CPU_min = minimum_CPU_use;
3   Begin
4       nodes = number_actif_nodes;
5       while (nodes > Node_min) then
6             CPU = utilisation courante du CPU
7             if (CPU <= CPU_min) then
8               state= verify_running_jobs;
9                if (state == false ) then
10                   delete_node();
11               end if
12            End if
13      end while
14  End
```

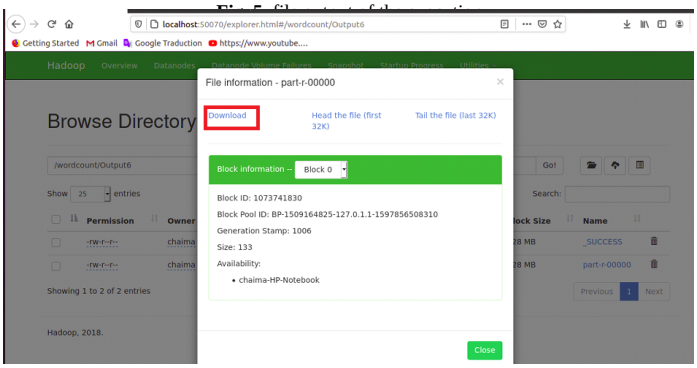
4 Implementation and results

In order to validate the proposed approach, we have installed Hadoop in a virtual machine environment to simulate the Cloud. We have chosen a Hadoop single node cluster to observe the results of the implemented algorithms. In addition, we have installed the cluster monitor Nagios and we have configured it with Hadoop so it can control its resources, Figure 4 shows Nagios interface after configuration.

Fig. 4. Cluster Monitor Nagios



As an execution scenario, we have chosen a file of more than 10000 tweets with multiple hashtags related to the corona virus. These tweets are processed with MapReduce to count the number of each hashtag and store it in HDFS under the supervision of Nagios, Figure 5 shows the results of the execution in HDFS.



The objective of this implementation is to retrieve cluster metrics from Nagios and pass them to the Auto-Scaler. This last compare them to the predefined thresholds and trigger the appropriate action (Scaling Up or Down). Since the approach is implemented in single node architecture, we have managed to display notifications rather than actually adding or removing instance to just validate the efficiency of the Auto-scaling Algorithm. The results showed that in a single node cluster The Auto-scaler module has displayed notifications in the exact time according to the controlled metric and the specified thresholds.

5 Conclusion

The cooperation between Big Data and Cloud Computing has led to a significant waste of energy and high costs, from where comes the need to propose an auto scaling mechanism to control dynamically the cluster according to workload. Hence, we have proposed an approach based on Hadoop to add resources or remove nodes whenever needed according to CPU, RAM and pending jobs metrics extracted by Nagios and sent to. We have implemented the approach in a single node cluster and a virtual machine environment. We have also chosen to process more than 10000 tweets to count hashtags related to corona virus using MapReduce. The simulation showed a promising execution of the proposed algorithm displaying notification about the appropriate scaling action needed.
We aim in the near future to test our approach in a real cloud environment under a various workload (high and low). We also aim to take in consideration other metrics and even predict the right number of resources required to execute jobs and tasks.

10

References

1. Wamba, S.F., Gunasekaran, A., Akter, S., Ren, S.J.-F., Dubey, R., Childe, S.J.: Big data analytics and firm performance: Effects of dynamic capabilities. Journal of Business Research (2017).
2. Hashem, I.A.T., Anuar, N.B., Mokhtar,S.: The rise of "big data" on cloud computing: Review and open research issues. Information systems (2015).
3. Talia, D.: Clouds for Scalable Big Data Analytics, Computer (2013).
4. Mell,P., Grance,T.,:he NIST definition of Cloud Computing,Nationa Institute of Standards and Technology, special publication (2012).
5. Balachandran, B.M.; Prasad, S.: Challenges and Benefits of Deploying Big Data Analytics in the Cloud for Business Intelligence. Procedia Comput. Sci (2017).
6. Barroso, L., & Hölzle, U.: The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, *Synthesis Lectures Comput. Archit.*, vol. 4, no. 1, pp. 1-108(2009).
7. Maheshwari, N., Nanduri, R., & Varma, V.: Dynamic energy efficient data placement and cluster reconfiguration algorithm for mapreduce framework, Future Generation Comput. Syst., vol. 28, no. 1, pp. 119-127(2012).
8. V. Shah, H. Trivedi et al., A Distributed Dynamic and Customized Load Balancing Algorithm for Virtual Instances (2015).
9. Z. Li , C. Yang , K. Liu , F. Hu ,B. Jin, Automatic Scaling Hadoop in the Cloud for Efficient Process of Big Geospatial Data (2016).
10. R. Jannapureddy, Q.Vien, P. Shah, R. Trestian. An Auto-Scaling Framework for Analyzing Big Data in the Cloud Environment (2019).
11. Q. Fu, N. Timkovich, P. Riteau, K. Keahey. A Step towards Hadoop Dynamic Scaling, (2018).
12. N. Maheshwari, R. Nanduri, V. Varma et al., Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework (2011).
13. J. Leverich, & C. Kozyrakis,. On The Energy (In)efficiency of Hadoop Clusters, Operating Syst. Rev., vol. 44, no. 1, pp. 61-65 (2010).
14. C.C. Chen, Y.T. Hasio, C.Y. Lin, S. Lu, , H.T. Lu,& J. Chou. Using Deep Learning to Predict and Optimize Hadoop Data Analytic Service in a Cloud Platform, pp. 909-916. (2017).
15. M.R Jam, L.M. Khanli, & M.K. Akbari,. Survey on Improved AutoScaling in Hadoop into Cloud Envionment, 15th Conference on Information and Knowledge Technology (IKT) (2013).
16. P. Kalagiakos & P. Karampelas. Cloud Computing Learning. The proceedings of IEEE International conference on Application of Informaion and Communication Technologies, Baku, pp. 1-4. (2011).
17. G. S. Domanal, & M.G.R. Reddy. Optimal Load Balancing in Cloud Computing by Efficient Utilization of Virtual Machines, in proceedings of the Sixth International Conference on Communication Systemsnd Networking (COMSNETS), pp. 1-4. (2014).
18. H.M. Mahalle, P.R. Kaveri, V. Chavan. Load balancing On Cloud Data Centres , International Journal of Advanced Research in Computer Science and Software Engineering: IJARCSSE, pp.1-4. (2013).
19. X. Wang, Z.Lu, J.Wu, T.Zhao, P.Hung., InSTechAH: An Autoscaling Scheme for Hadoop in the Private Cloud, IEEE International Conference on Services Computing. (2015).