Effective Resource Utilization in Heterogeneous Hadoop environment through a Dynamic inter-cluster and intra-cluster Load Balancing

Emna Hosni[1][0000−0003−3430−2966], Wided chaari[1], Nader Kolsi[2], and Khaled Ghedira[3]

[1] National School of Computer Sciences, LARIA, Tunisia
{emna.hosni,wided.chaari}@ensi-uma.tn
[2] Tunis School of Business, LARIA, Tunisia nader.kolsi@esct.uma.tn
[3] UIK, Université Ibn Khaldoun, Tunisia
Khaled.ghedira@uik.ens.tn

Abstract. Apache Hadoop is one of the most popular distributed computing systems, used largely for big data analysis and processing. The Hadoop cluster hosts multiple parallel workloads requiring various resource usage (CPU, RAM, etc.). In practice, in heterogeneous Hadoop environments, resource-intensive tasks may be allocated to the lower performing nodes, causing load imbalance between and within clusters and and high data transfer cost. These weaknesses lead to performance deterioration of the Hadoop system and delays the completion of all submitted jobs. To overcome these challenges, this paper proposes an efficient and dynamic load balancing policy in a heterogeneous Hadoop YARN cluster. This novel load balancing model is based on clustering nodes into subgroups of nodes similar in performance, and then allocating different jobs in these subgroups using a multi-criteria ranking. This policy ensures the most accurate match between resource demands and available resources in real time, which decreases the data transfer in the cluster. The experimental results show that the introduced approach allows reducing noticeably the completion time s by 42% and 11% compared with the H-fair and a load balancing approach respectively. Thus, Hadoop can rapidly release the resources for the next job which enhance the overall performance of the distributed computing systems. The obtained finding also reveal that our approach optimizes the use of the available resources and avoids cluster over-load in real time.

Keywords: Heterogeneous cluster · Resource allocation· Multi-criteria decision · Big Data · Hadoop

1 Introduction

Over the last decade, several parallel computing systems have been developed for massive data processing. Hadoop [4] is one of the most popular systems for

[4] https://hadoop.apache.org/

distributed computing and parallel processing. Indeed, the processing of data-intensive applications has become increasingly complex. The processing technique used is to divide large datasets into small partitions, so that each small task runs simultaneously on an individual partition [5]. Hadoop has three major layers, namely HDFS (Hadoop Distributed File System) and MapReduce, used for data storage and processing respectively, and Yarn (Another Resource Negotiator), used for resource management and job scheduling. In a heterogeneous Hadoop environment, over-allocation of resources for some jobs and under-utilization of cluster resources can occur [1]. In this situation, it may be necessary to move jobs from low-performance nodes to high-performance nodes in real time to finish fast. In order to improve the overall performance of Hadoop, effective load balancing is crucial to avoid the overhead of data transfer inter-cluster and intra-cluster, as well as optimize resource utilization. These issues are more common in heterogeneous Hadoop environments with different node and task characteristics. Moreover, if jobs cannot be executed effectively in heterogeneous clusters, a data transfer overheads may be incurred because available resources are not used efficiently to meet resource requirements. Therefore, to balance workloads according to the resources available in the cluster, data must be distributed accurately and efficiently to reduce data transfer costs. In this context, several contributions were developed [5] [9][1] to improve load balancing and resource utilization in Hadoop. Practically, in a heterogeneous Hadoop cluster the CPU cores, memory size and storage speed, etc are not similar. Considering this heterogeneity in the allocation of tasks allows for better load balancing and a reduction in execution time. However, the above-cited articles did not effectively handle node heterogeneity and multiple real-time resource requirements per job. This weakness can lead to sub-optimal load balancing where job requirements do not match the resources available in the Node Managers. Thus, to evaluate the load imbalance in Hadoop Yarn, we used in this study, different types of jobs running on heterogeneous nodes. The new load balancing model is designed for the Yarn architecture ensures an accurate match between the resource required and available ones. First, the deployed system profiles the available nodes by grouping them into clusters of similar capacity. The system then makes a dynamic, multi-criteria decision to rank each job based on its resource demand. It then uses the generated node groups and job rankings to dynamically allocate the most appropriate resources and reduce data transfers within the cluster. Experimental results show that the suggested approach improve load balancing considering a heterogeneous environment. The remaining paper is organized as Section .2 presents the related work. Section.3 formulates the problem statement. Section.4 presents the proposed load balancing policy. Section .5 illustrates the experimental results. Finally, Section. 6 provides some concluding remarks and presents a brief conclusion and future works.

2 Related Works

In recent years, heterogeneity of computing systems in clusters is one of the most critical challenges facing distributed computing systems. It has become an important area of research, especially in the Hadoop Yarn system, thanks to the development of various load balancing strategies and scheduling approaches. In fact, several studies have been conducted to deal with the heterogeneity problems in Hadoop system, mainly load imbalance and resource wastage. The heterogeneous environment may cause an imbalance in resource utilization between over-loaded and under-loaded hosts, which degrades resource usage. For this end, many authors [3][8][1][2][9] have developed a dynamic approaches to deal with the heterogeneity challenges, essentially in terms of resource utilization. In [8] suggested performance-based clustering of Hadoop nodes to load data among the cluster nodes. However, the author did not take into account the different resources requirements by job when loading data, which leads to load imbalance and inefficient resource allocation. Kairos [3], improved Hadoop scheduling decision. In contrast, he only took into account job heterogeneity and neglects the capabilities of nodes, which affect resource usage and result in poor cluster performance. Besides, TMSA algorithm [13] improved the load balancing in Hadoop Yarn. It considered the node heterogeneity using a prediction model integrated on each Node Manager to estimate the end time of a task. However, this algorithm was developed at a small-scale, which does not clearly reflect the scalability of the approach. In [7] the authors designed an approach to balance the workload based only on the computing capacity of the node. However, it uses the static resource to obtain the processing capabilities per node. It can also lead to load imbalance in a heterogeneous environment, which deteriorates the overall performance of the Hadoop system. In [9], the authors proposed a scheduler H-fair in a heterogeneous Hadoop environment. They selects jobs for scheduling based on a global dominant resource fairness heterogeneous strategy, and dispatches them on nodes with a same characteristics to the resource demands using the cosine similarity. This approach was compared to the Fair Scheduler. The authors focused only on the heavy workloads with highly resource. In contrast, in a heterogeneous Hadoop environment, when allocating tasks, it is important to consider multiple task requests to better satisfy the requirements in a parallel computing environment. In [1], the authors proposed a load balancing approach for a MapReduce Job running on a heterogeneous Hadoop cluster. However, they focus on balances the load among the heterogeneous nodes only in the Reduce phase of MapReduce. This work ignores multi-task assignment and does not monitor over-loading and under-loading of nodes while running tasks, which can lead to inefficient and inaccurate resource utilization between high performance and low performance nodes. Therefore, an optimal selection of nodes must be made to achieve a good match between the resource requirements and the nodes capabilities in real time. This is our main motivation for developing efficient load balancing in a heterogeneous Hadoop environment. On the other hand, IoT devices can be called upon to provide a wide variety of services. It is obvious that efficient allocation of IoT resources (processing power, storage capacities, etc.)

would enhance IoT performance. But the optimal allocation of resources for the IoT is difficult to achieve given its its distributed and heterogeneous nature [6]. In this paper, we combine a multi-criteria approach to rank jobs, and clustering of nodes for an accurate allocation decision. This combination improved the accuracy of assigning the most appropriate resources to job demand. We modify the analytical hierarchy process [10] to create dynamic scores according to the resources requirements, i.e., the most demanding jobs are executed in the most powerful nodes within a group of nodes.

3 Problem Statement

Given a heterogeneous Hadoop cluster H consists of a resource manager (RM) and n node managers (NMs) that have different performance, executing a J set of jobs, each with various resource requirements, our aim is to achieve the best match between available resources and resource requirements to improve resource utilization and subsequently reduce inter-cluster and intra-cluster data transfer. We classify the nodes according to the current resource usage (CPU, RAM, Disk I/O) in descending order within each node group. This classification is applied to select the most appropriate nodes for job processing, which minimizes data transfer between nodes and node groups. The node with the highest utilization of available resources has the highest priority for job execution. The proposed policy achieve the best load balancing that reduces the number of remote jobs, improves the locality rate and resource utilization. It also avoids under-loading and over-loading in heterogeneous Hadoop environments. This section formally describes the model used in this paper, which consists of jobs, nodes, and clusters.

3.1 Jobs

We consider that a set of n jobs $J = \{j_1, \ldots, j_m\}$ are assigned to to a heterogeneous Hadoop cluster H. Each job j_i requires different resources in terms of CPU usage, memory usage and I/O. A set of jobs J run in cluster H where each job j_i has a different resource demand. We can define it as follows : $j_i = \{id_i, Dcpu_i, Dm_i, D(I/O)_i, T_j\}, i = 1, \ldots, m$, where id_i is an unique number that identifies a job j_i, and $Dcpu_i, Dm_i, D(I/O)_i$, represent the different resources requirements by a given job. T_j is a set of tasks generated by a job j that must be completed for the job to finish its execution successfully. To evaluate the data transfer time per task T_t^{tran}, we can estimate it as follows [4]:

$$T_t^{tran} = \frac{|d_t|}{cap_t} \tag{1}$$

where, $|d_t|$ is the size of data block, cap_t is task transfer capacity.

3.2 Nodes

The heterogeneous Hadoop cluster H contains a set of nodes denoted as $H = \{N_1, \ldots, N_n\}$ having different capacities. We define heterogeneous hardware resources with various performance parameters (*CPU, RAM, Disk I/O*) provided

by each node. The cluster H is a group of related nodes including a Master Node, represented by RM, and slaves nodes which are the NM. Note that in the current Hadoop YARN system, only two resources are included, memory and CPU during resources allocation process. However, the proposed load balancing model take into account the CPU utilization, Disk I/O utilization and Memory utilization of nodes in H during job execution. The CPU performance of each node $P(cpu_i)$ is given bellow [14] :

$$P(cpu_i) = f_{cpu_{(i)}} \times nb_{co(i)}, \quad nb_{co} \succ 1. \tag{2}$$

Where $f_{cpu_{(i)}}$ is the CPU frequency of the i^{th} node, nb_{co} is the number of CPU cores. During Job processing, we can calculate the CPU utilization (%) as [12] :

$$U(cpu_i) = (100\% - Time\ spent\ \ in\ idle\ \ Task\ \%) \tag{3}$$

Where, $U(cpu_i)$ is the CPU utilization of the Node i. The memory capacity of each node i depending on the size of the RAM of each node i : $RAM_i = Size_{Ram}(i)$. We carry out the Memory Utilization (%) as :

$$U(M_i) = \frac{Rr_i}{T(M)} \times 100 \ \ i = 1 \ldots, n \tag{4}$$

Here $U(M_i)$ is the memory utilization of the i^{th} node, Rr_i is the size of used memory of N_i for the given job and $T(M)$ is a total memory available in the node. Then, we compute the Disk I/O utilization (%) as [12] :

$$U(D_i) = \frac{T_b}{T_{rq}} \times 100 \ \ i = 1 \ldots, n \tag{5}$$

$U(D_i)$ is the disk I/O of the i^{th} node, T_b is the total number of bytes transferred and T_{rq} is the total time from the first request until the end of the last transfer. The node's storage in cluster H is available for use in the HDFS, $S_N(i)$ represent the storage capacity of the node i. The global load of the node can be estimated as follows:

$$Load(N_i) = \alpha \ \ U(cpu_i) + \beta \ \ U(M_i) + \gamma \ \ U(D_i) \tag{6}$$

Here coefficients $\alpha + \beta + \gamma = 1$, which take probable values between $[0, 1]$. The value of α is relatively high for CPU-intensive jobs, the value of β will be increased if the computational capacity is memory-dependent and the value of γ will be high for jobs that require more Disk I/O. The node load ratio is calculated as follows:

$$L_{node}(N_i) = \frac{Load(N_i)}{cap_{(Ni)}} \times 100 \tag{7}$$

Where $cap_{(Ni)}$ is the resource capacity of node in terms of CPU, Memory and Disk I/O.

3.3 Cluster

The cluster load ratio $Load(H)$ can be calculated as follows:

$$Load(H) = \frac{\sum_{i=1}^{n} Load(N_i)}{\sum_{i=1}^{n} cap_{(N_i)}} \times 100 \tag{8}$$

Where $\sum_{i=1}^{n} cap_{(N_i)}$ is the total resource capacity of all nodes in the cluster cluster$H$ in terms of CPU, Memory and Disk I/O. As shown in formula (8), this value is an accurate measure of the cluster load balancing. This should be the used capacity of all nodes i in the cluster H. To calculate the over-loading of the cluster H, the threshold is taken as the sum of the maximum memory usage $max(U(M))$, maximum Disk I/O $max(U(D))$ and maximum CPU usage $max(U(cpu))$ in all the cluster, divided by the number of nodes n in the cluster. The equation for the over-load threshold $O_{(H)}$ is shown below :

$$O_{(H)} = \frac{max(U(cpu)) + max(U(M)) + max(U(D))}{n} \tag{9}$$

Hadoop clusters operating under heavy load have a significant influence on cluster performance. Thus, in order to ensure the success rate of task execution in Hadoop Yarn, the proposed approach sets a threshold $O_{(H)}$, represented by the maximum load defined in Eq.9. When the load of a node exceeds the threshold value, The Resource Manager RM should not allocate tasks to this node yet. Once the cluster is over-loaded, the maximum load is automatically set to 100%.

4 Effective Load Balancing Policy in Heterogeneous Hadoop Clusters

The main goal of the proposed policy is to improve the performance of Hadoop which is a distributed computing systems requiring an optimized utilization of its available resources at run-time. Therefore, efficient clustering of available nodes and multi-criteria job ranking are combined to effectively assign the cluster resources according to the job requirements. The major contributions of our policy is presented in three steps:

- iterative clustering of available nodes using efficient k-means algorithm. [11].
- The analytic hierarchy process [10] is used to assign score for jobs according to their resource requirements.
- Efficiently distributes the load in real time within and between groups of nodes.

4.1 Clustering of nodes

In the Hadoop Yarn cluster, the Node Managers NMs send nodes status and heartbeat messages, containing the resources availability in terms of CPU utilization, Memory utilization, Disk I/O utilization to RM. In this paper, we consider variations in resource utilization and hardware failures in the cluster. For

this reason, we start the clustering node phase, once the changes in the cluster and the failed nodes are identified by the RM. Groups of similar nodes are built in real time based on the cluster status obtained by the heartbeat messages. We applies the K-Means algorithm with an elbow [11] to optimize the effectiveness of k-means in the processing of massive data i.e. when the cluster scales up. Combining the elbow with K-Means resulted in a graph with error decreasing, increasing the value of K then graph will decrease slowly until a stable result of the K-value is achieved. Find K as the number of clusters to select the number of K groups to be used for clustering. The information about the state of the nodes cannot be processed directly, because there is a large difference between the variables in the dataset. Therefore, we use the Min-Max normalization [11] for the CPU speed, Memory and Disk I/O information's. The features we select for clustering will be expressed later as labels for groups of nodes. After creating the groups of similar nodes, the groups will be ordered for all features, where the performing node is among the group of nodes with a higher rank. After that, the system attribute values ranging from 1 to n to the respective labels, where n is the number of node clusters (groups). These labels are sent to the nodes tracked by the Resource Manager and can then be used during the dynamic resource allocation process. The node clustering phase is triggered according to the variation of resource usage i.e. the current status of each node in the whole cluster H. In the case of a failure node our algorithm ignores this node until its reactivation. We present below the clustering steps used in our load balancing policy:

- **1.** Set three performance features (CPU, Disk I/O, RAM) in the k-means algorithm.
- **2.** Receive the status information from all NMs (*if a node is failed, it will be temporarily disregarded*)
- **3.** *Min-Max Normalization* [11] is used before the calculating process (*all values are scaled between the range* $0, 1$ *where* 0 *is the minimum value and 1 is the maximum value*) :
- **4.** achieve the optimal number of clusters to create using the elbow method [11].
- **5.** Calculate the distance of each object to each centroid.
- **6.** Assign each item into the nearest centroid
- **7.** Perform iteration, then process find the position of new centroid
- **8.** Repeat Step.7 if the new centroid position with the previous centroid is not the same. (*If not, then return the result of clustering nodes.*)

4.2 Jobs Ranking

When, the client submits a job to the Resource Manager RM, the proposed strategy identifies continuously the requested resource of each job j.

At this stage, dynamic AHP rankings based on hierarchical job information are applied. The jobs raking step is developed using AHP method [10]: this step is carried out by improving the accuracy of the AHP model. This improvement

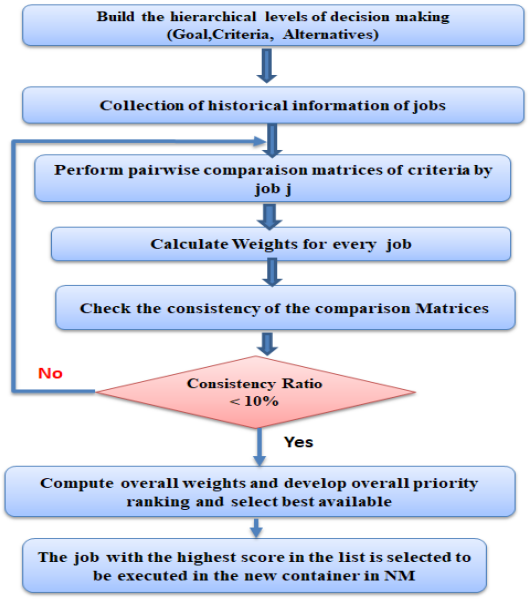

Fig. 1. Flowchart for analytical hierarchy process used for job raking

consists by collecting historical information using the APIs provided by Hadoop to assign weight for jobs according to their resource requirements. Each job has its own set of criteria (CPU,RAM,I/O) in terms of their resource demand. Once all scores of all jobs are calculated, our system select the job with the highest ranking means that it requires the highest amount of resources for processing as shown in Fig.1. This process is applied dynamically depending on the current state of the job submitted. The selected container is the most appropriate for the resource requirements of the job. Our system, provide the best trade-off between jobs J which is ranked based on Analytic Hierarchy Process AHP [10] and the resource available in the heterogeneous cluster H. The second phase aims to detect the most resource-intensive jobs with a higher score, based on the different current resource demands. The elements of the matrix m_{ij} are represented by the weighing of the different criteria of the jobs defined by $\frac{w_i}{w_j}$. We present the pairwise comparison matrix for criteria level as follows:

$$\begin{array}{c} \textbf{Criteria} \\ RAM \\ CPU \\ I/O \end{array} \begin{array}{c} \begin{array}{ccc} RAM & CPU & I/O \end{array} \\ \begin{pmatrix} 1 & m_{12} & m_{13} \\ \frac{1}{m_{12}} & 1 & m_{23} \\ \frac{1}{m_{13}} & \frac{1}{m_{23}} & 1 \end{pmatrix} \end{array}$$

The consistency of the comparison Matrix should be checked in order to ensure a high level of consistence. The consistency index CI was obtained by the formula

[10] :

$$CI = \frac{\lambda_{\max} - n}{n - 1} \tag{10}$$

where $\lambda_{max} \geq n$ and n is the number of criteria which representing the number of the compared elements. The consistency ratio CR was calculated using the formula below :

$$CR = \frac{CI}{RI} \tag{11}$$

For reliable result, CR value should be less than or equal to 0.1, Otherwise, there is an inconsistency matrix and the comparison must be recalculated. RI is a random consistency index presented by Saaty [10], where the number is different considering the number of attributes used.

4.3 Dynamic inter-cluster and intra-cluster load balancing

After the job ranking phase, the Resource Manager receives a list of job priorities. Our load balancing policy provides RM with the most appropriate nodes for each job. The proposed system uses the node clustering and the job ranking list currently provided to achieve the best match between job demand and available resources in Hadoop cluster. Thus, labels are created for node clusters according to their current capabilities. To evaluate the intra-cluster and inter-cluster load, we have created the list of node groups denoted L, l_i the i^{th} node group in this list and m the number of elements in L. We denote the total of CPU performance in node group l_i by l_{cpu} which is based on Eq 2. Then, the system sort the nodes in L in descending order in terms of the CPU performance.

$$Cpu_{(li)} = \frac{l_{cpu(i)}}{\sum_{k=1}^{m} l_{cpu(k)}} + Cpu_{(l_{i-1})} \tag{12}$$

where, $Cpu_{(li)}$, is the CPU utilization of l_i, $\sum_{k=1}^{m} l_{cpu(k)}$ is the total of CPU performance in the list of node groups L. $l_{cpu(i)}$ the total number of CPU cores available in the node group l_i, $Cpu_{(l_{i-1})}$ is the CPU utilization of l_{i-1}. The Resource Manager records the CPU usage received from the scoring jobs phase of the current workloads and their execution history. For example, let a CPU utilization for job j is 210% that mean a 100% utilization of **2** full CPU cores and 10% of a third. The allocation system applies Cpu_{l_i} according to the multi-criteria job ranking. The result obtained from $Cpu_{(li)}$ is used to define a CPU utilization range for each group of nodes l_i in terms of CPU performance (See Eq. 3). The *minimum* $U(cpu_i)$ is the *lower bound* of CPU usage per node in l_i and the value of $Cpu_{(li)}$ is the *upper bound* of CPU usage. This step is applied on all features to build intervals for Memory utilization and Disk I/O utilization using Eq.4 and Eq.5 in each group of node l_i. Therefore, the proposed system check in which range the CPU, RAM or disk I/O usage of such a job is situated. Then, the system give the CPU characteristic a value between 1 and n according to the rank of the range in which the job falls. Furthermore, the load balancing policy defines the current global load in terms of CPU, RAM and disk I/O

simultaneously of each node using $load(N_i)$, the load of the node group li with $load(H)$, and calculates the maximum resource usage thresholds of each node cluster by $O_{(H)}$. If the utilization rate of the node i is higher than the maximum resource usage threshold $O_{(H)}$ of the group of node (cluster). In this case, RM should not allocate any tasks to this node, a new node clustering will restart. As a result, this node will be placed in the low performance node group. The introduced system aims to improve load balancing on a heterogeneous Hadoop cluster so that the least resource-intensive jobs do not block the most powerful nodes. The Load balancing conditions are : **intra-cluster** $L_{node}(N_i) \leq load(H)$ and, **inter-cluster** $load(H) \leq S_{(H)}$. Let L_p be a list of pairs nodes-jobs where the intra cluster nodes meet the resource requests of the jobs. The pairing-rate for a nodes-jobs pairs is determined in 13 :

$$P(N, J) = \sum_{k=1}^{r} |N_k - J_k| \tag{13}$$

where r is the number of features (CPU, RAM, I/O), N_k and J_k are the feature labels for the job and node grouping pair. The feature label values are derived from the node grouping and job scoring from previous phases. After applying the pairing-rate $P(N, J)$ to all L pairs, the lowest pairing-rate indicates the best resource allocation. To determine the matching rate, we give an example of the matching between three group of nodes using the following matrices.

group of nodes 1

		RAM	*CPU*	*I/O*
		1	**1**	**1**
Rr	3	2	–	–
Rcpu	3	–	2	–
R(I/O)	2	–	–	1

group of nodes 2

		RAM	*CPU*	*I/O*
		3	**3**	**3**
Rr	3	0	–	–
Rcpu	3	–	0	–
R(I/O)	2	–	–	1

group of nodes 3

		RAM	*CPU*	*I/O*
		1	**1**	**2**
Rr	3	2	–	–
Rcpu	3	–	2	–
R(I/O)	2	–	–	0

Fig. 2. Example of three matrices of node groups

The example explain where job j has to be allocated to one of three available node groups (conducted during phase 4.1). The job j has the labels $l_1 = 3, l_2 = 3, l_3 = 2$, where l_1 is the Rr which is RAM label,l_2 the $Rcpu$ which is CPU label, and l_3 the $R(I/O)$ which is I/O label. The result of applying of $P(N, J)$ to all pairs of L is the sum of the main diagonal. The node groups **1** has 2+2+1 =5, the second has 1 and the third has 4 on the diagonal. When the sum of the diagonal is lower the load balancing on the Hadoop cluster is optimized. In this case, the group of nodes **2** is the best to fulfill the resource demand of the submitted job. If there is more than one node group with the same ranking,

we start with the most performing node group. The values obtained from this matching that are closer to **0** mean that the resource requirements of the job match the capabilities of the nodes. Since, higher resource demand leads to higher values J_k. Within the node groups, the proposed load balancing select the node according to the intra-cluster condition, which avoid the over-loaded situation in Hadoop cluster. It also prevents resource-intensive tasks from being allocated to lower performing nodes. The presented hybrid load balancing system performs iterative clustering of nodes using the efficient k-means algorithm, which creates groups of nodes with similar performance. After that, the system labels the jobs using AHP ranking while considering their resource demand. The jobs are sorted according to their overall weights. Then, the system combine the node groups and job scoring to achieve an accurate match between the required and available resources. It is based on the matching function explained by the pairing-rate for a nodes-jobs $P(N, J)$.

5 Experimental results

The evaluation of the load balancing approach proposed in this paper is performed with a different configuration of each node. The cluster consists of 10 heterogeneous nodes, one of which is defined as a resource management node and nine as worker nodes with heterogeneous capabilities. All nodes were deployed using VMware and were run on the Ubuntu 16.04 operating system. Our experimental environment was deployed in two physical machines: one has a (CPU Intel core I7-12700KF,3.6 GHz, 12-cores, 32 G of RAM), and the other has a (CPU Intel Core i9-7960X, 2.8GHz, 16 Cores 64G of RAM). We used Hadoop YARN 2.7.6 to run the experiments. We implemented our load balancing algorithm in all nodes. The HDFS block size and the replication level were set to 128MB and 3, respectively. First, the proposed approach classifies nodes according to their performance at run time. This node clustering is carried out according to three main features : CPU, RAM and Disk I/O. The k-means algorithm generates 3 groups of nodes that have an intra-group similarity in terms of resources availability. A high-performance node group including N2, N8 and N6, medium-performance nodes N3 and N1, and the low performance nodes including N7, N5,N9 and N4. The multi-criteria job scoring process dynamically provided the ranking of 10 jobs submitted at the same time, considering the heterogeneity of its resource demands is shown bellow. Apache Hadoop Yarn assumes that if the CPU usage is less than 20%, it is a low CPU intensive job, otherwise it is considered a high CPU intensive job. We can see that jobs 8, 6 and 2 are resource intensive in terms of three criteria *CPU, RAM and I/O*, compared to the others jobs. A different types of jobs such as WordCount and TeraSort are used to carry out the experiment. The job's configuration in the experiment are: WordCount $\alpha = 0.8$, $\beta = 0.2$,$\gamma = 0$, and for TeraSort $\alpha = 0.2$, $\beta = 0.8$, $\gamma = 0$ Our load balancing algorithm reduced noticeably the competition time of 10 jobs by 23% , 37% compared with H-fair and LB approach [1] respectively. This allows to rapidly release the containers for the next job which

enhance the overall performance of the distributed computing systems. We used different job type on different groups of nodes. Our approach achieved the minimum time spent while processing the job in the high performance group and in the medium performance group. The suggest load balancing system assign the resource-intensive jobs to the group of high-performance nodes, and inside each group the nodes are sorted in descending order. Thus, job will be dynamically allocated based on the load balancing conditions and the current capacity range of each node to avoid over-loading in the heterogeneous cluster.

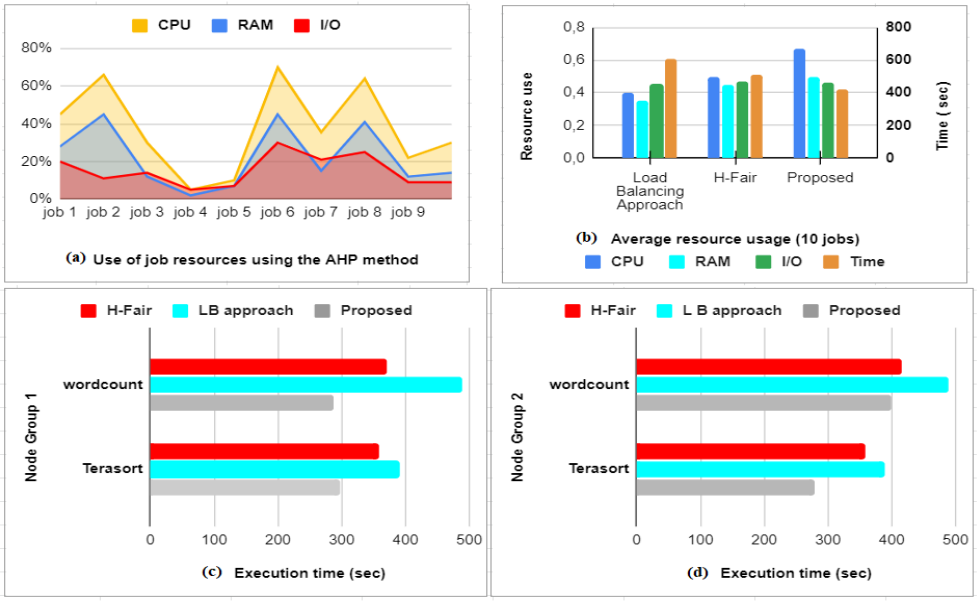

Fig. 3. (a). AHP Job Ranking, (b). Average resource usage, (c). Node Group 1, (d). Node Group 2

6 Conclusion

In a heterogeneous environment, several problems, such as under-loading and over-loading, may occur, which requires effective load balancing between and within clusters. In this paper, an efficient hybrid load balancing approach applied in a heterogeneous Hadoop cluster was proposed. It combines iterative clustering of nodes and a dynamic multi-criteria decision that scores jobs according to the required resource. Our main objectives were to improve the resources utilization, reduce the job competition time and avoid load imbalance in a heterogeneous cluster. In future work, we will improve our approach with higher scale while evaluating the energy consumption of Iot heterogeneous devices in real time.

References

1. Bawankule, K.L., Dewang, R.K., Singh, A.K.: Load balancing approach for a mapreduce job running on a heterogeneous hadoop cluster. In: International Conference on Distributed Computing and Internet Technology. pp. 289–298. Springer (2021)
2. Chen, W., Rao, J., Zhou, X.: Addressing performance heterogeneity in mapreduce clusters with elastic tasks. In: 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 1078–1087. IEEE (2017)
3. Delgado, P., Didona, D., Dinu, F., Zwaenepoel, W.: Kairos: Preemptive data center scheduling without runtime estimates. In: Proceedings of the ACM Symposium on Cloud Computing. pp. 135–148 (2018)
4. Jia, R., Yang, Y., Grundy, J., Keung, J., Li, H.: A highly efficient data locality aware task scheduler for cloud-based systems. In: 2019 IEEE 12th International conference on cloud computing (CLOUD). pp. 496–498. IEEE (2019)
5. Karun, A.K., Chitharanjan, K.: A review on hadoop—hdfs infrastructure extensions. In: 2013 IEEE conference on information & communication technologies. pp. 132–137. IEEE (2013)
6. Li, X., Da Xu, L.: A review of internet of things—resource allocation. IEEE Internet of Things Journal **8**(11), 8657–8666 (2020)
7. Naik, N.S., Negi, A., Br, T.B., Anitha, R.: A data locality based scheduler to enhance mapreduce performance in heterogeneous environments. Future Generation Computer Systems **90**, 423–434 (2019)
8. Paik, S.S., Goswami, R.S., Roy, D., Reddy, K.H.: Intelligent data placement in heterogeneous hadoop cluster. In: International Conference on Next Generation Computing Technologies. pp. 568–579. Springer (2017)
9. Postoaca, A.V., Pop, F., Prodan, R.: h-fair: asymptotic scheduling of heavy workloads in heterogeneous data centers. In: 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). pp. 366–369. IEEE (2018)
10. Saaty, T.L.: Decision making for leaders: the analytic hierarchy process for decisions in a complex world. RWS publications (1990)
11. Syakur, M., Khotimah, B., Rochman, E., Satoto, B.D.: Integration k-means clustering method and elbow method for identification of the best customer profile cluster. In: IOP conference series: materials science and engineering. vol. 336, p. 012017. IOP Publishing (2018)
12. Thu, M.P., Nwe, K.M., Aye, K.N.: Replication based on data locality for hadoop distributed file system. 9th International Workshop on Computer Science and Engineering (WCSE 2019 ... (2019)
13. Wang, M., Wu, C.Q., Cao, H., Liu, Y., Wang, Y., Hou, A.: On mapreduce scheduling in hadoop yarn on heterogeneous clusters. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 1747–1754. IEEE (2018)
14. Yan, W., Li, C., Du, S., Mao, X.: An optimization algorithm for heterogeneous hadoop clusters based on dynamic load balancing. In: 2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). pp. 250–255. IEEE (2016)