Autonomic Dominant Resource Fairness (A-DRF) in Cloud Computing

Amin Fakhartousi
Faculty of Science and Technology
Bournemouth University
Bournemouth, United Kingdom
s5222844@bournemouth.ac.uk

Sofia Meacham
Faculty of Science and Technology
Bournemouth University
Bournemouth, United Kingdom
smeacham@bournemouth.ac.uk

Keith Phalp
Faculty of Science and Technology
Bournemouth University
Bournemouth, United Kingdom
kphalp@bournemouth.ac.uk

***Abstract*— In the world of information technology and the Internet, which has become a part of human life today and is constantly expanding, Attention to the users' requirements such as information security, fast processing, dynamic and instant access, and costs savings has become essential. The solution that is proposed for such problems today is a technology that is called cloud computing. Today, cloud computing is considered one of the most essential distributed tools for processing and storing data on the Internet. With the increasing using this tool, the need to schedule tasks to make the best use of resources and respond appropriately to requests has received much attention, and in this regard, many efforts have been made and are being made. To this purpose, various algorithms have been proposed to calculate resource allocation, each of which has tried to solve equitable distribution challenges while using maximum resources. One of these calculation methods is the DRF algorithm. Although it offers a better approach than previous algorithms, it faces challenges, especially with time-consuming resource allocation computing. These challenges make the use of DRF more complex than ever in the low number of requests with high resource capacity as well as the high number of simultaneous requests. This study tried to reduce the computations costs associated with the DRF algorithm for resource allocation by introducing a new approach to using this DRF algorithm to automate calculations by machine learning and artificial intelligence algorithms (Autonomic Dominant Resource Fairness or A-DRF).**

Keywords*— *A-DRF, Resource Allocation, Cloud Computing, Fairness, Autonomic, Dominant Resource, DRF

I. INTRODUCTION

The allocation of resources in cloud computing is the most significant alarm. So many parameters will damage the virtual machine, such as time consumption, service quality, cost, etc., because the cloud is heterogeneous. When there are massive requests, it is necessary for a virtual machine to the allocation of resources in the shortest time [1].

In the literature and the industrial fields, the need for an efficient and fair resource allocation in cloud systems has long been recognised. Cloud computing is a promising platform that provides on-demand and scalable computing resources, as well as usage-based payment. However, the IT world's increasing complexity has made Quality of Service (QoS) in the cloud a complex topic and an NP-hard challenge over the last decade [2].

Fairness in resource allocation is recognised as a fundamental issue in computing systems, at least for the last two decades. The most conventional research in this area is computer networks in which the fair allocation of bandwidth among flows with various requirements, is a substantial concern [2-4]. By rapidly increasing the use of computing frameworks, resource allocation has become a more interesting and challenging concern. In modern computing frameworks, such as cloud computing, users show a significant interest in submitting multiple types of resources like CPU, RAM, etc. Therefore, fairly allocating resources, taking into account such diversity, is a complicated and NP-hard problem [4-6].

One of the proposed methods for fair resource allocation is the DRF algorithm. DRF employs the advanced filling algorithm as well as a linear optimisation method to allocate resources among users. Additionally, DRF satisfies some desirable fairness properties such as the sharing incentive, strategy-proof, envy-free, resource monotonicity, and Pareto-efficiency [3]. Among all these promising features, nonetheless, DRF suffers from autonomic decisions in calculating dominant resources for incoming requests. That is said, in each iteration, DRF tries to calculate dominant resources, which leads to increasing the response time in serving users' requests and degrading Service Level Agreement (SLA) [2, 4]. DRF although originally published in 2011, is a method that is still heavily used inside Apache Hadoop and Yarn scheduler which is widely used in many systems. A comparison with other state-of-the-art methods is detailed in the literature review.

To tackle this problem associated with DRF, we propose Autonomic DRF called A-DRF that empowers it to make real-time decisions. A-DRF, employs machine learning algorithms to monitor and classifies incoming tasks. More details of this algorithm will be discussed in the following sections.

The rest of this article is structured as follows. Section 2 presents the literature review of this article. Section 3 introduced related works. Section 4 describes the proposed method. Section 5 presents experimental result, including dataset data description and performance evaluation. Lastly, the conclusion and future work of this article are presented in Section 6.

II. LITERATURE REVIEW

This section first explains the cloud computing Service and its scheduling and resource allocation techniques and then reviews machine learning and its methods.

A. Cloud Computing

One of the newest platforms for mobile computing, IT enterprise, and business is Cloud Computing. Instead of purchasing resources such as Software, Memory, CPU, and Input or Output devices (I/O), etc. Because of dramatically increased cloud usage, a proper and effective allocation of resources has been led to a challenge. Diverse promising technologies were developed to enhance resource allocation process efficiency [7]. However, when the system is overloaded, there is some inefficiency regarding resource scheduling. So, to improve the performance of the resource allocation process, a proper scheduling algorithm is required.

The following will introduce three basic methods in allocating resources, namely First Input First Output (FIFO) Scheduling, Capacity Scheduling, and Fair Scheduling, and then compare them.

1) First Input First Output (FIFO) Scheduling: First Input First Output (FIFO) scheduling is a common scheduling algorithm that creates one job queue. Users submit all assignments to the queue according to the rule of first input, first output. This scheduler is easy to use and requires no configuration. However, it's not appropriate for shared clusters. Because massive applications consume all of the resources in a cluster, each application must wait its turn. Therefore, this scheduling method covers neither multiple user management nor cluster sharing [8].

2) Capacity Scheduling: Yahoo created Capacity Scheduler (CS) to handle a large number of resource sharing requests. Equal resource allocation according to the queue length is the fundamental thought behind the Capacity Scheduler. Once the job is executing, the priority is not supported by this algorithm. Therefore, some high preemption jobs may suffer from a longer lag than their deadline [9].

3) Fair Scheduling: Another popular scheduler offered for cloud systems is the Fair Schedule (FS) that is used to allocate a proportionate amount of resources to all requests. If there is only one request in the system, all the cluster's resources can be used until entering another job. The fair scheduler allows shorter jobs to finish in proper time, whereas long jobs are not starving [9]. One of the internal policies that can be set up with Fair scheduler is Dominant Resource Fairness or DRF. In a system with many kinds of resources, it is a just policy for allocating resources. In other words, DRF is a scheduling technique with different types of resources that is similar to the fair scheduling strategy, except that a new parameter called dominant resource is used to specify the most requested resource type. The instance's dominant resource is the most frequent type of requested resources (job or user). DRF attempts to balance instances' dominant resource usages by making scheduling decisions [3]. As shown in Algorithm 1, DRF keeps track of both the total resources allocated to each user and each user's dominant share, s_i. DRF selects the client with the smallest dominant share at each stage among those with tasks ready to execute. DRF assumes that a user can perform several tasks, each with a separate demand vector, and it uses attribute D to represent the demand vector for the next task the user wishes to start. To keep it simple, the pseudo-code does not catch the completion of a job. In this scenario, the task's resources are extracted by the user, then DRF chooses the user with the least dominant share to execute her task once again.

Algorithm 1: DRF [3]

$R = < r_1, r_2, \dots, r_n >$: Capacity of Resources.
$C = < c_1, c, \dots, c_n >$: Consumed Resources (initially 0).
$D = < d_1, d_2, \dots, d_m >$: Demanded Resources.
$A = < a_1, a_2, \dots, a_m >$: Allocated Resources (initially 0).
s_i (i = 1: n) : user i's dominant shares, (initially 0).

1. **pick** user i with lowest dominant share s_i
2. **if** $C + D_i \leq R$ **then**
3. $C = C + D_i$ (update consumed vector)
4. $A_i = A_i + D_i$ (update i's allocation vector)
5. $s_i = max_{j=1}^{m}\{\frac{a_{i,j}}{r_j}\}$
6. **else**
7. **Return** A
8. **end if**

B. Machine Learning

Machine learning focuses on applications that enhance their decision-making or predictive performance over time through learning from their past experiences. A machine learning algorithm is normally classified as supervised and unsupervised. the purpose of supervised learning is to predict the right answer for new data using training data that has been labelled with classes. In unsupervised learning, unlike supervised learning, there is no label for the data. The vast majority of algorithms in machine learning are supervised [10]. A machine-learning supervised algorithm can be used to accelerate resource allocation in cloud computing. Regression and Classification are two types of supervised machine learning techniques. The key difference between Regression and Classification algorithms is that we use Regression algorithms to predict the continuous values and use Classification algorithms to predict/classify the discrete values. A machine-learning supervised algorithm and specifically the regression type can accelerate resource allocation in cloud computing. Several types of regression analysis methods can be used based on the number of factors that contain the shape of the regression line, the number of independent variables, and the type of target variable. In the following, six regression methods have been examined in this project.

1) Linear Regression: In machine learning, Linear regression is one of the most fundamental forms of regression. A predictive feature and a dependent feature are linked linearly in the linear regression model. When there are multiple independent variables in the data, linear regression is referred to as multiple linear regression models [11].

2) Polynomial Regression: Another type of regression analysis method in machine learning is Polynomial Regression, which is like Multiple Linear Regression with a bit of alteration. The relationship between dependent and independent features in polynomial regression is marked by the n-th degree [12].

3) KNN Regression: K Nearest Neighbours (KNN) Regression is another technique to estimate the relationship among independent features and the continuous result by averaging the same neighbourhood observations. The neighbourhood size must be arranged using cross-validation to choose the size that minimizes the mean-squared error (It will be seen later). A large K value is generally more accurate because it reduces overall noise; however, it comes at the cost of blurring the distinct boundaries within the feature space [13].

4) Support Vector Regression (SVR): Support Vector Machine (SVM) can also be used as a regression technique called SVR while retaining all of the algorithm's main

characteristics, such as maximal margin. SVR is created according to the Support Vector Machine (SVM) concept and minimise the error by customising the hyperplane that maximises the margin while keeping in mind that some error is tolerable. Despite being less popular than SVM, SVR has been shown to be an effective tool for estimating real-value functions [14].

5) Bayesian Regression: Bayesian regression is a form of machine learning regression that employs the Bayes theorem to determine the regression coefficients values. After determining a model, the method calculates the model predictions and the posterior distribution of parameters. Bayesian Linear Regression is similar to both ridge and linear regressions, but it is more robust than basic linear regression. This statistical analysis enables the technique to specify complexity during training, resulting in a model with a lower likelihood of overfitting [15].

6) Multi-Layer Perceptron Regression (MLP) : MLP is a supervised learning algorithm capable of learning a non-linear function estimator for classification or regression. The difference between MLP and logistic regression is that there can be one or more non-linear layers called hidden layers among the input and the output layer. The loss function for MLP regression scenarios is a square error, and the loss function for classification is cross-entropy, and it can operate for single and multiple goal values regression [16].

III. Related Works

Many researchers have contributed by optimising the work-time and resources to improve resource schedulers' performance. Because multi-resource fairness in organisations has been an obstacle for hierarchical scheduling, within large organisations, DRF is extended for hierarchical scheduling [17]. However, if the number of nodes increases, the recalculation of dominant shares in each node would also need to be optimised. DRF, in particular, must sort request timestamps [3] and has an O(log n) complexity per allocation, where n is the number of users in each timestamp. It isn't easy to implement DRF at high speeds when n is big.

In [18], Tetris has been presented as a new scheduler that achieves a lower MakeSpan by using multidimensional bin packaging. Although the job resource requirements can be unpredictable, it assigns the minimum amount of resources to the jobs with the minimum amount of remaining work. They used heuristics to provide both at the same time because fairness is incompatible with high efficiency.

The purpose of [19] is to introduce HRSYARN (fair resource sharing based on Heuristics), a new resource scheduling that employs the Weighted Arithmetic Mean (WAM) to ensure fair resource sharing between tenants. Additionally, it includes a heuristic table that keeps track of resources that have been lent or leased to and from other tenants. In terms of resource usage efficiency and fairness, their scheduler outperforms other long-term YARN schedulers currently available. But there is no discussion or outcome regarding the scheduling of multiple resource types. Despite the fact that the majority of schedulers focus on short-term optimisations through greedy decisions, Altruistic Scheduling [20] takes a long-term optimisation approach. They demonstrated that it performs similarly to DRF while reducing completion time.

[21] is another research that works on long-term fairness. They introduced Long-Term Resource Fairness (LTRF) to allocate resources in a fair manner and Hierarchical Long-Term Resource Fairness (H-LTRF) as LTRF extension to solve the problem of memoryless resource allocation. LTRF is limited to fair allocation to a specific resource entity, which can be considered a constraint.

[22] some changes were made to the fair scheduling algorithm by considering fairness limitations to improve this algorithm. It proposed a system with several phases, including pool resource allocation, work classification, job sorting, job priority adjustments, and delay time adjustments.

Based on the experimental result section (Section 5), our scheduling method, A-DRF, shows better performance than DRF. It has been demonstrated to decrease resource allocation run time because the time complexity of A-DRF is O(1) across to DRF with O(log n) which is a radical improvement.

IV. Proposed Method

The main idea of job scheduling is maximising the throughput and minimising the jobs' run time under different requirements and resource constraints. The DRF algorithm for job scheduling has a high time complexity to calculate resource allocation; In other words, every scheduling decision takes O(log n) time, while n is the number of users. This issue complicates the resource allocation process when either there is a large number of simultaneous requests or a low number of requests while a large capacity of resources is available. Therefore, we have tried to automate the DRF algorithm without changing the concept of the DRF algorithm by eliminating the time-consuming calculation process using machine learning techniques. So the A-DRF method has been proposed to significantly reduce resource allocation time while having an almost fair strategy.

A-DRF employs machine learning algorithms to monitor and classifies incoming tasks. For this purpose, each training feature vector's solution to the resource allocation problem should be calculated in advance (In this study, the DRF algorithm is used). In A-DRF, and in each iteration, it is not required to calculate dominant resources; instead, A-DRF identifies dominant resources for incoming tasks based on the similarity, existing in the historical data which is obtained in the first iteration of requests. Then, when a new request arrives, using the trained model, the amount of resources allocated to that request is determined in a concise time. In other words, as shown in Algorithm 2, A-DRF consists of two steps. In the first step, we have a set of historical data to teach the resource allocation prediction model called the H set. This set includes the resource requests, the priority of the requests, and the amount allocated to these requests according to Algorithm 1. This dataset is then given to one of the machine learning methods, and then, we will have a trained model for allocating resources to future requests. In the second step, a new request is given to the trained model, and finally, the allotted amount is returned to that request.

According to this algorithm, every scheduling decision in A-DRF takes O(1) time for any number of users. Experimental results have shown A-DRF provides better efficiency compared to the DRF method.

Algorithm 2: A-DRF

$R = < r_1, r_2, \dots, r_n >$: Capacity of Resources.

Stage 1:

$H = < X, Y >$: Historical Data $< X$: Resource Requests and their Priority, Y : Allocated Resources>

M: Prediction Model

$TM = M(H)$: Training Model

Stage 2:

$D = < d_1, d_2, \dots, d_m >$: New Demanded Resources.

$A = < a_1, a_2, \dots, a_m >$: Allocated Resources (initially 0).

TM: Trained Model (was trained in previous stage)

$A = TM(D)$

Return A

V. Exprimental Result

To evaluate the proposed mechanism, the performance of the A-DRF algorithm is examined in terms of utilisation and allocation of resources. For this purpose, first, the data set used to review and evaluate the proposed method in Section 5.1 is introduced. Then in Section 5.2, the performance of the proposed method on this data is reviewed and compared with the DRF algorithm in terms of time and error rate.

A. Data Description

The dataset used in this research is related to version 3 of Google cluster-usage traces. This trace dataset contains data on eight distinct Borg cells during the month of May 2019 and focuses on resource requests and usage. Borg supports two types of resource requests: jobs (consisting of one or more tasks) that specify the computations a user wishes to run, and alloc sets (consisting of one or more allocs, or alloc instances) that define a resource reservation in which jobs may be run.

In this study, the InstanceEvents table is used to achieve the amount of resource requests. This table contains details about instances' events (tasks and alloc instances). We use fields “priority” and “resource_request” as attributes and calculate the resource allocation to these requests based on the DRF algorithm (Algorithm 1) as the amount of estimation. Due to a large amount of data, we have used only 200,000 records of this dataset in our experiments. Fig. 1 depicts the distribution of demanded resources (CPU & Memory) for the selected data.

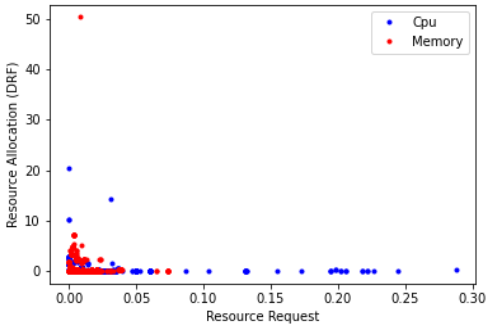

Fig. 1. Distribution of Demanded Resources

B. Performance Evaluation

Our assessments to evaluate the performance of A-DRF and, in general, the success of the final goal of the project is divided into two parts. In the first part, which is defined as an internal evaluation, we evaluate the performance of all the models used as the A-DRF method in comparison to each other. In this section, the efficiency of A-DRF is evaluated by performing experiments using the six regression methods introduced in Section 2.3.

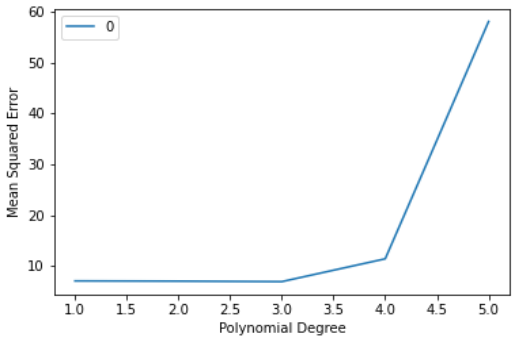

Fig. 2. Evaluate Mean Squared Error for different degree in Polynomial Regression Model

The first method that has been studied is Linear Regression method. The second is Polynomial Regression model. To determine the best degree in this model, Mean Squared Error had been examined for different degrees 1 to 5, as shown in Fig. 2. As you can see in the picture, the minimum error is related to grades 1 to 3, and we have chosen degree 2 to compare this method with other methods.

The third method that has been compared is the k nearest neighbour method or KNN. The analyst must set the neighbours' size, or it can be selected by cross-validation (which was used) to find the size that minimises the mean-squared error. Although this is an attractive method, it soon becomes unfeasible as the problem's dimension grows, especially when there are numerous independent features. Also, choosing the appropriate k has a great impact on building a proper model. Therefore, we evaluate Mean Squared Error for different K in this model. As depicted in Fig. 3, k with size 1 is the best number of neighbours to evaluate the amount of resource allocation.

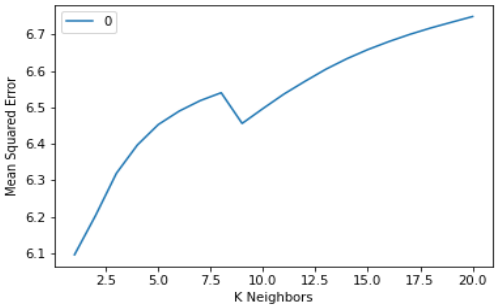

Fig. 3. Evaluate Mean Squared Error for different K in KNN Model

In these evaluations, methods Support Vector Regression (SVR), Bayesian Regression, and Multi-Layer Perceptron Regression (MLP) are also considered.

Before proceeding, it is necessary to mention that based on the type of data set used for testing, machine learning algorithms' performance varies. Therefore, to better evaluate the performance of the models, we have used three different types of test datasets.

1) Trained Test Data: In this type, a part of the data used in the model training stage is given as test data to the model,

which, as shown in Fig. 4, almost all models have performed better with this type of test data. This is a sign that the models are over-fit.

2) Cross-Validation: In this type, a part of the training data in each stage is kept and do not enter the training process and test it as test data, and finally consider the average error of all stages as the total performance error, but in fact, all test data have already been given to the model as training data in one of the steps. In this method, the over-fit is less that the previous one. However, the evaluations are still not completely reliable. It should also be noted that in both of these test methods, the KNN model had the best performance, but as you can see in Table I and Fig. 4, the error approaches zero that indicates the model was fully familiar with the test data.

3) Not Trained Test Data: The third type of test data is completely new for the model, and the model did not know about it in the training stage. In this method, the error rate provides the actual performance of the model, which, as we see, the SVR model was able to have the best performance. You can also see that the KNN model differs significantly from its performance in the two previous methods.

In the following, we calculated the training error, validation error, and test error using the Mean Squared Error. The results can be seen in Table I and Fig. 4. As shown in Table I, the lowest train and validation error is related to the KNN method, and the lowest test error is related to the SVR method. But in the average case, the Bayesian method is evaluated as the best method with the Least Mean Squared Error.

TABLE I. COMPARE MEAN SQUARED ERROR IN DIFFERENT MACHINE LEARNING METHODS

Method	**Training Error**	**Validation Error**	**Test Error**	**Average**
Linear Regression	1.0983	0.3810	49.6582	17.0458
Polynomial Regression	1.0579	0.8000	49.0501	16.9693
KNN Regression	**0.0001**	**0.0060**	37.1554	12.3871
SVR	0.5455	0.9469	**14.8967**	5.4630
Bayesian Regression	0.5112	0.5824	14.9266	**5.3400**
MLP	1.0279	1.2072	14.9266	5.7205

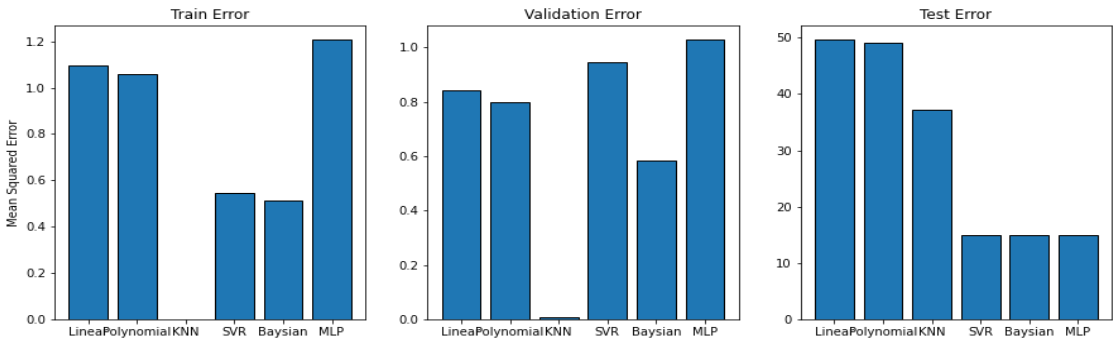

Fig. 4. Compare Mean Squared Error in Different Machine Learning Methods

The second part of the evaluation, which is actually the most important part of the project, is to evaluate the performance of the proposed idea (A-DRF) in achieving the goals. Our main goal has been to reduce the resource allocation time in the DRF method, so we need to compare the resource allocation time in the A-DRF method compared to the DRF method.

In this section, we have calculated the amount of resource allocation time in each of the models used as the A-DRF method and compared it with the DRF method (Fig. 5). This chart identifies two important issues for us. The first is the time complexity of the DRF algorithm It has been talked about before, and the second is the better performance of most models used in the A-DRF method, which indicates that the project goal has been achieved.

First, as can be seen from the Fig. 5, the DRF algorithm has a high fluctuation performance in the resource allocation process over the time. The most important reason for this is that when you have a large amount of resources but the number of requests for these resources is small and the amount of this request is also low, the DRF algorithm face problem. Because the DRF algorithm must continue the process of calculating the dominant source for each request until the full resource capacity is exhausted, and repeating this process at each step is time-consuming, which is well visible in the diagram and, as you can see, the highest time is for intervals of less than five simultaneous requests in the whole process. Then the oscillating performance of the DRF continues, and more rises and downs are observed. This is clearly visible in intervals 10, 20 and 30. To study this issue, it is necessary to examine the amount of resource requests along with the number of simultaneous requests. The study shows that when the volume of resource requests was higher in some periods, the DRF calculation process has depreciated a larger share of the total resource capacity at each stage.

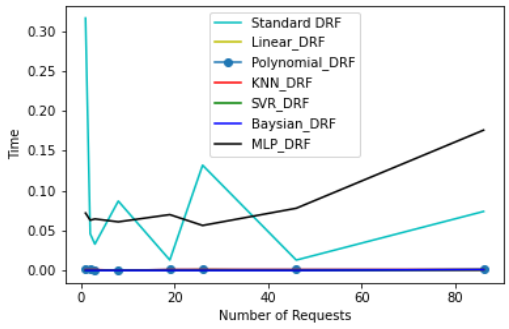

Fig. 5. Compare Runtime in Different Machine Learning Methods

As a result, the time of the calculation and resource allocation process has been reduced, although note that in any case, the performance of all A-DRF models except MLP has been better. According to Fig. 5, in the DRF algorithm, after the interval of 30 requests, the resource allocation time decreases with a steep slope and reaches its minimum in the 40-interval. From here on, there is a direct relationship in the number of requests. With the amount of resource allocation time in the DRF algorithm, we see that as the number of requests increases, the resource allocation time also increases slowly. This reminds us that DRF will not perform well on a large number of requests in terms of time as well. In contrast, A-DRF has been able to reduce resource allocation time drastically. And, in all models except MLP, it has seen a uniform and impressive performance compared to DRF.

Conclusion And Future Work

DRF algorithm is known as one of the most important algorithms for fair resource allocation. However, DRF suffers from autonomous decisions in calculating the dominant resources for incoming requests from users. Therefore, we have proposed a new A-DRF algorithm by considering this feature and using machine learning algorithms. Based on the experimental results, A-DRF shows great functionality of autonomic in calculating dominant resources and maintaining good fairness properties. But, the best characteristic of A-DRF rather than DRF is to accelerate resource allocation, especially in the huge amount of simultaneous requests.

In this project, we sought to improve the DRF algorithm using artificial intelligence techniques. but it might be interesting to take a step back for future work and look at using machine learning techniques to find more creative algorithms for resource allocation in the cloud. It may even be interesting to predict resource requests through these algorithms. For example, by considering a characteristic of each user, such as the user's location (based on IP), type of work, the relationship between features, etc., allocate resources appropriate to it.

We can use a converter to explain our new approach for scheduling to Hadoop and the Yarn scheduler in some cloud platforms such as Apache Hadoop. So, for future work, the resource allocation algorithm can be defined based on each of the machine learning algorithms as a built-in policy that can be changed by the user, such as select between KNN or SVM, etc. We also plan to experiment with more parts of the same and other available datasets as future work although we had ran some experiments and the results were similar.

References

[1] S. Nath, P. Bose, A. Mondal, and A.K. Das, "Cloud Allocation Technique: A Comparative Study," Fourth International Conference on I-SMAC (I-SMAC), Palladam, India, pp. 235-238, 2020.

[2] H. Hamzeh, S. Meacham, K. Khan, K.T. Phalp, and A. Stefanidis, "FFMRA: A Fully Fair Multi-Resource Allocation Algorithm in Cloud Environments," The 3rd IEEE Symposium on Software Engineering for Smart Systems (SSESS), pp. 19-23, 2019.

[3] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," inProc. USENIX NSDI, vol. 11, pp. 323–336, 2011.

[4] M. Ghanavatinasab, M. Bahmani, and R. Azmi, "SAF: Simulated Annealing Fair Scheduling for Hadoop Yarn Clusters," Journal of Grid Computing, 2020.

[5] Y. Yao, H. Gao, J. Wang, B. Sheng, and N. Mi, "New Scheduling Algorithms for Improving Performance and Resource Utilization in Hadoop YARN Clusters," IEEE Transactions on Cloud Computing, 2019.

[6] Y. Zhao, and H. Liu, "Cloud curriculum resource management platform based on Hadoop," Measurement and Control, 2020.

[7] J. Praveenchandar, A. Tamilarasi, "Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing," J Ambient Intell Human Comput, 2020.

[8] P. Shu-Jun, Z. Xi-Min, H. Da-Ming, L. Shu-Hui, and Z. Yuan-Xu, "Optimization and Research of Hadoop Platform Based on FIFO Scheduler," Seventh International Conference on Measuring Technology and Mechatronics Automation, China, pp. 727-730, 2015.

[9] I. Ullah, M.S. Khan, M. Amir, J. Kim, and S.M. Kim, "LSTPD: Least Slack Time-Based Preemptive Deadline Constraint Scheduler for Hadoop Clusters," IEEE Access, vol. 8, pp. 111751-111762, 2020.

[10] J.B. Wang, J. Wang, Y. Wu, J.Y. Wang, H. Zhu,M. Lin, and J. Wang, "A Machine Learning Framework for Resource Allocation Assisted by Cloud Computing," IEEE Network, vol. 32(2), pp. 144-151, 2018.

[11] D.A. Freedman, "Statistical Models: Theory and Practice," Cambridge University Press. vol. 26, 2009.

[12] J. Fan, I. Jianqing, "Local Polynomial Modelling and Its Applications: From linear regression to nonlinear regression," Monographs on Statistics and Applied Probability, 1996.

[13] N.S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," The American Statistician, vol. 46 (3), pp. 175–185, 1992.

[14] M. Awad, R. Khanna, "Support Vector Regression. Efficient Learning Machines," Apress, Berkeley, CA, pp. 67-80, 2015.

[15] CM. Bishop, and ME. Tipping, "Bayesian Regression and Classification," Computer and Systems Sciences, vol. 190, pp. 267-285, 2003.

[16] F. Murtagh, "Multilayer perceptrons for classification and regression," Neurocomputing, vol. 2, pp. 183-197, 1991.

[17] A.A. Bhattacharya, D. Culler, E. Friedman, A. Ghodsi, S. Shenker, and I. Stoica, "Hierarchical scheduling for diverse datacenter workloads," Proceedings of the 4th annual Symposium on Cloud Computing, vol. 4, 2013.

[18] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," ACM SIGCOMM Computer Communication Review, vol. 44, pp. 455–466, 2014.

[19] K.K. Pulamolu, D.V. Subramanian, "Heuristics based resource sharing with fairness in yarn: Hrsyarn," International Journal of Pure and Applied Mathematics, vol. 116(22), pp. 491–503, 2017.

[20] R. Grandl, M. Chowdhury, A. Akella, G. Ananthanarayanan, "Altruistic scheduling in multi-resource clusters," 12th Symposium on Operating Systems Design and Implementation, pp. 65–80, 2016.

[21] S. Tang, B.S. Lee, and B. He, "Fair resource allocation for data-intensive computing in the cloud," IEEE Transactions on Services Computing, vol. 11(1), pp. 20–33, 2016.

[22] Y.W. Cheng, S.C. Lo, "Improving fair scheduling performance on hadoop," International Conference on Platform Technology and Service, IEEE, pp. 1–6, 2017.