IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 11, Special Issue 2, March 2023

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

| International Conference on Recent Innovations in Engineering and Technology (ICRIET'23)|

| Sharadchandra Pawar College of Engineering, Pune, India |

|| Volume 11, Special Issue 2, March 2023 ||

HML-SVM: Detection of Faulty node with Hybrid Machine learning using SVM model

Atul V. Dusane[1], Dr. Krishnakant P. Adhiya[2]

Dept. of CSE SSBT COET, Jalgaon, Jalgoan, MH, India[1]

Dept. of CSE SSBT COET, Jalgaon, Jalgoan, MH, India[2]

ABSTRACT: The huge volume of data generated these days from a wide variety of sources has contributed to the widespread use of distributed data processing methods. In today's sophisticated huge computing systems, occupations are broken down into multiple smaller tasks that are carried out in parallel in order to improve work learning results and minimise consumption. Nevertheless, responding with straggler processes, which are processes that run slowly and contribute to an increase in the overall response time, is a common performance problem in such platforms. In this research, we suggested utilising a hybrid machine learning approach to find struggler nodes in a large distributed environment. These nodes would be identified by their inability to successfully complete tasks. In this paper, we proposed a hybrid machine learning model for detecting faulty nodes in large distributed machines with collaboration of reinforcement and supervised machine learning. The large Virtual Machine (VM's) log data has been collected from the distributed environment and proceeded with reinforcement learning algorithm for module training and supervised machine learning for module testing. According to extracted features, reinforcement learning encompasses an activation function that generates the label for the respective node, whether healthy or weak. In the testing phase, the natural world VM's log data has been collected and evaluated with supervised machine learning classifiers. Several machine learning classification algorithms have evaluated and acquired the results. The SVM provides higher accuracy over the other machine learning classifiers with our reinforcement learning model.

KEYWORDS: Faulty node detection, straggler node identification, hadoop, distributed computing, machine learning, name node, data node

I. INTRODUCTION

Consumptions of traditional computer services are made from data centres located in the cloud located in various locations around the world. These are internet-based virtual computing environments that are deployed across many locations. Previous work has mostly concentrated on locating the Stragglers, enhancing the programming level, and investigating the root cause of the thick grain. These approaches are not capable of supplying users with relevant information that may assist them in improving their programmes. Within the context of this system, we investigate the study of the big root, which is a standard technique to analysing root causes that incorporates both the framework and the characteristics of the system. The framework for processing the data splits it up into very small pieces and then processes those pieces in parallel. The programme will only be able to go to the next level after all of the actions inside the current level have been completed. The execution of the whole programme might be slowed down by these tasks, which are collectively referred to as "Stragglers," if some of the jobs within the same phase take longer than the others. Big Roots is a general strategy that encompasses process functionality as well as programme analysis of the underlying cause of stragglers.

This method covers a wider variety of causes and gives an easier route to performance optimization, which is why they advocate it. Our strategy is predicated on the idea that linking the appearance of slackers with the mundane would have the same consequence. If the usefulness of a suspending function is reduced in order to increase the richness of the normal work, then this article will be considered to be the primary cause of the disruption. The problems are solved by using this technique. Because of the work done in the past, we now have a useful reference for the optimization of future shows. In addition, statistical criteria are used to a wide variety of characteristics in order to reduce the number of false positives. As a consequence of our research into the major cause, we may choose to filter out the characteristics that suggest the blocking time is much less than the whole duration of the activity, for instance. This is due to the fact that even if just a little amount of time is spent focusing on such characteristics, the influence that they have on overall job performance is significant.

II. LITERATURE SURVEY

Bortnikov et al. [1] suggested a method that utilises machine learning in order to present a slowness predictor. The approach determines the speed at which sluggish jobs are being performed on a particular node. The atmosphere in which the experiment was carried out was consistent throughout. Those that lag behind are more likely to be exposed to a diverse ecosystem. Stragglers can be reduced in a heterogeneous environment by putting data based on the computational power of individual nodes. Furthermore, a machine-learning approach that predicts slowdowns in a heterogeneous environment might be helpful in mitigating the problem.

Accordign to Yadwadkar et al. [2] suggested a model to forecast the presence of stragglers in an environment that was considered to be homogenous. It makes judgments regarding the schedule in advance in order to avoid stragglers. The jobs are scheduled by the scheduler that is located in the master node, while the worker nodes are responsible for controlling their own scheduling.

Yadwadkar and Choi [3] conducted an investigation on Facebook's production cluster. It follows the traces and then does regression in order to forecast the task execution durations in an environment that is consistent throughout.

However, it failed to take into account the presence of a heterogeneous environment when attempting to anticipate stragglers, despite the fact that stragglers are more likely to be affected by a heterogeneous environment.

Ananthanarayanan et al. [4] presented a method to reduce the influence of stragglers in jobs estimation by carefully using speculation to reduce the number of stragglers. However, while it has provided a unified solution for ordinary jobs in an environment that is homogenous, it does not take into account an environment that is diverse. The atmosphere in which the study was carried out was consistent throughout. Stragglers are infrequent in such a setting.

Hadoop schedulers begin speculative work based on a straightforward algorithm that compares the overall average progress to the individual progress of each task. This approach performs admirably in environments that are relatively uniform and in which stragglers are uncommon [5]. The default schedulers in Hadoop operate exceptionally well in an environment that is homogeneous and where stragglers occur frequently [9,10]. When these assertions are breached, the scheduler's hypothesis prior to actually scheduling the straggler task additionally causes significant performance problems in the heterogeneous cluster [5]. These kinds of tasks often end up being a bottleneck in the overall progress of jobs, which shortens the makespan (i.e., the amount of time it takes the most recent work to finish) and brings down the entire throughput of the system. In the absence of stragglers, the performance of jobs that are constrained by a deadline is particularly bad. Because different nodes in a heterogeneous network can have varying levels of computational power, the progression of a job may not be linear due to the potential for differences in the level of processing power. When working in a diverse Hadoop environment, it is always going to be challenging to determine the right straggler in order to rearrange it.

An approach called longest estimate time to complete (LATE) was proposed by Dean et al. [5] to calculate the expected amount of time needed to execute certain jobs. It does this by increasing the speculative execution mechanism that runs on virtual machines, which in turn enhances Hadoop's performance in a heterogeneous environment. Nevertheless, a decline in performance is caused by the algorithm's default assumptions.

Using the tasks' progress and making an estimate of the total amount of time needed to complete the tasks is how this method identifies slow-moving tasks. In the year 2008, Zaharia et al. [6] offered a work that calculates the completion time of the activities in order to locate sluggish jobs. This technique was the first technique of its kind. To speed up the overall execution, the jobs that take the longest to finish are offloaded to other nodes.

A scheduling technique that dynamically estimates the progress of activities was proposed by Chen et al. [7]. This approach, which is autonomously adaptable to the changing environment as shown in Figure 6, was developed by Chen et al. In this step, the task tracker reads past data on currently executing tasks from every node in order to optimise the parameters. When a task has finished its execution, the task tracker receives input in the form of the completion time. This feedback causes the task tracker to update the historic data about the node. On the other hand, it
has just taken into account the heterogeneity of the hardware, and not any of the other elements that can have an effect, such as the various work types and job sizes. During the process of work scheduling, the disparities in stage weights are influenced by these elements.

Lin et al. [10] suggested a method that estimates fresh phase weight data based on the recently finished tasks of the work rather than on historical data that was gathered and stored in the past. This method is intended to be more efficient than the traditional method. The approach, on the other hand, does take into account the presence of several jobs in the cloud infrastructure. It is necessary to have an accurate count of the number of completed tasks associated with the occupations that are used for the process of learning. It has an impact on the estimated accuracy of the procedure that estimates things. The inaccurate estimation, on the other hand, makes the assumption that tasks are completed synchronously, which further lowers the accuracy of the projected result.

If a node is accessible, but it has a lower performance than other nodes, this is one of the most essential techniques. This kind of circumstance is referred to as a straggler. Hadoop is designed to detect lagging nodes automatically and then perform a speculative task execution on a faster node so that the computation can be completed more quickly [5]. Hadoop's speed is inextricably linked to the effectiveness of its task scheduler, which operates on the presumption that nodes linked across the cluster are identical and that tasks advance in a linear fashion. It exploits these assumptions to do a speculative re-execution of activities that seem to be lagging behind [12,13].

Pandey et al. [14] discusses the many types of heterogeneity variables that are frequently found in Hadoop. This paper investigates the numerous difficulties that can crop up throughout the process of creating schedulers for various types of heterogeneity. Nevertheless, it takes into account heterogeneity on three distinct levels: the degree of the user, the degree of the hardware, and the degree of the job. It doesn't talk about other critical aspects like data placement, data replication, or speculative implementation, which are the things that impact the heterogeneous Hadoop system the most.

Wang et al. [15] proposed a method to lessen the amount of needless speculative execution, which uses up system resources. It recognises the initial node failure at the very beginning of the process. It initiates a potentially risky work in a skillful manner in order to prevent a delay in the job's execution. However, it merely optimises speculative execution that will not totally alleviate the stragglers because of the nature of the optimization.Gupta et al. [16] suggested a method that dynamically schedules the jobs that run on the nodes depending on the capabilities of the processing units in those nodes. The amount of time needed to finish the job is cut down thanks to this method. However, because the compute capacity of the nodes are determined based on the static workload, it is unable to evenly distribute the load.

Zhang et al. [17] found that in heterogeneous clusters, load imbalance emerges as a result of the homogeneity assumptions that Hadoop makes while activities are being scheduled. It puts an end to the unreasonable task distribution that was taking place across the diverse Hadoop

cluster based on the computational capacity of the nodes. However, because it creates a heterogeneous setting on the simulator in order to imitate it, the outcomes are not accurate.An approach for Dynamic Scheduling for Speculative Execution (DSSE) was proposed by Jung et al. [18]. This technique optimises speculative execution by accelerating progress in a heterogeneous network, and it was developed by Jung et al. Despite the fact that this approach determines the right end time of activities, it is unable to position data according to the compute capability of each node in a heterogeneous environment. If it could do so, it would have been able to help reduce the number of stragglers even more.

III. PROPOSED SYSTEM DESIGN

A defective node detection using a suggested hybrid model is shown below in Figure 1. This model combines reinforcement learning and supervised learning model components. This body of work demonstrated the capacity to detect a faulty node in a decentralised system by using a variety of hybrid learning algorithms in conjunction with one another. It has been obtained from the log data of distributed virtual machines, and numerous parameters, including system memory, CPU load, activity RAC, and so on, have been recovered from the data. The removed portion served as an input for the classifier, which was being trained for a specific machine learning approach using that particular piece of data. Within the confines of the WEKA

3.7 framework, the application of classification has been made operational. We undertake an investigation of roughly six distinct machine learning approaches and assemble the anticipated results from each classifier in order to validate the work that was presented.

characteristics, the amount of CPU load, and the amount of memory load on each of the virtual machines. After that, we distribute the various tasks to be handled over all of the nodes in the cluster. When analysing load data, this information is taken into consideration so that the straggler may be located throughout the whole of the procedure. Simply using HML as a straightforward supervised classification algorithm was all that was required of us when it came to identifying and following individuals. After this step is complete, a method that is based on machine learning is used in order to train the classifiers. The material, together with its many categories, is presented right at the beginning. The HML approach makes use of a number of distinct decision trees while attempting to assess whether or not someone is lying about their identity on social media. Each of these decision trees is built by randomly picking a characteristic from a list of features. The HML method then chooses the decision tree whose outcomes are the most typical to serve as the final conclusion.

A. ALGORITHM DESIGN

Training

Input: Various activation functions[], Threshold Th, Training dataset TrainData[]

Output: For completed trained module Extracted Features Feature_set[].

Step 1: Set activation function, epoch size and input block of data d[]
Step 2 : Features.pkl ExtractFeatures(d[]) **Step 3 :**Feature_set[] optimized(Features.pkl) **Step 4 :** Return Feature_set[]

Testing

Input: Test Dataset contains various test instances TestDBLits [], Train dataset which is build by training phase TrainDBLits[] , Threshold Th.

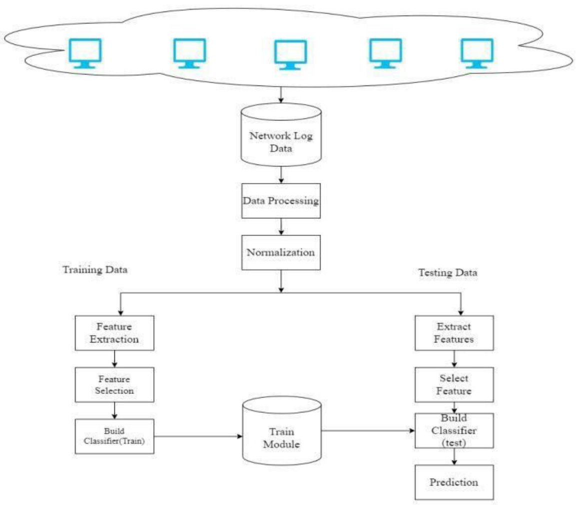
Output: HashMap <class_label, SimilarityWeight> all instances which weight violates the threshold score.

Step 1: For each read each test instances using below equation
$testFeature(m)$

= ∑ (. $featureSet[A[i] \ldots \ldots . [n]$ TestDBLits)

ୀଵ

Step 2 :extract each feature as a hot vector or input neuron from $testFeature(m)$ using below equation.



Extracted_FeatureSetx[t……n] = ∑
(t)$testFeature$

(m)

ୀଵ

Figure 1 : proposed system architecture for faulty node detection in distributed systems

The end result of the whole procedure is the identification of straggler nodes as the output, and after that, the system will automatically block any straggler nodes that it discovers. For the task at hand to be finished successfully, the step-by- step directions that are presented below must be carried out in their entirety. Before assigning any work to any of the nodes, we first collect information on the data proximity

Extracted_FeatureSetx[t] contains the feature vector of respective domain

Step 3: For each read each train instances using below equation

$trainFeature(m)$

$= \sum(.featureSet[A[i] \ldots\ldots..[n] \; \text{TrainDBList})$

Step 4 :extract each feature as a hot vector or input neuron from $testFeature(m)$ using below equation.

in order to improve the classification accuracy of the system using a variety of deep learning methods. The comparative

Extracted_FeatureSetx[t……n] = $\sum$

(t) $testFeature$

study and assessment of several categorization strategies for

(m)

the newly designed churn prediction module are shown in

Extracted_FeatureSetx[t] contains the feature vector of respective domain.

Step 5 : Now map each test feature set to all respective training feature set

$weight = calcSim$ (FeatureSetx || FeatureSety[y])

the figure that can be seen above labelled "Figure 2." The Naive Bayes model has a low degree of accuracy, but the HML-SVM classification has a very high degree of accuracy, reaching 96.5% in numerous cross-validation tests.

NB SVM SVM(HML)

Step 6 : Return Weight

IV. RESULTS AND DISCUSSION

Checking the validity of the matrices should be part of the process for evaluating the system. Python is used as the basis for the system, which also has an Intel i7 CPU running at 2.8 GHz, 16 gigabytes of RAM, and an open source environment. After the system was put into place, a comparison between a number of different current systems and the proposed system was carried out and appraised. The image that follows provides a detailed description of GUI testing together with data validation. A model for machine learning is sometimes referred to as just an error matrix in certain circles. In the realm of machine learning, and more specifically the problem of statistical classification, it refers to a table structure that enables the display of an application's output. This application is often one that engages in supervised learning. Each row of the matrix represents the predicted occurrences that should be found in a class, but each column of the matrix reflects the instances that are actually found in a class. In the context of supervised learning, an uncertainty matrix provides a straightforward method for assessing the results. It is used to characterise the result of the test that was anticipated for the model. Each row of the matrix represents a class in a class diagram, while each column in the matrix displays the instances that belong to a predicted class. In order to evaluate the discriminant function for the different dataset formats, a total of four separate experiments were conducted.

$accuracy =$ (1)

The accuracy (Eq. 1) is the percentage of accurate predictions out of an overall amount of projections.The equation is used to measure it:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

ACCURACY PRECISION RECALL F1 SCORE

Figure 2: detection and classification accuracy using numerous machine learning and proposed hybrid machine learning algorithm

The above Figure 2 describes a straggler node detection and prediction accuracy using proposed hybrid machine learning algorithms. The three different machine learning (ML) techniques has used for such as NB, SVM and hybrid SVM.

V. CONCLUSION

In this research, a straggler node identification and prediction method using a hybrid machine learning model is described. We have shown that the results obtained using our strategy are superior to those obtained using some of the more well- known alternatives. Our formulation for active learning is more accurate and generalizable than other strategies for active learning that need just a small portion of the learning algorithm, yet it only requires a third of it. As a consequence of this, the problem of socioeconomic inequality is addressed gradually. Our method is superior to others in terms of straggler detection due to the fact that it can more correctly portray the distribution of straggler nodes. The suggested models achieve an average accuracy of 96.50% for the many different log datasets when using hybrid SVM. Because of this performance, it is feasible to identify stragglers in operation with an extremely high degree of certainty. The framework that was presented may be used to a broad variety of workloads; it is not restricted to being utilised just with computer systems for big data (e.g., across several data centres). The node and task utilisation resources that are specified in this framework might provide additional information that can be learned from them. To prepare for future work, a variety of deep learning models will need to
be developed for the identification and prediction of straggler nodes in large dispersed systems

precision =
recall =*TP*

TP + *FP TP*
(3)

(4)
TP + *FN*
The suggested implementation has been carried out in an open-source environment for Windows, and the Python Platform has been utilised since open source is readily available. The data extraction from the file system application was accomplished with the help of the file system dataset. We generate a wide variety of data chunks

REFERENCES

1. [1] Edward Bortnikov,Ari Frank, Eshcar Hillel, Sriram Rao, Predicting execution bottlenecks in map-reduce clusters,Proceedings of the 4th
2. USENIX conference on Hot Topics in Cloud Ccomputing June 2012. [2]Yadwadkar, N.J., Hariharan, B., Gonzalez, J.E., Katz, R., 2016. Multi- task learning forstraggler avoiding predictive job scheduling. J. Mach. Learn. Res. 17, 3692–3728. Yang, S.J., Chen, Y.R., 2015. Design adaptive task allocation scheduler to improvemapreduce performance in heterogeneous clouds. J. Network Computer Appl.57, 61–70
3. Yadwadkar, N.J., Choi, W., 2012. Proactive straggler avoidance using machinelearning. University of Berkeley, White paper.
4. Ananthanarayanan,Yadwadkar, N.J. G., Katz, R.,Wrangler: Predictable and faster jobs using fewer resources. ACM Symposium on Cloud Computing. ACM,2014, pp. 1–14.
5. Dean, J., Ghemawat, S., 2008. Mapreduce: simplified data processing on largeclusters. Commun. ACM 51, 107–113.
6. Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R.H., Stoica, I., 2008. Improvingmapreduce performance in heterogeneous environments. In: Osdi, p. 7.
7. Chen, Q., Zhang, D., Guo, M., Deng, Q., Guo, S., 2010. Samr: A self- adaptive mapreduce scheduling algorithm in heterogeneous environment, in: 2010 10th IEEE International Conference on Computer and Information Technology, IEEE. pp. 2736–2743.

8. Shvachko, K., Kuang, H., Radia, S., Chansler, R., et al., 2010. The hadoop distributedfile system. In: MSST, pp. 1–10.
9. Zhao, X., Kang, K., Sun, Y., Song, Y., Xu, M., Pan, T., 2013. Insight and reduction ofmapreduce stragglers in heterogeneous environment. In: 2013 IEEEInternational Conference on Cluster Computing (CLUSTER). IEEE, pp. 1–8.
10. Lin, C., Guo, W., Lin, C., 2013. Self-learning mapreduce scheduler in multi-jobenvironment. In: 2013 International Conference on Cloud Computing and BigData. IEEE, pp. 610–612.
11. Sun, X., He, C., Lu, Y., 2012. Esamr: An enhanced self-adaptive mapreducescheduling algorithm. In: 2012 IEEE 18th International Conference on Paralleland Distributed Systems. IEEE, pp. 148–155. [13]Chen, Q., Guo, M., Deng, Q., Zheng, L., Guo, S., Shen, Y., 2013a. Hat: history-basedauto-tuning mapreduce in heterogeneous environments. J. Supercomputing 64,1038–1054.
12. Pandey, V., Saini, P., 2018. How heterogeneity affects the design of hadoopmapreduce schedulers: A state-of-the-art survey and challenges. Big Data 6,72–95.
13. Wang, J., Wang, T., Yang, Z., Mi, N., Sheng, B., 2016. esplash: Efficient speculation inlarge scale heterogeneous computing systems. In: 2016 IEEE 35thInternational Performance Computing and Communications Conference(IPCCC). IEEE, pp. 1–8.
14. Gupta, S., Fritz, C., Price, B., Hoover, R., Dekleer, J., Witteveen, C., 2013. Throughputscheduler: Learning to schedule on heterogeneous hadoop clusters, in: Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13), pp. 159–165.
15. Zhang, X., Wu, Y., Zhao, C., 2016. Mrheter: improving mapreduce performance inheterogeneous environments. Cluster Computing 19, 1691– 1701.
16. Jung, H., Nakazato, H., 2014. Dynamic scheduling for speculative execution toimprove mapreduce performance in heterogeneous environment. In: 2014 IEEE34th International Conference on Distributed Computing Systems Workshops(ICDCSW). IEEE, pp. 119–124.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.379

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details