IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 27, No. 4, August, 2022, pp. 621-627
Journal homepage: http://iieta.org/journals/isi

A Hybrid Supervised Learning Approach for Detection and Mitigation of Job Failure with Virtual Machines in Distributed Environments

Reshma S. Gaykar[1*], V. Khanaa[1], Shashank D. Joshi[2]

[1] Bharath Institute of Higher Education and Research, Chennai 600073, India
[2] College of Engineering, Bharati Vidyapeeth (Deemed to be University), Pune 411043, India

Corresponding Author Email: reshma.gaykar@gmail.com

https://doi.org/10.18280/isi.270412

Received: 31 May 2022
Accepted: 30 July 2022

Keywords:
hybrid machine learning, virtual machine, job failure, detection and mitigation, load balancing

ABSTRACT

Distributed data processing techniques are very popular nowadays due to high data generation from various resources. To increase work learning outcomes and to reduce consumption, modern massive computational systems divide jobs into several smaller tasks that perform in parallel. Nevertheless, responding with straggler processes, which are sluggish running processes that rise the total response time, is a typical performance issue in such platforms. In this paper, we proposed the detection of struggler nodes in a large distributed environment using a hybrid machine learning technique. Initially, the data has been collected from numerous virtual machine network logs. The entire data set has various fields such as Virtual Machine ID, CPU load, memory load, bandwidth utilization, etc. Memory utilization an input to the proposed system is collected from the garbage collection log files where the memory consumption on each VM and its timestamp is recorded. This is the most efficient way to get the memory consumption in web/desktop applications. Similarly, the CPU, I/O and bandwidth utilization is grabbed from the process monitoring functionality and SAR (System Activity Report) utility from the respective VM boxes. This data set is useful to identify weather-specific virtual machine is heated up or not. In this approach, we proposed three conventional machine learning algorithms and a hybrid machine learning algorithm for the identification of node status. Main purpose of the proposed system is to identify the slow performing node in an efficient way to prevent the other nodes from failures. This can provide effective load balancing and low response time for task execution from available VM's in distributed cloud environments. To create its training program, several extractions of features approaches were used. TF-IDF, correlational co-occurrence, and density-based features have been mined from the whole data set. With extensive experimental analysis, we evaluate our system with our proposed classification algorithm. As a result, the system produces higher classification accuracy of 94.5% over the traditional machine learning classifiers. If the proposed system is tested against the data set fields, memory load and CPU load on the homogenous machine configurations, we see more efficiency while detecting the underperforming node than the heterogenous machine configurations.

1. INTRODUCTION

Task implementation efficiency is critical for either system management or service customers when programs are built on parallel computing platforms like Hadoop. The former might result in lower reliability and a possible service level agreement gap, while the later would result in poor customer satisfaction due to the unexpected response. Node failures, such as machine crashes, are prevalent in large-scale production systems, and ways to cope with node failures have been extensively researched. However, in addition to node failures, node efficiency deterioration requires investigation since it may lead to major issues like the straggler problem. The group members or VM's who are running more slowly or making less progress than the others are known as stragglers. In distributed computing stragglers can be generate many times due to overhead of massive data processing. In many cases it enhances the overall execution time as well as generate data leakage issues. Core issue, we tried to address in the proposed system is to find out the efficient way to identify the straggler node. The Straggler issue illustrates the phenomena in which a tiny fraction of outlier activities performs exceptionally slowly compared to the other siblings' tasks in a concurrent job. The straggler issue is addressed by Speculator, a Hadoop built-in component. When a straggler is detected, a redundant replica job is initiated and executed on additional node. The outcome of the fastest task will be used, while the further task will be removed. Hardware heterogeneity, input output disharmony data skew, resource constraints, background traffic of network and Operating system or execution associated reasons are all examples of behaviors that might induce straggler formation in cluster systems. Node efficiency diversity is one of the most significant factors. The ability of a node to execute concurrent applications is referred to as node performance in this work. Machine learning approaches provide a strong light on node performance analysis, which is crucial for straggler reduction. The scheduling algorithm can choose appropriate nodes to

introduce tasks of latency-sensitive by categorizing nodes into separate classes and forecasting the connected performance categorization with high accurateness. This avoids assigning pure speculation tasks to nodes that are most probable to be in their poor operating province in the upcoming years. The following is a list of our major contribution:

- We proposed the hybrid machine learning framework called (HML) for classification of log data to achieve the high accuracy for straggler prediction.
- The hybrid feature extraction techniques for strong module building which achieves good accuracy.
- Validation of the proposed system with various machine learning and hybrid machine learning module.

In the proposed system for classification of data, we carried out HML as a supervised classification algorithm. Then supervised machine learning is applied in order to train the classifier. The class labelled data is present at the beginning. HML algorithm for Deep learning is applied to determine identity deception from the feature set and the majority output class of all the decision trees is taken as output of the HML. Proposed HML algorithm produces 94.5% detection and classification accuracy which is more as compare to the traditional algorithms which are mentioned in the result and discussion section.

The paper is described as, SECTION 2 demonstrates the literature review of a proposed model where the various existing system has been analysed and all the current systems are identified. In SECTION 3, research methodology of the presented system and implementation details are described, while in SECTION 4 the proposed algorithm is described in detail. SECTION 5 focuses on results and discussion of the experimental set-up, and various experimental results, and finally, the conclusion and recommendations are discussed in SECTION 6 of the proposed model.

2. LITERATURE REVIEW

The straggler issue, as well as the difficulty of overcoming it due to node performance heterogeneity, will be examined in this section. All observations are based on the distributed environment's data analytics findings.

Mou [1] uses a whole binary tree of code snippets to train a specialised convolutional neural network. An AST neural network breaks huge ASTs into a group of tiny statement trees to alleviate the gradient constraint. The tree-based RNN is then applied. Overall, tree-based techniques enhance source code representation by including syntactic information. Additionally, arbitrary code fragments may be processed to AST, expanding the range of applications for tree-based techniques. Though, tree-based approaches greatly increase the complexity of code fragments. Furthermore, they are prone to issues like long-term dependency and gradient fading. To improve productivity and lessen dependency on human employees, pattern-based techniques are offered. On the one side, flaws are detected using principles derived from established software. Infringement of these rules is viewed as potential security flaws. Fabian [2] established anomalous patterns as a result of variable initializations using information or control reliance upon security-sensitive functions. A range of crucial expressions or API usages are captured, including a typical API usage behaviour, imports, as well as function calls. Bian et al. [3] created odd patterns using syntactic information. They create AST from sliced source codes. The vector representation is then obtained using a hash method. Anomaly detection provides the benefit of uncovering previously undiscovered vulnerabilities while also reducing reliance on labelled vulnerability datasets. Anomaly detection, on the other hand, is project specific or task specific and has a low recall.

The unsupervised and supervised approaches are the two main concepts describes in details according to Nasteski [4]. In supervised Machine learning, sometimes known as "learning with a teacher," the train data set (condition) and testing dataset (required answer) are formed. Obtaining such training sets from industry and labs is difficult. A few numbers of damaging tests may be done for training reasons owing to the restricted number of malfunctioning equipment running in the industry due to regular maintenance and in labs. Furthermore, in both circumstances, data gathering with numerous failures (composite errors) in the similar machine is difficult. The mathematical models of electrical machinery can train AI algorithms by rising computational capacity of computers and cloud computing. Unsupervised machine learning, commonly known as "learning without a teacher," is a method of learning in which patterns are identified from unstructured data [5]. The goal is to cluster objects and/or minimise the quantity of data available. Semi-supervised algorithms are sometimes used in industrial systems to get a more exact result. Some research is using both training and test data, while others just contain training data.

Unlike traditional techniques, reinforcement distributed deep learning focuses on recognising and generalising patterns in reinforcement configuration environment [6] by Mousavi. The goal is to reduce mistakes and improve precision; the machine learns to assess data before each step. Furthermore, the machine seeks to maximise the reward (benefit) from the learning, which is predetermined, such as minimal resource consumption, obtaining the required value, analysing time, and so on.

According to Tranchevent et al. [7] Unsupervised machine learning refers to algorithms that can learn to do a job on their own, without the assistance of a teacher. When a result is known and a link between system responses is needed, unsupervised learning is typically compared with supervised learning. If there are comparable patterns, the algorithm attempts to detect similarities between items and divides them into groups in unsupervised learning. Clusters are the names for these groupings. The following supervised methods are the most extensively used namely k means, cluster analysis as well as fuzzy c-means.

The particle swarm optimization [8] is one of the techniques that may be utilised in data mining and cluster analysis. SI is an optimization approach that explains the collective behaviour of a decentralised and self-organized system. The SI system is made up of agents (or boids) that interrelate with one another and with their surroundings. Swarm intelligence is a multiple agent system capable of self-organization and exhibiting rational behaviour. This algorithm can adapt to changes and reach some optima quickly. In complicated problems, solutions might be locked in local minimums because they are dependent sequences of random options.

At the same time, the genetic algorithm [9] is the most often employed reinforcement algorithm in condition monitoring. A GA (genetic algorithm) is an optimization methodology that provides natural selection concepts to simulate random selection in the environment. The focus on employing the "crossing" operator, which exploits the instrumental function

of crossing in wildlife, is a distinguishing aspect of the GA. In the case of GA, the issue is formalised to the point that its solution may be recorded as a vector of genes, each of which has a value. The genotype is expected to have a constant length in traditional GA implementations. There are, however, GA variations. Liang and Znati [10] suggested and CICIDS2017 assessed a deep-learning-based technique based on long short term memory (LSTM) and a recurrent ANN. They alternated training and testing the system on different parts of the dataset that included types of malicious. As a result, both the test and training data come from the very same network contexts. In the first experiment, the recall of their proposed methodology was about 300 percent larger than the percentage of the DT, Support vector machine, and artificial neural network has the accuracy of roughly 6% lower. In the second trial, the training and testing sets were swapped, and the recall improved by roughly 59 percent while the precision declined by about 25 percent.

Yang et al. [11] developed a dimensionality reduction approach based on the auto encoder, a form of unsupervised ANN. They trained the technique on a dataset that was a distinct subset of the network log than what we use, and then used network simulation to test it on a fake dataset. Deepa et al. [12] combined Support vector machine, a supervised methodology, with self-organized mappings which is an unsupervised technique also called as SOM. An artificial neural network called the SOM is utilized to reduce data dimensionality. The connection will be stopped if the support vector machine algorithm determines that it is harmful; else, it will be forwarded to the SOM for judgement. Li et al. [13] suggested a real-time technique for detecting high-rate DDoS assaults that employs entropy analysis and ANN. Because of the massive botnet, the entropy of source IP is likely to rise during a DDoS assault. However, this may not be the case in reality due to variables such as the number of objectives or policies in play. To identify DDoS assaults, Idhammad et al. [14] suggested a hybrid learning technique. An entropy calculation phase, a co-clustering step, and a classification technique are all included in their approach. First, the mean entropy of 4 characteristics for every dataset's record is determined using a temporal frame.

As a result, it is critical to recognise and neutralise them as soon as possible. Furthermore, centres with huge computational infrastructure may encounter delays, resulting in poor task execution. Large data centres also have a high volume of service generation, making them susceptible to stragglers. Background network traffic, resource congestion, hardware heterogeneity as well as OS related issues are all root reasons for stragglers [15]. Stragglers have been the subject of much research. The effect of stragglers has grown considerably as the quantity of computer infrastructure and tasks done has expanded over time. Stragglers are known to significantly delay task execution, compromising the "Consumer Service Level Agreement" and QoS performance standards. Bortnikov offers two approaches for dealing with stragglers that is tolerance and prevention [16]. Stragglers, on the other hand, are difficult to avoid because pursuing them is impracticable. As a result, most stakeholders choose the straggler tolerance strategy. In the tolerance of straggler, a percentage score consists of values in the range (0 – 1) that reflect beginning and ending and it is used for tracking execution progress of the task. Currently, straggler identification methods may be classified as either offline or online analytics [17] by Abasi. Though, it's worth mentioning that detecting online might happen too late in a task's execution cycle. Consequently, even after the installation of speculative copies, it will slowly run. Offline methods, on the other side, are often used to prevent stragglers. This method is less prevalent since it is considered less viable. Combining online and offline tactics, on the other hand, may provide greater outcomes. They may considerably boost the efficacy of "straggler detection" when used simultaneously. Bangare et al. [18-22] have contributed Machine learning projects for medical images. Shelke et al. [23] and Gupta et al. [24] also worked on the similar domain of research. Pande et al. [25-27] worked on the spline curve etc. Basic concept of staggler and ML are referred from the papers [28-31].

3. PROPOSED SYSTEM DESIGN

As machine learning (ML) algorithms are more common and training models becomes more difficult, a simple, versatile, and straggler-resistant distributed ML framework becomes more important. With a parallel processing communication pattern, proposed framework aims to reduce the effects of stragglers in large-scale training jobs. Stragglers are workers who are behind the rest of the workers in synchronous distributed computing. This section outlines our suggested methodology for reducing straggler effects in distributed machine learning. Straggler nodes cause synchronous processing delays. This occurs when the results of all workers must be combined before moving on to the next stage of the calculation.

Due to the repetitive nature of Stochastic Gradient Descent-like algorithms, the issue of stragglers is especially prominent in distributed machine learning. As a result, even a few stragglers in each loop may drastically reduce performance as measured by throughput and time to accuracy. In the world of distributed computing, stragglers have long been a problem. They squander valuable calculation cycles, which must be recovered by reproducing machines or redoing the process. The difficulty is worsened in machine learning since the underlying training methods are iterative and synchronous. State-of-the-art straggler mitigation solutions in distributed computing depend on replication, which may be an expensive approach due to the extra resources required. Straggler mitigation was previously presented as a primary-backup design.

Figure 1 describes a Hybrid Machine Learning (HML) for classification of detection and prediction of straggler in large distributed environment. Data has been validated according to the rules and norms defined during pre-processing. Each property has a lower and upper limit for specific values, and when one of these values exceeds or violates the bounds, the system instantly destroys the instance. Collection of data gathering, cleaning, filtering and normalizing are all part of pre-processing. Cleaning and repairing false or incorrect information from documents, records, and datasets entails locating and changing (or eliminating) lost mistaken, wrong or nonsensical info, as well as replacing, updating, or eliminating dirty or sensitive info. Proposed system uses scripting software or transaction processing to sanitize data interactively. We used consistent sampling procedures to balance data and filtered the standardized dataset to exclude the incorrectly classified occurrences. In the feature extraction, the normal and numerical values from text data are extracted.

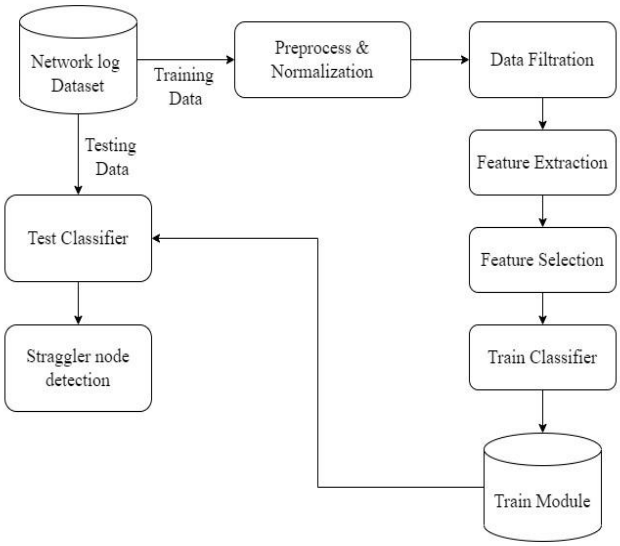

Figure 1. Proposed system architecture using HML for straggler node detection and classification

The various feature selection and extraction techniques have been applied to the entire data set. The TF-IDF, correlation coefficients, bi-tagged, N-Gram and density-based features have been extracted from the entire data set. Once the feature extraction has been done, we generate a unique feature vector that contains heterogeneous features from the extracted features. This feature vector gives the assurance that it does not have redundant data and provides minimum redundancy and maximum relevance (mRmR). Finally, the system detects each record, either attack or normal using a supervised classification technique. Moreover, the system also demonstrates an unknown attack classification of normal real time dataset.

For the detection and classification, we carried out HML as a supervised classification algorithm. Then supervised machine learning is applied in order to train the classifier. Here class labelled data is present at the beginning. HML algorithm for Deep learning is applied to determine identity deception on social networks where multiple decision trees are created using randomly selected features from the feature set and the majority output class of all the decision trees is taken as output of the HML. The results have been evaluated according to confusion matrix and generated the F1 Score, PR-AUC (area under the (precision-recall) curve).

4. ALGORITHM DESIGNING

Training Procedure (HML): According to machine learning strategy we divide our dataset into two phases as train dataset and test dataset. The 10-fold cross validation has utilized for data splitting and then 70-30% data feed for raining and testing respectively. The following algorithm describes a training function which used for module training during the execution.
Input: The training dataset *train_data* [], No. of epoch size n, name of classifier *cls_name*.

Output: Building a training model *train_matrix* []

(1) Read each attribute *cls_name* form training dataset as *t* [] into *train_data* []
(2) $feature\_set$ []← $extract\_features$ (t[])
(3) new_matrix [] ← $optimization(feature\_set)$

return $train\_matrix$ [] ← new_matrix [] as training dataset.

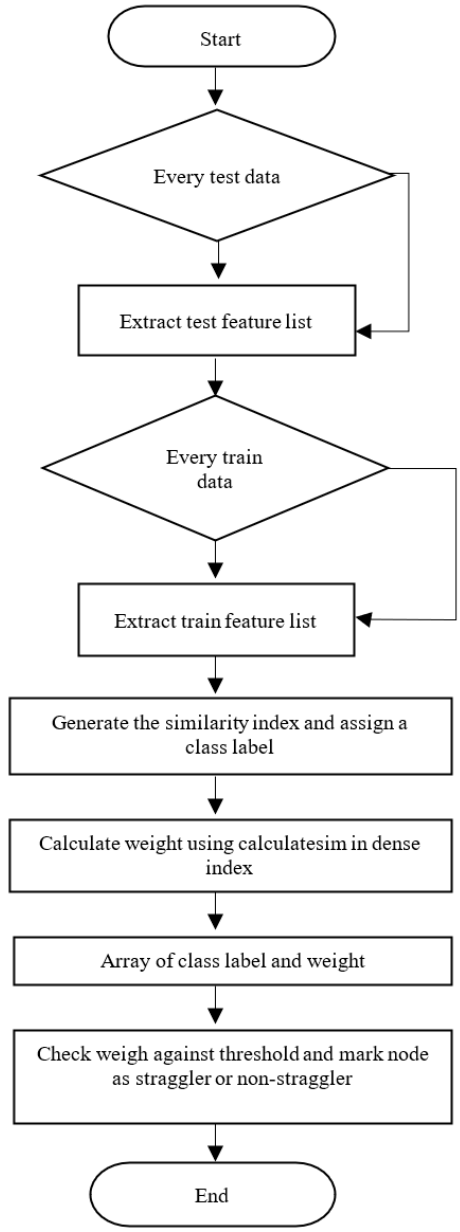

Figure 2. Flowchart for straggler node detection and classification

System testing algorithm (HML): Once training has been successfully done, it generates the background rules for entire system. In testing phase of step 1 and 2, it extracts the features from testing dataset. The step 3 and 4 demonstrates the extraction of rules from training repository. Finally steps 5 describe evaluation and generate the similarity index and assign the class label to all instances as straggler or non-straggler. Figure 2 show the flowchart for the proposed algorithm.

Input: Train dataset *trainingDBList* [], Test dataset *testingDB-List* [] and Threshold Th.

Output: Whose weight is heavier than Th is determined by *rslt-set <cls_name, sim_wt>*

(1) As described in the following equation, for every testing data, it operates in a convolutional layer with invader training and test data.

$$testing\_feature(k) = \sum_{k=1}^{n} (.\ featset[A[1] \ldots\ldots A[n] \leftarrow testingDBList\ )$$

(2) By using below code, generate a feature vector from *testing_feature (k)*.

$$extract\_featset\_x\ [1 \ldots\ldots n] = \sum_{x=1}^{n} (t) \leftarrow testing\_feature(k)$$

The output of every pooling layer collected by every convolutional layer and forwarded on to the total convolutional layer is *extract_featset_x[t]*. Each layer keeps the features extracted of every instance in the test dataset.

(3) For every train instance, use the following function:

$$training\_feature(l) = \sum_{l=1}^{n} (.\ featset[A[1] \ldots\ldots A[n] \leftarrow trainingDBList\ )$$

(4) Produce new feature vector from *training_feature(l)* by using following function

$$extract\_featset\_y\ [1 \ldots\ldots n] = \sum_{y=1}^{n} (t) \leftarrow traning\_feature(l)$$

The output of every pooling layer collected from every convolutional layer and forwarded on to the total convolutional layer is *extract_featset_y[t]*. Each layer stores each derived feature from each occurrence in the train dataset.

(5) Compute every testing record with whole train dataset, in the dense layer

$$weight\_wt = calculatesim\ (featsetx || \sum_{i=1}^{n} featsety[i])$$

(6) Return *rslt-set* (*cls_name, weight_wt*)

Both the training and testing algorithms are used for training and testing respectively. The extracted heterogeneous features ensure robust module training, supervised classifiers achieve higher accuracy in detecting such faulty nodes. In module testing when any input is assigned to a classifier, it generates a specific weight with all training rules. Finally, optimization techniques are used for the selection of the best prediction.

5. RESULTS AND DISCUSSIONS

For the system's evaluation process, we have computed the matrices for correctness. The system is built on python framework with INTEL 2.8 GHz i7 processor and 16 GB RAM with open-source environment. After the implementation of system, comparison between numerous existing system and proposed system has been evaluated. The following figure describes testing result in details with data validation. A machine learning model is also known as just an error matrix. It is a table structure that lets the visualization of an application's output, classically a supervised learning one, in the machine learning field, and particularly the issue of statistical classification. Every row of the matrix indicates the expected occurrences in a class, while every column represents the instances in an actual class. Supervised learning, an uncertainty matrix is a simple tool for evaluating outcomes. It is used to characterise the projected model's test outcome. Every row in the matrix shows the classes in a class diagram, whereas every column indicates the examples in a predicted class. Four independent experiments were performed to test the discriminant function for various dataset formats.

$$accuracy = \frac{(TP + TN)}{(TPtable + TN) + (FP + FN)} \quad (1)$$

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

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

$$precision = \frac{TP}{TP + FP} \quad (3)$$

$$recall = \frac{TP}{TP + FN} \quad (4)$$

Table 1. Faulty nodes detection analysis

Training data	Test data	Classifier	Accuracy	Precision	Recall	F-Measure
10%	10%	HML	94.5%	0.96	0.97	0.97
10%	10%	NB	81.0%	0.86	0.92	0.89
10%	10%	RF	90%	0.92	0.96	0.94
10%	10%	SVM	93%	0.95	1.0	0.97

The proposed implementation has done in Windows opensource environment, python Platform has used due to availability of open source. The file system dataset has used to extract the data from file system application. We create various data chunks to perform the system classification accuracy with different deep learning algorithms. Table 1 illustrates the comparative analysis of evaluation of different classification techniques for developed churn prediction module. The Naïve Bayes provides minimal accuracy thus HML categorization gives maximum accuracy with 94.5 percent on several cross validation. The results also demonstrate that the HML classifies logging data and achieve high-precision discrete prediction of maximum value of 0.96 as compare to the other algorithm.

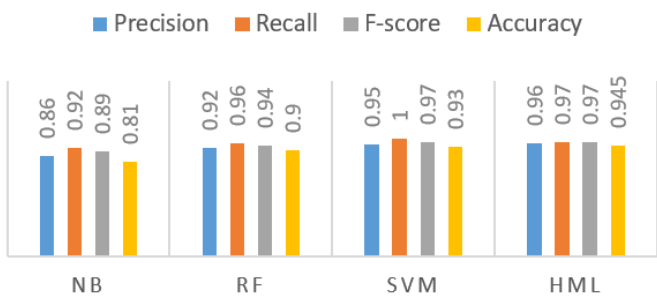

Figure 3. Detection and classification accuracy using numerous machine learning and proposed hybrid machine learning algorithm.

The above Figure 3 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, RF and SVM. The HML produces highest accuracy The NB gives 81% accuracy; RF produces 90% while SVM gives 93% and proposed HML produces 94.5% detection and classification accuracy.

6. CONCLUSIONS

In order to ensure effective task execution, it is necessary to analyse and anticipate node performance. It makes task deployment easier by avoiding and allocating them to nodes that are expected to be in low-performance or execution. It gives information on the optimal node choices that are good for starting replications for straggler mitigation strategies such as fault injection. The following is a list of the paper's key contributions. To begin, we looked at the efficiency heterogeneity of nodes based on dispersed data for parallel task execution. We characterise and assess the nodes' effectiveness using straggler numbers because they directly indicate the effect of heterogeneity which has on efficient task replies, in contrast to work that focuses only on physical capacity disparities or utilisation variances. Second, we looked at a number of factors that might be used to define node performance and created an automatic labelling system that could give unbiased and precise labels for various performance segments. We generated the features to represent node execution ability using standardised execution speed of the task and task quantity per node numbers, statistical properties, and timing aspects. Finally, we introduced ML, a framework for analysing node performance that categories machine nodes. The results of the predictions demonstrate that different machine learning algorithms can accurately predict node efficiency subcategories with an accuracy rate of 94.5% percent as shown in Table 1. This may help the scheduler even more by blacklisting endpoints that are anticipated to be in low-performance conditions in the next scheduled window. In addition to the aforementioned contributions, future research will include integration of the suggested deep learning techniques into cluster work schedule decision-making elements such as the task scheduler in distributed systems, as well as improving its ability to handle limitation situations when no tasks are forwarded.

REFERENCES

[1] Mou, L., Li, G., Zhang, L., Wang, T., Jin, Z. (2016). Convolutional neural networks over tree structures for programming language processing. arXiv:1409.5718. https://doi.org/10.48550/arXiv.1409.5718

[2] Yamaguchi, F., Maier, A., Gascon, H., Rieck, K. (2015). Automatic inference of search patterns for taint-style vulnerabilities. In 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, pp. 797-812. https://doi.org/10.1109/SP.2015.54

[3] Bian, P., Liang, B., Zhang, Y., Yang, C., Shi, W., Cai, Y. (2018). Detecting bugs by discovering expectations and their violations. IEEE Transactions on Software Engineering, 45(10): 984-1001. https://doi.org/10.1109/TSE.2018.2816639

[4] Nasteski, V. (2017). An overview of the supervised machine learning methods. Horizons, 4: 51-62. https://doi.org/10.20544/HORIZONS.B.04.1.17.P05

[5] Cord, M., Cunningham, P. (2008). Machine learning techniques for multimedia: Case studies on organization and retrieval. Springer Science & Business Media. https://doi.org/10.1117/1.3207770

[6] Samsami, M.R., Alimadad, H. (2020). Distributed deep reinforcement learning: An overview. arXiv preprint arXiv:2011.11012. https://arxiv.org/abs/2011.11012.

[7] Yu, S., Tranchevent, L., Liu, X., Glanzel, W., Suykens, J. A., De Moor, B., Moreau, Y. (2011). Optimized data fusion for kernel k-means clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(5): 1031-1039. https://doi.org/10.1109/TPAMI.2011.255

[8] Abdmouleh, Z., Gastli, A., Ben-Brahim, L., Haouari, M., Al-Emadi, N.A. (2017). Review of optimization techniques applied for the integration of distributed generation from renewable energy sources. Renewable Energy, 113: 266-280. https://doi.org/10.1016/j.renene.2017.05.087

[9] Beg, A.H., Islam, M.Z. (2016). Advantages and limitations of genetic algorithms for clustering records. In 2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA), Hefei, China, pp. 2478-2483. https://doi.org/10.1109/ICIEA.2016.7604009

[10] Liang, X., Znati, T. (2019). A long short-term memory enabled framework for DDoS detection. In 2019 IEEE global communications conference (GLOBECOM), Waikoloa, HI, USA, pp. 1-6. https://doi.org/10.1109/GLOBECOM38437.2019.9013450

[11] Yang, K., Zhang, J., Xu, Y., Chao, J. (2020). DDoS attacks detection with autoencoder. In NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, pp. 1-9. https://doi.org/10.1109/NOMS47738.2020.9110372

[12] Deepa, V., Sudar, K.M., Deepalakshmi, P. (2018). Detection of DDoS attack on SDN control plane using hybrid machine learning techniques. In 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, pp. 299-303. https://doi.org/10. 1109/ICSSIT.2018.8748836

[13] Li, J., Liu, M., Xue, Z., Fan, X., He, X. (2020). RTVD: A real-time volumetric detection scheme for DDoS in the Internet of Things. IEEE Access, 8: 36191-36201. https://doi.org/10.1109/ACCESS.2020.2974293

[14] Idhammad, M., Afdel, K., Belouch, M. (2018). Semi-supervised machine learning approach for DDoS detection. Applied Intelligence, 48(10): 3193-3208. https://doi.org/10.1007/s10489-018-1141-2

[15] Aktas, M.F., Peng, P., Soljanin, E. (2018). Straggler mitigation by delayed relaunch of tasks. ACM SIGMETRICS Performance Evaluation Review, 45(3): 224-231. http://dx.doi.org/10.1145/3199524.3199564

[16] Bortnikov, E., Frank, A., Hillel, E., Rao, S. (2012). Predicting execution bottlenecks in map-reduce clusters. In 4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 12).

[17] Abasi, A.K., Khader, A.T., Al-Betar, M.A., Naim, S., Makhadmeh, S.N., Alyasseri, Z.A.A. (2020). Link-based multi-verse optimizer for text documents clustering. Applied Soft Computing, 87: 106002. https://doi.org/10.1016/j.asoc.2019.106002

[18] Bangare, S.L. (2022). Classification of optimal brain

tissue using dynamic region growing and fuzzy min-max neural network in brain magnetic resonance images. Neuroscience Informatics, 2(3): 100019. https://doi.org/10.1016/j.neuri.2021.100019

[19] Bangare, S.L., Pradeepini, G., Patil, S.T. (2006). Implementation for brain tumor detection and three dimensional visualization model development for reconstruction. ARPN Journal of Engineering and Applied Sciences (ARPN JEAS), 13(2): 467-473.

[20] Bangare, S.L., Dubal, A., Bangare, P.S., Patil, S.T. (2015). Reviewing Otsu's method for image thresholding. International Journal of Applied Engineering Research, 10(9): 21777-21783. http://dx.doi.org/10.37622/IJAER/10.9.2015.21777-21783

[21] Bangare, S.L., Pradeepini, G., Patil, S.T. (2018). Regenerative pixel mode and tumour locus algorithm development for brain tumour analysis: A new computational technique for precise medical imaging. International Journal of Biomedical Engineering and Technology, 27(1-2): 76-85. https://doi.org/10.1504/IJBET.2018.093087

[22] Bangare, S.L., Pradeepini, G., Patil, S.T. (2017). Neuroendoscopy adapter module development for better brain tumor image visualization. International Journal of Electrical & Computer Engineering (2088-8708), 7(6): 3643-3654. http://dx.doi.org/10.11591/ijece.v7.i6.pp%25p

[23] Shelke, N., Chaudhury, S., Chakrabarti, S., Bangare, S. L., Yogapriya, G., Pandey, P. (2022). An efficient way of text-based emotion analysis from social media using LRA-DNN. Neuroscience Informatics, 100048. https://doi.org/10.1016/j.neuri.2022.100048

[24] Gupta, S., Kumar, S., Bangare, S.L., Nuhmani, S., Alguno, A.C., Samori, I.A. (2022). Homogeneous decision community extraction based on end-user mental behavior on social media. Computational Intelligence and Neuroscience, 2022: 3490860. https://doi.org/10.1155/2022/3490860

[25] Pande, S.D., Chetty, M.S.R. (2018). Analysis of capsule network (Capsnet) architectures and applications. J Adv Res Dynam Control Syst, 10(10): 2765-2771.

[26] Pande, S.D., Chetty, M.S.R. (2019). Position invariant spline curve based image retrieval using control points. Int J Intell Eng Syst, 12(4): 177-191. http://dx.doi.org/10.22266/ijies2019.0831.17

[27] Pande, S.D., Patil, U.A., Chinchore, R., Chetty, M.S.R. (2019). Precise approach for modified 2 stage algorithm to find control points of cubic Bezier curve. In 2019 5th International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, pp. 1-8. https://doi.org/10.1109/ICCUBEA47591.2019.9128550

[28] Mandhala, V.N., Bhattacharyya, D., Midhunchakkaravarthy, D., Kim, H.J. (2022). Detecting and mitigating bias in data using machine learning with pre-training metrics. Ingénierie des Systèmes d'Information, 27(1): 119-125. https://doi.org/10.18280/isi.270114

[29] Govindarajan, M. (2022). Effective intrusion detection system using classifier ensembles. Ingénierie des Systèmes d'Information, 27(1): 151-156. https://doi.org/10.18280/isi.270118

[30] Gaykar, R.S., Khanaa, V., Joshi, S.D. (2021). Detection of faulty nodes in distributed environment using machine learning. 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), pp. 228-232. https://doi.org/10.1109/ICAC3N53548.2021.9725478

[31] Gaykar, R.S., Khanaa, V., Joshi, S.D. (2021). Identification of straggler node in distributed environment using soft computing algorithms. 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), pp. 1-5. https://doi.org/10.1109/ESCI50559.2021.9396825