A smart method for spark using neural network for big data

Md. Armanur Rahman[1], J. Hossen[2], Aziza Sultana[3], Abdullah Al Mamun[4], Nor Azlina Ab. Aziz[5]
[1,2,4,5]Faculty of Engineering and Technology, Multimedia University, Melaka, Malaysia
[3]Faculty of Computing and Engineering, Dhaka International University, Dhaka, Bangladesh

Article Info

Article history:

Received Aug 24, 2020
Revised Oct 6, 2020
Accepted Oct 27, 2020

Keywords:

Apache spark
Big data
Configuration parameters
Machine learning
Self-configuration

ABSTRACT

Apache spark, famously known for big data handling ability, is a distributed open-source framework that utilizes the idea of distributed memory to process big data. As the performance of the spark is mostly being affected by the spark predominant configuration parameters, it is challenging to achieve the optimal result from spark. The current practice of tuning the parameters is ineffective, as it is performed manually. Manual tuning is challenging for large space of parameters and complex interactions with and among the parameters. This paper proposes a more effective, self-tuning approach subject to a neural network called Smart method for spark using neural network for big data (SSNNB) to avoid the disadvantages of manual tuning of the parameters. The paper has selected five predominant parameters with five different sizes of data to test the approach. The proposed approach has increased the speed of around 30% compared with the default parameter configuration.

Corresponding Author:

Md. Armanur Rahman
Faculty of Engineering and Technology
Multimedia University
Melaka, 75450, Malaysia
Email: arman.bdmail@gmail.com

1. INTRODUCTION

Around the world, the number of online users is increasing at a rapid rate with the advancement of social communication and e-commerce business. Besides, a lot of users are storing their content constantly for future use. As indicated by International data corporation (IDC), digital space is projected to increase more than 44 Z.B. in volume by 2020 [1-3]. In the era of digital data, big data is something that can't be overlooked. Therefore recently, the big data era, different industries and governments have given emphasis on big data technologies. Since the conventional computing techniques could not provide the expected result and efficiency to manage big data. The different distributed frameworks like hadoop [4], spark [5], and storm [6] have been introduced to satisfy the prerequisite of taking care of the big data.

Apache spark is one of the most notable and broadly used frameworks because of its high performance and flexibility [7]. Apache spark has over 180 parameters with default values. The appropriate values of the parameter can be selected by the user manually while processing different sizes and types of data. The performance becomes unsatisfactory due to the inappropriate selection of parameter values. Therefore, additional tuning of the parameter is required for each particular application [8]. The users require appropriate knowledge for manual tuning of the parameters in the spark framework, however, manual tuning is very tedious due to the complex interaction between them.

As per the current practice, parameter tuning in big data is performed in 2 ways. Firstly, manual tuning of the parameter by trial and error. This process is very complicated as it requires a long time and depth knowledge due to a large number of parameters and its internal correlation with each other. To address the manual tuning problem, [9] author proposed a cost-based model for the hadoop system. However, the model needs to be persevered by users based on different policies. Secondly, self-tuning parameter when it requires. This paper proposes an approach based on a neural network to minimize the drawback of manual tuning. The research developed a self-tuning approach that can perform self-tuning of the parameter range based on the neural network model. This approach has three key advantages compared to the existing approaches. Firstly, all tasks are processed by the neural network model. Secondly, all types of datasets that consist of structured data, semi-structured data, and unstructured data can be processed. Thirdly, any volume of the dataset can be processed.

The training data has been collected for the selected five parameters by changing the parameter range and various input of data sets. The training process is only for one time to learn the machine learning model, which then can predict the numerical values for the selected parameters. The method has been implemented on a testbed that uses Dell PowerEdge R 720 server, hosting spark framework, and runs as spark nodes. The test results provide that our proposed method can perform effective self-tuning based on the neural network model so that it meets maximum resource usage capability and saves processing time. The key commitments of the method are as follows:

- It has implemented an artificial neural network in the approach that processes spark jobs using its application service based on the neural network model. Hence, users do not require in-depth knowledge of the internal system function. Thus, they can save time by avoiding manual tuning.
- The self-tuning facility of the approach integrates parameter range allocation. It helps to meet task deadlines and improves the overall performance of spark.
- In our evaluation using spark workloads with five different input datasets, the approach achieved an average performance speedup of about 30% performance.

The remains of the paper are organized as follows. Section 2, presenting the background of the study. Section 3, the related work, is discussed. Section 4 presents the details of the artificial neural network. Section 5 presents the architecture of SSNNB. The methodology is presented in section 6. Section 7 presents results and analysis-finally, Conclusions and future work presented in section 8.

2. BACKGROUND OF THE STUDY

2.1. Spark

In the area of big data, “Apache Spark” is the most accepted open-source platform that supports the idea of resilient distributed datasets (RDDs). The RDDs allow rapid treating of the massive size of data leveraging distributed memory. Data operation in memory is appropriate for repetitive applications such as graph algorithms and reiterative machine learning. RDD is considered as the main feature of spark. It characterizes a read-only collection of entities allocated among several machines. An RDD explicitly stores in the cache memory by the user over several machines and can be reused as the parallel operation in multiple MapReduce. RDD has the fault tolerance ability over a notion of extraction. Whenever a partition of RDD is lost, it can rebuild it since it has sufficient information regarding its origin. Though RDDs do not have shared memory construction, on the one hand, they can represent reliability and scalability and, on the other hand, a sweet-spot among expressivity. RDDs are well-suited for a diversity of applications. Figure 1 presents the spark-cluster framework [10]. A spark comprises a driver node that is equivalent to a master node and several worker nodes that are correspondent to slave nodes. The driver node manages all worker nodes through the worker node process. The worker nodes communicate with the driver node through the worker node process and manage local executors. Each application consists of multiple executors and one driver. All the jobs in an application come from the same executors. The spark context is creating by the main jobs of the application that are run by the driver process. Each of the worker nodes accomplishes one or more executor backend process during launching, and a single executor backend does managing executor instance. An executor manages a thread group that runs each of the tasks as a single thread. Nevertheless, the time of execution of a specific task in the platform of Apache depends on various factors such as input data volume, data type, CPU speed, memory size, number of nodes, configuration parameters, design and implementation of the system and so on. Based on these factors, the time of execution time of a specific job in apache spark may differ conspicuously [11]. There is more than 180 configuration parameter in apache spark that user can tune according to the need of a specific application to enhance the performance. It is the modest and most operative approach to enhance the enactment. Users tune these parameters physically by experiment [12]. At present, the parameters are manually tuned by experimentation that is not effective. It needs complicated interactions with the parameters and takes a larger parameter space. Again, these parameters must be re-tuned for various applications and clusters.

Artificial neural networks (ANN) is a mathematical processing method that can be used for both classification and regression [13, 14]. The neurons make it a powerful learning model for this reason for regression analysis. It is the best choice, including multiple inputs and output data [15, 16]. A neural network can predict numerical values correctly, and it can prevent overfitting easily. ANN is much suitable in several areas, including natural language and image processing, prediction as well as emotion recognition [17-19].

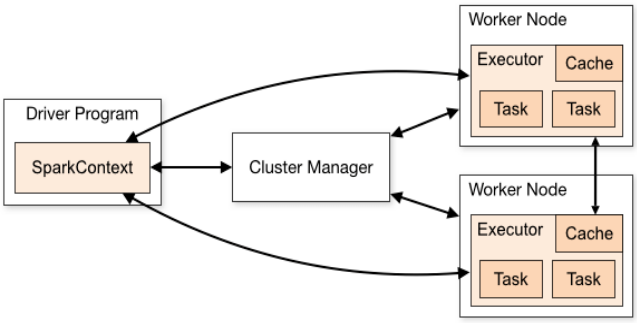

Figure 1. A common layout of apache spark

3. RELATED WORK

In recent years, one of the keenest research is in the optimization of the performance of big data system. However, almost all the existing researches have been done on the Hadoop platform or the framework of MapReduce computing. Starfish [9] utilizes simulation and a cost-based model to seek the required job configuration for the workload of MapReduce. AROMA [20] uses an optimization framework and two-phase ML to automate resource distribution and job configurations considering heterogeneous clouds. The authors of [21], indicated that hadoop scheduler in the heterogeneous environment, the performance reduction and proposed another scheduler named longest approximate time to end. In [22] a different work concentrated on examining the different resource consumption effects for variant set for the Reduce slots and Map. These problems have been addressed in [23], through a framework called “Profiling and Performance-based System” (PPABS), which can atomically tune the configuration of hadoop setting by deducting the requirements of application performance. Modifying the popular KMeans++ clustering along with the simulated Annealing algorithm are the main contributions of [24], which were needed to adjust to the MapReduce paradigm. Reference [23] recommends easing this issue by an engine that suggests the configurations for a new analytical job timely and intelligently. This engine is embedded in an adapted k-nearest neighbor (KNN) algorithm to discover the appropriate configuration based on the past job experience that is executed well. However, the research of optimizing apache spark performance is still in the beginning stage. The authors of [24], present a simulation driven forecast model to anticipate the performance of a job with high correctness for Apache Spark. Their proposed model can predict memory usage and execution time of spark systems in the case of default parameters. [25] Showed that the support vector regression (SVR) model is computationally efficient with high accuracy. According to their findings, it can be concluded that using the auto-tuning method can offer comparable or better performance compared to starfish with a fewer number of parameters.

4. ARTIFICIAL NEURAL NETWORK (ANN)

The scikit-learn is an essential tool since it allows only a few lines of coding and prevalent data groundwork. In order to proceed with the evaluation, the Keras wrappers need to be provided with a defined function to create ANN. In fact, the function is formulated to create a base model that is the subject of evaluation. The base model is connected with three neurons through a hidden layer, as illustrated in Figure 2. The hidden and output layer is activated with ReLU and softmax activation functions. Furthermore, an efficient optimizer "Adam" can be used to update network weights iteratively based on training data. The object in the Keras wrapper, known as KerasRegressor, is used as a regression estimator in the scikit-learn. The function of ANN is then created immediately to pass parameters including the batch size and epochs number along with the function of the model, both of which are set to default. Furthermore, a process of arbitrary number creator with a constant arbitrary seed has been initialized to compare the consistency of the models. In this research, the process of arbitrary number creators is repeated for the evaluation of each model. A neuron takes inputs, does some math with them, and produces an output. A simple neuron looks like what is shown in Figure 3.

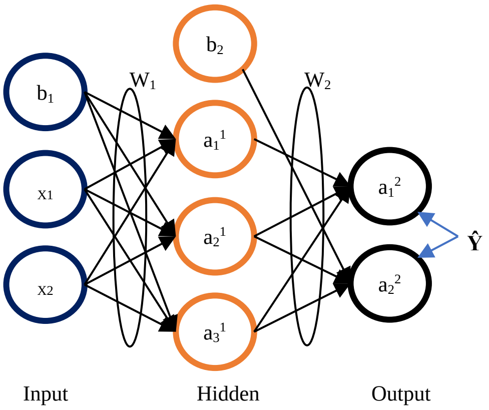

Figure 2. A neural network with hidden layer

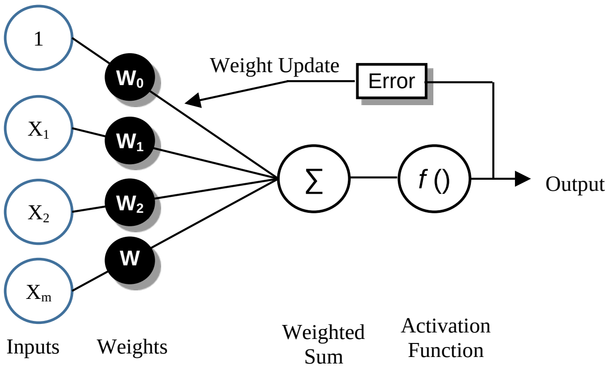

Figure 3. Layout of a simple neuron

Three things are happening here. First, each input is multiplied by a weight:

$$x_1 \rightarrow x_1 * w_1 , x_2 \rightarrow x_2 * w_2 , x_m \rightarrow x_m * w_m \qquad (1)$$

Next, all the weighted inputs are added together with a bias *b*:

$$(x_1 * w_1) + (x_2 * w_2) + (x_m * w_m) + b \qquad (2)$$

Finally, the sum is passed through an activation function:

$$y = f(x_1 * w_1 + x_2 * w_2 + x_m * w_m) + b \qquad (3)$$

4.1. Activation functions ReLU and softmax

Rectified linear unit (ReLU), is a recently popular activation function in neural networks [26-28]. It is well-defined as $f(x) = max(0, x)$. One of the advantages of the function is, it is also non-linear and can run backward for error minimization. Additionally, the function activates multiple neuron layers. Figure 4 shows the rectified linear unit (ReLU) activation function.

Softmax is a type of logistic function in mathematics. The softmax function accommodates outputs of each unit in between 0 to 1, displayed in a K-dimensional vector of random real numbers [29-31]. The function is used as an activation function due to its categorical probability distribution characteristic. The function is used for any number of classes and able to estimate the probability that any of the tested classes are true. The softmax function provided by

$$\sigma(Z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \qquad (4)$$

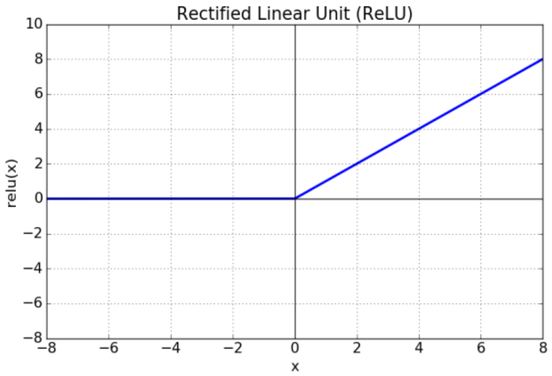

Figure 4. Rectified linear unit

5. SSNNB FRAMEWORK

The spark configuration parameters are tuned by the predicted values from the self-tuning approach SSNNB, which architecture is shown in Figure 5. SSNNB considers two input values, which are dataset size and execution time. From Figure 5, there are several blocks such as:

- Training data is obtained from a database
- The data has been received, and the model is generated by the “Model Training” block
- Generated model has been stored in a fixed location by the 'Store Model on Disk' block
- “Predicted Parameter Value”, this block provides the predicted optimum parameter value
- Finally, the predicted optimum values are received and updated in the “Spark System” block

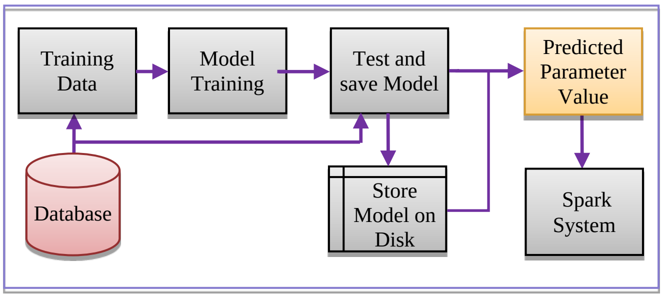

Figure 5. SSNNB architecture

6. METHODOLOGY

6.1. Parameter selection

The selected five parameters are shown in Table 1. The column 'Default value' displays the default parameter values, and the column 'Range value' displays the range of the selected parameters in the spark method [32-34]. Self-tuning is required when processing various sizes and different types of data to minimize processing time and achieve maximum performance from spark [35]. This paper selected five predominant parameters of the spark, based on the review of the authors [36]. The notable reason is: firstly, the selected five parameters are covered, including CPU, memory and disk of the resource in a cluster. Secondly, in schedule and shuffling modules, it has a great impact. Thirdly, this parameter also has a significant impact on the machine and cluster level [37].

Table 1. Default parameter value of spark with range

Spark Parameters	Spark Parameter	Range Value	Default Value
driver.cores	driver cores for a driver process	1-8	1
driver.memory	driver memory for a driver process	1g-4 g	1 g
executor.cores	cores are for executor process	10-40	1
executor.memory	executor of memory for per executor process	2g-8 g	1 g
reducer.maxSizeInFlight	Max size of the map outputs	24m-96 m	48 m

6.2. Data collection

Training data has been collected by the spark job, which is completed by changing the parameter and values and various dataset sizes and types. Finally, the sum of 3,000 sample data have been collected for training and testing the neural network model. For the high accuracy of the model, the normalization has been done.

6.3. Training and testing

For training, the neural network model has randomly sleeted 80% and the remaining 20% data have been used for testing. To get the best accuracy from the model, the training cycle has been repeated several times. In training, the epoch size has increased up to 250, and the model accuracy level was 97.1% and 96.7% for testing. It has observed that the accuracy has been increased during training and testing with the number of epochs is increased. It is observed from Figure 6 that, after 250 epochs, there is no significant improvement in both model accuracy and model loss.

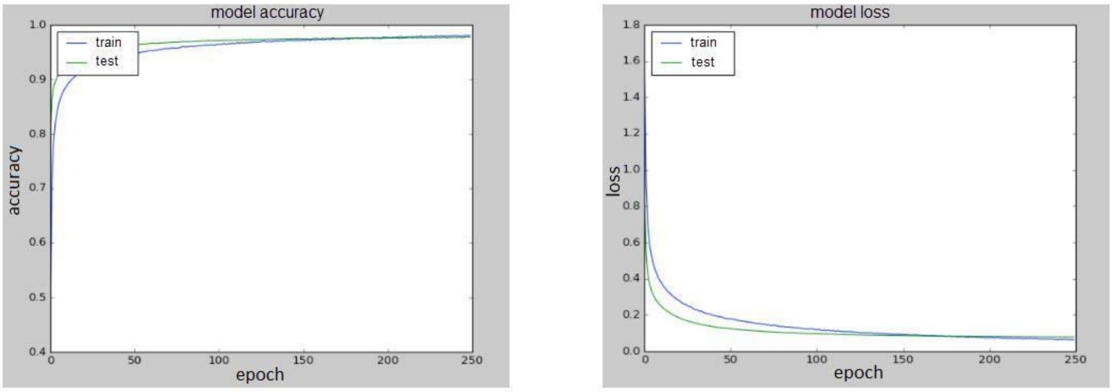

Figure 6. Model accuracy and loss in training

6.4. Test bed

The SSNNB approach has used the Dell PowerEdge R720 server as a testbed. The server is equipped with Intel® Xeon® CPU E5-2650 version 2.0 @ 2.60 GHz 16-core processor and 32 GB PC3 memory. The operating system was Ubuntu, and the version was 17.10 and hadoop version 2.8.1 with spark version 2.2.0. The self-tuning task can be run using an independent or a different VM. As listed in Table 2, the spark job is run with five different datasets ranging from 5 GB, 10 GB, 15 GB, 20 GB and 50 GB, which is collected from the Puma Benchmark suit. In order to facilitate a fair comparison with the default system, the five parameters are selected. Datasets ranging from 1 GB to 5 GB have been used during training, and the rest of the datasets up to 50 GB have been used during the evaluation process.

Table 2. Considered datasets

Spark	Size of dataset	Source of dataset
Word count	5 GB	Puma Benchmark
	10 GB	
	15 GB	
	20 GB	
	50 GB	

6.5. Artificial neural network model development

In ANN model development, the ML libraries are required, which are imported from Keras. One of the well-known libraries of Keras and behind it TensorFlow, is supported. Keras framework is much easier to use instead of directly using Tensorflow. In some respects, the variables X, Y, and Z are used to load and store the train and test data. Thus, X and Y comprise two training data; execution time and dataset size obtained by manual parameter tuning. Similarly, the variable Z holds the size and time of execution of the test data. The test dataset, as well as the train, are filled into the system. The necessary hidden layer is built from the base model. Furthermore, functions for activation are also added. In the base model, the dropout function (0.02) is added to prevent overfitting. It passed the optional learning rate of 0.0001 for the compilation of the model, and the designated learning rate is 0.01. After that, the optimizer Adam and the mean squared error (loss function) are compiled with the base model. X and Y data are then fitted with a scale function. To predict the accuracy of Z data, the base model combines batch size and epoch. The validity and loss of analysis are printed. The activation function or the number of epoch or the optimizer must be changed if the accuracy is lower than the expected result. The accuracy of 96.9% for testing and 97.8% for training data could be accomplished by utilizing 250 epoch and appropriately changing the others-the accuracy of increments in training and testing segments when the quantity of epochs is increased. Figure 6 shows that beyond 250 epochs, accuracy or loss is not substantially improved. The model will be saved for every parameter. It has five models built by modifying the Y with five distinct parameters, which is illustrated in Figure 7.

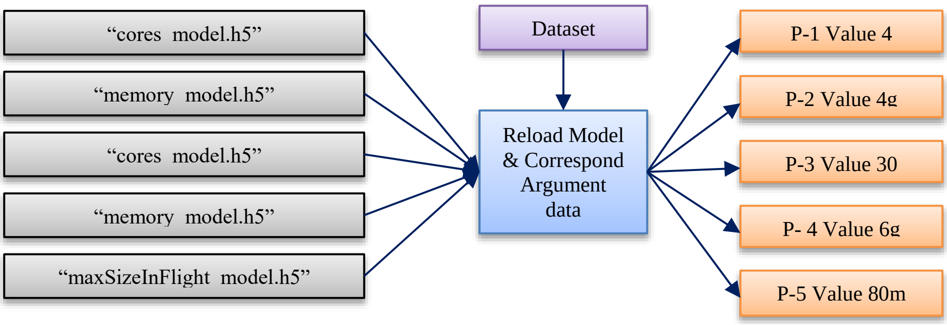

Figure 7. ANN models to predict the optimized parameter (P for the parameter)

7. RESULTS AND ANALYSIS

7.1. SSNNB model efficiency

Figure 8 represents the computational time of spark work independently for both default design and SSNNB. For various sizes of input datasets. It has been seen that the time necessary in executing spark job is essentially lower with SSNNB rather than the default parameter boundary settings free of information size in the scope of 5 GB to 50 GB.

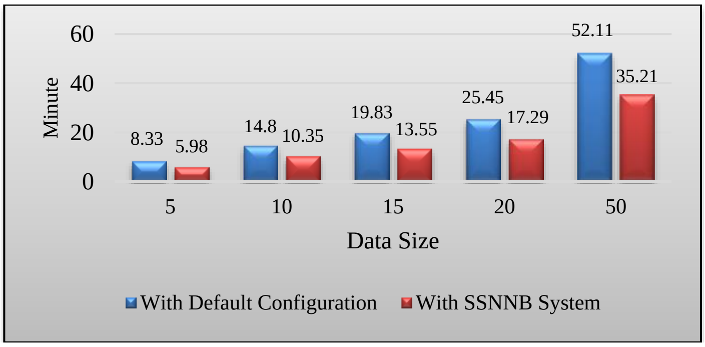

Figure 8. Comparison with SSNNB approach and default configuration

7.2. Ability of Self-tuning and execution time speedup

To assess the ability of the SSNNB framework, a spark job has been evaluated for five distinct sizes of input data extending from 5 GB, 10 GB, 15 GB, 20 GB to 50 GB independently with both the SSNNB and the default design. The predicated ideal parameters value has been introduced in Figure 9. Referring to Figure 8, with the default configuration, for dataset sizes of 5, 10, 15, 20, and 50 GB, spark takes 8.33, 14.8, 19.83, 25.45, and 52.11 minutes separately. Notwithstanding, the SSNNB framework takes 5.98, 10.35, 13.55, 17.29, and 35.21 minutes separately. In Tables 3 and 4, it can be seen from the result that the SSNNB approach achieved an average 30% faster compared to the default configuration with independent dataset size.

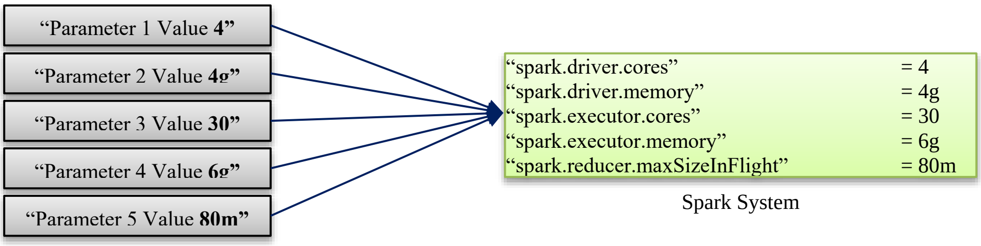
"Parameter 1 Value **4**"
"Parameter 2 Value **4g**"
"Parameter 3 Value **30**"
"Parameter 4 Value **6g**"
"Parameter 5 Value **80m**"
"spark.driver.cores" = 4
"spark.driver.memory" = 4g
"spark.executor.cores" = 30
"spark.executor.memory" = 6g
"spark.reducer.maxSizeInFlight" = 80m
Spark System

Figure 9. Detected optimum values for spark parameters

Table 3. Processing time reduce for different dataset

Data Input	Process with Default Configuration Execution Time (Min)	Process with SSNNB system Execution Time (Min)	Time Saved In Min
5 GB	8.33	5.98	2.35
10 GB	14.8	10.35	4.45
15 GB	19.83	13.55	6.28
20 GB	25.45	17.29	8.16
50 GB	52.11	35.21	16.9

Table 4. Predicted optimum parameter value using SSNNB approach

Configurable Parameters	Default Parameter Value	With SSNNB 5 GB	With SSNNB 10 GB	With SSNNB 15 GB	With SSNNB 20 GB	With SSNNB 50 GB
Number of cores of driver process	1	3	4	6	6	8
Driver process memory size in Giga Bytes	1 g	2 g	4 g	4 g	4 g	4 g
Number of cores of executor process	1	20	20	30	30	40
Executor process memory size in Giga Bytes	1 g	3 g	4 g	4 g	5 g	6 g
Maximum number of the map to each reducer task	48 m	48 m	60 m	60 m	65 m	80 m

8. CONCLUSION

This research introduces a novel way to deal with the self-tuning approach for spark predominant parameters to speed up the execution while handling big data, including the different sizes of the dataset and variety of data. Moreover, estimation of optimum parameter value for five selected parameters is enabled by the approach. The approach received the optimum value from the neural network model and updated it in the spark system before processing. Dell Poweredge R70 server, including five different datasets, has been used in the procedure. The performance of SSNNB is compared with the default configuration, and the result shows the performance improvement is 30% on an average. It has also been observed that the performance was improving while increasing the dataset size. Future research will focus on how to select a more appropriate number of parameters and use better servers to obtain better outcomes. Metaheuristics algorithms are to be considered for this optimization.

ACKNOWLEDGEMENTS

This research is funded by the Ministry of Higher Education, Malaysia, under the Fundamental Research Grant Scheme FRGS/1/2019/ICT02/MMU/02/15. The authors also would like to acknowledge the anonymous reviewers for their valuable comments and insights.

REFERENCES

[1] Archana, R. A., Ravindra S. Hegadi, and T. N. Manjunath, "A Study on Big Data Privacy Protection Models using Data Masking Methods," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 5, pp. 3976-3983, 2018, doi: 10.11591/ijece.v8i5.pp3976-3983

[2] Anagnostopoulos, Ioannis, Sherali Zeadally, and Ernesto Exposito, "Handling big data: Research challenges and future directions," *The Journal of Supercomputing*, vol. 72, no. 4, pp. 1494-1516, 2016.

[3] Salkuti, Surender Reddy, "A survey of big data and machine learning," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 1, pp. 575-580, 2020, doi: 10.11591/ijece.v10i1.pp575-580.

[4] Jankatti, S., Raghavendra, B. K., Raghavendra, S., and Meenakshi, M., “Performance evaluation of Map-reduce jar pig hive and spark with machine learning using big data,” *International Journal of Electrical and Computer Engineering (IJECE),* vol. 10, no. 4, pp. 3811-3818, 2020, doi: 10.11591/ijece.v10i4.pp3811-3818.

[5] Hasan, R. A., Alhayali, R. A. I., Zaki, N. D., and Ali, A. H., “An adaptive clustering and classification algorithm for Twitter data streaming in Apache Spark,” *TELKOMNIKA Telecommunication, Computing, Electronics and Control,* vol. 17, no. 6, pp. 3086-3099, 2019.

[6] Nivash, J. P., et al., “Analysis on enhancing storm to efficiently process big data in real time,” *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT),* 2014. pp. 1-5.

[7] Raswitha Bandi, and G. Anitha, "Machine Learning with PySpark-Review," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS),* vol. 12, pp. 102-106, 2018.

[8] Kalyani K., and Pathrikar, “Review on apache spark technology,” *International Research Journal of Engineering and Technology (IRJET),* vol. 04, pp. 1386-1388, Oct. 2017.

[9] Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F. B., and Babu, S., “Starfish: A Self-tuning System for Big Data Analytics,” *Cidr,* vol. 11, no. 2011, pp. 261-272, 2011.

[10] Riza, L. S., Pratama, F. D., Piantari, E., and Fashi, M., “Genomic repeats detection using Boyer-Moore algorithm on Apache Spark Streaming,” *TELKOMNIKA Telecommunication, Computing, Electronics and Control,* vol. 18, no. 2, pp. 783-791, 2020.

[11] Salloum, S., Dautov, R., Chen, X., Peng, P. X., and Huang, J. Z., “Big data analytics on Apache Spark,” *International Journal of Data Science and Analytics,* vol. 1, no. 3-4, pp. 145-164, 2016.

[12] Jonnalagadda, V. S., Srikanth, P., Thumati, K., and Nallamala, S. H, “A review study of apache spark in big data processing,” *International Journal of Computer Science Trends and Technology,* vol. 4, no. 3, pp. 93-98, 2016.

[13] Naser and M. Abd Ulkareem, “Prediction prices of basrah light oil using artificial neural networks,” *International Journal of Electrical and Computer Engineering (IJECE),* vol. 10, no. 3, pp. 2682- 2689, 2020.

[14] Mahmood, Maha, Belal Al-Khateeb, and Wisam Makki Alwash, “A review on neural networks approach on classifying cancers,” *Int. J. Artif. Intell.,* vol. 9, no. 2, pp. 317-326, 2020.

[15] Kwiatkowski, B., Bartman, and Mazur, “The quality of data and the accuracy of energy generation forecast by artificial neural networks,” *International Journal of Electrical and Computer Engineering (IJECE),* vol. 10, no. 4, pp. 3957-3966, 2020.

[16] Anh, Q. H., Tan, P. T., and An, N. T., “A hybrid Artificial neural network-genetic algorithm for load shedding,” *International Journal of Electrical and Computer Engineering (IJECE),* vol. 10, no. 3, pp. 2250-2258., 2020.

[17] Goay, C. H., et al., “Progress in neural network based techniques for signal integrity analysis–a survey,” *Bulletin of Electrical Engineering and Informatics (BEEI),* vol. 8, no. 1, pp. 276-282, 2019.

[18] Souri, A., El Maazouzi, Z., Al Achhab, M., and El Mohajir, B. E., “Neural network dealing with Arabic language,” *International Journal of Informatics and Communication Technology (IJ-ICT),* vol. 9, no. 2, pp. 73-78, 2020.

[19] Abougarair, A. J., “Neural Networks Identification and Control of Mobile Robot Using Adaptive Neuro Fuzzy Inference System,” *Proceedings of the 6th International Conference on Engineering & MIS,* 2020, pp. 1-9.

[20] Lama, P., and Zhou, X., “Aroma:Automated resource allocation and configuration of mapreduce environment in the cloud,” *Proceedings of the 9th international conference on Autonomic computing,* 2012, pp. 63-72.

[21] Zaharia, M., Konwinski, A., Joseph, A. D., Katz, R. H., and Stoica, I., “Improving MapReduce performance in heterogeneous environments,” *Osdi,* vol. 8, no. 4, pp. 29-42, 2008.

[22] Wu, D., and Gokhale, A., “A self-tuning system based on application profiling and performance analysis for optimizing hadoop mapreduce cluster configuration,” *20th Annual International Conference on High Performance Computing,* 2013, pp. 89-98.

[23] Zhang, R., Li, M., and Hildebrand, D., “Finding the big data sweet spot: Towards automatically recommending configurations for hadoop clusters on docker containers,” *IEEE International Conference on Cloud Engineering,* 2015, pp. 365-368.

[24] Wang, K., and Khan, M. M. H, “Performance prediction for apache spark platform,” *IEEE 17th International Conference on High Performance Computing and Communications,* 2015, pp. 166-173.

[25] Yigitbasi, N., Willke, T. L., Liao, G., and Epema, D., “Towards machine learning-based auto-tuning of mapreduce,” *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems,* pp. 11-20, 2013.

[26] Sahana, H. P., Sanjana, M. S., Muddasir, and N. M., “Apache Spark Methods and Techniques in Big Data-A Review,” *Inventive Communication and Computational Technologies,* pp. 721-726, 2020.

[27] Nazmul Haque, and Md. Hasnat Riaz., "Autonomous Vehicle Control System as a Mobile Robot by Artificial Neural Network," *International Journal of Robotics and Automation (IJRA),* vol. 6, no. 3, pp. 200-206, 2017.

[28] Shatha A. Baker, Hesham H. Mohammed, Hanan and A. Aldabagh, “Improving Face Recognition by Artificial Neural Network Using Principal Component Analysis,” *TELKOMNIKA Telecommunication, Computing, Electronics and Control,* vol. 18, no. 6, pp. 3357-3364, 2020.

[29] Shaikh, E., Mohiuddin, I., Alufaisan, Y., and Nahvi, I., “Apache Spark: A Big Data Processing Engine,” *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM),* 2019, pp. 1-6.

[30] Al-Azzawi, D. S., “Application and evaluation of the neural network in gearbox,” *TELKOMNIKA Telecommunication, Computing, Electronics and Control,* vol. 18, no. 1, pp. 19-29, 2020.

[31] Abd Rahman, N. H., and Lee, M. H., “Artificial neural network forecasting performance with missing value imputations,” *IAES International Journal of Artificial Intelligence (IJ-AI),* vol. 9, no. 1, pp. 33-39, 2020.

[32] Bhattacharya, A., and Bhatnagar, S., “Big data and apache spark: A review,” *International Journal of Engineering Research & Science,* vol. 2, no. 5, pp. 206-210, 2016.

[33] Patil, N. S., et al., “A survey on graph database management techniques for huge unstructured data,” *International Journal of Electrical and Computer Engineering (IJECE),* vol. 8, no. 2, pp. 1140 -1149, 2018.

[34] Nair, L. R., Shetty, S. D., and Shetty, S. D., “Streaming big data analysis for real-time sentiment based targeted advertising,” *International Journal of Electrical and Computer Engineering (IJECE),* vol. 7, no. 1, pp. 402-407, 2017.

[35] Vijayarekha, K., “Activation Functions, NPTEL-Electron,” *Commun. Eng.-Pattern Recognit,* pp. 1-6, 2015.

[36] Md. Armanur Rahman, J. Hossen and Venkataseshaiah C., “SMBSP: A Self-Tuning Approach using Machine Learning to Improve Performance of Spark in Big Data Processing,” *7th International Conference on Computer and Communication Engineering,* 2018, pp. 274-279.

[37] Md. Armanur Rahman1, Abid Hossen, J. Hossen, Venkataseshaiah C., “Towards Machine Learning based Self-tuning of Hadoop-Spark System,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS),* vol. 15, no. 2, pp. 1076-1085, 2019.

BIOGRAPHIES OF AUTHORS

Md. Armanur Rahman received B.Sc. degree in computer science and engineering from Asian University of Bangladesh (AUB) in 2010, Masters (MEngSc.) degree in Big data and Machine Learning from the Multimedia University (MMU), Malaysia in 2019. Now he is persuing Ph.D. in Facial Expression Recognition using Machine Learning at Multimedia University (MMU). His research interest include performance optimization of big data system, data mining, machine learning and image processing.

Jakir Hossen is graduated in Mechanical Engineering from the Dhaka University of Engineering and Technology (1997), Masters in Communication and Network Engineering from Universiti Putra Malaysia (2003) and PhD in Smart Technology and Robotic Engineering from Universiti Putra Malaysia (2012). He is currently a Senior Lecturer at the Faculty of Engineering and Technology, Multimedia University, Malaysia. His research interests are in the area of Artificial Intelligence (Fuzzy Logic, Neural Network), Inference Systems, Pattern Classification, Mobile Robot Navigation and Intelligent Control.

Aziza Sultana received the B.Sc. degree in computer science and engineering from Dhaka International University (DIU) in 2016. She is currently persuing Masters degree in Computer Science and Engineering at the same university. Her research interest include performance optimization of big data system, data mining, machine learning and image processing.

Abdullah Al Mamun has received B.Sc. degree in Electrical and Electronic Engineering from Pabna University of Science and Technology in 2018. Now he is pursuing M.Eng.Sc.at Multimedia University (MMU) in the Faculty of Engineering and Technology since 2019. His research interest includes computer vision; image processing, signal processing, deep learning and machine learning.

Nor Azlina Ab Aziz she is currently a Senior Lecturer in the Faculty of Engineering and Technology at Multimedia University, Melaka. She is interested in the field of soft computing and its application in engineering problems. More specifically, her focus is in the area of swarm intelligence and nature inspired optimization algorithm.