A Smart Method using Machine Learning for Spark to Process Big Data

Md. Armanur Rahman[1*], J.Hossen[2], Aziza Sultana[3], Md. Akhtaruzzaman[4], Jesmeen M. Z. H.[5], K. Tawsif [6], E.M.H. Arif [7]

[1, 2, 5,6,7] Faculty of Engineering and Technology, Multimedia University, Melaka, 75450, Malaysia
[3]Faculty of Computing and Engineering, Dhaka International University, Dhaka, 1205, Bangladesh
[4]Faculty of Computing and Engineering, Asian University of Bangladesh, Dhaka, 1230, Bangladesh

[*]Corresponding e-mail: armanur.rahman@gmail.com

Keywords: Big data, machine learning, Hadoop-Spark

ABSTRACT – Apache Spark is a distributed open source framework for big data processing. The performance of Spark is greatly affected by parameter configuration settings. To get the best performance from Spark is still a big challenge because of a large number of parameters. This parameter is tuned manually by experimentation which is not effective. Besides, these parameters must be re-tuned for various applications. In this work, a method based on machine learning is proposed and developed to effectively self-tune Spark parameters. The results show that the performance is speeded up by 33.4% on an average, compared to the default configuration.

1. INTRODUCTION

The advancement of the social network, mobile network and e-commerce are rapidly and stubbornly growing which is continuously increasing the figure of internet users. These massive number of internet users are donating huge content for future uses. According to IDC, the volume of digital data will go beyond 44ZB by 2020. [1-3]. The presence of big data cannot be declined in the current situation of the digital world. Currently, with the appearance of big data, a big devotion is seen for big data technologies which are being applied in various areas such as industry, governments, and academia. The traditional computing system is unable to offer the required proficiency and achievement. Thus, various distributed platforms such as Spark [4], Hadoop [5] and Strom have been ascended to deliver the demands of big data processing [6]. Among these accessible frameworks, Apache Spark has the global popularity for its dimensions to support rich applications with high processing performances.

Spark has over 180 parameters which control the system performance [7]. These parameters have default values. The User can manually select the suitable values for each parameter. Improper choice of the parameter value leads to poor performance. Manual tuning of the parameters in the Spark system requires the user to have in-depth knowledge of the system. Because of large parameter space, manual tuning is very time consuming and inefficient. Re-tuning of the parameters may be required for each different application [8].

To overcome this manual tuning problem, we developed a method using Linear Regression with Neural Network that self-tunes the parameter range when it requires while processing big data.

2. METHODOLOGY

a. Parameter Selection

In this work, five parameters have been nominated as these are able to cover most of the existing properties of a cluster which generally includes memory, CPU, hard disk and so on.

Table 1 Spark parameters and their default values

Parameter	Default	Range
spark.driver.cores	1	1-8
spark.driver.memory	1g	1g-4g
spark.executor.cores	1	10-40
spark.executor.memory	1g	2g-8g
Spark.reducer.maxSizeInFlight	48m	24m-96m

b. Data Collection

For training and testing, two input data, size of data and execution time have been collected. To accumulate input data, wordcount job has been employed by altering parameters and their values within the given variety for different sizes of the dataset and 3000 sample of data have been collected.

c. Training and Testing

In order to train and test model, the data was split into 70% and 30% for training and testing respectively and the training phase has been repeated for a number of times.

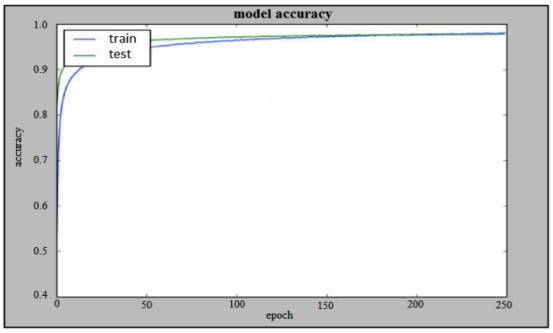

Figure 1 Model accuracy and loss in train and test cases

It is noticed that, by increasing the number of the epoch, training and testing can achieve 95.9% and 95.6% correspondingly. The observation shows that the accuracy of both training and testing increases the increase of epoch. Figure 1, described that epoch beyond 250 does not increase the accuracy.

d. Built Model Using Linear Regression (LR) with Neural Network

To develop this method we used LR with Neural Network (NN).

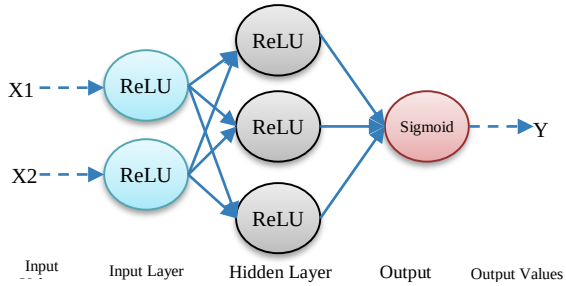

Figure 2 A Standard Neural Network

In NN have one input and a hidden layer where we use ReLU activation function and sigmoid activation function is used in the output layer. The equation 1 and 2 shows the ReLU and Sigmoid function and equation 3 is for LR. For optimization the NN we used "adam" that optimizer calculates both propagations that is feedforward and backpropagation.

Keras wrapper object is known as KerasRegressor, which is treated as a regression estimator in scikit-learn. The function of the Linear Regression model is created instantly and passed some parameters along with the model such as batch size, number of epochs and so on. To certify the evaluation of model consistency, it has initialized a process of random number generator with a constant random seed.

For evaluating each model, this process will be carried out constantly. After that, the model is evaluated based on train and test data and finally, the weights are saves on the hard disk utilizing the hdf5 file for prediction. Running the process gives us a prediction of the model's performance on the problem for unprocessed data. The result includes the mean squared error the average and standard deviation.

$$f(x)=\max(0,x) \quad (1)$$

$$f(x) = \frac{1}{1+e^{-x}} \quad (2)$$

$$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} + \varepsilon_i \quad (3)$$

e. Test Bed

The experiment has been implemented on a testbed which consists of a server named Dell PowerEdge R720. The server is equipped with Intel (R) Xeon (R) CPU E5-2650 v2 storage 2.6GHz 16 core processor 32 GB PC3 memory. Linux operating system and Spark version 2.2.0 with Hadoop version 2.8.1 are used by the server. In spark System, the wordcount job was run as shown in the following Table 2 and the PUMA benchmark datasets were used for the experiment.

Table 2 Datasets considered in this research

Size of Data sets	Data sets Collected From	Spark Program
50 GB	Puma Benchmark	Wordcount
100 GB	Puma Benchmark	Wordcount
150 GB	Puma Benchmark	Wordcount
200 GB	Puma Benchmark	Wordcount
250 GB	Puma Benchmark	Wordcount

3. RESULT AND DISCUSSION

To assess the capability of methods to self-tune the parameters of Spark based on the different size of dataset, wordcount job was executed five times with five variant dataset sizes which are 50, 100, 150, 200 and 250 GB. From Figure 3 and Table 3 we can see that the execution time with default configuration for 5 chosen dataset are 41.25, 80.13, 118.13, 156.48 and 194.01 minutes whereas our proposed method takes 28.31, 54.29, 79.28.101.18 and 125.7 respectively.

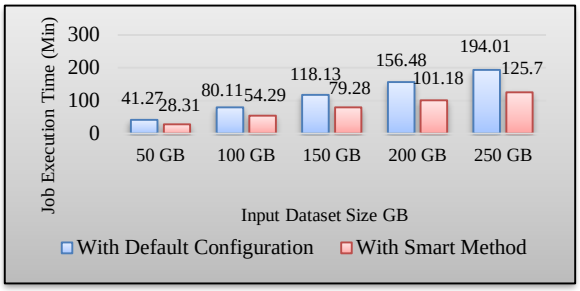

Figure 3 Comparison with Default Configuration

Table 3 Predicted optimal parameter value for each dataset

Configuration Parameters	Default	Range	Predicted Value for 50GB	Predicted Value for 100GB	Predicted Value for 150GB	Predicted Value for 200GB	Predicted Value for 250GB
spark.driver.cores	1	1-8	2	3	6	6	7
spark.driver.memory	1g	1g-4g	3g	3g	4g	4g	6g
spark.executor.cores	1	10-40	25	35	35	40	40
spark.executor.memory	1g	2g-8g	3g	3g	5g	6g	6g
Spark.reducer.maxSizeInFlight	48m	24m-96m	48m	65m	65m	75m	80m

4. CONCLUSION

A smart method is proposed in this paper for self-tuning configuration of Spark to increase its performance for processing big data. It enables optimum range estimation for five designated parameters and updates the system of Spark before starting of processing. The results indicated that the average performance is increased up to 33.4% compared to the default configuration. It is noticed that the improvement increases with the dataset size. Further research is being continued for selecting the more appropriate number of parameters utilization for better servers.

REFERENCES

[1] Profile US 2013 The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East- United States 1–7.

[2] Anagnostopoulos, I., Zeadally, S., & Exposito, E. (2016). Handling big data: research challenges and future directions. *J. Supercomput. 72*, 1494–516.

[3] McKinsey & Company. (2011). Big data: The next frontier for innovation, competition, and productivity McKinsey Glob. Inst. 156.

[4] Bhattacharya, A., & Bhatnagar, S. (2016). Big Data and Apache Spark : A Review, 206–10.

[5] Kaur. I., Kaur, N., Ummat, A., Kaur, J., & Kaur, N. (2016). Research Paper on Big Data and Hadoop, 8491, 50–3.

[6] Van Der Veen, J. S., Van Der Waaij, B., Lazovik, et, al., (2015) Dynamically scaling apache storm for the analysis of streaming data Proc. - *2015 IEEE 1st Int. Conf. Big Data Comput*. Serv. Appl. BigDataService, 154–61.

[7] Wang, G., Xu, J., & He, B. (2016). A Novel Method for Tuning Configuration Parameters of Spark Based on Machine Learning.

[8] Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F.B., & Babu. S. (2011). Starfish: Self-tuning System for Big Data Analytics, Fifth Bienn. Conf. Innov. Data Syst. Res. Asilomar, CA, USA, January 9-12, 2011, Online Proc. 261–72.