# Towards machine learning-based self-tuning of hadoop-spark system

**Md. Armanur Rahman[1], Abid Hossen[2], J. Hossen[3], Venkataseshaiah C[4], Thangavel Bhuvaneswari[5], Aziza Sultana[6]**
[1,3,4,5]Faculty of Engineering and Technology, Multimedia University, Malaysia
[2]Faculty of Computing and Engineering, Khulna University, Bangladesh
[6]Faculty of Computing and Engineering, Dhaka International University, Bangladesh

**Article Info**

***Article history:***

Received Sep 21, 2018
Revised Feb 1, 2019
Accepted Feb 25, 2019

***Keywords:***

Apache spark
Big data
Machine learning
Self-tuning
Spark parameter

**ABSTRACT**

Apache Spark is an open source distributed platform which uses the concept of distributed memory for processing big data. Spark has more than 180 predominant configuration parameter. Configuration settings directly control the efficiency of Apache spark while processing big data, to get the best outcome yet a challenging task as it has many configuration parameters. Currently, these predominant parameters are tuned manually by trial and error. To overcome this manual tuning problem in this paper proposed and developed a self-tuning approach using machine learning. This approach can tune the parameter value when it's required. The approach was implemented on Dell server and experiment was done on five different sizes of the dataset and parameter. A comparison is provided to highlight the experimented result of the proposed approach with default Spark configuration system. The results demonstrate that the execution is speeded-up by about 33% (on an average) compared to the default configuration.

***Corresponding Author:***

Md. Armanur Rahman,
Faculty of Engineering and Technology,
Multimedia University, Jalan Ayer Keroh Lama,
Melaka, 75450 Bukit Beruang, Malaysia.
Email: arman.bdmail@gmail.com

## 1. INTRODUCTION

The improvement of the mobile network, e-commerce, the social network is continuously and vastly increasing which results in the increment of a number of internet users. The increasing number of internet users constantly generates huge content of data for future use. According to an indication by IDC by the end of 2020, the amount of digital data will be more than 44 ZB [1-3]. The existence of big data cannot be denied with the current state of the digital world. Recently, big data technologies gained huge concentration with the evolving of big data in the sectors such as government, academia, and industries. The traditional computing system cannot offer the necessary efficiency and performance. Therefore, the big data industries have seen various platforms such ad Spark [4], Haddoo [5, 6] and Strom [7] to entertain the demands of a large amount of big data processing. Apache spark is one of the most widespread frameworks among the prevailing distributes framework, due to its great capability to sustenance heavy applications and for complex data processing performance [2, 4]. The most popular processing platforms in is Apache Spark which offers high-level API in scalar, python, and Java [8]. In the spark, the system has more than 180 parameters that require to adjust manually for each individual application in order to increase applications functionalities [9]. It is the only operational and retiring method to enrich the capability. In one hand, the huge number of parameter space offers a lot of chances to improve prominent proficiency by tuning parameter carefully. Then again, it is very tough to tune abundance of parameters because of the complex interaction among parameters [10].

We have developed a novel approach using machine learning which can self-tune the parameter value while processing big data.

Whatever remains of this paper is structured as follows. The details of Apache spark and related work are provided in section 2 and 3 accordingly. Section 4 described the steps of methodology. The Block diagram of the novel approach are presented in section 5. The result and analysis is presented in section 6. The conclusion of the paper and suggestion is described in section 7.

## 2. BACKGROUND OF APACHE SPARK

In the big data industries, Apache Spark is the greatest acknowledged open source platform which declares the great idea of using "Resilient Distributed Datasets (RDDs)" [4]. RDDs permits rapid considering of the huge data size extracting distributed memory. The key feature of Apache Spark is RDD that is characterized by a read-only entities collection assigned among various machines. An RDD can explicitly store data in the cache memory set by the user for several times and reuse it in parallel nature as the MapReduce does. RDD has the characteristics of tolerating fault through an extraction notation. RDD can rebuild the lost partitions of data as it has the sufficient information about the origin of the data. RDDs are considered as the well-suited for diverse of applications [11-14]. The spark cluster framework is demonstrated in the Figure 1 framework.

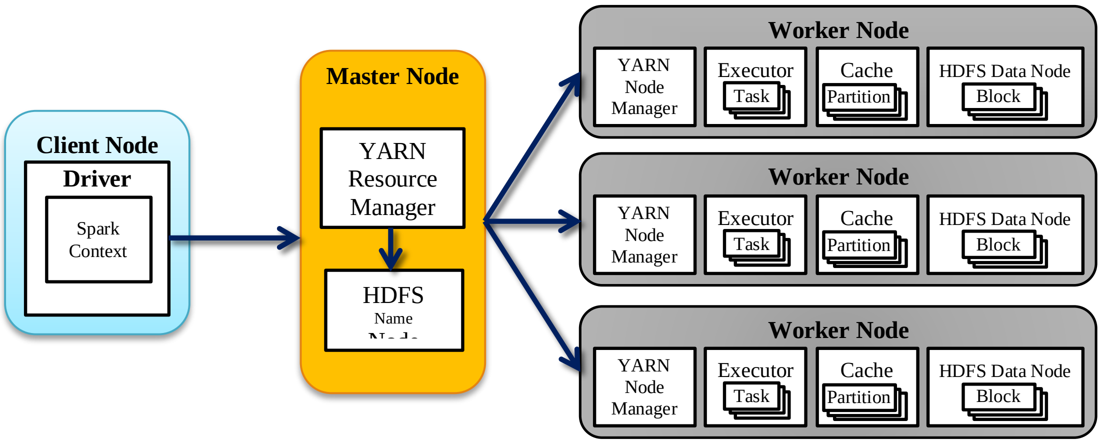

Figure 1. Spark Physical Cluster

The Apache Spark consist with a driver node which is corresponding to a master node and a number of workers node which are a reporter to slave nodes. All the worker nodes are managed by the driver node through a process named worker daemon process. The worker daemon process helps worker's nodes to communicate with driver node as well as to manage local executors. Each application comprises of multiple one driver and multiple executors. Each application comprises with one drive and a number of executors. The driver process runs the key jobs of the application and generates SparkContext. Each worker nodes perform either one or more Executor backed process while initiating and supervising instance is accomplished by a single Executorbacked. An executor accomplishes a group of the thread which tracks each of the jobs as a single thread. However, the execution time of a definite job in the Apache platform rely on various aspects such as the volume of input, CPU speed, data type, size of memory, the number of nodes, design and implementation of the system, parameter configuration, and so on. The execution time in Apache Spark platform may differ obviously in each individual job based on these aspects.

## 3. RELATED WORK

In the present years, one of the hottest research is performance optimization of big data due to the wide big data transformation analytics platform. Nevertheless, most of the prevailing researches have been conducted on either MapReduce (MR) computing framework or Hadoop-Spark platform. Starfish uses simulation and a cost-based model to look for required employment setup for the workload of MR. AROMA [15] exploits an optimization context as well as a two-stage ML to reset resource distribution and job configuration keeping in mid heterogeneous clouds. The authors of [16], point out that Hadoop scheduler in a heterogeneous situation can result in serious performance reduction and thus they proposed an alternative scheduler titled Longest Approximate Time to End.

In [17] a concentration was seen for examining diverse resource consumption consequence for a different set for Map and reduce slots. These difficulties have been solved by [18] over a system named “Profiling and Performance-Based System (PPABS)” that can auto-tune tune Hadoop configuration setting by reducing the needs of application performance. The key contribution of [18] is modifying widespread KMeans++ cluster along with simulated strengthening algorithm that was required to adjust to MR paradigm. Reference [18] recommends simplifying this issue by an engine which proposes the configuration for a new analytical task intelligently and timely. In order to discover the correct configuration in which the past job performed well, this engine was embedded into the modified k-nearest neighbor (KNN). However, researching the Apache Spark performance optimization is still the beginning stage. A simulation driven prediction model to estimate job performance with high perfection for Apache Spark is presented in [19]. Their proposed model predicts the execution time and memory usage of the Spark system in the situation of defaults parameters. The authors of [20] presented that “Support Vector Regression” (SVR) is computing effective with high accurateness. Based on their findings, it allows concluding that utilizing automatic parameter tuning can provide improved performance compared to Starfish with using relatively few parameters.

## 4. METHODOLOGY

### 4.1. Data Collection

In order to train and test our proposed system, we have collected two input data, target time and dataset size by processing Puma beanchmark. Wordcount job was initiated to collect the input data by changing parameter and their values for different dataset sizes. A number of 3000 data sample data was collected for training and testing to the machine learning model. The data was separated into 80:20 for training and testing respectively.

### 4.2. Parameter Selection

In this proposed work, we have selected five configuration parameters which are exposed in Table 1. In this table defaults parameter shows the values with default setting and the range of the parameters are shown by the column named range. Parameter range is used to reduce the process time and to maximize performance when the parameters can not tune automatically as needed. Parameter selection is an important issue for the research of this arena. By considering the notable facts, five parameters have been selected. First, these five parameters are available almost in all existing resources of the clusters that generally includes memory, CPU disk and so on. The second thing that the selected parameter can play a significant role for both scheduling and shuffling modules. Thirdly, different levels of clusters are impressively affected by these parameters [21, 22].

Table 1. Selected predominant parameter

| Parameter | Description | Default | Range |
|---|---|---|---|
| “spark.driver.cores” | Number of cores to use for the driver process | 1 | 1-8 |
| “spark.driver.memory” | Amount of memory to use the driver process | 1g | 1g-4g |
| “spark.executor.cores” | Number of cores to use for the executor process | 1 | 10-40 |
| “spark.executor.memory” | Amount of memory to use per executor process | 1g | 2g-8g |
| “Spark.reducer.maxSizeInFlight” | Maximum size of map outputs to fetch simultaneously from each reduce task | 48m | 24m-96m |

### 4.3. Flowchart

To develop this approach we made two process one is for model making and another one is the-prediction. Figure 2 shows the model making flowchart. The flowchart includes the required machine learning libraries, train data, test data, model defined, model compile, model fit with train data, predict the model with test data and model save. Figure 3 Prediction flowchart shows how the optimum parameter values are predicted using the model generated earlier and saved in the disk. It incorporates the following steps: load desired dataset; provide input values of predefined target time and dataset size; load the generated model; predict the optimum parameter range using model; receive and update the optimum parameter values in Spark system; start processing the desired dataset and after execution is done reset default values in Spark.

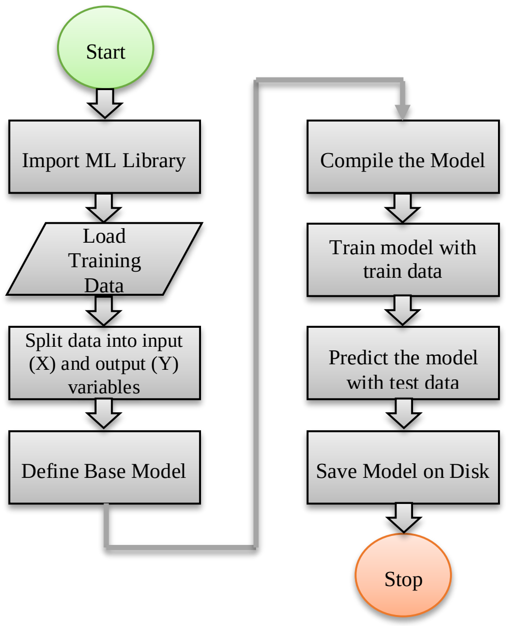
Start
Import ML Library
Load Training Data
Split data into input (X) and output (Y) variables
Define Base Model
Compile the Model
Train model with train data
Predict the model with test data
Save Model on Disk
Stop

Figure 2. Flowchart of Model Making

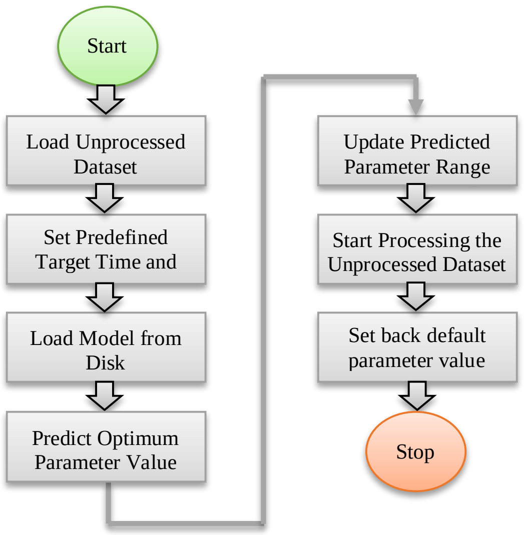
Start
Load Unprocessed Dataset
Set Predefined Target Time and
Load Model from Disk
Predict Optimum Parameter Value
Update Predicted Parameter Range
Start Processing the Unprocessed Dataset
Set back default parameter value
Stop

Figure 3. Flowchart of Prediction

### 4.4. Linear Regression (LR)

LR is one in all the foremost ordinarily used strategies for prediction. the method is precisely specified by an equation [23-27]:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} + \varepsilon_i \qquad (1)$$

where $Y_i$ is the output in the ith trail with i = 1,…,n, where n denoted as the trial size, the values $X_{i1}, X_{i2}, \ldots, X_{ij}$ is the observed value of the jth of p, j=0,…,p independent variables related with the ith output, the non-observable random variables $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ are random error term with E {ε} = 0 and variance 2 $\sigma^2\{\varepsilon_i\} = \sigma^2$ and $\beta_j$ are unidentified parameters to be assessed. The procedure can be demonstrated by:

$$\widehat{Y_i} = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} \qquad (2)$$

Different type of methods subsist to get the regression coefficient, $\beta_j$, the most well-known approach is "ordinary least squares" (OLS). By using OLS, the regression coefficient approximations are completed by expressing the measurements in the matrix form, for expediency, (*p* +1) is defined as p′.

$$\underset{(n*1)}{\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}} = \underset{(n*p')}{\begin{bmatrix} 1 & X_{11} & X_{12} & X_{13} & \cdots & X_{1p} \\ 1 & X_{21} & X_{22} & X_{23} & \cdots & X_{2p} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & X_{n2} & X_{n3} & \cdots & X_{np} \end{bmatrix}} * \underset{(p'*1)}{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}} + \underset{(n*1)}{\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}} \qquad (3)$$

The OLS yields to equation (4), which gives the least squares estimate β^ of the parameter set.

$$\hat{\beta} = (X^T X)^{-1} \; X^T Y \qquad (4)$$

To calculate the coefficient of the regression model, it is necessary to weigh the model's validity. In order to examine this, we have employed commonly used $r^2$ fit. The quantity of $r^2$ provides the degree for which the linear relationship among variables and a set of predictors are able to justify the variance in the variable. In other words, $r^2$ demonstrates the total proportion of variation in y which is described by the fitted model. For example, 0.9 means that the variation of 90% can be expressed by the LR model. The $r^2$ is calculated as follows

$$r^2 = \frac{SSR}{SST} = \frac{\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} \tag{5}$$

where The Sum of Squares Regression (SSR), The Total Sum of Squares (SST), $\hat{Y}_i$ is the estimated value for $Y_i$ , i.e., $\hat{Y}_i = \beta^T X_i$ · $r^2$ is always between 0 and 1. We utilized to evaluate whether the statement in which the linear model can be fitted historic data was effective or not. As we know that a regression model adopts that the quantity errors are independent and Gaussian. Thus, it is projected that the fragments are typically distributed [28-29].

### 4.5. Model development

We develop this linear regression model by using Keras, Tensorflow and Pycharm. Keras is a library of Tensorflow. In development, the required machine learning libraries from Keras are imported. The train and test data are loaded and kept in X_train, Y_train, and X_test variables respectively. X_train contains two training values which are data size and execution time which are collected manually by parameter tuning. Likewise, X_test variable holds the test data size and execution time. The train and test dataset are loaded into the model. After that, the base model is compiled. Then X_train and Y_train data are fitted. After that, the base model predicts the accuracy of X_test data. The accuracy, and loss are printed for analysis (Figure 4). The model accuracy is 95.7% for training data and 94.3% for testing. After satisfactory accuracy we saved the model on disk for prediction (Figure 5). One model is saved for one parameter. Therefore, it has altogether 5 models that were constructed by changing Y_train with five different parameters.

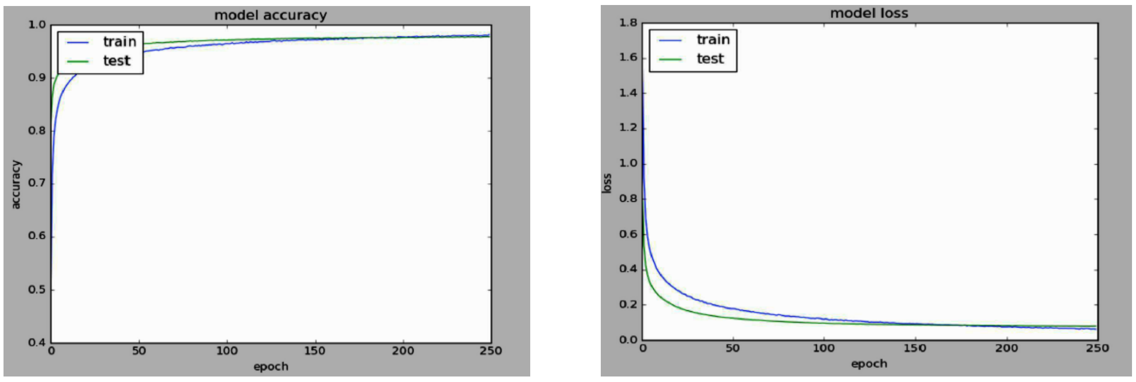

Figure 4. Model Accuracy and Loss in Train and Test Cases

driver_cores_model.h5

driver_memory_model.h5

executor_cores_model.h5

executor_memory_model.h5

reducer_maxSizeInFlight_mode

Figure 5. Generated Machine Learning Model

### 4.6. Prediction

For prediction of optimum parameter values using the stored models, a prediction algorithm is developed. For prediction, machine learning libraries are imported and the desired dataset is loaded. After that, two inputs (dataset size and predefined target time) are given as argument values. Then, the stored model is loaded and input data is fitted into the model to predict the optimum parameter value. The Optimum parameter value is predicted based on input data (dataset size and predefined target time) using the stored model (Figure 6). In this process, only one parameter value is predicted. Therefore, the same process is to be repeated for the remaining parameters.

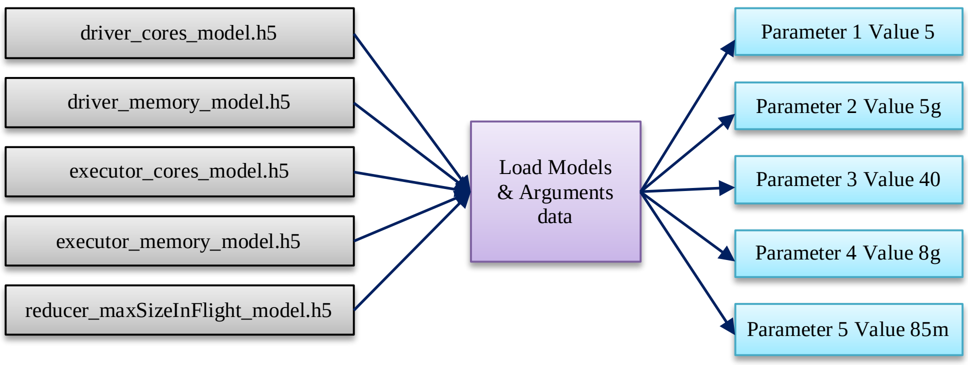

Figure 6. Prediction of Parameter Value Using Stored Model and Arguments

Once the optimum parameter value is obtained, the corresponding default value in the Spark system is updated with this value (Figure 7). Then, the Spark system starts processing the given dataset using the predicted optimum parameter values. After completing the process, another function is developed for resetting the parameter to its default value in Spark.

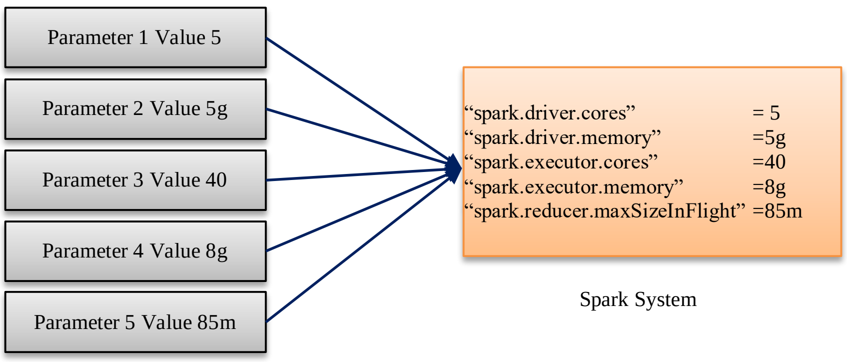

Figure 7. Predicted Optimum Parameter Value Updated in Spark

### 4.7. Test Bed

The novel approaches have been implemented on a testbed comprising of Dell PowerEdge R720 server. The server is furnished with Xeon(R) CPU E5 v2 @ 2.6Ghz 16 core processor 32 GB PC3 memory, Intel(R). Ubuntu Linus version 17.10 is used by the server with Hadoop 2.8.1. The novel approach can be run either in an independent system or on a virtual machine (VM). As listed in Table 2, the wordcount job was run in the Spark system for five different datasets which are 70, 120, 170,220, and 270 GBs. The datasets were chosen from PUMA benchmark. In Table 1 five predominant parameter configuration is displayed.

Table 2. Selected datasets for this work

| Datasets Size | Benchmark | Spark Program |
|---|---|---|
| 70 GB | | |
| 120 GB | | |
| 170 GB | Puma Benchmark | Wordcount |
| 220 GB | | |
| 270 GB | | |

## 5. BLOCK DIAGRAM

Training block gathers training data from a dataset and formulates these data for model creation. The model generation block obtains this data and creates the model according to the definition of model LR and stored the model on the disk in the predefined location for later use. The block named 'model assessment' is used to evaluate and test the saved model. 'Predicted parameter value' is used to get output that is the parameter range as a real value. The update block in the Spark system is responsible for receiving and updating real values in the Spark system. The block diagram of approach is illustrated in Figure 8.

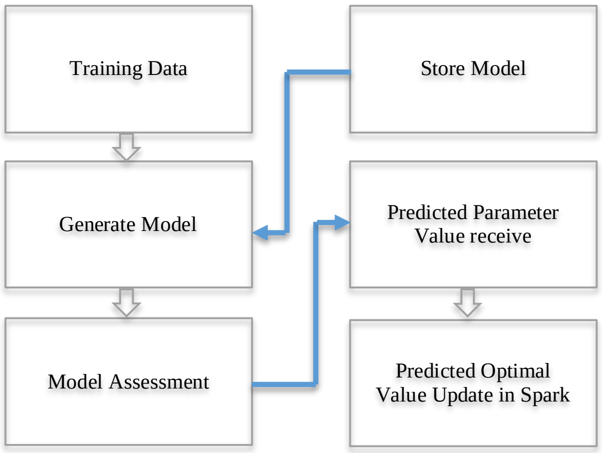
Training Data
Store Model
Generate Model
Predicted Parameter Value receive
Model Assessment
Predicted Optimal Value Update in Spark

Figure 8. The block diagram of this approach

## 6. RESULT DISCUSSION

In this part consist of the novel approach efficiency, capability and system performance speedup.

### 6.1. Efficiency of Novel Approach

Figure 9 illustrations wordcount job execution time with the novel approach and default configuration for a variety of dataset sizes. It is observed that the execution times of wordcount are significantly lower compared to default parameter configuration which is independent of dataset size such the range is from of 70 to 270 GB.

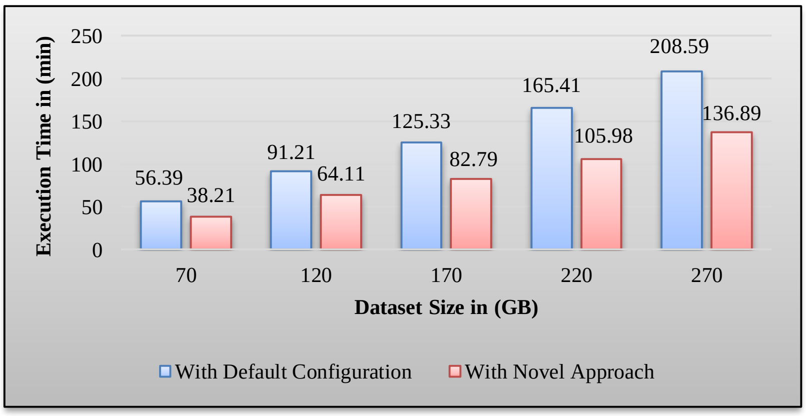

Figure 9. The figure shows a comparison between novel approach and default configuration

### 6.2. Self-tuning capability and execution time speed-up

To assess novel approach ability of self-tune, the Spark parameters as per the different of input dataset size, we run a Spark wordcount programme for five datasets such as (70, 120, 170, 220 and 270 GB) with default parameter value and with our developed novel approach. Table 3 presents the execution times for each of the 5 input dataset and the corresponding predicted optimum parameter value. It can be observed from Table 3, to process mention datasets, the default parameter configuration of Spark takes 56.39, 91.21, 125.33, 165.41 and 208.59 minutes for dataset sizes 70, 120, 170, 220 and 270 GB respectively. But our novel approach takes 38.21, 64.11, 82.79, 105.98 and 136.89 minutes accordingly. The results show (Table 4) that independent of the dataset size and the execution times obtained with the novel approach are significantly lower than default configuration.

Table 3. Predicted optimum parameter value

| Parameters Name | Parameter Default Value | Parameter Range Value | Predicted Parameter Value for 50GB | Predicted Parameter Value for 100GB | Predicted Parameter Value for 150GB | Predicted Parameter Value for 200GB | Predicted Parameter Value for 250GB |
|---|---|---|---|---|---|---|---|
| “spark.driver.cores” | 1 | 1-8 | 2 | 3 | 6 | 6 | 6 |
| “spark.driver.memory” | 1g | 1g-4g | 3g | 3g | 4g | 4g | 6g |
| “spark.executor.cores” | 1 | 10-40 | 20 | 30 | 30 | 35 | 40 |
| “spark.executor.memory” | 1g | 2g-8g | 3g | 3g | 5g | 5g | 6g |
| “Spark.reducer.maxSizeInFlight” | 48m | 24m-96m | 48m | 60m | 60m | 70m | 80m |

Table 4. Time Saved

|  | Executed with default Configuration | Executed with ASSPM system | Time Saved |
|---|---|---|---|
| Data Size | Execution Time (Min) | Execution Time (Min) | In Min |
| 70 GB | 56.39 | 38.21 | 18.18 |
| 120 GB | 91.21 | 64.11 | 27.1 |
| 170 GB | 125.33 | 82.79 | 42.54 |
| 220 GB | 165.41 | 105.98 | 59.43 |
| 270 GB | 208.59 | 136.89 | 71.7 |

## 7. CONCLUSION

A novel approach is presented in this paper, for self-tuning configuration of Spark parameter to increase its performance for big data processing. Our developed approach estimates the optimal range for five nominated parameters and updates Apache Spark beforehand the start of processing. The method was applied on Dell PowerEdge R720 server using different sizes of datasets. The experimented result shows, typical performance is increased 33% related to the default configuration. The improvement is noticed with the increase of dataset size. For selecting more suitable parameters utilizing better servers we are still doing research.

## REFERENCES

[1] Profile U S, “The Digital Universein 2020: Big Data,” *Bigger Digital Shadows, and Biggest Growth in the Far East-United States*, pp. 1-7, 2013.

[2] Anagnostopoulos I., *et al.*, “Handling big data: research challenges and future directions,” *J. Supercomput.*, vol. 72, pp. 1494-516, 2016.

[3] McKinsey and Company, “Big data: The next frontier for innovation, competition, and productivity,” *McKinsey Glob. Inst.*, vol. 156, 2011.

[4] Bhattacharya A. and Bhatnagar S., “Big Data and Apache Spark : A Review,” pp. 206-10, 2016.

[5] Kaur I., *et al.*, “Research Paper on Big Data and Hadoop,” vol. 8491, pp. 50-3, 2016.

[6] Rahman M. A., *et al.*, “A Survey of Machine Learning Techniques for Self-tuning Hadoop Performance,” *Int. J. Electr. Comput. Eng.*, vol. 8, pp. 1854, 2018.

[7] V. D. Veen J. S., *et al.*, “Dynamically scaling apache storm for the analysis of streaming data,” *Proc. - 2015 IEEE 1st Int. Conf. Big Data Comput. Serv. Appl. BigDataService*, pp. 154-61, 2015.

[8] Drabas T. and Lee D., “Learning PySpark,” vol. 273, 2017.

[9] Wang G., *et al.*, “A Novel Method for Tuning Configuration Parameters of Spark based on Machine Learning,” 2016.

[10] Herodotou H., *et al.*, “Starfish: {A} Self-tuning System for Big Data Analytics {CIDR},” *2011 Fifth Bienn. Conf. Innov. Data Syst. Res. Asilomar, CA, USA, Online Proc.*, pp. 261-72, 2011.

[11] Gupta A., *et al.*, “Big Data Analysis Framework Using Apache Spark and Deep Learning.”

[12] Jonnalagadda V. S., *et al.*, “A Review Study of Apache Spark in Big Data Processing,” vol. 4, pp. 93-8, 2016.

[13] Karau H., *et al.*, “Learning.”

[14] Parsola J, *et al.*, “Post Event Investigation of Multi-stream Video Data Utilizing Hadoop Cluster,” *Int. J. Electr. Comput. Eng.*, vol. 8, pp. 5089, 2018.
[15] Lama P. and Zhou X., “AROMA: Automated Resource Allocation and Configuration of MapReduce Environment in the Cloud,” *Proc. 9th Int. Conf. Auton. Comput. - ICAC’12*, vol. 63, 2012.
[16] Zaharia M., *et al.*, “Improving MapReduce Performance in Heterogeneous Environments,” pp. 29-42.
[17] Wu D. and Gokhale A., “A self-tuning system based on application profiling and performance analysis for optimizing hadoop mapreduce cluster configuration,” *20th Annu. Int. Conf. High Perform. Comput. HiPC*, pp. 89-98, 2013.
[18] Zhang R., *et al.*, “Finding the Big Data Sweet Spot : Towards Automatically Recommending Configurations for Hadoop Clusters on Docker Containers,” pp. 365-8, 2015.
[19] Wang K. and Khan M. M. H., “Performance prediction for apache spark platform,” *Proc. - 2015 IEEE 17th Int. Conf. High Perform. Comput. Commun. 2015 IEEE 7th Int. Symp. Cybersp. Saf. Secur. 2015 IEEE 12th Int. Conf. Embed. Softw. Syst. H*, pp. 166-73, 2015
[20] Yigitbasi N., *et al.*, “Towards machine learning-based auto-tuning of MapReduce,” *Proc. - IEEE Comput. Soc. Annu. Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst. MASCOTS*, pp. 11-20, 2013.
[21] Gounaris A. and Torres J. “A Methodology for Spark Parameter Tuning Big Data Res,” vol. 11, pp. 22-32, 2018.
[22] Angelov P., *et al.*, *Conference I and Data B 2016 Advances in Big Data.*
[23] Sunthornjittanon S., “Linear Regression Analysis on Net Income of an Agrochemical Company in Thailand,” 2015.
[24] Gustafsson A. and Wogenius S., “Modelling Apartment Prices with the Multiple Linear Regression Model,” 2014.
[25] Fallis A., “A Multiple Linear Regression Model to Predict the Student’s Final Grade in a Mathematics Class,” *J. Chem. Inf. Model*, vol. 53, pp. 1689-99, 2013.
[26] Cook R. D. and Weisberg S., “Simple Linear Regression,” pp. 97-138, 2008.
[27] Joseph P. J., *et al.*, “Construction and Use of Linear Regression Models for Processor Performance Analysis,” *Twelfth Int. Symp. High-Performance Comput. Archit.*, pp. 99-108, 2006.
[28] Baran M. E., *et al.*, “Load estimation for load monitoring at distribution substations,” *IEEE Trans. Power Syst.*, vol. 20, pp. 164-70, 2005.
[29] Lim H. L. and Brown R. H., “Gas Load Forecasting Model Input Factor Identification Using A Genetic Algorithm,” *IEEE*, pp. 670-3, 2001.

## BIOGRAPHIES OF AUTHORS

Md. Armanur Rahman received the B.Sc. degree in computer science and engineering from Asian University of Bangladesh (AUB) in 2010. He is currently working toward the MEngSc degree at the Multimedia University (MMU), Malaysia. His research interest include performance optimization of big data system, data mining, machine learning and image processing.

Abid Hossen is a Serving as Vice President at Information Technology division of National Bank Limited of Bangladesh. He received a B.Sc. In Computer Science and Engineering from Khulna University (KU) and pursing his M.Sc in Industrial and Production Engineering from Bangladesh University of Engineering and Technology (BUET). During his 18 years carrier in IT of different bank, he implemented different IT project successfully. His area of interest is IT Security, Big data, Artificial Intelligence and Cloud Computing.

Dr. 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.

Dr.Chinthakunta Venkata Seshaiah received his Bachelor of Engineering (B.E.) Degree in Electrical Engineering from S.V. University, Andhra Pradesh, India, in the year 1964.He received Master of Engineering (M.E) degree in High Voltage Engineering from Indian Institute of Science, Bangalore in 1966. He received his Ph.D. degree in Electrical Engineering (in the area of Power Systems) in 1976 from I.I.T. Madras. Later, he worked in the same institute till 2005. He was appointed as Professor of Electrical Engineering in Jan.1993. In 2006, he joined the Faculty of Engineering and Technology, Multimedia University (Melaka) Malaysia and is with them presently as Associate Professor. His research interests are in the areas of Electrical Power Systems, High Voltage Engineering and Instrumentation,Power Electronics and its application to green technology solutions,Electic Power quality improvement and Electrical energy conservation, Power efficient devices and Big data analytics.

Dr.T.Bhuvaneswari is a Lecturer in the Faculty of Engineering and Technology, Multimedia University (MMU), Melaka. She obtained her PhD in Electronics Engineering from Multimedia University in 2013. She earned her Master of Engineering in Applied Electronics with Distinction from Bharathiar University, India in 2001 and Bachelor of Engineering in Electrical and Electronics Engineering (First class) from Bharathiar University, India in 1998.She has 22 years of overall teaching experience.Her research interests are digital system design, VLSI, FPGA, Solar power controller design and bioinformatics.

Aziza Sultana received the B.Sc. degree in computer science and engineering from Dhaka International University (DIU) in 2016. She is currently searching an oopertunity to continue her higher study. Her research interest include performance optimization of big data system, data mining, machine learning and image processing.