Science Arena Publications
Specialty Journal of Electronic and Computer Sciences
ISSN: 2412-7485
Available online at www.sciarena.com
2018, Vol, 4 (2): 1-13

Provide a Distributed and Scalable Method for the Intrusion Detection in Computer Networks

Elahe Mohseni[1*], Alireza Bagheri[2], Sam Jabbehdari[3]

[1]Department of Software Engineering, Islamic Azad University, North Tehran Branch, Tehran, Iran,
[2]Assistant Professor, Department of Software, Amir Kabir University of Technology, Tehran,
[3]Associate Professor, Department of Software, Islamic Azad University, North Tehran Branch, Tehran.

***Corresponding Author**

Abstract: *The big data security and its management are significantly important given the increasing volume of data in computer networks. In this regard, data mining algorithms have been introduced as the applicable data analysis tools. Classification algorithms are important techniques in data mining. Decision tree algorithms have many applications in intrusion detection problems due to producing classified results in a meaningful and tree structure. Given the importance of intrusion detection velocity in networks, the parallel processing of classification algorithms on a big data context has been a challenging issue. In this research implemented a distributed and scalable method based on C5.0 decision tree for the intrusion detection problem using a new and advanced Spark framework in the field of data processing and used the most important feature selection to improve the system efficiency. Due to the use this framework and its high ability in in-memory processing. Finally, the proposed algorithm was evaluated using the standard KDDCUP99 dataset. The evaluation results indicated the scalability and high speed of the proposed algorithm.*

Keywords: *Intrusion Detection, Big Data, Data Mining, Decision Tree, Parallel Processing, Spark.*

INTRODUCTION

Due to the increasing use of computers for research in research centers, universities and businesses, the need for information security and faster processing has increased and become a basic need. Therefore, the parallel processing and the use of intrusion detection systems based on data mining algorithms play major roles in fulfilling this need. Information security of computer networks has become a critical issue (Swamy and Lakshmi, 2012).

The unexpected growth of the global data has created a concept called the big data in recent years; hence, most data analysis methods and tools have been changed to be adapted to size, velocity, and variety of today's data and deal with challenges of the present era. Big data processing requires the power of processing a large number of machines together and parallel implementation of algorithms on different clusters to increase data processing velocity.

In this research, the distributed classification algorithm of C5.0 decision tree was implemented to increase the algorithm velocity because the velocity is an important parameter in the intrusion detection. There was a fundamental challenge in providing this algorithm, so that the implementation of a decision tree algorithm in the big data was not optimal since an increase in features enhanced the number of exponential tree production situations. Therefore, the feature selection algorithm was used to select the most important

features for improving the efficiency and increasing the precision of decision tree algorithm. To evaluate the efficiency and accuracy of the proposed algorithm, it was tested and evaluated with standard criteria in most of similar works as well as the standard dataset KDDCUP99 in real terms. Results indicated the scalability, and increased velocity and precision of the intrusion detection algorithm.
In the second section of research, conducted studies on data mining and processing of big data are presented, and the section provided the proposed algorithm. In the fourth section, the proposed algorithm was implemented and evaluated, and finally, the fifth section presented the summary, conclusion and future work.

Research Background

The intrusion detection began in 1980 after a research on methods of improving the security of computer systems and monitoring customer sites (Anderson, 1980). Since then, various techniques have been introduced to build intrusion detection systems one of which is the use of data mining algorithms.

- In 2012, a classification model was introduced for abnormal behavior and signature-based intrusion detection based on decision tree algorithms. The efficiency of decision tree algorithms ID3, C4.5, and C5.0 was evaluated using KDDCUP99 datasets in two parameters, the detection rate and false positive. According to results, C5.0 decision tree algorithm had a better performance than other algorithms in the intrusion detection (Manish and Hanumanthappa, 2013).
- In 2014, studies were conducted on decision tree algorithm, Spark and MapReduce framework indicating that the MapReduce framework was inappropriate for repetitive algorithms, while the Spark-based memory framework is suitable for repetitive and interactive algorithms. In the present paper, the decision tree algorithm C4.5 is applied and evaluated on MapReduce and Spark framework. The evaluation results indicated better efficiency of developed algorithm in a small volume of data on Spark framework than MapReduce framework (Wang et al., 2014).
- In 2016, a big dataset was processed using decision tree algorithms and three popular classification tools, Weak, MOA[1] and Spark Mlib that ran on the Hadoop framework. Results of the comparison indicated that Weak tool was not a suitable classification for large volumes of data if Spark Mlib accurately and quickly classified large volumes of data. In general, this tool represents a high scalability for classification since increasing the time is constant compared to the increment of samples (Wisesa et al., 2016).
- In 2016, five machine learning algorithms namely logistic regression, Support vector machines, Random Forests, Gradient Boosted Decision Trees, and Naive Bayes were implemented on SPARC framework; and algorithms were evaluated in terms of education time, prediction time, precision, and sensitivity using KDDCUP99 dataset. Results of evaluation indicated the precision and sensitivity of the random forest algorithm compared to other algorithms (Manish Kulariya, 2016).
- In 2016, a paper was published on how to select a feature to classify a large volume of network traffic within the Spark framework. Finally, the implementation results indicated that this feature selection method was an important step in the classification saving the model construction and classification time. Furthermore, it indicated that the combination of this method with the Spark computing framework had a better efficiency than the MapReduce model (Wang et al., 2016).

Suggested Method

In this research, C5.0 decision tree algorithm, which was introduced as the best intrusion detection algorithm (Abhaya et al., 2014), was distributed; and the best features were selected to optimize the accuracy and velocity of algorithm using the random forest algorithm.

[1] Massive Online Analysis (MOA)

In the proposed method, input data was separately given in parallel among system nodes and random forest algorithm was separately implemented on each node to select the most important features for constructing a final tree.

The decision tree algorithm was then separately implemented on the data of each system node and attacks are detected. In the proposed algorithm, solutions were proposed to increase the velocity of tree creation, and ultimately, the algorithm was evaluated using a standard KDDCUP99 dataset. The general procedure for the proposed algorithm can be expressed in four steps:

- **First step: Data segmentation**: The dataset is first equally distributed among system nodes to be processed separately and in parallel in the next steps. At this stage, the main task of this algorithm is to segment and distribute data effectively and equitably among nodes.
- **Second step: Data preprocessing**: This step aims to pre-process data and convert the nominal data to a numerical values in parallel on each part of RDD to prepare target data to build the tree and select features.
- **Third step: Feature selection**: This step aims to identify a subset of data with minimum possible size for features in order to provide the necessary and sufficient information for the purpose. Obviously, this subset is the goal of all algorithms and feature selection methods.
- **Fourth step: Algorithm Training and Evaluation**: In this step, the decision tree algorithm is implemented using selected features in the Spark framework and is trained using KDDCUP99 training dataset. The goal of this step is to reduce the tree build time, increase the precision, reduce the false detection rate, and run the parallel decision tree algorithm to detect the network intrusion. In this regard, decision tree algorithm parameters are determined in such a way that the tree is created in the shortest time and eventually an implemented algorithm is evaluated using the experimental KDDCUP99 dataset.

Algorithm Implementation

This section provides issues related to the implementation of the proposed algorithm for big data classification. We first examined the algorithm implementation method, applied libraries and details of implementing the proposed method.

The suggested algorithm was implemented on a cluster with 1 Master node and 4 Salve nodes by CentOS Linux operating system. The proposed method was implemented through five different Python Language Packages in big data, spark framework and machine learning as shown in Table 1:

Table 1: Applied Packages in Spark

Packages	Methods and classes	Description
Pyspark	SparkContext SparkConf RDD	Pyspark package is a Python API for spark framework with several classes. SparkContext Class is the main point of contact with the spark cluster applied to build RDD and distributed variables in a cluster.
Scikit-learn	Ensemble Feature_extraction Feature_selection	Scikit-learn package is a simple effective tool for data mining and data analysis that is used in a variety of domains, is build based on Scipy, Nippy and matplotlib modules and includes various methods for classifying, clustering, selecting and reducing features.
Pyspark.mllib	Pyspark.mllib. claassification pyspark. mllib. tree pyspark. mllib. util pyspark.mllib.regression	Mlib package is a machine learning library for Spark aiming to make machine learning easy and scalable and includes various tools for clustering and classifying algorithms, reduction algorithms, feature selection and pipeline operations.
Numpy	Array Objects Universal function Routing	The Numpy package is the main package for scientific calculations in Python which includes multi-dimensional arrays, advanced functions, combination tools of C and C++ and tools for use in linear algebra, Fourier transform and random features.
Pandas	Pd	Panda package is an Open Source tool with high and easy

	Pandas	function to use in data building and analysis in Python.

- **Data segmentation**

For parallel dataset processing, data should be segmented into equal parts and distributed among system nodes. Frameworks such as Spark and Hadoop usually work with distributed file systems and databases, and distributed databases. This segmentation is automatically performed. HDFS is one of these file systems (Gadeka, 2014), but since we installed Spark Standalone, then we used Spark features to distribute data among nodes.

A RDD is in fact a set of distributable objects which can be segmented into several parts and can be measured in different nodes in Spark. Any RDD includes objects of different languages such as Python, Java and Scala defined by users. RDD is created by users in two ways: loading an external dataset or with a set of distributed objects in the driver program. In this method, we first converted an input dataset to RDD, and then distributed them equally among system nodes to implement the parallel selection of the most important features in each part of data (Wang, 2014).

- **Data Preprocessing**

The first step before implementing the algorithm is to select features and algorithm of the pre-processing tree construction. The data conversion method was used since the standard KDDCUP99 dataset was applied as a reference dataset for training and testing in the proposed algorithm, and this dataset consisted of nominal and numerical data, so that we could make the use of data in the feature selection algorithm and tree building.

In this research, we used duplicated data cleaning of then conversion of nominal to numerical data by an indexing method. To this end, the following two steps were performed:

1) **Creating a list of unique data of nominal columns**
2) **Scanning each data of rows and comparing data with the created list:**
   2-1) If the data is available in the list, the index number of data is considered as the output and stores in that row for the nominal data.
   2-2) If the data is unavailable in the list, the list length is stored as the number of that data.

Implementing this algorithm for all rows of dataset, all nominal column data is converted to numerical; and the dataset is prepared for the next use.

- **Feature Selection**

In this step, the random forest algorithm is implemented in parallel on the data for each section, and finally the most important features are selected based on obtained weights for each feature. The random forest algorithm process is as follows.

- **Random forest algorithm**

Random forests are popular methods of machine learning due to their relatively good precision, reliability, and ease of use. Two methods of impurity reduction and average precision reduction were provided for the feature selection.

We selected the impurity reduction method in this research. On the basis of this method, nodes with the highest impurity reduction occur at the beginning of trees, and those with the least impurity at the bottom of trees. Therefore, it is possible to build a subset of the most important features by pruning sub-trees of a particular node. The random forest algorithm includes a number of decision trees and each tree node divides the dataset into two parts. During the tree training, it is possible to calculate how much weight impurity is reduced in a tree by each feature. The features can be averagely ranked based on this amount (Chen, 2016).

The amount of impurity refers to the optimality of any condition. Impurity rate is measured by impurity methods of Gini, Information gain, and entropy in the classification method; and by variance method in the regression tree method.

Details of algorithm of selecting the most important features are presented in the algorithm (1):

```
Procedure FeatureSelection
Input: Transformed KDDCUP99 T, attributes S
Output: List of importance feature
Begin
P ← convert T to float
L ← copy of label column of KDD CUP 99 dataset
    /* Convert Label to number */
for each Row of L
 if (L! = 'Normal.')
    L=1
 else
    L=0
end if
end for
t0 = time ()
R ← create random forest with Extratreesclassifier ()
tt = time () - t0
F ← Compute feature important for any feature
S ← Select feature importance
End
```

Algorithm 1: Selection of the most important features

- **Model training**

At this stage, C5.0 algorithm of decision tree is fully investigated and then the tree height is replaced with the the best values to create a decision tree with the highest efficiency and the least time to detect the intrusion of different tree parameters including the purification method.

- **C5.0 decision tree classification algorithm**

As mentioned earlier, the decision tree algorithm presents the output knowledge as a tree of different states. Generally, the designed decision tree is not unique for a training dataset; and different decision trees can be created based on a dataset. This algorithm is basically a greedy algorithm starting from the root node, and selecting an attribute for sample testing in each non-leaf node, and then the sample set is divided into several sample subsets according to the test result. Each sample subset forms a new node, and eventually the segmentation process is repeated until it reaches the end-specific conditions (Jain and Srivastava, 2013). As a result, the complexity of a tree is measurable using the criteria namely the total number of leaves, total number of nodes, total number of applied attributes, and the tree depth. Decision tree C5.0 substitutes for decision tree C4.5. Therefore, the velocity, precision, memory usage, and creation of smaller decision trees, and Boosting support are better than C4.5 tree. The information gain criterion is used to select features (Wang et al., 2014). It can be calculated by the following Equation (1):

- **Information gain calculation**

Information gain specifies which properly attributes segment the set of examples into the target class; hence, an attribute with higher information gain is selected as the tree root. Information gain of a feature refers to the amount of entropy reduction that is obtained by separating examples through this feature. In other words, the information gain (S, A) for a feature (A) towards a set of examples (S) is defined as follows (Manish Kulariya, 2016):

$$\text{Gain(S,A)= Entropy(S)} - \sum_{v\in \text{Values(A)}} \frac{|sv|}{|s|} * \text{Entropy}(S_v) \quad (1)$$

Where, Values(A) is the set of all features of A; and SV is a subset of S for which A has a value of V. In the definition above, the first term is the amount of data entropy which is calculated by the Equation 2. The second term is the expected entropy after the data separation. The entropy of the dataset is calculated by the following equation:

$$\text{Entropy(S)} = \sum_{i=1}^{m} p_i \, log_2(p_i) \quad (2)$$

Pi is the probability of an event belonging to the class Ci (Galathiya et al., 2012). If the set S includes positive or negative examples of a target concept, Entropy S of this Boolean category is defined as follows:

$$\text{Entropy(S)} = -P\oplus \log2\ P\oplus - P\ominus \log2\ P\ominus \quad (3)$$

P⊕ is the ratio of positive examples to the whole examples and P⊖ is the ratio of negative examples to the whole examples. Log0= 0 in entropy calculations.

Details of C5.0 decision tree training algorithm and its evaluation are presented in Algorithm (2).

As shown in Algorithm (2), the selected set of features from the algorithm (1) is given as input to the algorithm, and then 70% of data is considered as the training data. Since any feature in each record is separated by the quotation marks (","), features are separated using the function of Split () to put any feature in a column, and then TrainingData set is given to Trainigdecision () function as an input; and a tree is built using the DecisionTree.trainClassifier () function. Determining the parameters of DecisionTree. trainClassifier class is important for building a tree. NumClasses Parameter indicates the number of existing classes in the dataset. Here, the problem is converted to a two-class problem. Nominal features and their values are considered in CategoricalFeaturesInfo parameter. In the Impurity parameter, we choose a purification method in which Gini method is use in C5.0 tree algorithm. Considering different states for MaxDepth and MaxBins parameters, we consider the best state increasing the velocity and precision. Finally, the output is shown as a tree.

```
Procedure Train DecisionTree

Input: New data with feature selection T, attributes S

Output: Decision tree

(Trainingdata, Testdata) ← Split dataset T to 70% trainigdata, 30% testdata

Traningdata ← Split any row of training with "," to column

Testdata ← Split any row of test with "," to column

Trainigdecision (Traningdata , Label Column)

   Create labeledPoint (Labelclass, array of any row for training data)

   t0 = time ()
```

```
    Create Decision tree with trainClassifier Function
    tt = time () - t0
  return Decisiontree
End
```

Algorithm 2: Decision tree training

- **KDDCUP99 dataset**

KDDCUP99 dataset is a widely used standard datasets for evaluating intrusion detection systems. It was the first applied in the third competition on data mining tools and exploration of international knowledge. This dataset is a part of collected data by the Intrusion Detection Evaluation Program, DARPA 1998, that was collected by IST group from MIT University through a wide range of intrusion influence in a military network (Chen, 2016). This dataset has several types each of which has a unique feature, and it has a total of 41 network features and a class label. Educational Dataset, kddcup.data_10_percent, contains 10% of the total dataset, and KDDcup.corrected dataset is used as the experimental data (Gunes Kayacık et al., 2005).

Table 2: KDDCUP99 Datasets according to their types (Patel et al., 2014)

Dataset	DoS	Probe	u2r	r2l	Normal
" 10% KDD"	391458	4107	52	1126	97277
"Corrected KDD"	229853	4166	70	16347	60593
"Whole KDD"	3883370	41102	52	1126	972780

Model Evaluation

This section presents the proposed algorithm evaluation for big data classification. The generated knowledge at the model learning stage should be analyzed at the evaluation stage in order to determine its value, and subsequently determine the efficiency of learning algorithm of model. These criteria can be calculated both for the training dataset in the learning phase and the set of test records in the assessment phase. The confusion matrix is used to calculate the efficiency of classifier algorithm. Table (3) presents Matrix elements:

Table 3: Classifier Evaluation criteria

Real \ Prediction	Positive	Negative
Positive	TP	FN
Negative	FP	TN

- **True Negatives (TN):** It refers to the number of records with negative real class; and the classification algorithm truly recognizes them as negative.

$$TN = \frac{TN}{TN+FP} \quad (4)$$

- **True Positives (TP):** It refers to the number of records with positive real category; and the classification algorithm truly recognizes them as positive.

$$TP = \frac{TP}{TP+FN} \quad (5)$$

- **False Positives (FP):** It refers to the number of records with negative real category; and the classification algorithm falsely recognizes them as positive.

$$FP = \frac{FP}{TN+FP} \quad (6)$$

- **False negatives (FN):** It refers to the number of records with positive real category; and the classification algorithm falsely recognizes them as negative.

$$FN = \frac{FN}{TP+FN} \quad (7)$$

Other criteria are also used to evaluate a classifier, as follows:

- **Accuracy**: It is the most important criterion for determining the efficiency of a classification algorithm indicating the match of predictions of a model with the modeling reality. It is calculated by the Equation (8):

$$CA = \frac{TN+TP}{TN+FN+TP+FP} \quad (8)$$

- **Error Rate**: It is the opposite of classification accuracy criterion. Its lowest value is zero when we have the best efficiency; and on the contrary, its highest value is seen in the least efficiency:

$$ER = \frac{FN+FP}{TN+FN+TP+FP} = 1\text{-}CA \quad (9)$$

- **Precision**: It is basically based on the precision of predicting the classifier and indicates to what extent we can trust in the output.

$$Precision = \frac{TP}{TP+FP} \quad (10)$$

- **Recall**: It is a complementary parameter of precision and is equal to the total number of records with a desired label (Rastgari & Sivandian, 2014; Wang et al., 2014).

$$Recall = \frac{TP}{TP+FN} \quad (11)$$

- **F-measure**: It is a combination of precision and recall for system efficiency evaluation.

$$F\text{-}measure = \frac{2*Recall*Percision}{Recall+Percision} \quad (12)$$

Algorithm (3) provides details of evaluated C5.0 decision tree algorithm using experimental datasets and various functions in Python.

Procedure DecisionTree (T, S)

Input: 30% of New data with feature selection T, attributes S

Output: Result Test

Begin

Testdata ← Split any row of test with "," to column

Call Function Testdecision (Testdata)

end

Testdecision (Testdata, DecisionTree)

Predications ← Test Decisiontree with Testdata & Predict Function

CompareLabel ← Compare predicted Label & actual Label with Zip ()

/* Compute the parameters of Confusion matrix */
TP ← Count the predicted Label that are equal the actual Label & Label is Normal

TN ← Count the predicted Label that are equal the actual Label & actual

Label is Attack

FP ← Count the predicted Label that are opposite of actual Label & actual

Label is Normal

TP ← Count the predicted Label that are opposite of the actual Label &

Label is Attack

Compute Accuracy, Precision, Recall, Error_rate, F-measure, TPR, TNR, FPR, FNR return Result of test
End Function

Algorithm 3: Decision tree evaluation

Table 4 presents evaluation results in various evaluation parameters.

Table 4: Comparison of proposed and original algorithms

	F-Measure	Recall	Precision	FP Rate	TP Rate	Accuracy
Original algorithm	0.993	0.987	0.999	0.0002	0.987	99.729
Proposed algorithm	0.995	0.996	0.996	0.0001	0.994	99.804

As shown in Table (4), the proposed to has higher precision than the original algorithm; and the false Positive (FP) rate is reduced in the proposed algorithm; hence, the proposed algorithm has better performance than the original algorithm. Results of comparing two algorithms are presented in Figure 1.

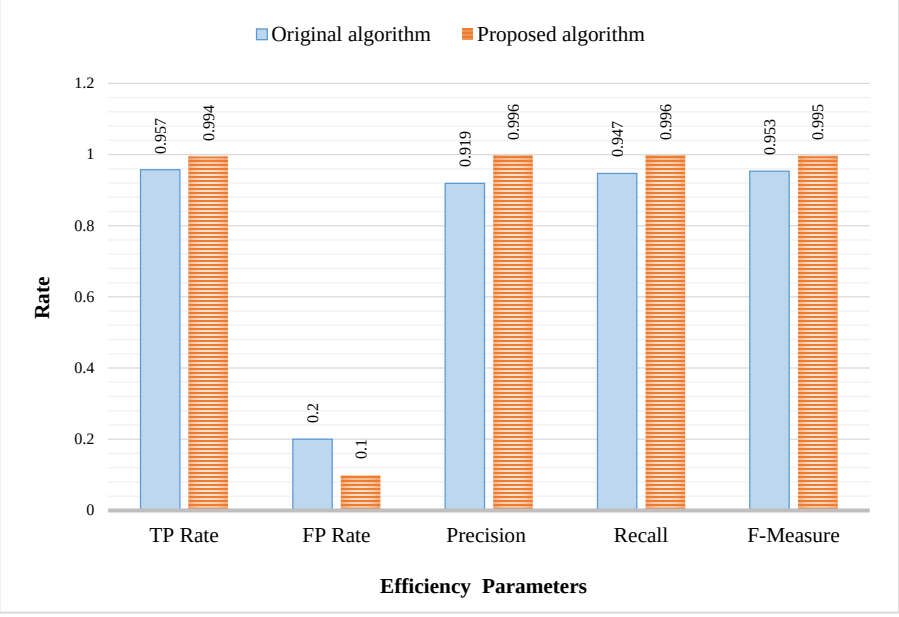

Figure 1: Comparison of original and proposed algorithms

Table 5 presents results of evaluating the proposed method in four different attacks types. According to the results, the proposed method fully recognizes a denial-of-service attack (DoS attack):

Table 5: Detection of attacks with the proposed method

	Recall	Precision	F-measure
Normal	99.80%	99.80%	99.80%
Dos	100%	100%	100%
PRB	98.30%	98.50%	98.40%
U2R	89.90%	94.30%	92.00%

- **Scale**

Scalability is a parameter that is used to check the efficiency of an intrusion detection system. In fact, the scalability is the ability of a system to adapt to the increased demand for data processing. From a wider perspective, large information systems are divided into two types of scalability:

Scale Out: Scale Out includes the distribution of workloads among a number of servers, in which a large number of independent machines sit together to improve the processing ability. In general, several operating systems are running on different devices.

Scale Up: Scale Up covers installing more processors, longer memory, and faster hardware on a separate server that typically includes an instance of the operating system (Singh and Reddy, 2014).

- **Speedup**

Suppose that a parallel algorithm, which works with P processor, ends working at the time Tp(n). In the Equation (13), T*(n) is the optimal time of a serial algorithm for problem solving and indicates the advantage of increasing the speed of a parallel algorithm to the best serial algorithm.

$$Sp(n) = \frac{T*(n)}{Tp(n)} \quad (13)$$

However, since the optimal time of a serial algorithm is unknown, other definitions are provided for T*(n) for instance, the necessary time for a processor to run a parallel algorithm. If Sp is the speedup for a p processor, an ideal or a linear speed is increased when Sp= p. In other words, when we perform an algorithm with increase linear speed, doubling the number of processors doubles the speed. However, it is usually difficult to achieve an increased linear speed in increased number of processors. To calculate the speedup, T_{old} is considered as the time of classification on a processor; and T_{new} as the classification time. As shown in Figure 2, the speedup occurs almost linearly until four processors. In general, the duration of operation in states one, two and four of processor is very long, and thus a delay of few minutes has a very little impact on the speedup.

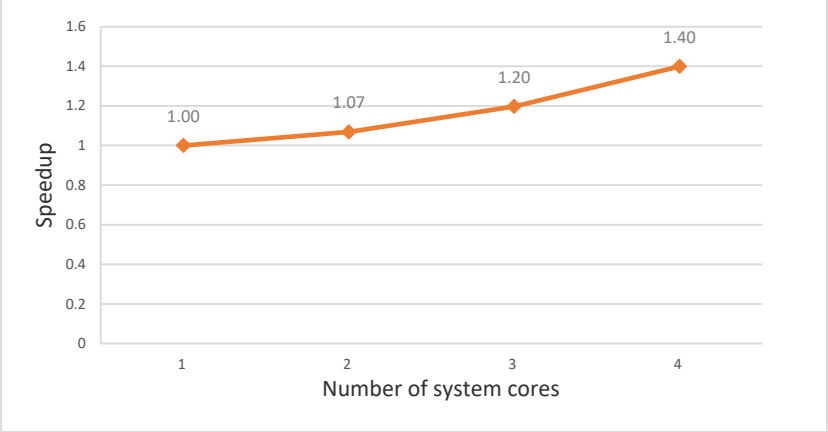

Figure 2: Results of speedup

- **Scaleup**

The scaleup criterion evaluates the growth ability of an algorithm against system sizes and the dataset. In other words, the scaleup can be defined as follows: A system with a size of m times higher can do a task of m times higher with a same execution time as the initial system. More precisely, this criterion can be obtained by Equation (14):

$$\text{Scaleup(DC,m)} = \frac{T(DC,1)}{T(DC\times m,m)} \quad (14)$$

Where, T (DC, 1) is equal to the execution time of a program on the initial dataset with a processor; and T (DC× m, m) is equal to the execution time of a program on the dataset m equated with m processor. The more this value is low, the less the scale is increased. KDDCUP99 dataset was used to calculate this criterion; and the algorithm was implemented on a dataset of 125, 250, 375, and 500 thousand records with four, three, two, and one processors. Results of this criterion are shown in Figure (3). According to results, the proposed algorithm has good scalability in the big data.

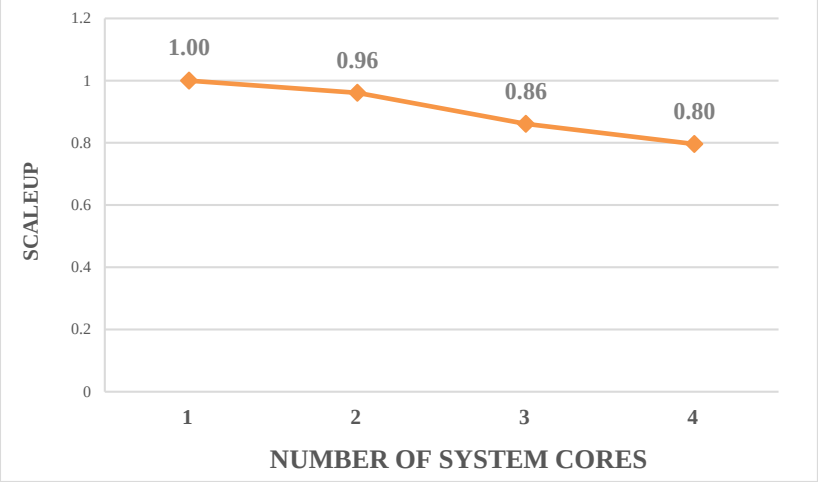

Figure 3: Scaleup increase results

Summary and Conclusion

Classification is the most common and practical issue in data mining; and various methods have been proposed for this issue. Decision tree algorithm is a set of these methods with better efficiency in intrusion detection problems than other algorithms as considered in this project.

Decision tree algorithm problem is that it fails to run with the increasing amount of data; and the tree size increases in intensified features, and thus it cannot be simply analyzed. Therefore, the focus of current research is not only on classification, but also on parallel and scalable algorithms, which are capable of working with big data, in most data mining issues.

In this research, we proposed a parallel and scalable decision tree algorithm within the Spark framework, and used feature selection algorithms to optimize the algorithm and the Random Forest to select the most important features. The main idea of this algorithm is to adapt the decision tree algorithm with feature selection method in spark framework.

So that the input dataset was processed in parallel and the most important features were selected for tree building. The size of this selected dataset would be much less than size of the original dataset; thus, the algorithm construction will have greater speed.

We evaluated the proposed algorithm by a variety of methods with high-volume datasets and high dimensions. In evaluating the speedup, the algorithm had high efficiency, but there was a slight low speedup due to the low volume of input data per processor and the increase in numbers of processors. It was normal, and can be solved by doing the test with larger datasets. Evaluations on scalability and increased size also indicated good performance of algorithm in the big data.

- **Future work**

Some issues, which may be considered and investigated in future studies, are as follows.

- **Online data flow processing**

Data flow processing technology is not an emerging technology in the field of data analysis, but the increased data volume, significant advances and changes in this field, increasing the efficiency of MapReduce model and the emergence of highly innovative frameworks such as Spark have made this technology mature. Given the large volume of network traffic, there is a need for online data analysis for instantaneous and fast detection of intrusions and preventing them.

- **Combination of Intrusion Detection Software and Spark Framework**

The problem of intrusion detection software is that it fails to run on a system and cannot be in parallel. Due to the increased data volumes, this is a major challenge in the intrusion detection problem; hence, combining the intrusion detection software with the big data framework can increase the precision and efficiency of the intrusion detection system.

References

1. A. S. Galathiya, A. P. Ganatra and C. K. Bhens, "Improved Decision Tree Induction Algorithm with Feature Selection, Cross Validation, Model Complexity and Reduced Error Pruning," International Journal of Computer Science and Information Technologies, vol. 3, pp. 3427-3431, 2012.
2. Abhaya, K. Kumar, R. Jha and S. Afroz, "Data Mining Techniques for Intrusion Detection: A Review," International Journal of Advanced Research in Computer and Communication Engineering, vol. 3, no. 6, pp. 6938-6942, June 2014.
3. D. Singh and Ch. K Reddy, "A survey on platforms for big data analytics," Journal of Big Data, 2014.
4. G. Kayacık, A. Nur Zincir-Heywood and Malcolm I. Heywood, "A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets", 2005.
5. H. A. Wisesa, A. Ma'sum and M. Petrus, "Processing big data with decision trees: A case study in large traffic data," Big Data and Information Security (IWBIS), International Workshop on, pp. 115-120, 2016.
6. H. Wang, B. Wu, S. Yang, B. Wang and L. Yang, "Research of Decision Tree on YARN Using MapReduce and Spark," IEEE, 2014.
7. Harshawardhan S. Bhosale, Devendra P. Gadekar, "A Review Paper on Big Data and Hadoop," International Journal of Scientific and Research Publications, vol. 4, no. 10, pp. 1-7, 2014.
8. J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng and K. Li "A Parallel Random Forest Algorithm for Big Data in a Spark Cloud Computing Environment," IEEE, 2016.
9. J. P. Anderson, "Computer Security Therat Monitoring and Surveilance," 1980.
10. J. Parekh and R. Pate, "An Intrusion Detection based on Data mining technique and its," International Journal of Modern Trends in Engineering, 2014.
11. J. Rastgari & Z. Sivandian, A Study of the Application of Genetic Algorithm in the Diagnosis of Influence of Computer Networks, Second National Conference on Sciences and Computer Engineering, 2014, pp. 1-8.

12. K. Abd Jalill, M. H. Kamarudin and M. N. Masrek, "Comparison of Machine Learning Algorithms Performance in Detecting Network," in International Conference on Networking and Information Technology, 2010.
13. K. Manish and Hanumanthappa, "Intrusion Detection System Using Decision Tree," IEEE, 2013.
14. K. Swamy and K. Vijaya Lakshmi, "Network Intrusion Detection Using Improved," International Journal of Computer Science and Information Technologies (IJCSIT), vol. 10, pp. 26-32, 2012.
15. N. Jain and V. Srivastava, "Data minig technques: A Survey paper," International Journal of Research in Engineering and Technology, pp. 4622-4626, 2013.
16. P. Saraf, R. Ranjan, G. P. Gupta, M. Kulariya, "Performance Analysis of Network Intrusion Detection Schemes using Apache Spark," International Conference on Communication and Signal Processing, IEEE, pp. 1973-1977, 2016.
17. Y. Wang, W. Ke and X. Tao, "A Feature Selection Method for Large-Scale Network," MDPI, pp. 2-11, 2016.