Vol.2013-MPS-94 No.11
2013/7/22

Implementation of Multiple Classifier System on MapReduce Framework for Intrusion Detection

Masataka Mizukoshi[1,†1,a)] Bando Shintaro[2,b)] Martin Schlueter[2,c)] Masaharu Munetomo[2,d)]

Abstract: Since the data volume from various facilities keeps growing rapidly in recent years, "big data" processing frameworks such as Hadoop have been developed as a scalable architecture to process large amount of data in cloud computing environment. We focus on intrusion detection problems which require large amount of data to be processed in order to detect malicious attacks. In this paper we discuss a Hadoop implementation of a multiple classifier system to enhance performances of the learning process in intrusion detection.

Keywords: Multiple Classifier System, Intrusion detection, Hadoop, Distributed computing, Big data

1. Introduction

Currently, the amount of data that has to be processed by many companies has reached Petabytes and is expected to keep growing in the future. Under these circumstances, it is critical that these large amounts of data be efficiently treated. This requires that useful information be gleaned from such data. To do this, various Machine Learning approaches are being proposed. This is also true in the field of Intrusion Detection Systems (IDS). Many machine learning approaches have been proposed in the field of IDS. These approaches include Support Vector Machines (SVM), Neural Networks (NN), and classifier systems (see [1]-[5]). IDSs need to to analyze large-scale data quickly and respond promptly to any new intrusion detected. However, if machine learning is applied to large scale data, time becomes a critical issue.Here, we suggest an early learning technique by executing multiple classifier systems simultaneously (parallel) on Hadoop MapReduce framework (see [8],[9]). Classifier systems are valid for a variety of input environments. A multiple classifier system extends this concept to facilitate the processing of big data. Recently, classifier systems have been studied extensively as a powerful way to learn large-scale data. In this paper, a proposed implementation of such a multiple classifier system on the Hadoop MapReduce Framework is presented. Hadoop is an open source software for large-scale distributed data processing that enables users to easily perform distributed processing (see [10]).

1 Information Processing Society of Japan, Chiyoda, Tokyo 101–0062, Japan
2 Graduate School of Information Science and Technology Hokkaido University Sapporo
Sapporo 060-0811, Japan
†1 Presently with Hokkaido University
a) m.mizukoshi@ist.hokudai.ac.jp
b) dontforget3-7@ec.hokudai.ac.jp
c) schlueter@midaco-solver.com
d) munetomo@iic.hokudai.ac.jp

2. Classifier systems

Classifier systems are the learning process of rule-based machine learning system. They are composed of a set of rules, called a classifier, which consists of IF/THEN statements. The method by which a classifier system learns is described below (Fig. 1).

2.1 Flow of learning in a Classifier System

Classifier systems can roughly be divided into five modules: i) action module, ii) renumeration assignment module, iii) discovery module, iv) effector module, and v) detector module. A classifier system first converts the input from the environment to adapt it for use by the system. In the action module, the system determines the classifier that should act on the input data. In the renumeration assignment module, the classifiers chosen in the action module return some rewards for the classifiers. In the discovery module, a Genetic Algorithm (GA) is executed in order to receive a better population of classifiers.

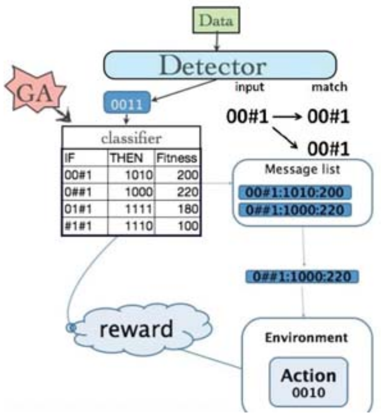

Fig. 1 Structure of a typical Classifier System

3. Flow of learning in a multiple classifier system

In a multiple classifier system (MCS), input data is distributed to each machine and a classifier system is executed by each machine. One classifier population is made by the classifier from which it is shifted to execution by each machine and is summarized in one place. Although learning big data is attained in a short time using this method, the accuracy of the learning depends on the composition of the population collected in one place. Thus, the method that is used to conduct the merge is very important. The efficiency of learning changes significantly according to which classifier is entered into a population and which classifier is removed.

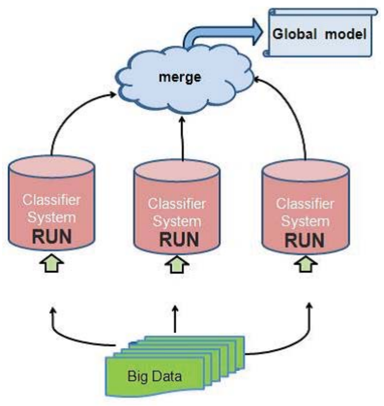

Fig. 2 Structure of a typical Multiple Classifier System

3.1 Merge technique

In the merge technique, classifiers, which are sent sequentially, are sorted by the system according to intensity and number of hashes (＃) in order to make a high-quality population. The decision as to whether the system leaves a classifier that has a high correspondence power or a classifier that has high intensity is dependent on the learning objective or contents. The performance of the population changes in accordance with the kind of population constituted. There is no absolute way that is good to constitute a population from any kind of technique since many methods have been proposed. Figure 2 ilustrates the system-wide flow in a typical MCS (Fig. 2).

4. Proposed implementation

4.1 Hadoop

Hadoop is an open source large-scale distributed processing software that uses the MapReduce Framework. Hadoop performs Terabyte to Petabyte levels of big data processing that ordinarily takes a lot of time and is not dependent on an expensive computer as the work is distributed over thousands of nodes comprising ordinary servers that can be obtained cheaply. Hadoop simplifies the construction of distributed processing programs. The typical layout of the Hadoop MapReduce Frameowork is depicted in Fig. 3.

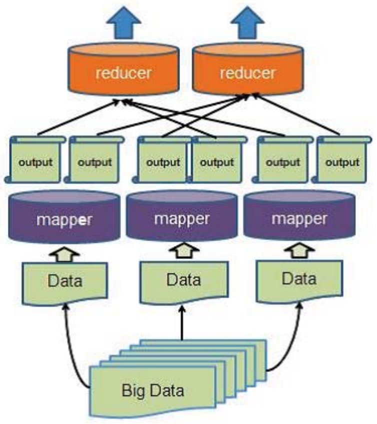

Fig. 3 Typical layout of the Hadoop MapReduce Framework

4.2 MapReduce Framework

A MapReduce Framework is divided into three phases: Map, Shuffle, and Reduce. By passing data to each phase using the value from the key/value pair, processing is performed by each phase. Users can carry out distributed processing by describing the processing in the Map and Reduce functions. In Map processing, the input data are received as key/value pairs and processing by analyzing the contents generates middle data in the form of key/value pairs. The generated data is gathered by key in Shuffle processing, and is sent to Reduce processing. In Reduce processing, the value processed for every key is generated and output as a result.

4.3 Mounting of the multiple classifier system on Hadoop

In Hadoop, classifier systems are individually mounted in Map processing and merge is performed in Reduce processing. The contents of mounting within each phase of processing is explained below.

4.3.1 Map processing

A classifier system is individually mounted in Map processing. A classifier is tagged by the kind of environment the classifier is matched to by setting this tag to key. The classifier is treated as value and outputted. The processesed classifier is outputted with a parameter such as intensity or others and is then sent to Shuffle processing.

4.3.2 Shuffle and Reduce processing

In Shuffle processing, data is classified according to the middle value generated by Shuffle processing for every tag, and the classified classifier is put into Reduce processing. In Reduce processing, merge is performed by each Reducer and a classifier population generated. In Hadoop, Reduce processing is also arranged in parallel by two or more machines. Therefore, if the number of Reducers increases,the computation time in the merging method is reduced.

A group forms the colony of the classifier, which reacts to similar inputs so it is not subject to or influenced by learning advance of classifiers with a far relation. With this method, learning about rare inputs from which learning is seldom made repeatedly can be performed efficiently.

5. Experiment

5.1 Object problem

In our experiment, a classifier system was built to study data for intrusion detection using the network communication data of KDD Cup 1999 Data (international data-mining contest). The contents of the data are the study data used by a teacher which is accompanied by data for result judging. The environment was set up by binary coding this data to 0 and 1.

5.2 Execution environment

One hundred and thirty virtual machines (VM) were launched on a CloudStack cloud platform and Hadoop version 1.0.4 mounted on them.

5.2.1 Key setup

An execution classifier was tagged according to the kind of input data it processes. Keys were setup according to attacks from 22 kinds of intrusion attack data. The data were divided according to intrusion data and groups made.

5.2.2 The merge method performed

In the classifier system using Hadoop, the fall of the merge speed by hypertrophy of a global model is avoidable by increasing the number of Reducers. Therefore, even if the group becomes very large, computation time is seldom influenced. The maximum global size was set at 500. The groups were setup such that an individual with a higher specialty nature (individuals with a small number of hashes, '#') might remain since it is highly possible to setup a maximum. In this experiment, classifiers were sorted using conditions 1 and 2:

1) Keep those classifiers with a low number of hashes, #'s, in the system.

2) If two classifiers have an identical number of hashes, #'s, keep the one with the higher intensity in the system.

6. Experimental results

The system was mounted and two or more Mappers started learning in the multiple classifier system. Learning with a simple classifier system was compared with learning distributed with the MCS using the method outlined in Section IV.

The number of nodes used by the Mapper was varied from 3 to 93 and execution time measured. The resulting execution time is depicted in Fig. 4. Block count was also changed in increments of 8 MB, 16 MB, 32 MB,... and so on in Hadoop, when distributing a file. Like the upper graph, it turned out that execution time decreased linearly.

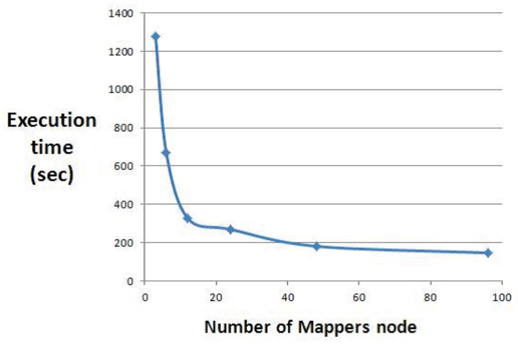

Fig. 4 Execution time versus number of Mappers

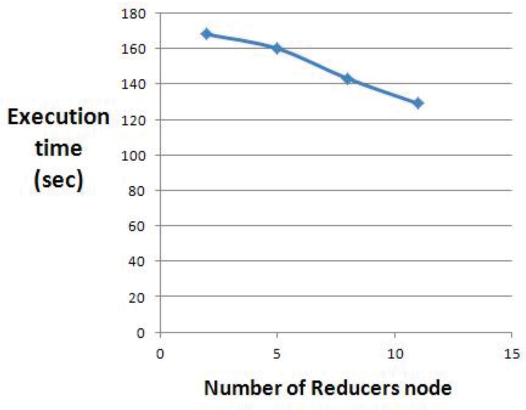

Fig. 5 Execution time versus number of Reducers

As shown in Fig. 5, execution time decreased as the number of Reducers increased. However, the change is not linear because the execution time resulting from the change in the number of Reducers is influenced by the items of learning data.

Next, we examined the change in the accuracy of learning in terms of the change in the number of nodes. Here, the accuracy is determined by the probability of the classifier group generated to learn and judge a result correctly according to the test data.

Figure 6 illustrates learning accuracy with respect to the number of Mappers. If the number of Mappers is increased by too much, a state will be reached where there is hardly any learning. Also, if a file is divided too much, learning by each machine will become insufficient, and as a result, the overall learning rate will fall. In this experiment, the size of the learning data used was 740 MB. The accuracy of the learning became low for sizes smaller than 64 MB. Thus, it was necessary to search the MCS on Hadoop to determine the optimal file assignment size.

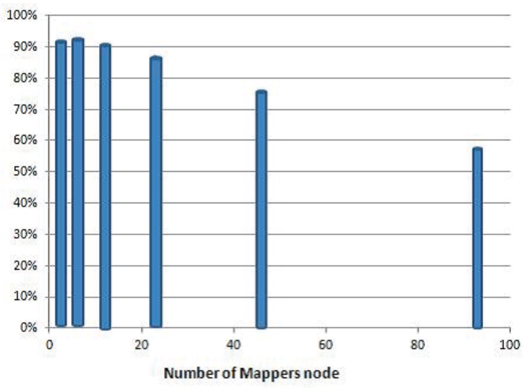

Fig. 6 Accuracy of learning versus number of Mappers.

Next, learning with a simple classifier system was compared with learning using our proposed MCS technique.

The column on the right of the graph shows learning with the simple classifier system, while that on the left represents learning by our proposed MCS. As can be seen in the graph, the accuracy

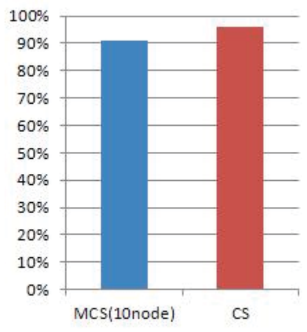

Fig. 7 Accuracy of learning in simple and multiple classifier systems

of the overall learning fell in the MCS. Since a classifier system is what is originally risen and learned in one classifier group, when it merges, as the whole group's work, it is inferior to the simple classifier system.

The pie chart depicted in Fig. 8 illustrates the items of learning data used in the experiment. In the learning data used (740 MB), attack intrusion data of a very rare kind comprising only about 20 KB (less than 1 % of the whole), such as guess passed and nmap, was included. Next, we examined how the learning rate varied with the small amount of attack data (Fig. 9). It was

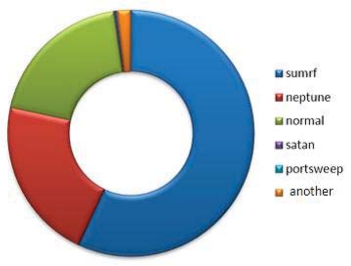

Fig. 8 Learning data items, displayed according to their proportional likelihood to be attacked.

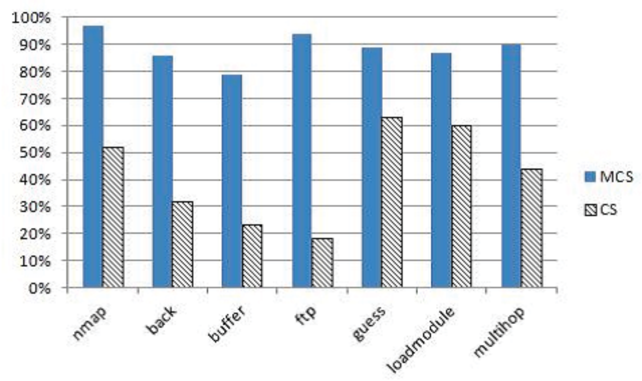

Fig. 9 Left: Learning of multiple classifier system. Right: Learning of simple classifier system

difficult for the above data to update the learning intensity of the simple classifier system. Because there was the possibility of obtaining learning by other inputs, leaking and being removed from a group was high, so there was hardly any learning. However, in our proposed technique, since the classifier was gathered by distributing classifiers that received and reacted to each small input in learning to each Reducer, it was possible to keep the learning rate high.

7. Conclusion

Learning by a multiple classifier system, compared with learning with a simple classifier system, can significantly reduce the learning time and the effectiveness of the learning method on large-scale problem data sets. Moreover, a time reduction in the merge process was gained when executing the proposed multiplier classifier system on the Hadoop Map Reduce framework. However, it turns out that some accuracy of learning is lost when using a MCS. In order to raise the accuracy of learning, we have to consider a better merging method and to further study the method of classifiers. When learning data was distributed, It turned out, that the optimal distribution of data is problematic. Learning by each machine is inefficient, if the processed data is too small. Also, one has to be careful with splitting data where the distribution of important of data is unknown.To consider that Hadoop divides a data file automatically, it seems that this issue becomes a big subject to the proposed technique.

By the proposed technique, it was possible to learn without leaking small input data. Even in large scale data, the proposed technique was able to learn in a reasonable fast time without leaking small input data.

Recently, an improvement of the MCS technique, such as sharing classifiers and other parameters between individual machines, has been proposed (see [2]). Such improvements seems also to be promising for our proposed technique and should be investigated further in the future.

Acknowledgments This work is supported by KAKENHI (No.22500196), Japan Society for the Promotion of Science.

References

[1] Igino Corona, Roberto Tronci, Giorgio Giacinto : *A Multiple Classifier System for the Protection of Web Services*, 21st International Conference on Pattern Recognition (2012).
[2] Szymon Sztajer, Michal Wozniak : *Untrained Multiple Classifier System for Designing Security Scanner for Web Applications*, 5th International Conference on Information and Automation for Sustainability (2010).
[3] Battista Biggio, Goirgio Fumera, Fabio Roli : *Multiple Classifier System for Adversarial Classification Tasks*, Dept. of Electrical and Electronic Eng, Univ. of Cagliari Piazza d'Armi, 09123 Cagliari, Italy.
[4] David E. Goldberg : *Genetic Algorithms in search, Optimization and Machine Learning*, (1989)
[5] Biggio, B., Fumera, G., Roli, F.: *Supervised and Unsupervised Ensemble Methods and Their Applications*, Studies in Computational Intelligence. Springer, Heidelberg (2009).
[6] John H. Holland and Judith S. Reitman : *Supervised and Unsupervised Ensemble Methods and Their Applications*, In Donald A. Waterman and Frederick Hayes-Roth, editors, Pattern-Directed Inference Systems, pages 313-329, Orlando, (1978).Academic Press.
[7] John H. Holand: *Adaptation in Natural and Artificial Systems*, MIT Press (1989).
[8] Henrique Santos, Manuel Filipe Santos and Wesley Mathew, University of Minho, Portugal : *Supervised Learning Classifier System For Grid Data Mining*, (2009).
[9] ajai Terano, Hasnat Elias Mohammad Abu, Mhd Irvan : *Acquiring Plant Operation Knowledge through Learning Classifier Systems*, (2011).
[10] Tom White : *Hadoop*, O'Reilly Media (2010).
[11] *Future view: Web navigation based on learning user's browsing patterns*, (2003).
[12] Apache CloudStack, available at: http://incubator.apache.org/cloudstack/ (2013)
[13] Apache Hadoop, available at: http://hadoop.apache.org (2013)