

Big Data Analytic based on Scalable PANFIS for RFID Localization

1st Choiru Za'in

*Department of Computer Science and Information Technology
La Trobe University
Melbourne, Australia
C.Zain@latrobe.edu.au*

2nd Mahardhika Pratama

*School of Computer Science and Engineering
Nanyang Technological University
Singapore, Singapore
mpratama@ntu.edu.sg*

3rd Andri Ashfahani

*School of Computer Science and Engineering
Nanyang Technological University
Singapore, Singapore
andriash@e.ntu.edu.sg*

4th Eric Pardede

*Department of Computer Science and Information Technology
La Trobe University
Melbourne, Australia
E.Pardede@latrobe.edu.au*

5th Huang Sheng

*Singapore Institute of Manufacturing Technology
Singapore, Singapore
shuang@SIMTech.a-star.edu.au*

Abstract—RFID technology has gained popularity to address localization problem in the manufacturing shopfloor due to its affordability and easiness in deployment. This technology is used to track the manufacturing object location to increase the production's efficiency. However, the data used for localization task is not easy to analyze because it is generated from the non-stationary environment. It also continuously arrive over time and yields the large-volume of data. Therefore, an advanced big data analytic is required to overcome this problem. We propose a distributed big data analytic framework based on PANFIS (Scalable PANFIS), where PANFIS is an evolving algorithm which has capability to learn data stream in the single pass mode. Scalable PANFIS can learn big data stream by processing many chunks/partitions of data stream. Scalable PANFIS is also equipped with rule' structure merging to eliminate the redundancy among rules. Scalable PANFIS is validated by measuring its performance against single PANFIS and other Spark's scalable machine learning algorithms. The result shows that Scalable PANFIS performs running time more than 20 times faster than single PANFIS. The rule merging process in Scalable PANFIS shows that there is no significant reduction of accuracy in classification task with 96.67 percent of accuracy in comparison with single PANFIS of 98.71 percent. Scalable PANFIS also generally outperforms some Spark MLlib machine learnings to classify RFID data with the comparable speed in running time.

Index Terms—Big data stream analytic, Rule merging strategy, Scalable machine learning, Distributed evolving algorithm, PANFIS

I. INTRODUCTION

Radio Frequency Identification (RFID) localization systems (RFID localization) has become a popular technology in

manufacturing or other production base industries to optimize the production output by providing the position/location of the Manufacturing Objects (MOs) (e.g. workstations, tools, staffs, and support services). In the manufacturing process, obtaining the accurate location of the MOs is essential because this information can represent the current Work-In-Process (WIP) condition in the industry. The WIP becomes the basis reference for decision making to avoid unexpected failure and in turn will increase the production output.

RFID localization technology is considered to be chosen among other localization technologies such as Wireless Sensor Networks (WSNs), Ultrasound, Infrared, and Video Camera due to its simplicity and price affordability in the deployment process [1]. These benefits allow industry to apply RFID localization devices easily in many areas of the manufacturing location in order to provide the accurate MOs to the Manufacturing Execution Systems (MES).

RFID localization systems consist of some devices and components: RFID tag, RFID reader, and data processing subsystems. RFID tag transmits beacon messages to RFID reader. RFID reader captures the Received Signal Strength (RSS) along with the tag ID from RFID tag. These signals arrive continuously and they need to be processed by the MES in the real-time mode.

Processing and analyzing MOs locations from the received RSS in RFID localization are challenging tasks due to two factors: 1) RSS information are not always reliable because its value varies due to environmental changes (e.g. minor movement); 2) RSS information arrives in the real-time mode

rapidly, which can be considered as a data stream. Thus, it will cause the generation of large-volume of data which is stored in the cloud/server, and this phenomenon can be considered as a big data problem. All of these factors underlie the needs of the advanced analytic algorithms to discover the knowledge of a big data stream.

It has been a common practice that machine learning algorithms are used for knowledge discovery of the data for decision making purposes. Many new business requirements are growing based on big data stream analytic services. However, discovering big data stream knowledge for decision making is challenging due to its 4V's characteristics: volume, velocity, variety, and veracity. For this reason, in RFID localization task, an advanced big data analytic is required to enable faster and accurate decisions in terms of determining the MOs locations to catch up the expected production speed.

Evolving machine learning techniques have gained popularity to process data stream, such as the work conducted in [2] in web news mining classification. These techniques aim to learn non-stationary data by learning the data in the single pass mode, enable to update data pattern for every incoming datum, without retraining all history data [3]–[6]. This feature is beneficial in handling data stream to cope with the velocity characteristic of big data. Furthermore, there are many distributed platforms such as Hadoop [7] and Spark [8] to scale the limited resource of CPU in processing and storing big data. With many challenging issues in big data, most of the research directions in this area are growing towards a development of the advanced distributed big data analytic to discover big data stream knowledge.

Due to the increasing demand in big data knowledge discovery, many efforts have been made on building large-scale distributed machine learning. Some recent works have been conducted in [9], [10]. Both frameworks incorporate distributed processing platform to scale the machine learning capability by dividing data stream into many chunks. However, while the first only processes the chunk of big data in batch mode as a standard platform for distributed learning, the latter extends the big data analytic framework to address the velocity characteristic of big data, enabling data chunk to be processed as a data stream, using evolving algorithm namely PANFIS [11]. The main property of the work in [10] is the model fusion, where all models generated from all data chunks are merged into a final model. This final model represents the current knowledge big data.

Based on the previous work in [10], we propose a novel big data stream analytic based on PANFIS, a seminal evolving neuro fuzzy system, to classify RFID big data stream for localization task. The distributed big data stream in RFID localization analytic framework has the following characteristics:

- 1) It processes the rapid rate of RSS signals using big data analytic framework by incorporating PANFIS, where PANFIS processes data stream in the single pass mode, and distributed machine learning framework using Spark platform.

- 2) This framework is equipped with rule' structure merging and rule parameters update to cope with the changing environment.

The rest of the paper is organized as follows. Section 2 discusses the related researches: RFID localization systems and a technique to capture the RFID data from sensors. Section 3 describes our proposed approach which specifically describes the data streams' flow processing in Spark platform. Section 4 discusses experimental setup and results in evaluating big data analytic and section 5 will conclude the paper.

II. RFID LOCALIZATION PROBLEMS

A. RFID Localization Systems

On the manufacturing shopfloor, there are many MOs required to be stored, received, installed, and tracked. The conventional approach to manage MOs drawn a lot of human resources. Thus, advanced techniques are required to identify and track MOs location in the large manufacturing shopfloor. Radio Frequency Identification (RFID) localization technology is one of many technologies which has been put forward for localization system and tracking due to its price affordability and the easiness of deployment.

The RFID localization system consists of three main components: RFID tag, RFID reader, and data processing subsystem. The main functionality of an RFID tag is to send the beacon message containing the tag ID and radio signal strength (RSS) information. The difference between active and passive tags lies on its power. RFID tag is categorized into two types: active tags and passive tags. Active tag is battery powered, which actively sending the beacon message periodically to the RFID reader, whereas the passive tag does not have a power source. Thus, the signal range of active tag can reached up to 300 m compared to 1 m for a passive tag. This experiment utilizes a passive tag, which only sends a beacon message once it receives the signal from the RFID reader.

RFID reader, as a second component in RFID localization system, captures the signals (data) sent by the tags and transmits it to the data processing subsystem. These signals are then processed by using the algorithm embedded in the data processing subsystems to do the RFID localization task. As a result, data processing subsystem provides the accurate MOs in the manufacturing shopfloor.

The most difficult task in the RSS processing task is the noise attached in the RSS. In an ideal condition, MOs location can be determined using the normal measured RSS value. However, due to the severe interference and multi-path effect, the RSS information becomes unreliable and keeps changing overtime. A minor change in the environment can cause a significant change in the RSS information. Thus, only calculating distances from RSS observations does not solve the localization problem. Therefore, an advanced machine learning algorithm is employed to train the value of RSS based on reference tags information to accurately perform the localization task. The experiment is set up by deploying the reference tags on the fixed and known locations. The

RSS information from reference tags are exploited to assist in locating the MOs. Suppose there is an RFID network which consists of n tags $T = \{T_i : i = 1, \dots, n\}$ and m readers $R = \{R_j : j = 1, \dots, m\}$. The RSS observations of tag T_i at time-step t is given as $x_i(t) = [x_{i1}, \dots, x_{im}]^T$, where x_{im} is the RSS observation of the tag T_i by reader R_m . The position of reference tag is denoted as class label $y(t) = [y_1, \dots, y_n]$, and thus the localization can be formulated into a multi-class classification problem. The Scalable PANFIS embedded in data processing subsystem is then employed as the machine learning algorithm to determine the MOs locations in the real-time mode.

B. Capturing RFID Data From Sensor

This localization setup were conducted at SIMTech laboratory, Singapore. The environment is arranged to represents the RFID smart rack system. The system aims to improve the production efficiency in manufacturing process by locating the right material for production. Thus, the accurate information of MOs location is highly required. As can be seen in Fig. 1, one RFID reader and four RFID reference tags along with data processing subsystem are set up. The objects are placed at a steel storage rack with the dimensions of $1510 \text{ mm} \times 600 \text{ mm} \times 2020 \text{ mm}$. The rack has 5 different shelves, each of them 360 mm apart. The rack shelf can contain up to 6 test objects, and each of them is attached with a passive RFID tag. The antenna is installed close to the rack at a distance of 1000 mm from the rack. The height of the antenna is 2200 mm above the ground and it is connected to the RFID receiver. The receiver is then connected to a data processing subsystem, where the localization algorithm is executed, through the ethernet links. The number of reference tag indicates that there are 4 classes representing the location of the objects considered in this experiments.

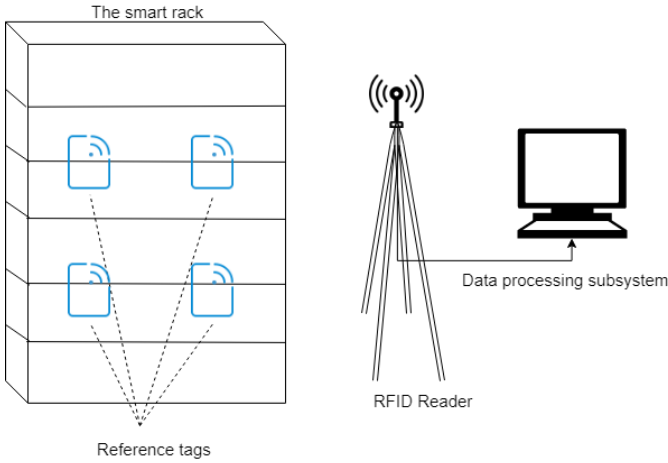


Fig. 1. Data collection procedure in RFID Localization Systems

The data processing subsystem has two main components: data acquisition component and algorithm execution component. The Microsoft Visual C++ based PC application is developed to acquire the RSS information from all tags data,

while the localization algorithm is executed using big data analytic based on PANFIS. The reader is also configured to record the RSS information for every 1 s . During 20 hours of experiment recording, the number of 283,100 observations are obtained, where every instance has the information of RSS and its reference tag which represents the MOs object location. Based on the recorded RFID data, Scalable PANFIS can learn this data stream behavior to predict the future data.

III. DISTRIBUTED BIG DATA ANALYTIC BASED ON PANFIS APPROACH

This section presents the details of our distributed big data analytic based on PANFIS. The briefly overview of PANFIS algorithm is described in the subsection III-A. Big data analytic architecture and data processing are elaborated in the subsection III-B. Furthermore, subsection III-C will describe the merging process of all rules generated from PANFIS learning algorithm of every data chunk.

A. PANFIS Algorithm

PANFIS is built based on Neuro Fuzzy System (NFS), a hybrid intelligent system, which combines Fuzzy Logic and Artificial Neural Network. The fuzzy systems capable to mimic human-like reasoning through the use of the fuzzy sets and fuzzy linguistic rule, whereas the neural network plays important role as a basis to develop algorithm to model complex patterns and prediction problems.

In the real world applications, systems are usually assembled with shifts and drifts, which cannot be handled by the common NFSs. These NFSs usually really depend on expert knowledge [12] and in turn requires tedious manual interventions. This problem leads to a static rule base, which unable to be adjusted once initial setting is setup to gain a better performance. PANFIS is built to further develop the disadvantage of common classical NFS to overcome the degree of nonlinearity which exists in the real world systems.

PANFIS is able to assemble a complex system's rule base autonomously by learning system from scratch with an empty rule base. Fuzzy rules and its parameters can be generated, updated and pruned during the learning process. Furthermore, the rule merging process is also executed during the learning by identifying identical fuzzy rule sets to simplify the rules complexity.

The main characteristics of PANFIS is the generation of ellipsoids in arbitrary direction. These ellipsoids describe the local correlation between variables. The antecedent parts of fuzzy sets are formed by the ellipsoids connected with a new projection. Thus, it is more interpretable for an expert/user in comparison with the non-axis parallel model applied in [13].

PANFIS system evolution is initialized by the generation of rules which is driven by datum significance (DS) criterion. DS algorithm was proposed in [14] and [15] to identify the datum's high-potential managing troublesome data streams. This initialization of rule growing must assures that ε -completeness criterion to ensure that rule base and the fuzzy partitions have a sufficient coverage of the input space. During the system's

evolution, when the new datum is covered by the current rules, the rule parameters (focal points and radii) of the fuzzy rules and fuzzy consequences are updated based on the new datum attributes. For focal points and radii, original PANFIS applies Extended Self-organizing Map (ESOM) theory by adjusting neighboring rules. However, in this work, rule adaptation utilizes GENEFS [16], pClass [4], and GEN-SMART-EFS [5]. This is due to the ESOM's drawbacks in regards to the possibility of instability issues when the inverse covariance matrix is ill-conditioned (e.g., due to redundant input features). Fuzzy consequences are also adjusted by using the enhanced recursive least square (ERLS), an extension of conventional recursive least square (RLS). In the ENFS community, ERLS has become the common method which is proven to support system errors' convergence and the update of weight vector.

The rule pruning of PANFIS is inherited from rule base simplification technology namely generalized growing and pruning radial basis function (GGAP-RBF) in SAFIS [15]. However, rule base simplification in SAFIS cannot be applied in PANFIS as it is only suits to zero-order TSK fuzzy system and the unidimensional membership function environments. PANFIS extends this method using extended rule significance (ERS) concept by defining the rule significant as statistical contribution of the fuzzy rule when the number of observation approaches to infinity. Furthermore, the statistical contribution of the fuzzy rule over all fuzzy rules is determined by the fuzzy rule volume over the all rules volume. If the contribution vector value of such rules less than the threshold $kerr$, these rules are regarded as outdated rules and will be discarded to reduce the network complexity.

PANFIS also applies fuzzy set merging process to form effective (economical and interpretable) rule base, and also to achieve a good predictive quality. This is done due to the identical/overlapping fuzzy sets (e.g., due to their similarity in membership functions). This similarity can be measured by benefiting a kernel-based metric method comparing the center and the widths of two fuzzy sets in one joint formula [17]. In the case of the perfect match, the two fuzzy sets are identical, and has the degree of similarity measure equal to 1. Conversely, if the two fuzzy sets have the similarity degree above the tolerable value $Sker \geq 0.8$ based on [17], the two fuzzy sets could be merged.

B. Big Data Analytic Architecture and Data Flow Process in Spark Platform

Apache Spark (Spark) is regarded as the latest framework for big data analytic to support advanced in-memory programming model. Spark ecosystem consists of two main parts: lower-level libraries known as Spark-core and upper-level libraries known as extension of Spark-core. Spark-core consists of some programming interfaces such as Scala, Java, Python, R, and SQL as an integrated Spark APIs. Upper-level libraries consists of Spark's MLlib for machine learning purposes, GraphX for graph analysis, Spark Streaming for stream processing, and Spark SQL for structured data processing. All of these Spark's components enable Spark to perform scalable

distributed machine learning, graph analysis, and structured data processing.

In this work, we utilize SparkR, an R package distributed with Spark to provide R front-end to Spark. It enables R to manipulate Spark DataFrames (DataFrames), a Spark data abstraction that is fault-tolerant for in-memory cluster computing, and operates Spark functionality in R as shown in Fig. 2. Fig. 2 shows the control flow of distributed big data analytics in this framework.

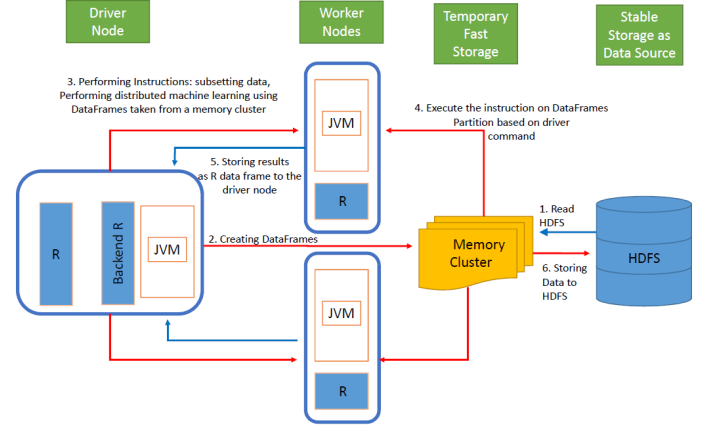


Fig. 2. Data control flow and operation in Distributed Machine Learning based on PANFIS

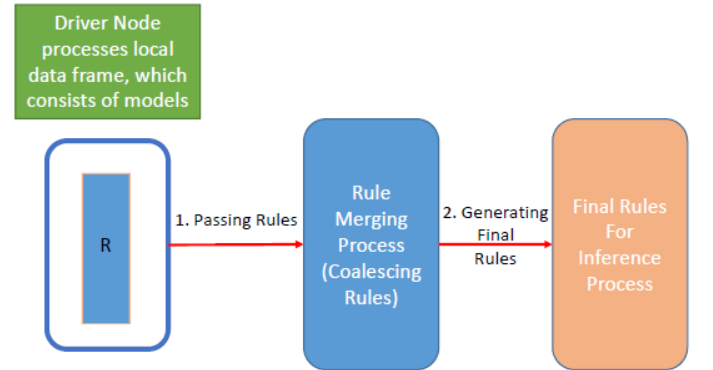


Fig. 3. The Rule Merging Scheme of processing all models into a single model in local R data frame

The big data analytic framework consists of nodes (driver node and some worker nodes) and the data sources (e.g., Hadoop Distributed File System (HDFS)). In this framework, distributed machine learning process is initialized by loading RFID dataset from HDFS storage, which is shown at step 1 Fig. 2. The second step, the driver sends an instruction to create the DataFrames (Spark's data abstraction) then stores the DataFrames in a memory cluster, a shared memory in the cluster. DataFrames is the only data type that can be distributed in the Spark platform. At the third step, the driver node sends many instructions to manipulate the DataFrames. These instructions can be repetitively created by the driver node to manipulate DataFrames, for example, performing distributed

algorithm (PANFIS) to all worker nodes to learn the chunk of DataFrames. Worker will process a chunk with a function (e.g. PANFIS) applied to them and send the result (rule(s)/model) to the driver node, which is shown at step 4 Fig. 2. At fifth step, all models generated from the worker nodes will be converted into local driver node memory as R data frame as a collected models, which can be seen by user. Step 6, is an optional step, whether driver node want to stores all models in the HDFS. Please be noted that all the instructions created by the R front-end is translated by JVM to be able to access Spark API to manipulate the DataFrames.

As explained in Fig. 2, driver node receives all rules (models) generated from many worker nodes when they process all data chunks using PANFIS algorithm. Fig. 3 depicts the data frame of R is captured in driver node as a learning results (a collected models). These results (rules) are then merged become the final model for inference purpose based on the process described in subsection III-C.

C. Rule Merging of Final Rules after All Data Chunk's Learning

Distributed big data analytic generates rules for every data chunk processed in the processors/nodes. In the distributed big data analytic architecture, processors/nodes are embedded by machine learning algorithm, which is in this case PANFIS algorithm. Rules generated by PANFIS represent the clusters of the data chunk. With so many rules generated in the system, some rules may become overlapping. Some rules can be merged to reduce the redundancy between the rules [18]. This merging process is essential in order to ensure the readability and transparency and avoiding the ambiguous rules. This rule merging scenario refers to the explanation in [5] without similarity criterion to extend the PANFIS rule merging scenario. The implementation of rule-merging follows two criterions: 1) Inspecting the 2 rules to be merged whether they are lying nearby, touching, or even slightly overlapping; 2) Calculating the homogeneity of the adjacent rules.

The first criterion in merging process is calculating the degree of overlapping for two clusters which measured by Bhattacharyya distance formula [19] [20]. This criterion process is calculated after the rules collection process in the learning phase. The overlap degree between target cluster (rule) expressed by win and the other clusters, $k = \{1, \dots, C\} \setminus \{win\}$ is expressed as follow :

$$olap(win, k) = \frac{1}{8} (c_{win} - c_k)^T \Sigma^{-1} (c_{win} - c_k) + \frac{1}{2} \ln \left(\frac{\det \Sigma^{-1}}{\sqrt{\det \Sigma_{win}^{-1} \det \Sigma_k^{-1}}} \right) \quad (1)$$

where $\Sigma^{-1} = (\Sigma_{win}^{-1} + \Sigma_k^{-1})/2$. The highest value of statistical contribution is chosen among the clusters, which based on the equation explained in PANFIS [11] as the target cluster win . The distance between two ellipsoidal clusters has the value of 0 if they both are touching, greater than 0 in the case of overlapping, and less than zero for disjoint situation. The minimum threshold for clusters to be merged is 0, which

is the condition where both clusters are touching each other in regard to the conditions stated.

The homogeneity of adjacent rules is the second criterion of the merging process. This criterion ensures the homogeneous shape and direction of two merged rules. This criterion also reflects the two local data clouds. By utilizing the blow up effect to trigger homogeneous joint regions, the homogeneity is defined as follows:

$$V_{merged} \leq p(V_{win} + V_k) \quad (2)$$

where p is input attribute's dimension. V_{merged} , V_{win} , and V_k depict merged rules' volume, winning rule, and compared rule respectively. Rule's volume can refer to the formula explained in PANFIS [11]. In conclusion, the merging criterions can be simplified by the conditions as follow:

$$(Eq.(1) \geq thr) | Eq.(2) \quad (3)$$

where thr is set to 0.8 based on empirical experience in [17].

1) *Rule merging phase's policy*: Merging process will be conducted if two criterions are satisfied refer to the winning rule. The winning rule refers to the rule which has more data points than another rule $N_{win} > N_k$. The updating parameters of rule merging is executed based on the work in [16] which is formulated as follows:

$$c_{win}^{new} = \frac{c_{win}^{old} N_{win}^{old} + c_k^{old} N_k^{old}}{N_{win}^{old} + N_k^{old}} \quad (4)$$

$$\Sigma_{win}^{-1}(new) = \frac{\Sigma_{win}^{-1}(old) * N_{win}^{old} + \Sigma_k^{-1}(old) * N_k^{old}}{N_{win}^{old} + N_k^{old}} \quad (5)$$

$$N_{win}^{new} = N_{win}^{old} + N_k^{old} \quad (6)$$

$$w_{win}^{new} = \frac{w_{win}^{old} * N_{win}^{old} + w_k^{old} * N_k^{old}}{N_{win}^{old} + N_k^{old}} \quad (7)$$

Where c_{win}^{new} and $\Sigma_{win}^{-1}(new)$ are the new antecedent parameters of the merged rule and w_{win}^{new} is the consequent parameter of the merged rule.

IV. EXPERIMENTAL SETUP AND RESULTS

The environmental setup in which Scalable PANFIS is applied in the distributed framework along with the experiment results will be described in this section. The framework performance between PANFIS distributed framework (Scalable PANFIS), PANFIS, and some other machine learning distributed frameworks are compared in order to measure the running time and accuracy. The data used in this experiment is RFID data, generated from real application based on sensor in classification problem. This dataset consists of 283,100 instances with only 1 dimension of input describing the RSS frequency of RFID to infer the location of Manufacturing Object (MOs).

A. Experiments

NeCTAR Cloud, which provides flexible scalable computing is used as the computing environment for this experiment. The framework consists of 7 nodes (1 driver and 6 workers). Each node has the specification as following: 30 GB Disk Capacity, 6GB RAM, and NeCTAR Ubuntu 16.04 LTS (Xenial) amd64 as operating system. The total cluster memory used in this framework is 30 GB as we use 5 GB of memory for 6 worker nodes, leaving another 1 GB for every node for other processes in the nodes. We use spark 2.2.1 release for cluster computing system at the time of writing. In this experiment 5 algorithms are compared: Scalable PANFIS, PANFIS, generalize linear regression model (GLM), Gradient Boost Tree (GBT) and KMeans. For comparison purpose with other seminal evolving algorithms, we also conduct the experiment against gClass [21], eT2Class [3], pClass [4], and eT2ELM [21]. For this purpose, we use the the personal computer with the following specifications: Intel(R) Core(TM) i7-6700 CPU @3.4 GHz 16GB RAM and Matlab 2017b Software specification.

These algorithms are chosen to perform the experiments based on the following reasons:

- 1) PANFIS is a seminal evolving algorithm which has capability to learn data stream in online manner. In this case, this base algorithm is compared against the proposed method, big data analytic based on PANFIS (Scalable PANFIS).
- 2) Random Forest (RF), Generalized Linear Model (GLM), Gradient Boost Tree (GBT), KMeans, are chosen as part of Machine Learning Library (MLib) on sparkR as a scalable machine learning for knowledge discovery. Thus we use these algorithms to compare their performance against scalable PANFIS. However, as their nature are binary classifier algorithm, we apply one vs rest strategy to be able to act as multi-class classifier.
- 3) gClass, eT2Class, pClass, and eT2ELM are compared in terms of running time and accuracy to perform the classification task as they are also classified as a seminal evolving algorithms.

The classification scenarios will measure the running time and the accuracy to evaluate the performance of all algorithms. The details of the experiment results are explained in the subsection IV-B.

B. Results

This subsection details the performance of every classifier in classifying the RFID dataset as described in subsection II-A. The dataset is divided into two parts: 1) 200,000 of training data and 2) 83,100 of test data for validation. We conduct the experiment for five different algorithms: The first algorithm, PANFIS, is executed in single CPU without distributed processing with the specification of 30 GB Disk Capacity, 6GB RAM, and NeCTAR Ubuntu 16.04 LTS (Xenial) amd64 as operating system. The software used for running PANFIS is R version 3.4.3. The second to fifth algorithm are the Scalable PANFIS and four other MLib algorithms provided in the Spark

library, which are executed with distributed processing in the Spark platform. We utilize SparkR to provide a lightweight front end for using Apache Spark from R. The number of data partition used for experiment in the distributed environment system is 50 partitions.

The result depicted in Table I shows that PANFIS algorithm (single machine learning) performs significant result with 98.71 percent of accuracy in learning the RFID data. The Scalable PANFIS, a distributed big data analytic based on PANFIS machine learning in the Spark environment, yields 96.67 percent in accuracy, which is still comparable with PANFIS single machine learning. The merging process as shown in Table II, reduces the number of rules from 55 into around half of 28 rules. The comparison of Scalable PANFIS with three other algorithms provided in MLib shows that Scalable PANFIS can handle the nonstationary data stream with the accuracy higher than GLM, GBT, and Kmeans with 96.67, 50.03, 75.07, and 47.66 percent for Scalable PANFIS, GLM, GBT, and KMeans respectively. Scalable PANFIS is slightly outperformed by RF with 96.67 and 98.56 percent for Scalable PANFIS and RF respectively. However, Scalable PANFIS better than RF in terms of running time. Please be noted that all of the MLib algorithms are binary classifier in nature and are converted as multi-class classifier by applying one vs rest strategy.

Another highlight is also depicted in the Table I in terms of running time. Scalable PANFIS performs very significant result with more than 20 times faster than PANFIS executed in the single CPU with 104 seconds and 2130 seconds for Scalable PANFIS and PANFIS respectively in terms of running time.

To further evaluate the Scalable PANFIS performance, this algorithm is benchmarked with other state-of-the-art algorithms: eT2Class, eT2ELM, gClass, and pClass. Table III shows that the accuracy and the running time of other benchmarked algorithms. This experiment is conducted in the same computational environment. The result shows that Scalable PANFIS has a comparable performance in terms of accuracy with other evolving algorithms.

TABLE I
THE PERFORMANCE COMPARISON OF SCALABLE PANFIS AGAINST
SINGLE PANFIS AND OTHER MLIBS ALGORITHMS

Algorithm	Accuracy (%)	Running Time (s)
PANFIS	98.71	2130
Scalable PANFIS	96.67	104
RF	98.56	149
GLM	50.03	264
GBT	75.07	272
Kmeans	47.66	88

V. CONCLUSION AND FUTURE WORKS

Scalable PANFIS is a big data analytic framework which processes high-volume of big data by distributing data stream into many partitions thus accelerate the learning process.

TABLE II
THE EVOLUTION NUMBER OF RULES IN THE MERGING RULE PROCESS OF
BIG DATA ANALYTIC FRAMEWORK

Number of Rule	
Before Merging	After Merging
55	28

TABLE III
PERFORMANCE OF OTHER SEMINAL EVOLVING ALGORITHMS RUNNING
STANDALONE IN THE SINGLE CPU USING MATLAB ENVIRONMENT

Algorithm	Accuracy (%)	Time (s)
eT2Class	98.77	1223
eT2ELM	95.38	350
gClass	92.89	1253
pClass	98.05	604

PANFIS algorithm, the base algorithm of Scalable PANFIS learns the data chunk in online manner. The models generated from the learning process of many data partitions can be merged become a single model without reducing the accuracy performance. For the future work, we will further evaluate the performance of Scalable PANFIS by testing this algorithm to many other high-dimensional big data with using some classification techniques.

VI. ACKNOWLEDGMENT

This project is fully supported by NTU start up grant and MOE tier 1 research grant. This research is also supported by use of the Nectar Research Cloud, a collaborative Australian research platform supported by the National Collaborative Research Infrastructure Strategy (NCRIS).

REFERENCES

- [1] L. M. Ni, D. Zhang, and M. R. Souryal, "Rfid-based localization and tracking technologies," *IEEE Wireless Communications*, vol. 18, no. 2, 2011.
- [2] C. Za'in, M. Pratama, E. Lughofer, and S. G. Anavatti, "Evolving type-2 web news mining," *Applied Soft Computing*, vol. 54, pp. 200–220, 2017.
- [3] M. Pratama, J. Lu, and G. Zhang, "Evolving type-2 fuzzy classifier," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 3, pp. 574–589, 2016.
- [4] M. Pratama, S. G. Anavatti, M. Joo, and E. D. Lughofer, "pclass: an effective classifier for streaming examples," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 2, pp. 369–386, 2015.
- [5] E. Lughofer, C. Cernuda, S. Kindermann, and M. Pratama, "Generalized smart evolving fuzzy systems," *Evolving Systems*, vol. 6, no. 4, pp. 269–292, 2015.
- [6] M. Sayed-Mouchaweh and E. Lughofer, *Learning in non-stationary environments: methods and applications*. Springer Science & Business Media, 2012.
- [7] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*. Ieee, 2010, pp. 1–10.
- [8] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.
- [9] A. Fernández, S. del Río, V. López, A. Bawakid, M. J. del Jesus, J. M. Benítez, and F. Herrera, "Big data with cloud computing: an insight on the computing environment, mapreduce, and programming frameworks," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 5, pp. 380–409, 2014.
- [10] C. Zain, M. Pratama, E. Lughofer, M. M. Ferdous, Q. Cai, and M. Prasad, "Big data analytics based on panfis mapreduce."
- [11] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, "Panfis: A novel incremental learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 55–68, 2014.
- [12] W. L. Tung and C. Quek, "efsm-a novel online neural-fuzzy semantic memory model," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 136–157, 2010.
- [13] A. Lemos, W. Caminhas, and F. Gomide, "Multivariable gaussian evolving fuzzy modeling system," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 1, pp. 91–104, 2011.
- [14] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 57–67, 2005.
- [15] H.-J. Rong, N. Sundararajan, G.-B. Huang, and P. Saratchandran, "Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction," *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260–1275, 2006.
- [16] M. Pratama, S. G. Anavatti, and E. Lughofer, "Genefis: toward an effective localist network," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 3, pp. 547–562, 2014.
- [17] E. Lughofer, J.-L. Bouchot, and A. Shaker, "On-line elimination of local redundancies in evolving fuzzy systems," *Evolving Systems*, vol. 2, no. 3, pp. 165–187, 2011.
- [18] E. Lughofer, *Evolving fuzzy systems-methodologies, advanced concepts and applications*. Springer, 2011, vol. 53.
- [19] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distribution," *Bull. Calcutta Math. Soc.*, 1943.
- [20] A. Djouadi, O. Snorrason, and F. Garber, "The quality of training sample estimates of the bhattacharyya coefficient," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 92–97, 1990.
- [21] M. Pratama, J. Lu, S. Anavatti, E. Lughofer, and C.-P. Lim, "An incremental meta-cognitive-based scaffolding fuzzy neural network," *Neurocomputing*, vol. 171, pp. 89–105, 2016.