1st International Conference on Applied Soft Computing Techniques
22 & 23.04.2017
In association with
International Journal of Scientific Research in Science and Technology

Incremental-Parallel Data Stream Classification in Apache Spark Environment

A.Anantha Babu, J.Preethi
Department of Computer Science and Engineering, Anna University Regional Campus Coimbatore, Tamil Nadu, India

ABSTRACT

With notorious domain of big data age, the challenging task on data stream classification is high velocity conceptual infinite stream and perspective statistical properties of data which differs periodically. In this paper, we propose an Incremental Parallel Random Forest (IPRF) algorithm for data streams in spark cloud computing environment. The algorithm incrementally estimates the accuracy for classifying the data streams, which priors to parallelization process in order to reduce the training time and prediction process using random sampling and filtering approach, that improves the dynamic-data allocation and task-scheduling mechanism in a cloud environment. From the perspective of dynamic-data allocation, dynamically changes the data in a data stream environment, to reduce the communication cost, volume data using vertically data-partitioning, data-multiplexing method. From the perspective of task scheduling, an incremental-parallel technique is carried out in the training process of Random Forest and a task directed acyclic graph depends upon resilient distributed data objects as static, redundant, and least data object appending to re-organize the mapping relationship between successor task and slaves. The details and the results of evaluating the proposed mechanism using benchmark datasets are presented in this paper.

Keywords: Big Data, Data Stream Classification, Incremental-Parallel Technique

I. INTRODUCTION

A. *Motivation*

With the unprecedented growth and profuse appliance of cloud computing and Internet of Things (IoT), the transmission, sharing, and collection of data reached very high level. Data is produced and gathered continuously at a puzzling speed in such areas social networks, remote sensor network monitoring, on-line banking, and solar power system. Unlike traditional data sets, these emerging and popularizing data sets are enormous, temporally ordered, quickly changing, and potential with these characters are called as data stream [1].

Big data can be considered as one of the main source data streaming. The well-known challenging problems are data streaming likes Infinite length, concept-draft, and concept-evolution and feature evolution. One problem emerges when a data stream is assumed to infinite length, yet it is speed and rapid phenomenon. Another issue is the concept draft presumption that streaming data can be underlying concepts of the stream change over time. A third assumption, when the concept and feature evolution can be used to predict the future behavior and also has been undermined as the technology continuously changed [2].

To refresh mining results are used to the promising approach in the incremental techniques of the high speed and analysis for large–scale data streams. Unfortunately, the new programming models are supported by the incremental processing. Data mining is processed to the hadoop famous cloud platform [3]. In the MapReduce model can be implemented to the machine learning algorithm, the Hadoop Distributed File System (HDFS) can be written in the intermediate results in all iterations. Enormous resource communication, storage and also disk I/O operation is taken much time and cost [4]. Data mining is suitable to be processed the cloud platform apache-spark. Apache spark is supported to the Resilient

Distributed Datasets (RDD) and Directed Acyclic Graph (DAG) in compared to hadoop. It can be performed to the stored data cache, perform computation and iterative computation for same data directly from memory. The slaves can be processed disk I/O operation to take the huge amount of time. Therefore, it can be more suitable for data mining with incremental computation [5].

L.Breiman is introduced to the Random Forest (RF) algorithm [6] at 2001. It can be supported the ensemble machine learning to data streaming classification, decision tree and bagging, weighted voting method to construct large number decision trees using probabilistically. Decision trees are constructed concurrently to be supported parallelization.

B. *Our Contributions*

In this paper, we propose an Incremental Parallel Random Forest (IPRF) algorithm for big data that is implemented on the spark cloud computing environment. The IPRF algorithm is optimized based on an incremental-parallel technique combining the data–parallel and task–parallel optimization. To improve the classification accuracy of IPRF, an optimization is proposed prior to the parallel process. Our contributions in this paper are summarized as follows.

- An incremental-parallel learning technique is proposed to improve the accuracy of IPRF, which combines the ideas of streaming decision trees and Random Forests.
- A dynamic duration data streams, in which the duration at each time step may dynamically change, this is in contrast to the traditional data–parallel optimization where the duration is fixed and static.
- Based on the dynamic data allocation, a memozation task parallel optimization is proposed and implemented on spark. Training task DAG of the data stream is constructed based on the RDD model and Memozation task schedulers are involved in perform the tasks in the DAG.

This paper we divide into different section: the section I is the introduction. Section II reviews the related work that has been done towards data stream mining and gives the limitations of the conventional methods and fundamental techniques for constructing dynamic data allocation and task scheduling. In Section III, we introduce two fundamental techniques for constructing our Random Forest in data streaming classification: Filter method and weighted voting method. Section IV describes data stream implementation of the RF algorithm on spark. Section V presents the experimental results and evolutions, with respect to the classification accuracy and performance. Finally, Section VI presents the conclusion and future work.

II. RELATED WORK

Although traditional data processing techniques have achieved good performance for static data, they are difficult to be handled to dynamic data efficiently. When a dataset becomes the data arrive at high speed and in huge volumes, the deemed concept of a data stream is frequently subject change along the time and large size, the accuracy and performance data mining algorithm are significantly denied [7, 8].

Due to the need of address the underlying concepts change over a time and noisy data, various improvement method introduced by researchers. Hulten et al. [9] and Bifet et al. [10] introduced some data streaming classification algorithm for mining time–changing data streams to address the issue of window size. These algorithms use Concept–Adapting Very Fast Decision Tree (CVFDT) and Hoeffding Adaptive Tree using Adaptive Windowing (HAT–ADWIN) and achieve efficient variable–length window in binary classification problems. Wang et al. [11] proposed a general framework for classification of data streams with concept–drift. Kanoun et al. [12] and Ghazikhani et al. [13] focused on the defect of adaptive ensemble algorithm is that the basic classifier learning model must have the ability to learn incrementally for new data blocks. Tarkoma et al. [14] proposed a theory and practice of bloom filters for distributed systems. Based on the existing research results, we propose a new optimization approach in this paper to address the problem of infinite length, concept–draft, concept–evolution, feature evolution and noisy data, which reduce the training time of the data according to the structure of the RF and improves the algorithm's accuracy with a low computational cost.

Focusing on the performance of classification algorithms for streaming data, numerous studies on the intersection of parallel / distributed computing and the learning of decision tree model were proposed. Palit et al. [15]

proposed a parallelized boosting with Map-Reduce, in which parallel ADABOOST and LOGITBOOST algorithms that achieve parallelization in both time and space. Svore et al. [16] carried out a boosted decision tree ranking algorithm, which addresses the speed and memory constraints by distributed computing systems.

Focusing on dynamic resource allocation and task scheduling execution in a parallel and distributed environment, Warneke et al. [17] proposed a dynamic resource allocation for efficient parallel data processing in a cloud environment. Chen et al. [18] carried out locality-aware and energy-aware job pre-assignment for MapReduce. Liu et al. [19] proposed a dynamic resource allocation and task scheduling strategy with uncertain task runtime on IaaS clouds. Zhang et al. [20] proposed an evolutionary scheduling of dynamic multitasking workloads for big data analysis in an elastic cloud.

Apache Kafka provides a distributed server for message queuing and then it works on the underlying concept of publish–subscribe mechanism. To provide large scalability for storing and consuming data are various machines in the cluster to be distributed on the streaming data. Due to replicated and persistent storage, it curbs up the risk of data loss [21]. Spark streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Spark is programmable framework for massive distributed processing of datasets, receives input data streams, divides the data into batches and then processed by the spark engine to generate the results.

III. Dynamic Random Forest Algorithm Optimization

We proposed an optimization approach for the RF algorithm to the improvement of the reduce time and space, classification accuracy for data streams. First, a pre-processing is performed in the classification. Second, a base classifier is model constructed in the pre-processed data. After ensemble approach is used to the output of multiple algorithms combined with the streaming data.

A. Pre–Processing: Sampling and Filtering for Streaming data

To improve the accuracy of the RF algorithm for streaming data, we present a new pre-processing method to reduce the time and memory the processing input data according to the importance of the feature variables. In a sampling techniques are constructing subsample from income streaming data according to some criterion. The stratified random sampling is used to create the sample of streaming data constructed to the models in ordered to subsample created into equal size. Filtering or feature selection is one of the common processes on the data streams. Filter methods can be performed rank individual feature or evaluate entire feature subsets. When the stream of data arrives, then the tuples which meet the specified condition will be accepted and others are dropped. The purpose of bloom filter in streaming analytics is the space and time efficient data structure. It deals with hash functions and makes the process of filtering more efficient. It can be solved by choosing an optimal number of hash functions and bit array size on the false probability. The process of filtering is presented in Fig. 1.

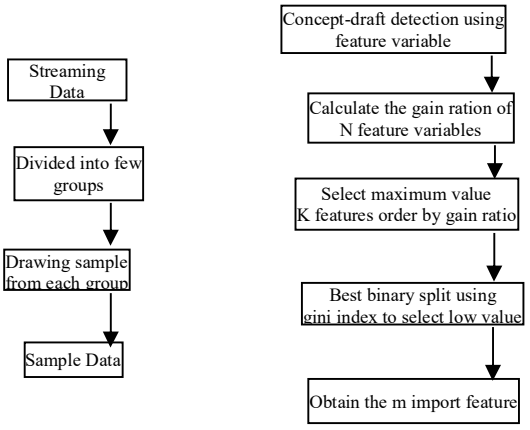

Figure 1. Stratified Random

Sampling and Filtering Process.

First, in the training process of each decision tree, Shannon's entropy is a measure of the indecorum associated with a random variable. It is defined as

$$H(y) = \sum_{i=1}^{n} p_i \log_2(p_i) \quad \text{- (1)}$$

Where y is a discrete random variable, and p_i is the probability of occurrence of y_i. To measure the entropy

for a dataset with known distribution requires using following equation.

$$H(y) = \sum_{y \to i=1}^{n} P(y) \log_2 (P(y)) \quad \text{- (2)}$$

Where P(y) is the probability mass function of y. A Two–window paradigm is used entropy in the context of detecting concept changes in data streams.

Second, the self-split information $I(y_{ij})$ of each input variable is calculated, as defined

$$I(y_i) = \sum_{k=1}^{o} - p_{(k,i)} \log_2 (p(k,i)) \quad \text{- (3)}$$

Where o is the number of different values y_i and P(k,i) is the probability of the type of value k with all type variables in y_i.

Then, the information of each feature variable calculated,

$$IG(y_i) = H(y) - H(y_i) \quad \text{- (4).}$$

By using information gain to measure the attribute, with the maximum gain ratio selected as the splitting attribute, it can be avoid the over fitting problem. As the split information approaches 0, the ratio becomes unstable. The gain ratio is defined as

$$GR(y_i) = \frac{IG(y_i)}{I(y_i)} \quad \text{- (5)}$$

Third, Gini index to select the best attribute, to measure how well separate on the feature variable. Decision tree is growing at each level to measure the best split and resume operation continue on the best split.

$$GI(y) = 1 - \sum_{i=1}^{n} P_i^2 \quad \text{- (6)}$$

In a binary split compute a results on the partitioning to calculate on the weighted sum of the impurity. It can be split on y partitions y1 and y2 on the gini index of y that partitioning is

$$Gini(y) = \frac{y_1}{y} Gini(y_1) + \frac{y_2}{y} Gini(y_2) \quad \text{- (7)}$$

For each attribute possible to binary split is considered. The splitting subset can select on the minimum gini index for that attribute. The splitting attribute selected on the reduction of impurity to maximized on the attribute. In a splitting criteria can be considered on the attribute, splitting point and splitting subset.

$$Gini(A) = Gini(y) - Gini_A(y) \quad \text{- (8)}$$

Fourth, the hoeffding bound is used to accumulate an enough records in a robust decision for attribute test to be made. The hoeffding bound states that, given a random variable s in the range L, m independent values of s having mean (s) and the true mean of r is at least (s)-e, where upon

$$E = \sqrt{\frac{L^2 \ln (1/g)}{2m}} \quad \text{- (9)}$$

With probability 1-g and g is a user-defined threshold.

B. **Random Forest on Streaming data classification**

Random forest ensemble process is constructs on the basis of two techniques stratified random sampling and bloom filtering. The pre-processed data requires less amount of space and time when compared to the processing input data. Decision trees store the data in the leaf node and it is a balanced tree which is often linked to one another. Decision tree makes incrementally updated on the evolution of concept draft. Random forest is used for classification which consists of multiple decision trees. The output of each instance is calculated based on the averaging or weighted voting method the predictions of trees using ensemble techniques.

First, a pre-processing is performed in the classification and it can be handled to the huge and high–velocity data. Data streaming scenarios are considered to the most important parameters (Memory and Time). Pre-processed data can produce more or less equal to that of accuracy obtained through original data on the pre-processing techniques. To reduce the training and testing

time is used to the stratified random sampling and bloom filtering.

Second, after pre-processed data is constructed to the classification on the decision tree. Data streaming is considered to the concept–draft, feature selection, changes over a time, and model stability. Classification process can be performed to the perfectly predicts the unseen data, we can say there is a model stability. The classification process is constructed to the multiple decision tree model.

Third, Ensemble approach has been used for classification and it has been used to the supervised machine learning models. Machine learning algorithm has been trained and predictions to the unseen data than the models build using single classifier. Ensemble represents a single hypothesis and it seems that ensemble models have more flexibility.

C. The Standard Random Forest Algorithm

L.Breiman has developed the random forest algorithm for ensemble classification techniques based on the decision tree model. A number of binary decision tree grows as with any tree ensemble classifier and each new record can predict the class using a plurality of the class predictions from the set of trees. However, it can differ from standard ensemble techniques in the way to grow each tree are selected in which records and the way at each internal node are selected in which attributes.

Suppose that a data set contains p records, each with q attributes. The steps for each tree growing of the random forest algorithm are as follows:

Step 1: Choosing k different subsets of the p records from a training set in a stratified random sampling manner.

Step 2: For each internal node, a subset of q ($q<<p$) stratified randomly chosen attributes are selected, and the way in which attribute and the split point is a decision made using the standard Gini index algorithm only the selected attribute.

Step 3: The tree is left pruned.

The stratified random selection of records are replacement leave off some records, are never used in the building this tree. To estimate the error rate of each tree can be used these records. This process can be repeated with a new subset of the records to produce a user specified number of trees. Each tree is made the choice of the attribute and split point at each internal node. This tree is being built to depend on the other records and other attributes that were chosen to build this internal node.

IV. Incremental-Parallel learning of the Random Forest algorithm on Spark

We propose Incremental-parallel learning of the Random Forest algorithm on apache spark, to address the problems on the real-time streaming data communication cost and workload imbalance problem of large scale data in a parallel and distributed environment. The IPRF algorithm is optimized based on the dynamic data allocation and task scheduling for dynamic parallel optimization. The dynamic data allocation optimization on the vertical data-partitioning and data multiplexing method are performed each iteration results to summarized. The filter method reduced the data size and improve communication cost on the distributed and parallel environments. In a task scheduling based on data allocation to run the task carried out in the training process of the IPRF and RDD objects based on the DAG task created. Then task schedulers are minimize the data communication cost.

1. Dynamic data communication on spark cluster

Apache spark is a fast and general engine for large-scale data processing in a data streams with unbounded sequence. The apache spark can be perform two primitives: a Master and a Slave. It can be performed in a Resilient Distributed Datasets (RDDs) collection objects to compute a results. RDDs can be contains two actions: Transformations (convert into new form on existing data) and Operations (computing results). It can be supported SQL operation and can be avoid on the redundancy execution using Directed Acyclic Graph (DAG). The spark is an advantages vertically partitioned data, data multiplexing method and dynamic data allocation on a streaming environment. Then it can be achieved into workload balance, reduce communication cost achieved on the spark.

The vertical data partitioning method is a gini index computing task of each feature variable take up dynamic of the training time. In a task can be changed dynamically based feature variables. IPRF can performed reduced volume of data and distributed cost in a parallel and distributed environment. The training datasets are divided into dynamic feature subsets.

The training datasets are vertical partitioned into subsets by using feature subsets. There is a subset construct to the individual RDD objects and independent of other subsets. There is a dynamic feature variable added on end of the RDDs objects. The volume of datasets are sampling problems arise on the linearly increment on the RF scale and can be created Data-Sampling-Index (DSI) on the table. In data multiplexing methods are perform: allocate to slaves on the dynamic sampling times, allocate feature subsets to the clusters and generate DSI table by using gini index computing on the dynamic nature of datasets. Gini index is performed into dynamic feature variable selection on the training samples.

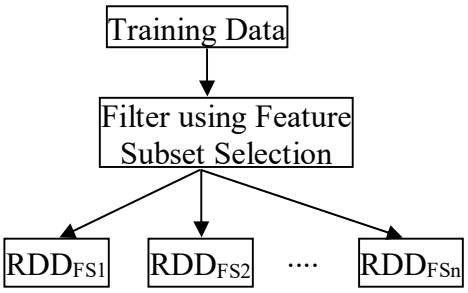

Figure 3. Vertical Data Partitioning Method

i. **Dynamic workload and task allocation strategies**

To better dynamic workload allocation strategies on data-partitioning and data-multiplexing method is performed on the feature subsets. In a dynamic nature of datasets are varied on each time based on the data type and volume. The datasets can be allocated into multiple slave nodes in a volume data. In a data allocation method have a three types: greater than slave capacity, less than slave capacity and equal amount to slave capacity. There is a data allocation method into local and global communication of data in the cluster environments. The RDD objects are allocated to the particular slave node and after execution results can be stored on persistent ways. The dynamically objects are changed to the feature subsets and same object allocation to the same worker node to avoid the execution.

The decision trees are built using IPRF on the dynamic independent feature variables construct the sub-node. There is some gini index computing task change on dynamic feature selection. It can be performed: training process and testing process. In a training process are trained to the filtering and sampling into dynamic data to be a feature selection, and testing process are performed into DAG create to construction on the process execution independent slaves. In a compute task scheduling can divided into two types: gini-index splitting task and node-splitting tasks. Gini-index is a computing task split into multiple slave nodes and impurity levels are reduced on the construction of the decision tree. Node splitting process are task executed on the worker nodes and momzation is stored results on the execution of the cluster worker nodes.

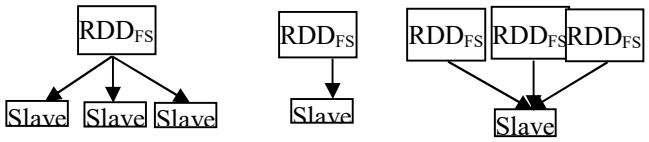

Figure 4. Dynamic Data Allocation and Task execution strategies

V. Extensive Experimental Results

A. Experimental Setup

The apache spark cloud platform performed all experiments and it is built of one master and three slave nodes. Each node contains the Linux Mint 18, Intel Pentium inside 2GHz GPU and 8GB memory. All the nodes are connected at high-speed Gigabit network and are configured with hadoop 2.7.3, Spark 2.1.0 and the algorithm is implemented in scala 2.11.0. The streaming datasets with time and memory is used in the experiments.

We have used bench mark data name Solar Power System which consists 1,88,835 instances with 40 attributes which contains 5 quantitative variables, 4

binary wildness, 6 solar power system attributes and no missing attributes. In a solar power system datasets used to detect the income amount energy or power, the usage of energy or power and amount of energy or power saved in the feature usage. In this three types of operation can be executed on spark cloud platform. In a training data not load into spark cluster. The spark cluster can be processed the datasets results to be stored. Then each iteration can execute the single process on the RDD objects process the slave node and result can be stored to the master node. There is a next iterations data to check before process the data, then conditions are

1. the incoming data is new to be process
2. some modification on the existing data to process
3. Existing data can't processed then already processed results to retrieve in the master.

B. Classification accuracy

The classification accuracy is not affected on the model construction using samples and filtered tuples. To evaluate the actual, sample and filter data has been implemented on the random forest using spark. The spark can be converted into libSVM format (label index: value index: value) for pre-processing. The algorithm is mainly considered to the some parameters: fixed (number of class) and dynamically changing parameter (maximum number of bins, maximum depth of tree and maximum number of trees). The majority voting methods are used to prediction on test data. The streaming data is mainly considered to the factors on time and memory. Then time taken to test the incoming data for model construction.

The test error is calculate using confusion matrix contains the information on the actual and prediction classifications done by random forest ensemble method. Then a binary prediction can be considered:

- X and U be number of correct and incorrect predictions on the instance of class a.
- Y and V be number of correct and incorrect predictions on the instance of class b.

$$Errorrate = D+U \,/\, X+D+U+Y \qquad \text{- (10)}$$

The error rate can be identified to varies size of data execute on the spark environment and then the results are represented to the Fig. 4.

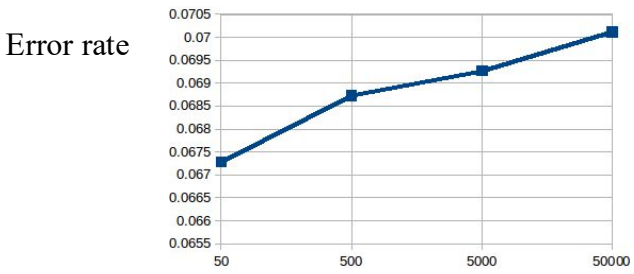

Figure 5. Test accuracy for different streaming data **size**

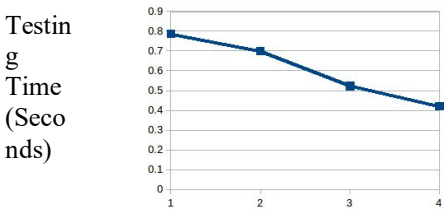

Figure 6. Testing Time analysis

The random forest is execute the data on the test and model time varies for actual, sample, filtered, both into represents Fig. 5. The curves are represented on the model building time and test on the instance. The model build and testing time can be reduced on the actual data execution process to sampling and filtering process data.

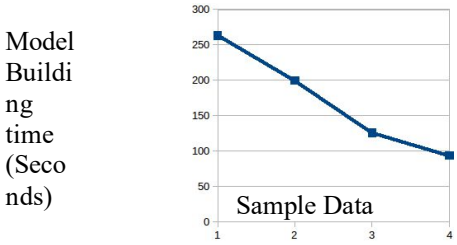

Figure 7. Model Construction analysis

C. Spark Cluster Scaling

The system generate amount of data is based on the available data reported by master on each slave node, the dynamic data optimization problem, can be solved using linear programing methods. Then, the slave requests

master data based in the optimization problems solutions. If the slave request is accept or reject on the master node to be process the data. Then some tasks are not assigned to the successfully, the master assign the task to large on small number slave processed.

Figure 8. Dynamic Data allocation and Task

D. **Data Communication Cost Analysis**

In the data communication cost can be analyzed on the scaling on the spark cluster in the IPRF algorithm. In the master node monitor the data size, then shuffled to the data among slave nodes. In this case, data size reduced ways to communicate the slave nodes in the cluster. Here, data is not transfered full size, the specified condition data of user request data to only.

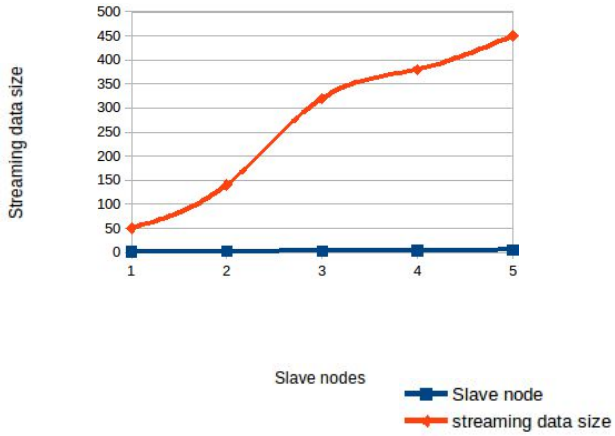

Figure 9. Data Communication Cost analysis of IPRF

VI. Conclusion

In this paper, an incremental-parallel random forest algorithms has been implemented for data streams. The classification accuracy of the IPRF algorithm is optimized through sampling and flitering method using weighted-voting method. Then, a incremental-parallel technique combining dynamic data allocation and task scheduling optimization is performed and implemented in a spark cloud computing environment. The feature subset selection have designed a framework for application oriented dynamic resource allocation, task scheduling and scalable data allocation algorithm to minimize the cluster deployment cost constrained on capacity and service delay bound and the size of the reduce task with task duration.

VII. Reference

[1] X. Wu, X. Zhu, and G.Q. Wu, "Data mining with big data," Knowledge and Data Engineering, IEEE Transactions on, vol. 26, no. 1, pp. 97-107, January 2014.

[2] M. M. Masud, Q. Chen, L. Khan, C. C. Aggarwal, J. Gao, J. Han, A. Srivastava, and N. C. Oza, "Classification and adaptive novel class detection of feature-evolving data streams," Knowledge and Data Engineering, IEEE Transactions on, vol. 25, no. 7, pp. 1484-1497, July 2013.

[3] Apache, "Hadoop," Website, January 2017, http://hadoop.apache.org.

[4] S. del Rio, V. Lopez, J.M. Benitez, and F. Herrera, "On the use of mapreduce for imbalanced big data using random forest," Information Sciences, vol. 285, pp. 112-137, November 2014.

[5] Apache, "Spark," Website, January 2017, http://spark-project.org

[6] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, October 2001.

[7] P. Domingos and G. Hulten, "Mining high speed data streams," In: Proceedings of the 6th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD), August 2000, pp 71–80.

[8] A. Tsymbal, "The problem of concept drift: definitions and related work," Technical report TCD-CS-2004-15, Computer Science Department, Trinity College Dublin, Ireland.

[9] G. Hulten, L. Spencer, and P. Domings, "Mining time-changing data streams," In: Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD), August 2001, pp 97–106.

[10] A. Bifet and R. Gavalda, "Adaptive Parameter-free learning from Evolving Data Streams,"

Technical report, Polytechnic University of Catalonia, 2009.

[11] H. Wang, W. Fan, P. Yu and J. Han, “Mining concept-drifting data streams using ensemble classifiers,” In: 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), August 2003, pp. 226-235.

[12] K. Kanoun and M. van der Schaar, “Big-data streaming applications scheduling with online learning and CDF detection,” In: Proceedings of the 2015 Design, Automation and Test in Europe Conference and Exhibition, EDA Consortium, March 2015, pp. 1547-1550.

[13] A. Ghazikhani, R. Monsefi, and H.S. Yazdi, “Online neural network model for non-stationary and imbalanced data stream classification,” International Journal of Machine Learning and Cybernetics 5, no. 1, pp. 51-62, February 2014.

[14] S. Tarkoma, C.E. Rothenberg, and E. Lagerspetz, “Theory and practice of bloom filters for distributed systems,” Communications Surveys Tutorials 14, IEEE, no. 1, pp.131–155, First 2012.

[15] I. Palit and C. K. Reddy, “Parallelized boosting with map-reduce,” In: Data Mining Workshops (ICDMW), 2010 IEEE International Conference on, pp. 1346-1353, December 2010.

[16] K. M. Svore and C. J. Burges, “Distributed stochastic aware random forest efficient data mining for big data,” in Big data (BigData Congress), 2013 IEEE International Congress on, Cambridge University Press, 2013, pp. 425-426.

[17] D. Warneke and O. Kao, "Exploiting dynamic resource allocation for efficient parallel data processing in the cloud," IEEE transactions on parallel and distributed systems 22, no. 6, pp. 985-997, June 2011.

[18] L. Chen, J. Zhang, L. Cai, Z. Deng, T. He, and X. Wang, "Locality-Aware and Energy-Aware Job Pre-Assignment for Mapreduce," In: Intelligent Networking and Collaborative Systems (INCoS), 2016 International Conference on, pp. 59-65. IEEE, 2016.

[19] S. Liu, K. Ren, K. Deng, and J. Song, "A dynamic resource allocation and task scheduling strategy with uncertain task runtime on IaaS clouds," In: Information Science and Technology (ICIST), 2016 Sixth International Conference on, pp. 174-180. IEEE, 2016.

[20] F. Zhang, J. Cao, W. Tan, S. Khan, K. Li, and A. Zomaya, "Evolutionary scheduling of dynamic multitasking workloads for big-data analytics in elastic cloud," IEEE Transactions on Emerging Topics in Computing 2, no. 3 pp. 338-351, August 2014.

[21] Apache Kafka, A high-throughput distributed messaging system source: http://kafka.apache.org/ accessed July 2015.