EasyChair Preprint

№ 2411

Performance Evaluation of Distributed Machine Learning for Load Forecasting in Smart Grids

Dabeeruddin Syed, Shady S. Refaat and Haitham Abu-Rub

January 18, 2020

Performance Evaluation of Distributed Machine Learning for Load Forecasting in Smart Grids

Dabeeruddin Syed*, Shady S. Refaat†, Haitham Abu-Rub†
Department of Electrical & Computer Engineering
*Texas A&M University, College Station, Texas, U.S.A.
†Texas A&M University at Qatar, Doha, Qatar
Emails: {dsyed}@tamu.edu,{shady.khalil,haitham.abu-rub}@qatar.tamu.edu

***Abstract*—Load forecasting in smart grid is the process of predicting the amount of electrical power to meet the short, medium and long term demands. Accurate load forecasting helps electrical utilities to manage their energy production, operations, control and management. Most of the state-of-the-art forecasting methodologies utilize classical machine learning algorithms to predict the electrical load. There is a need that big data platforms and parallel distributed computing are utilized to their potential in the available solutions. In this paper, the Apache Spark and Apache Hadoop are utilized as big data platforms for distributed computing in order to predict the load using available big data. In this paper, MLib, Apache Spark library for machine learning algorithms, is utilized for distributed computing. Using MLib allows testing the classic regression algorithms such as linear regression, generalized linear regression, decision tree, random forest and gradient-boosted trees in addition to survival regression and isotonic regression. The obtained results show that Apache Spark produces high accuracy while parallelizing the process of load forecasting in highly competent training and test times. Actual big data are used in the load forecasting process.**

***Keywords*—Apache Spark, Load Forecast, Distributed Machine Learning, Smart Grids, Distributed Computing.**

I. Introduction

With the prominence of information and communications technologies, the traditional electrical grid is being transformed to smart grid. Smart grid is supported by a huge number of smart meters, sensors, measurement units, etc. Those elements provide a continuous, massive amount of data at a high velocity, variety, and volume to support smart grid performance. The success of the future smart grid depends mainly on the effective utilization of the huge amount of the big data flow [1]. The data obtained from different smart grid sources satisfy all the big data characteristics. Owing to the peculiar characteristics of big data in smart grid, the data requires specialized technologies for proper management, storage, processing and visualization.

The data can reveal the hidden patterns and with the use of big data analytics tools, the correlation between different features and label can be revealed. Apache Hadoop is one of the big data analytics and management platform which is designed to overcome the challenges including scalability, storage bottleneck, handling variety of unstructured data, handling high rate of incoming data, etc. in big data analysis. It is used for the batch processing of offline data and it has its own inherent storage system called Hadoop Distributed File System (HDFS). This makes Hadoop ecosystem highly fault-tolerant by storing the data across several nodes in a systematic order eliminating all possible data losses in the event of system crash. Hence, applications using Hadoop framework have capability of parallel processing. Apache Spark is an in-memory, iterative and streams-processing platform that works on top of Hadoop or other big data systems and can help in the streams processing of data in real-time.

There are many aspects related to the management of generation and consumption of energy which should be taken care by electrical utilities. Load forecasting is a crucial decision support tool that allows for efficient energy management in smart grid. Demand information can be utilized by the customers in the near future, while the electrical utilities can make important decisions based on it like purchase and sale of electrical energy, load switching and capacity planning. Even a little improvement in load forecasting accuracy for the energy domain leads to big savings with great economic and environmental benefits. Load forecasting can be divided into three categories: short-term (ranges from an hour to 7 days), medium-term (ranges from 7 days to 1 year) and long-term (ranges longer than one year) forecasting, depending on the period of forecasting. Highly accurate load forecasting will enable the utilities to minimize the risks by understanding the future long-term energy demand and planning their capacity and infrastructure to meet the demand.

However, there are many challenges related to load forecasting. Weather plays a prominent role in energy consumption and could be unpredictable. Currently, load forecasting is based on methodologies which are not sustainable. Moreover, acquiring accurate data on consumer behavior is extremely difficult. It is extremely difficult to accurately fit large number of complex features into the forecasting model using classical modeling methods. These challenges call for a real-time streaming load forecasting platform that can predict load in real-time which is the goal of this paper.

The effectiveness of the application of distributed machine learning (ML) algorithms using Apache Spark is employed for load forecasting in this paper. A cluster of Hadoop, managed by HDFS and Yet Another Resource Negotiator (YARN) along with Spark, is set up for executions and testing. MLib library of Spark is utilized to implement the ML algorithms and the datasets analyzed are the English smart meters data [2]. Sequential forward search is used for feature selection. Therefore, the main aim of this paper is to obtain the metrics of application of ML techniques in distributed way using Apache Hadoop and Spark. The paper proposes a big data management platform to perform the load predictions while streaming the

data in real-time and it also evaluates the performance of MLib library of the proposed platform.

The rest of the paper is organized as follows. Section II describes the relevant literature while Section III presents the different components of our work and how these relate to each other. Furthermore, Section V illustrates the results of the metrics of distributed computing using ML for load forecast. Finally, Section VI discusses the conclusion and future work.

II. Related Work

Many ML techniques have been developed to provide efficient load forecasting in smart grid. In [3], Zhang et al. applied artificial neural network (ANN) for load forecasting. They compared the mean square error (MSE) of load forecasting with respect to the number of neurons in the three hidden layers of the model. A 5.5% error (M.S.E.) has been achieved with their model. Their work considered the energy demand and weather data in Ontario province of Canada.

In [4], Dong et al. worked with the clustering of the load datasets based on the k-means clustering algorithm. The different clusters of datasets were passed to train the convolutional neural network. This methodology performed well with the dataset containing even 1.4 million of energy records in terms of accuracy (with a Mean Absolute Percentage Error of 3.05 %). However, the execution times of the modeling for load forecasting have not been provided.

In [5], Gajowniczek et al. proposed a short-term load forecasting model using ANN and support vector machines (SVM) for day-ahead predictions at individual household level. The accuracy of the proposed ANN model and SVM has been reported as 62% and 60% respectively. Edwards et al. [6] reported that SVM performed better than ANN based methods for predictions of hourly electrical consumption at residential buildings level. Whereas, the ANN model performed better in case of commercial buildings. The load in residential buildings is more unpredictable than the load in commercial buildings. Commercial buildings have occupancy during specific times and those have more or less similar patterns of daily occupancy. These might be the reasons for the differences in performance of ANN based models and SVM. Other research [7], [8] has also indicated that SVM based regression outperforms ANN models in predicting load especially on smaller data sizes. Also, SVM has lower running times than neural-network based models on larger datasets because SVM are conceptually simpler and yield better to post-hoc analysis [9].

Few of the methods utilized to enhance the forecasting accuracy include detection of regular patterns using cluster-based aggregate forecasting [10], neural-network based predictions after clustering of customers [11], consideration of socio-economic features like population, inflation and national energy statistics [12], etc. Indeed, most of the works have focused on increasing the accuracy of the model, however, the execution times (training time and testing time) have not been given much consideration.

This paper is focused on the optimization of trade-off between the accuracy of the model, execution time, and computation resources. Considering the nature of load data from smart grids, the proposed big data management platform is intended to perform the load predictions while streaming the data in real-time.

III. Methodology for Performance Evaluation

This section provides the information of the obtained data and the applied methodologies. The first step in the methodology is the data acquisition and pre-processing of data. The first subsection III-A describes the data and the pre-processing steps in detail. Performing the load forecasting with distributed and parallel computations, it is required that the methods are performed on adequate and competent platforms. The subsequent subsection III-B describe the big data platforms that can be used for distributed ML. The concluding subsection III-C discusses the job schedulers that are generally used to optimize the jobs executed on big data platforms.

A. Data Description

The smart meters energy consumption data from London households [2] has been used in this work. The energy consumption values have been recorded between November 2011 and February 2014 with frequency of half-hour for 5567 households. However, it was noted that the energy consumption values for all the households are not available since the start of November 2011. Hence, the data has been aggregated to energy consumption values per day per household (which is the label in our case). At data acquisition stage, the only features in data are time and energy consumption values. However, the weather data is a crucial factor for the energy consumption. Thus, we scraped the weather data using a web API darksky.net and merged it with the England dataset [13]. This gave us many helpful weather features. Next step was the application of generalized forward feature selection algorithm which helps to select the best features. The features finally used in the analysis include the following: maximum temperature, minimum temperature, dew point, UV Index, humidity, cloud cover, visibility, wind speed, pressure, wind bearing, moon phase, time, holiday index and censor (used only for survival regression).

The weather data collected from weather API contained missing values for the features such as dew point, pressure, wind speed, etc. Dealing with missing values was one of the steps in pre-processing of data. The imputation methods like mean, mode, median imputation methods and forward or backward fill are used to fill the missing values [14]. The pre-processing steps on the data is illustrated in the Figure 1.

As seen in the Figure 1, feature selection is performed to select the best features that return the least Root Mean Square Error (RMSE). values after prediction. In this work, Sequential Forward Search method is used for feature selection. Also, outlier detection is performed to remove the records which has zero value as energy consumption. The zero energy consumption shows either mis-recording of energy values or blackout. In any case, the zero values are ignored as the aim is to generalize the correlation between features and label. For this, specific events like faults, blackouts or mis-recording are ignored. In the next step, one of the models among linear regression (LR), generalized linear regression (GLR), decision tree (DT), random forest (RF), gradient-boosted trees (GBT), survival regression (SR) or isotonic regression (IR), is selected

for experiment. When an experiment is performed, the parameters, associated with the regression technique, are optimized. The final step is the evaluation of the regression techniques using performance metrics like RMSE, Mean Absolute Error (M.A.E.), training time, prediction time, etc.

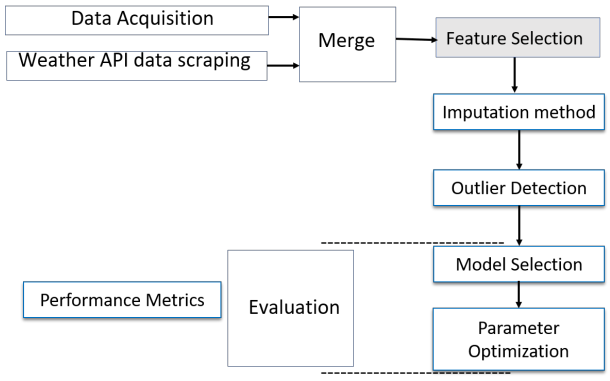

Fig. 1: Pre-processing steps in the load forecasting modeling

B. Software Platforms

1) Hadoop: Apache Hadoop is an open-source big data management and computing platform useful for batch processing of offline data [15]. It works on the MapReduce programming paradigm. It is highly scalable, reliable and its library is suited for distributed processing of large data sets that can be partitioned across multiple nodes in a cluster which suits smart grid data character. It has the following modules, namely:

a) Hadoop Common: It is a common utility which supports all the other modules of Hadoop through its inbuilt libraries. It is also called Hadoop Core since it is the base of the Hadoop framework through its essential services and base processes like abstraction of the operating system, abstraction of the file system, etc. It contains Java Archive (JAR) files and scripts which are essential for the Hadoop startup.

b) Hadoop Distributed File System (HDFS): HDFS is a distributed file system that furnishes high throughput access to the data across different nodes in the cluster. HDFS is a primary storage system for Hadoop and can also be used with Apache Spark. It is based on master-slave architecture. A master name node distributes the data across compute nodes throughout the cluster, maintains the file system namespace and also regulates the access to the data by clients.

c) Hadoop YARN: YARN is a resource manager and job scheduler, which is part of Hadoop environment. It will be discussed in detail in the subsequent section.

d) Hadoop MapReduce: MapReduce is a programming paradigm of Hadoop environment. It is based on two functions map and reduce. Here, map refers to a base function that applies to each element in a data array and reduce refers to an aggregate function on a data array. MapReduce paradigm is responsible for the distributed processing of datasets partitioned across the data nodes. It provides fault tolerance and abstraction of data. It identifies data corruption and quickly applies fix for an automatic recovery solution. Abstraction of data into specific and appropriate data structure is fundamental to speed up the execution of jobs in Hadoop and to take advantage of internal optimizations provided by the data platform.

2) Spark: Spark is an open source java-based distributed data processing framework that is useful for big data applications and as per the description, it is ten times faster than Apache Hadoop MapReduce paradigm when used for in-memory processing. The high speed of Spark is due to the fact that the data processing happens in the main memory of the worker nodes and thus, it completely avoids the I/O operations from disk. The major components of Apache Spark include library to build ML models called MLib, stream processing component (Spark Streaming) and graph processing component called GraphX. Apache Spark satisfies different types of processing like batch processing, interactive queries, iterative applications and streaming applications. These types of processing previously were dealt by separate engines but Spark provides these functionalities using one engine. When it comes to big data, the speed and processing time are of high importance and Spark ensures this with large datasets and its ability to run computations in memory [16].

Spark uses Resilient Distributed Datasets (RDD) to renders distributed datasets partitioned across computing nodes. The information of the partition locations and metadata are also stored in RDDs. RDDs are abstraction of distributed datasets. When transformations are applied on RDDs, these are converted to new RDDs in which the transformations are stored as function object. The dependency is also registered in the new RDDs. When multiple RDDs are joined together to form new RDD, the new one will inherit all the dependencies from the parent RDDs. The most often utilized transformations on RDD are mapping, joining and filtering. These transformations render parallel processing of the RDD partitions, joining of RDD partitions and filtering of partitions based on any conditions [17].

C. Job Schedulers

Usually, the job schedulers are classified into two families namely, High Performance Computing (HPC) for long duration simultaneous distributed simulation and modeling jobs and Big Data schedulers (BD) for short running simultaneous high-performance data processing jobs. The classical HPC schedulers [18] [19] include Grid Engine, HTCondor, Load Sharing Facility (LSF), OAR and Portable Batch System (PBS).

The newer high-performance computing schedulers are Cray Simple Linux Utility for Resource Management (SLURM) [20] and Application Level Placement Scheduler (ALPS) [21]. Whereas the big data schedulers include commercial schedulers [19] like Google Borg, Google Omega and Google MapReduce, and open source schedulers like YARN [22] and Mesos [23].

The most used job schedulers are described as below:

1) LSF: LSF is Load Sharing Facility resource manager and job scheduler developed by Platform Computing for distributed computing processes [19]. It is a full-featured scheduler and has been succeeded by the open-source version of itself called OPENLAVA.

2) Mesos: Apache Mesos is an open-source job scheduler used as manager of resources in computer clusters [23]. It was developed by University of Berkeley and it uses Linux Control Group (cgroup) to isolate CPU, file system, I/O and memory. Apache Mesos works on two levels - firstly, it partitions the compute nodes into scheduling domains and each scheduling domain is managed by Mesos framework which then allocates the resources to each of the scheduling domains.

3) SLURM: SLURM is a free and open-source resource manager for linux and unix-like kernels [20]. It is widely used on about 60% supercomputers in the world. It is highly scalable and can manage up to 1000 job submissions per second. It has support for job profiling and job arrays.

4) YARN: YARN is job scheduler and cluster manager in Hadoop environment [22]. The high efficiency and speed of YARN exist due to the fact that it evokes different daemons to perform the job scheduling and cluster management tasks. For every application, there is a resource manager and application master assigned. This improves scalability and utilization efficiency in the cluster.

IV. Case Study with Experimental Model

The experiment cluster used in this case study of load forecasting in smart grids, consists of fourteen nodes. The hardware specifications of the cluster are described in the Table I:

TABLE I: Cluster Setup

Hardware Specification	**Value**
Nodes	16
Interconnect	Onmi-Path
CPU Architecture	Intel Broadwell x86_64 CPU operating at 2.4 GHz
CPU cores	8 per node
Memory	16 GB RAM per node
Job scheduler	Slurm

The operating system in all the nodes is CentOS Linux 7 (Core) and JAVA version installed is 1.8.0. The installed version of Hadoop is 2.7.0 and the version of Apache Spark is 2.4.0. The architecture of Spark used with Slurm job scheduler is shown in Figure 2. The master node will drive the Spark Context which is the main program. The job scheduling and cluster management are performed by Slurm. We specify the number of nodes, the number of cores and the memory to be utilized in the slurm job while the job scheduler assigns the appropriate and desired resources from the cluster to the job. The worker nodes are evoked and the job is performed.

The cluster of nodes described above provided a platform for the performance evaluation of distributed load forecasting using distributed ML. Also, the use of spark provides library called MLib which has inherent capabilities for distributed ML. The parallel processing is required in real-world forecasting application as smart grids generate high volume of energy consumption data at high velocity from variety of sensors connected to them.

The checkpointing and profiling have been performed to estimate time taken for reading, training, testing and evaluation of the models, the results of which will be described in the subsequent section.

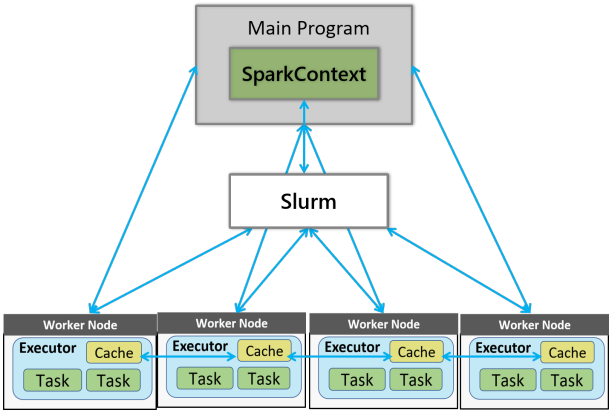

Fig. 2: Spark architecture with Slurm Job Scheduler

A. ML Algorithms in Apache Spark

For the applications of regression, not all the ML algorithms are implemented in the Apache Spark library. For example, ANN, recurrent neural networks (RNN), etc. are not available with MLib library. The following are the ML algorithms available for regression with Spark MLib [24]:

1) Linear Regression (LR): LR is one of the most popular methods of predicting values as a label. The output with LR is assumed to follow a gaussian distribution. The hypothesis function in LR is given by:

$$h_\theta(x) = \theta_0 + \theta_1 x \tag{1}$$

where x is a vector of features,
θ_0 and θ_1 are parameters of the model.

The cost function quantifying the error is given by:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{k=1}^{m} (h_\theta(x^{(k)} - y^{(k)})^2)) \tag{2}$$

where m denotes the number of training samples and
y^k denotes the vector of target variables.

The optimization objective of LR is given by the following:

$$Objective : min_{\theta_0,\theta_1} J(\theta_0, \theta_1) \tag{3}$$

2) Generalized linear regression (GLR): The output in the GLR follows distribution from the exponential family of distributions like normal, poison, exponential, gamma, chi-squared, bernoulli, dirichlet, geometric, wishart, etc.

3) Decision tree: DT is a non-parametric supervised learning method which predicts the variables based on the splitting conditions it has learned from the features of data. The deeper the tree is, the complicated the decision rules are and the fitter the model is. The advantages of DT are that it is simple to interpret, visualize, requires little data preparation and can handle both numerical and categorical features. The split points or decision rules for DTs are given by the following equation:

$$h(x) \leq 0 \tag{4}$$

where h(x) is the hyperplane that specifies the split point and h(x) is given by

$$h(x) : \theta^T x + b = 0 \tag{5}$$

where θ^T represents the transpose of weight matrix and b represents the bias.

At each of the split points, the binary partition which gives the maximum purity as per the threshold set is finally confirmed as the node of a DT. The purity of a region is defined as the fraction of points with the majority label.

4) Random Forest: RF works well with larger datasets and is a collection of trees which each yield predicted values. It is aptly named random because it randomly selects a portion of dataset for training. Each of the DTs in the forest makes decision which is then voted to yield the final predictions. The greater number of trees a model has, the higher the accuracy of regression will be.

5) Gradient-boosted trees: GBT are bagging collection of DTs. However, unlike RF, the predictions made by one tree are not independent of predictions made by other trees. The process of prediction is iterative i.e. predictions made by one tree are picked by other trees to make their predictions. Intuitively, this reduces the RMSE.

6) Survival regression: SR makes use of additional data apart from the covariates that we regress against a variable. It makes use of censoring vector that describes the event status of an observation (value 1 indicates that the event has occurred and value 0 indicates that the event is censored). In our work, we have used all the values in censor vector to be 1 assuming all the events have occurred. There are three major types of models in SR namely, Aalen's additive model, accelerated failure model and Cox's model.

7) Isotonic regression: IR is curve fitting technique closely similar to LR. However, it is free-form curve fitting such that the fit is not non-increasing or non-decreasing everywhere and is as close to the actual observations. The advantage is that IR does not constraint itself to present a target function for the data such as LR presents linearity as a target function.

The Spark platform and its component MLib have been evaluated for performance accuracy and execution time using the following metrics:

1) Mean Absolute Error (MAE): It is given by the average of the error values iterated over all the predicted values. It does not represent error as good as RMSE under the condition that data follows gaussian distribution.

$$M.A.E. = \frac{1}{m} * \sum_{k=1}^{m} |e_k| \tag{6}$$

2) Root Mean Square Error (RMSE): It is a measure of average error. However, it represents three characteristics of set of errors namely, variance in the distribution of inaccuracy values and with the square root of the number of records ($\sqrt{m}$) and average-error. It is considered to be very sensitive to outliers when compared to MAE.

$$R.M.S.E. = \sqrt{\frac{1}{m} * \sum_{k=1}^{m} e_k^2} \tag{7}$$

V. Evaluation Results

In this section, we present and discuss the results of the load prediction in smart grids. The models or predictors have been evaluated in terms of RMSE, MAE, training time (s) and testing time (s) using the smart meter energy consumption dataset from England households.

TABLE II: RESULTS

ML Algorithm	RMSE	MAE	Training time (s)	Prediction time (s)
LR	0.3729	0.3135	1.6114	0.2288
GLR	0.3679	0.2865	0.8782	0.2362
DT	0.4825	0.3415	1.2334	0.2715
RF	0.3652	0.2821	1.6614	0.2780
GBT	0.5098	0.3554	6.2548	0.2873
SR	0.4279	0.3649	3.5008	0.3649
IR	0.6836	0.5123	0.8599	0.2741

In this work, we have assessed the performance of load prediction models on big data processing platform and these models are based on classical ML models namely LR, GLR, DT, RF, GBT, SR and IR. The results of performance in terms of RMSE, MAE, training and testing times are illustrated in the Table II, Figure 3 and Figure 4.

It is evident that the RF performs better for load prediction than all the other regression models. This is due to the fact that the RMSE of RF is 0.3652 while the regressors like GLR and LR have the RMSE of 0.3679 and 0.3792 respectively. Although, GLR and RF have similar accuracy in terms of RMSE, it is evident that GLR performs in less duration of time. The training time is the least for IR (0.85 s) followed by GLR which has training time of 0.878 s. And it is observed that the most accurate RF regression has a higher training time of 1.66 sec which is about 2 times greater than the fastest regression models.

When it comes to the prediction time, the linear regression is the fastest of all the regression models with a prediction time of 0.228 sec. However, it is closely followed by GLR with a prediction time of 0.236 sec. It is clear that the Apache Spark platform provides a parallel and distributed computing platform for the ML models to perform in faster times than in standalone machines. However, it is to be noted that the data size should be of larger size so that the time to distribute the resources is less when compared to the actual processing time. In such environment, Apache Spark with its distributed computing and parallel processing abilities can provide real-time processing platform for load prediction in smart grids.

VI. Conclusions and Future Work

This paper presents the performance of ML algorithms using MLib library of Apache Spark. According to the results, it is clear that the distributed computing of load forecasting provides satisfactory performance in terms of accuracy and computation times.

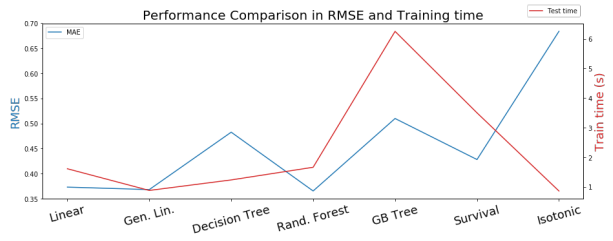

Fig. 3: Performance comparison in terms of RMSE and Training time

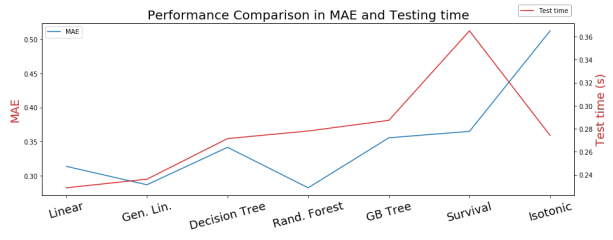

Fig. 4: Performance comparison in terms of MAE and Testing time

In future work, the experiments would be extended to deep learning techniques in Apache Spark. However, currently there are no in-house implementations of deep learning libraries in MLib library of Apache Spark for regression. We would also like to extend our analysis using different job schedulers like Apache Mesos, YARN and LSF to improve the performance of load forecasting in terms of computation time. Also, the work can be further extended towards the optimization of the number of nodes, number of cores and memory provided. The comparison of performance in terms of accuracy and computation times against different number of nodes, cores and memory will help in the computation resource management for distributed computing.

VII. Acknowledgment

This publication was made possible by NPRP grant [NPRP10-0101-170082] from the Qatar National Research Fund (a member of Qatar Foundation), the co-funding by IBERDROLA QSTP LLC and sponsorship by Texas A & M Energy Institute Fellowship. Portions of this research were conducted with the advanced computing resources provided by Texas A & M High Performance Research Computing. The statements made herein are solely the responsibility of the authors.

References

[1] Amira Mohamed, Shady S Refaat, and Haitham Abu-Rub. A review on big data management and decision-making in smart grid. *Power Electronics and Drives*, 4(1):1–13, 2019.

[2] London datastore. https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households.

[3] Hao-Tian Zhang, Fang-Yuan Xu, and Long Zhou. Artificial neural network for load forecasting in smart grid. In *2010 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3200–3205. IEEE, 2010.

[4] Xishuang Dong, Lijun Qian, and Lei Huang. Short-term load forecasting in smart grid: A combined cnn and k-means clustering approach. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 119–125. IEEE, 2017.

[5] Krzysztof Gajowniczek and Tomasz Zabkowski. Short term electricity forecasting using individual smart meter data. *Procedia Computer Science*, 35:589–597, 2014.

[6] Richard E Edwards, Joshua New, and Lynne E Parker. Predicting future hourly residential electrical consumption: A machine learning case study. *Energy and Buildings*, 49:591–603, 2012.

[7] Maheen Zahid, Fahad Ahmed, Nadeem Javaid, Raza Abid Abbasi, Zainab Kazmi, Hafiza Syeda, Atia Javaid, Muhammad Bilal, Mariam Akbar, and Manzoor Ilahi. Electricity price and load forecasting using enhanced convolutional neural network and enhanced support vector regression in smart grids. *Electronics*, 8(2):122, 2019.

[8] Changhao Xia, Mi Zhang, and Jin Cao. A hybrid application of soft computing methods with wavelet svm and neural network to electric power load forecasting. *Journal of Electrical Systems and Information Technology*, 5(3):681–696, 2018.

[9] Hai-xiang Zhao and Frédéric Magoulès. A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, 16(6):3586–3592, 2012.

[10] Tri Kurniawan Wijaya, Matteo Vasirani, Samuel Humeau, and Karl Aberer. Cluster-based aggregate forecasting for residential electricity demand using smart meter data. In *2015 IEEE international conference on Big data (Big data)*, pages 879–887. IEEE, 2015.

[11] Abbas Shahzadeh, Abbas Khosravi, and Saeid Nahavandi. Improving load forecast accuracy by clustering consumers using smart meter data. In *2015 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2015.

[12] Arnaud Grandjean, Jérôme Adnot, and Guillaume Binet. A review and an analysis of the residential electric load curve models. *Renewable and Sustainable energy reviews*, 16(9):6539–6565, 2012.

[13] Dark sky api - weather conditions. https://darksky.net/dev/.

[14] Jadran Sessa and Dabeeruddin Syed. Techniques to deal with missing data. In *2016 5th international conference on electronic devices, systems and applications (ICEDSA)*, pages 1–4. IEEE, 2016.

[15] Apache Hadoop. Apache hadoop & yarn, 2016.

[16] Wei Huang, Lingkui Meng, Dongying Zhang, and Wen Zhang. In-memory parallel processing of massive remotely sensed data using an apache spark on hadoop yarn model. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(1):3–19, 2016.

[17] Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia. *Learning spark: lightning-fast big data analysis*. " O'Reilly Media, Inc.", 2015.

[18] Albert Reuther, Chansup Byun, William Arcand, David Bestor, Bill Bergeron, Matthew Hubbell, Michael Jones, Peter Michaleas, Andrew Prout, Antonio Rosa, et al. Scheduler technologies in support of high performance data analysis. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2016.

[19] Albert Reuther, Chansup Byun, William Arcand, David Bestor, Bill Bergeron, Matthew Hubbell, Michael Jones, Peter Michaleas, Andrew Prout, Antonio Rosa, et al. Scalable system scheduling for hpc and big data. *Journal of Parallel and Distributed Computing*, 111:76–92, 2018.

[20] C Hollowell, W Strecker-Kellogg, J Barnett, A Zaytsev, C Caramarcu, and A Wong. Mixing htc and hpc workloads with htcondor and slurm. In *J. Phys. Conf. Ser.*, volume 898, page 082014, 2017.

[21] Travis Newhouse and Joseph Pasquale. Alps: An application-level proportional-share scheduler. In *2006 15th IEEE International Conference on High Performance Distributed Computing*, pages 279–290. IEEE, 2006.

[22] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM, 2013.

[23] Roger Ignazio. *Mesos in action*. Manning Publications Co., 2016.

[24] Apache Spark. Classification and regression. https://spark.apache.org/docs/latest/ml-classification-regression.html.