Forecasting Smart Meter Energy Usage using Distributed Systems and Machine Learning

Chris Dong*, Lingzhi Du*, Feiran Ji*, Zizhen Song*, Yuedi Zheng*, Alexander J. Howard*,
Paul Intrevado, Diane Myung-kyung Woodbridge
{cadong,ldu4,fji3,zsong11,yzheng41,ajhoward7,pintrevado,dwoodbridge}@usfca.edu
Data Science Program
University of San Francisco

***Abstract*—In this research, we explore the technical and computational merits of a machine learning algorithm on a large data set, employing distributed systems. Using 167 million (10 GB) energy consumption observations collected by smart meters from residential consumers in London, England, we predict future residential energy consumption using a Random Forest machine learning algorithm. Distributed systems such as AWS S3 and EMR, MongoDB and Apache Spark are used. Computational times and predictive accuracy are evaluated. We conclude that there are significant computational advantages to using distributed systems when applying machine learning algorithms on large-scale data. We also observe that distributed systems can be computationally burdensome when the amount of data being processed is below a threshold at which it can leverage the computational efficiencies provided by distributed systems.**

***Index Terms*—Distributed computing, Distributed databases, Machine learning, Data processing, Smart grids.**

I. Introduction

A smart meter is an electronic device that records energy consumption in real time and transmits that information to a utility company for monitoring and billing. Smart grids, when coupled with smart meters and weather forecasts, monitor aggregate energy consumption, and can dynamically manage energy supply and set pricing. The number of smart meter installations has grown significantly over the last decade. The European Union (EU) aims to replace at least 80% of traditional electric meters with smart meters by 2020, as part of a set of measures to upgrade energy supply and tackle climate change [1].

Leveraging smart meter data to help raise consumer awareness of energy consumption patterns can be a crucial part of smart-grid analytics applications. Using demand-side management, utility companies can incent consumers to lower energy consumption during peak demand periods by analyzing residential smart meter data. Critical in this effort to smooth demand is the development of an accurate and efficient forecast model for energy consumption.

Utilizing smart meter data and other contributing factors—weather, consumer activity patterns, inflation, national energy statistics, and demographics—researchers have employed smart meter data for various applications. Gajowniczek used artificial neural networks and support vector machines in conjunction with time series analysis techniques to forecast residential consumption using high resolution smart meter data and weather data [2]. Edwards compared various machine learning algorithms including regression, neural networks, and support vector machines to predict consumption using data collected from individual smart meters and weather data [3]. Wijaya sought to improve forecast accuracy of residential energy consumption by detecting habitual patterns using Cluster-Based Aggregate Forecasting (CBAF) [4]. Shahzadeh improved the accuracy of neural network-based prediction methods by clustering consumers [5]. Grandjean evaluated international consumption patterns by considering additional external variables including population, inflation, and national energy statistics in a top-down model [6]. Bottom-up models employed regional variables such as household demographic, appliance ownership and use information [7].

Smart meter systems generate energy consumption records in real time and transmit data several times daily. It is therefore important to establish an efficient method by which to store and retrieve such large volumes of data. Moreover, the development of forecasting models that provide accurate, real-time predictions is critical to the efficient operation of a smart grid. We develop and detail a scalable machine learning model that is able to predict residential energy consumption using distributed databases, distributed computing and machine learning.

In this research, we develop a forecasting model capable of predicting single-day energy consumption. Data is stored in Amazon Web Service (AWS) Simple Storage Service (S3), a scalable storage infrastructure. Distributed database systems and distributed computing allow multiple systems to work together and function in an efficient, affordable, scalable, and reliable manner. Most modern distributed databases and computing frameworks are based on a map-reduce model, which allow data to be mapped to different nodes, processed in parallel, and subsequently merged using a reduce function [8]. We used MongoDB, a non-relational distributed database (NoSQL) for storing and accessing data in a distributed manner. Data processing and machine learning was performed using Apache Spark on AWS Elastic Map Reduce (EMR). The performance of EMR was tested and compared under differing configurations. Forecasting was

* These authors contributed equally.

based on a Random Forest model which reuses data, enabling Spark to significantly reduce computational time.

II. System Overview

A. System Workflow

If each of the anticipated 53 million smart meters to be installed in homes across the United Kingdom [9] were to report a residence's kilowatt-hour (kWh) every half hour, this would equate to 2.5 trillion records daily. Collecting, managing and analyzing data on this scale necessitates efficient, reliable and scalable algorithmic infrastructure. To develop such a system, we stored data in Amazon Web Service (AWS) Simple Storage Service (S3), loaded the data into MongoDB on AWS Elastic Compute Cloud (EC2), and processed data using Apache Spark on AWS Elastic MapReduce (EMR). Figure 1 depicts an overview of the system workflow.

AWS S3 is a robust and scalable object storage system that is automatically extensible if additional storage is required. S3 uses the world's largest cloud service, and data is automatically distributed over multiple physical facilities. AWS S3 facilities are located in over 17 regions throughout the world including several European cities such as Frankfurt, Ireland, Paris and London. S3 additionally provides a high level of data security, including encryption and machine learning for anomaly detection [10].

Equally as important as scalable data storage is a scalable data management solution. Moreover, a database that supports schemaless data is valuable given the propensity for structural changes in the data. MongoDB is an open source, non-relational and schemaless database that runs on distributed systems. MongoDB automatically loads data across a cluster and, if necessary, can redistribute and rebalance it [11]. MongoDB represents data in a format called JavaScript Object Notation (JSON) which is both human and machine-readable, and is commonly used for data exchange on the web. JSON also supports various data types including numbers, strings, boolean, and even arrays and hashes, which relational databases do not support.

In this study, we transfer data from S3 to MongoDB, installed on a Linux EC2 instance. AWS EC2 provides a cloud-based virtual machine instance with auto-scaling features, scaling the number of instances to ensure high throughput [12]. Figure 2 depicts a sample document loaded into MongoDB.

To query and process data stored in MongoDB, we utilized Apache Spark, an extension of the MapReduce model. Spark partitions data into multiple nodes in a cluster to process data, including the parallelization of machine learning algorithms across the cluster. Spark represents the data as a Resilient Distributed Dataset (RDD), which is an abstraction of a distributed collection of data with operations applicable to the data, and provides networking, security, scheduling and data shuffling functions. Spark plans stages of tasks using a directed acyclic graph (DAG) scheduler for efficient data processing across nodes.

MapReduce is a programming model which distributes data into a number of machines and utilizes CPUs for processing data in parallel. As MapReduce can be implemented using commodity machines, it is cost-efficient and also recovers quickly from a single node failure by replicating data or adding a new machine. Hadoop MapReduce, an implementation of MapReduce, is widely used. By keeping data in random access memory (RAM), using RDD, and planning and managing tasks efficiently, Spark outperforms Hadoop MapReduce by upto 100 times [13]. This efficient data processing and scheduling led us to employ Spark for our iterative machine learning algorithms [14].

Several Spark modules, including Spark SQL and Spark ML were employed. Spark SQL is a module that allows for manipulating large sets of distributed and structured data called DataFrames. Similar to a table in a relational database management system (RDBMS), a DataFrame includes columns and observations, and additionally facilitates running queries in a parallelized fashion [13][15]. Moreover, Spark ML—the machine learning library in Spark— requires DataFrames as input. We therefore loaded our data from MongoDB into a Spark DataFrame and applied feature engineering and machine learning algorithms.

AWS Amazon Elastic MapReduce (EMR) provides a Hadoop framework for processing vast amounts of data easily, quickly, and cost-effectively across dynamically scalable Amazon Elastic Computing Cloud (EC2) instances. In addition, EMR uses YARN (Yet Another Resource Negotiator) as a cluster manager. YARN is Hadoops resource manager and execution system, and is able to run not only Hadoop MapReduce but also Spark. On each cluster node, EMR launches a system agent that administers YARN components, communicates with the EMR service to coordinate jobs, and tracks the status of a cluster [16].

B. Algorithm

Traditional parametric time series models, such as an autoregressive integrated moving average (ARIMA), deal with a singular time series. As the energy usage of each household is itself a singular time series, these models do not work well when forecasting in the aggregate. We therefore employed a Random Forest model using a set of 79 engineered features to predict single-day consumption, which worked well to capture general energy usage patterns for residential customers.

1) Feature Engineering: Data collected within a sliding time window (w) prior to the given day to be predicted were used as features. The time horizon of the sliding time window, w (in days), is a hyper-parameter in the model that can be tuned to improve model accuracy. The reader may assume $w = 10$ days for ease of exposition.

The first set of engineered features are daily means of electrical consumption. Each day, energy consumption is collected on a half-hourly basis, resulting in 48 readings per day. We take the mean of these 48 readings each day to generate a daily

Fig. 1: System Workflow

```
> db.energy.findOne()
{
        "_id" : ObjectId("5a5ff09353609bf85037a7c2"),
        "" : 0,
        "LCLid" : "MAC000002",
        "day" : "2012-10-13",
        "hh_0" : 0.263,
        "hh_1" : 0.269,
        "hh_2" : 0.275,
        "hh_3" : 0.256,
        "hh_4" : 0.211,
        "hh_5" : 0.136,
        "hh_6" : 0.161,
        "hh_7" : 0.119,
        "hh_8" : 0.16699999999999998,
        "hh_9" : 0.109,
        "hh_10" : 0.168,
        "hh_11" : 0.107,
```

Fig. 2: Example document, where "energy" is the name of the collection.

Energy Use per Half-Hour (Wide)

id	day	hh_0	hh_1	...	hh_47
MAC000002	2012-02-01	0.263			

(a) Daily energy usage which include 48 half-hourly data.

Household Info

id	block
MAC000002	0
MAC000003	1

(b) Household information.

Weather

timestamp	temp_max	temp_min
11/11/12 23:00	11.96	3.29

(c) Weather history and forecast information.

Fig. 3: Input data: 79 features are generated after feature engineering and used for bi-hourly predictions.

mean, and repeat this over the sliding time window w—which in this case we assume to be the past ten days—to generate our first ten features. The second set of features captures the minimum and maximum temperature over the previous ten days, resulting in twenty additional features.

The third feature set is mean energy use in a fixed half-hourly period across the sliding time window. To compute this, we choose a fixed half-hourly time period, e.g, 1:30pm-2:00pm. We then computed the mean energy consumption in that half-hour period by averaging the energy consumption in that time period across the previous 10 days, generating a single feature. This feature is meant to capture seasonality across the time series. There are 48 half-hour increments in a day, resulting in 48 additional features.

A final categorical feature was also included indicating household income. Studies show a strong relationship between wealth (per capita GDP) and energy consumption (per capita kWh/year) across 168 countries [17]. We used Acorn data, a widely-known geodemographic segmentation of the UK population developed by CACI Limited, which categorizes households based, in part, on income. The three income classifications are Affluent, Comfortable, and Adversity, representing wealth in decreasing order. Features are summarized in Table I.

2) Machine Learning Model: Random Forest is an ensemble learning method that generates multiple decision trees and aggregates their results. Each decision tree is generated independently from different bootstrapped samples of data, and their outputs are aggregated either by simple averaging or majority voting [18].

For a Random Forest regressor, we denote X as input and y as the response. We bootstrap n samples $(\tilde{X}_1, \tilde{y_1}), (\tilde{X}_2, \tilde{y_2}), \cdots, (\tilde{X}_n, \tilde{y_n})$ from (X, y) where n is a hyper-parameter defining the number of trees. For each bootstrapped sample, a decision tree h_k is trained to minimize the mean square error $\sum(\tilde{y_k} - h_k(\tilde{X}_k))^2$. A trained Random Forest regressor consists of n trained decision trees $h_1, h_2, \cdots, h_n$.

To generate a prediction for new input (test) data x_{new}, the Random Forest algorithm returns the average of predictions of all trees in Equation 1.

TABLE I: Features of the Random Forest Model

# of Features	Description
w	Daily Mean Electrical Consumption
w	Daily Minimum Temperature
w	Daily Maximum Temperature
48	Half-Hourly Mean Electrical Consumption
1	Neighborhood
$49 + 3w$	**TOTAL**

$$\hat{y} = \frac{h_1(x_{new}) + h_2(x_{new}) + \cdots + h_n(x_{new})}{n} \quad (1)$$

Random Forest has been proven to improve performance and reduce variance compared to a single decision tree, and is also robust to outliers and missing values [19]. For these reasons, Random Forest is a widely-used algorithm in many applications.

To avoid overfitting on a specific day, we repeated the aforementioned feature building procedure on different days and combined them into a singular aggregate dataset. As we were predicting the energy usage for every two-hour increment for the following day, we employed 12 labels. Twelve independent Random Forest models were built for every two-hour interval. These models shared the same features but were trained separately.

III. Experimental Output

1) Data: The dataset used to build our model comes from the Low Carbon London project for households in London and supplied by London Datastore [20]. This data contains energy usage records for 5,567 London households, from November 2011 to February 2014. Energy consumption records were collected every half hour in kWh, resulting in 48 daily observations per household. The data is 10 GB in size, and includes 167 million rows. Additionally, weather data for corresponding periods was collected using Dark Sky API [21], and household income data was incorporated from the CACI Acorn group [22].

2) Results: We stored data generated by feature engineering in AWS S3. Then, we transfered 3,469,352 observations from S3 to MongoDB. For machine learning, we loaded data from MongoDB to Spark and used Spark ML to implement independent Random Forest models for 12 periods (every two hours for the next day). We employed Root Mean Squared Error (RMSE) as the evaluation metric (see Equation 2). Figure 4 shows RMSE values on test data for the corresponding 12 periods.

$$RMSE(y, \tilde{y}) = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(\tilde{y_k} - y_k)^2} \quad (2)$$

The result in Figure 4 shows that the model fits the test data well based on RMSE values ranging between 0.0945 and 0.2201. Figure 5 depicts predicted and actual energy consumption for two distinct users.

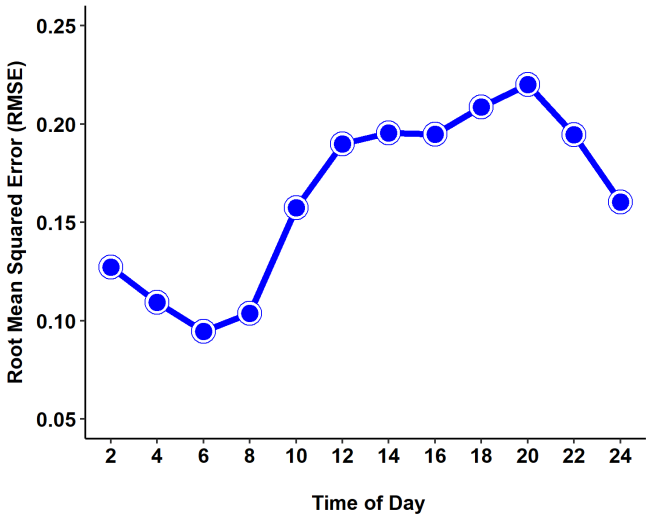

Fig. 4: RMSE values of 12 Random Forest models

To compare the performance of the Random Forest algorithm on Spark clusters with different system configurations, we launched several AWS EC2 clusters. Table II shows the system specifications and cost of a single EC2 instance in a Spark Standalone cluster that uses a cluster manager optimized for Apache Spark [23] and 3 different YARN clusters. In this experiment, we launched YARN clusters with a different number of instances to validate the influence of varying configurations on algorithmic performance.

We configured the number of trees in our Random Forest to 10 and 300 respectively using different cluster settings, as shown in Figure 6. We concluded that Spark YARN clusters do not have an advantage over a single instance, when the number of trees is small. However, as the number of trees increases, a single instance quickly runs out of memory. Additionally, for YARN clusters with instances with the same number of cores, an increased number of instances negatively correlates with reduced run times. Finally, the YARN cluster with 5 instances was the most cost efficient option for the data set that we employed (Table III).

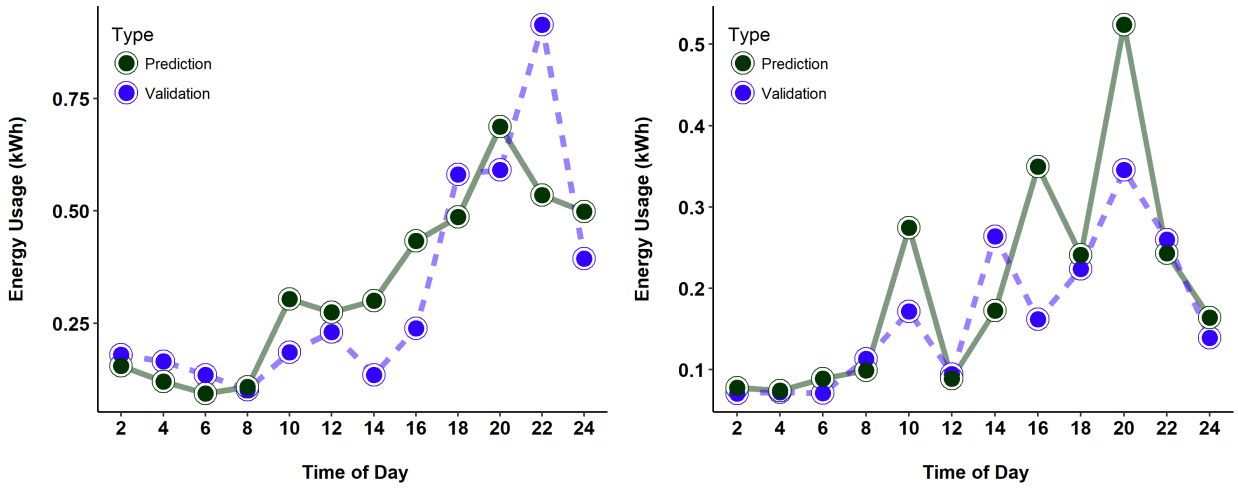

Fig. 5: Bi-hourly predicted and actual energy usage for a single consumer

TABLE II: Cluster types and specifications.

	Single EC2	YARN Cluster	YARN Cluster	YARN Cluster
Type of EC2 instance(s)	T2.large	C3.2xlarge 7	C3.4xlarge	C3.8xlarge
Number of instances	1	5	3	1
Number of workers	1	4	2	1
Number of cores per worker	2	8	16	32
RAM per worker (GB)	8	15	30	60
Disk Storage per worker (GB)	300	160	160	640
Cost per hour ($)	0.09	1.89	2.27	1.51

TABLE III: Cluster types and cost for building a model.

	YARN 5 Instances	YARN 3 Instances	YARN 1 Instance
Time per model (min)	8.3	9.1 7	15.7
Cost per model (USD)	0.29	0.38	0.44

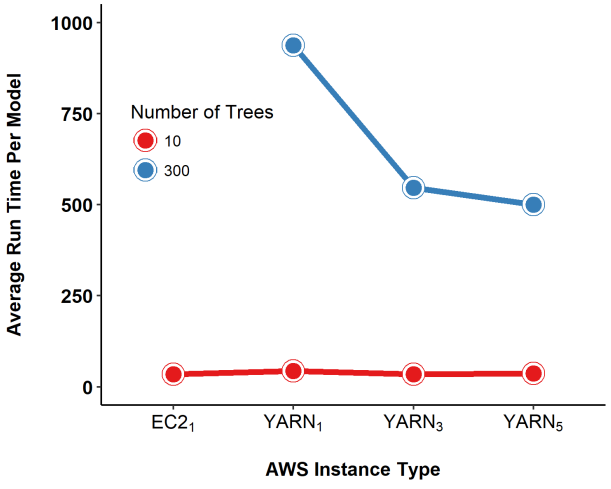

Fig. 6: Average run time with a different number of trees in Random Forest on different cluster settings.

IV. Conclusion

In this research, we employed a data set with 167 million smart meter observations, predicted future energy consumption with accuracy, and demonstrated the computational merits of leveraging distributed systems using a Random Forest machine learning algorithm.

Although we used a very large data set to generate our results, this data is not nearly as large as the data set would be if this model were to be implemented on all data collected in real-time by UK smart meters. In our circumscribed case, we were able to generate efficiencies by piping data directly from AWS S3 directly to Spark, bypassing MongoDB. This result is a function of the relatively small size of data. Were this an actual, full-scale commercial implementation, with data being collected and processed from all UK smart meters, the proposed distributed system—including the use of MongoDB–would exhibit computational efficiencies.

REFERENCES

[1] European Commission. (2018) Smart grids and meters. [Online]. Available: https://ec.europa.eu/energy/en/topics/markets-and-consumers/smart-grids-and-meters

[2] K. Gajowniczek and T. Zabkowski, "Short term electricity forecasting using individual smart meter data," *Procedia Computer Science*, vol. 35, pp. 589–597, 2014.

[3] R. E. Edwards, J. New, and L. E. Parker, "Predicting future hourly residential electrical consumption: A machine learning case study," *Energy and Buildings*, vol. 49, pp. 591–603, 2012.

[4] T. K. Wijaya, M. Vasirani, S. Humeau, and K. Aberer, "Cluster-based aggregate forecasting for residential electricity demand using smart meter data," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 879–887.

[5] A. Shahzadeh, A. Khosravi, and S. Nahavandi, "Improving load forecast accuracy by clustering consumers using smart meter data," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–7.

[6] A. Grandjean, J. Adnot, and G. Binet, "A review and an analysis of the residential electric load curve models," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 9, pp. 6539–6565, 2012.

[7] B. Yildiz, J. Bilbao, J. Dore, and A. Sproul, "Recent advances in the analysis of residential electricity consumption and applications of smart meter data," *Applied Energy*, 2017.

[8] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[9] L. Alejandro, C. Blair, L. Bloodgood, M. Khan, M. Lawless, D. Meehan, P. Schneider, and K. Tsuji, "Global market for smart electricity meters: Government policies driving strong growth," *US International Trade Commission, Office of Industries Working Paper No. ID-037. http://www.usitc. gov/publications/332/id-037smart_meters_final. pdf*, 2014.

[10] Amazon Web Service. (2018) Amazon s3. [Online]. Available: https://aws.amazon.com/s3/

[11] MongoDB. (2018) Mongodb for giant ideas. [Online]. Available: https://www.mongodb.com/

[12] Amazon Web Service. (2018) Amazon ec2. [Online]. Available: https://aws.amazon.com/ec2/

[13] Apache Spark, "Apache spark: Lightning-fast cluster computing," 2018. [Online]. Available: http://spark.apache.org

[14] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, "Mllib: Machine learning in apache spark," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016.

[15] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi *et al.*, "Spark sql: Relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1383–1394.

[16] Amazon Web Service. (2018) Overview of amazon emr architecture. [Online]. Available: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-arch.html

[17] U. Soytas and R. Sari, "Energy consumption and gdp: causality relationship in g-7 countries and emerging markets," *Energy economics*, vol. 25, no. 1, pp. 33–37, 2003.

[18] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[19] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] London DataStore. (2018) Smartmeter energy consumption data in london households. [Online]. Available: https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households

[21] Dark Sky API. (2018). [Online]. Available: https://darksky.net/dev

[22] CACI International Inc. (2018) Acorn - the smarter consumer classification — caci. [Online]. Available: https://acorn.caci.co.uk/

[23] Apache Spark. (2018) Spark standalone mode. [Online]. Available: https://spark.apache.org/docs/latest/spark-standalone.html