A BIG DATA ANALYTICS ARCHITECTURE FOR MULTI TENANT ENERGY OPTIMIZATION SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

OĞUZ CAN KARTAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2017

Approval of the thesis:

A BIG DATA ANALYTICS ARCHITECTURE FOR MULTI TENANT ENERGY OPTIMIZATION SYSTEMS

submitted by **OĞUZ CAN KARTAL** in partial fulfilment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Dr. Cevat Şener _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Prof. Dr. Ahmet Coşar _____
Computer Engineering Department, METU

Dr. Cevat Şener _____
Computer Engineering Department, METU

Asst. Prof. Dr. Pelin Angın _____
Computer Engineering Department, METU

Asst. Prof. Dr. Tansel Dökeroğlu _____
Computer Engineering Department, University of Turkish Aeronautical Association

Assoc. Prof. Dr. Pınar Karagöz _____
Computer Engineering Department, METU

Date: 08.09.2017

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name:

Signature:

ABSTRACT

A BIG DATA ANALYTICS ARCHITECTURE FOR MULTI TENANT ENERGY OPTIMIZATION SYSTEMS

Kartal, Oğuz Can
M.S., Department of Computer Engineering
Supervisor: Dr. Cevat Şener

September 2017, 67 pages

Efficient energy consumption is a trending topic nowadays, which has serious effects both environmentally and financially. Commercial and industrial buildings waste huge amounts of energy because of lack of integrated optimization systems. In this thesis, a big data analytics architecture for large-scale multi-tenant energy optimization systems is proposed, which is capable of doing various near-real time analyses on sensor data with the help of machine learning models created from old sensor data.

In order to build big data analytics handling subsystem there are several steps during the flow of the sensor data. Raw data collected from the sensors in the field to the system is parsed and turned into meaningful data containing required features. This meaningful data is used for predicting the forth-coming energy consumption values. Prediction feature of the system is carried out with a machine learning model created from old sensor data. This meaningful data is also used for updating this machine learning model, to improve the accuracy and provide compatibility of model with live sensor data. Prediction and model update analyses are implemented on the streaming sensor data, without first storing it to a database or file system to provide near-real time feature of system. A very important feature of the system is scalability, which means adding new tenants or increasing the frequency of sensor data arrival is handled by system.

Keywords: Big data analytics, Scalability, Internet of Things, Apache Spark

ÖZ

ÇOK KULLANICILI ENERJİ OPTİMİZASYON SİSTEMLERİ İÇİN BÜYÜK VERİ ANALİZİ MİMARİSİ

Kartal, Oğuz Can
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Danışman: Dr. Cevat Şener

Eylül 2017, 67 sayfa

Verimli enerji tüketimi çevresel ve mali açıdan ciddi etkilere sahip olan, her geçen gün önemi artmakta olan konulardan biridir. Ticari ve endüstriyel binalar, entegre optimizasyon sistemlerinin eksikliği nedeniyle büyük miktarda enerji harcarlar. Bu tezde, eski sensör verilerinden yaratılan makine öğrenme modelleri yardımıyla sensör verisi üzerinde çeşitli gerçek zamanlı analizler yapabilen büyük ölçekli çok kullanıcılı enerji optimizasyon sistemleri için bir büyük veri analiz mimarisi önerilmiştir.

Büyük veri analitiği işleme alt sistemi oluşturmak için, sensör verisinin akışında birkaç adım vardır. Sahadaki sensörlerden sisteme gelen ham veriler ayrıştırılır ve gerekli özellikleri içeren anlamlı verilere dönüştürülür. Bu anlamlı veriler, ilerleyen süreçte gelecek olan enerji tüketim değerlerini tahmin etmek için kullanılır. Sistemin tahmin etme işlevi, eski sensör verilerinden yaratılan bir makine öğrenme modeli ile gerçekleştirilir. Bu anlamlı veriler, aynı zamanda kullanılan makine öğrenme modelinin güncellenmesini ve doğruluğun iyileştirilmesini sağlamak için de kullanılır. Tahmin ve model güncelleme analizleri, sistemin gerçek zamanlı özelliğini sağlamak için önce bir veri tabanı ya da dosya sistemine yazmadan akan sensör verileri üzerinde yapılır. Sistemin çok önemli bir özelliği ölçeklenebilirliktir; yani yeni kullanıcılar eklemenin veya sensör verilerini alma sıklığını artırmanın performans üzerindeki etkisi sistem tarafından ayarlanmalıdır.

Anahtar sözcükler: Büyük veri analizi, ölçeklenebilirlik, nesnelerin interneti, Apache Spark

To my family

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude towards my supervisor Dr. Cevat Şener for his patience and support through my graduate studies. Throughout this journey, he always guided and motivated me in the best possible way. I am glad and honoured that I had the chance to work with him.

I would also like to thank Şahin Çaglayan, CTO of Reengen Energy, for his support and guidance about the domain that I was not familiar with. With his help, I was able to comprehend the problems in the energy domain. In addition, he provided me data and information for my experiments.

Last but not least, I am truly grateful for the effort the administrators of METU Computer Engineering Department put in this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IoT	Internet of Things
RDBMS	Relational Database Management System
RFID	Radio-Frequency Identification
NoSQL	Not only SQL
CPU	Central Processing Unit
DAG	Directed Acyclic Graphic
JMS	Java Message Service
PMV	Predicted Mean Vote
PPD	Predicted Percentage Dissatisfied
SME	Small and medium-sized enterprises
ECU	Electronic Control Unit
MLlib	Machine Learning Library
SQL	Structured Query Language
SSD	Solid-State Disk
HUE	Hadoop User Experience
RAM	Random Access Memory
XML	Extensible Markup Language
RSS	Rich Site Summary
HTTP	Hyper-Text Transfer Protocol
API	Application Programming Interface
HDFS	Hadoop Distributed File System
HVAC	Heating, Ventilating and Air Conditioning
EOS	Energy Optimization System
BMS	Building Management System
RDD	Resilient Distributed Dataset
DSTREAM	Discretized Stream

CHAPTER 1

INTRODUCTION

We are living in a world, where a large number and variety of data sources are available from various "smart" devices. These devices, which have the ability to collect and exchange data, together with their communication network form what is known as the "Internet of Things" (IoT). The Internet of Things creates opportunities for integrating physical world into computer based-systems by making collecting data from remote devices and controlling them across existing networking infrastructures possible. Thanks to rapid advances in underlying technologies, the Internet of Things is becoming increasingly prevalent in various domains of modern life, such as finance, telecommunications, surveillance, manufacturing, health-care, energy, network security and so on. According to Gartner, Inc. (a technology research and advisory corporation), there will be nearly 20.8 billion devices on the Internet of Things by 2020 [1]. Considering all these devices collecting and sending data, massive amount of data flows around the world. Such data withholds valuable knowledge, previously hidden because of the amount of work and computational resources required to extract it. In each of the IoT domains there is a need to gather, process and analyse this data, detect emerging patterns and outliers, extract valuable insights, and generate actionable results.

One of the largest subset of IoT network is energy domain. In the United States, commercial buildings account for nearly 20% of the nation’s energy usage, with more than half that figure being used simply for heating and lighting. However, the Environmental Protection Agency (EPA) estimates that the typical commercial facility wastes 30% of its energy [2]. The problem is getting more serious with more and more buildings getting constructed every day. In order to stop the energy waste, energy optimization systems should operate in buildings with the ability to handle the continuously increasing data. Good news is that innovations in data storage and

processing methodologies enable the processing of large amounts of data in a scalable manner, and generation of actionable insights in near real-time for a better operation and predictive maintenance of energy assets.

Electricity is not economically storable, and this in turn requires maintaining the supply/demand balance in real time by building a large-scale energy optimization system [3]. Currently the minimum interval of doing both baseline energy forecast and HVAC (Heating, Ventilating and Air Conditioning) set point regression analyses are an hour for most energy optimization systems. Decreasing this minimum interval and doing more frequent analyses has serious advantages over hourly analyses because nowadays, accurate energy consumption forecast for the upcoming few minutes for balancing sudden changes in demand against energy delivery has become a critical research field as a result of privatization and deregulation of the power industry [4]. Electricity distribution companies use baseline energy forecast for making a decision about how much electricity to produce, while building managers use baseline energy forecast for making a decision about how much electricity to consume. This decision-making process is currently working on hourly analyses, but more frequent analysis results will improve the precision of overall hourly analyses, and more precise results will help balancing demand response of the electricity. In times with the high demand for electricity, electricity distribution companies need to curtail some load from some of the customers. In such a case if the building manager has more frequent HVAC set point regression analysis results, consumption can be reduced by adjusting set points accordingly.

The concept of real-time pricing is surging nowadays. In this type of pricing, electricity rates fluctuate in very frequent intervals, like an hour, within a day. A signal for specifying the costs of generating electricity in the current time interval is transmitted to the customers for each interval. Real-time pricing gives the chance of having higher control over their electricity consumption to the consumers. Customers can use the opportunity of consuming electricity at cheaper prices as a result of using less electricity throughout times when it is costlier to produce electricity. Moreover, carbon emissions can be reduced with the help of electricity use timing by environmentally-conscious consumers [5]. In a scenario with real-time pricing both the amount of electricity consumed and produced affect the price of electricity in

spot market. So, the more precise results from baseline energy forecast analysis are profitable for both producers and consumers. Also in the scenario with real-time pricing, building manager would be able to consume less electricity in the time period where electricity prices are high and vice versa by adjusting set points accordingly if more frequent HVAC set point regression analysis results are used.

In this thesis, we propose a big data analytics architecture for multi-tenant energy optimization systems which builds machine learning models with batch accumulated sensor data, and use these machine learning models to predict consumption values in near real time by analysing streaming sensor data. Our architecture also updates the machine learning models on the fly by analysing streaming sensor data. Apache Spark is selected as the processing framework since it has support for both batch and streaming processing and also has a powerful, in terms of implemented algorithms, machine learning library. In this thesis, several variations and parameter changes are also analysed for better performance and accuracy of the energy optimization system. Performance of the system stands for analysing the data coming from multiple tenants in near-real time. Accuracy of the system stands for making better consumption predictions to determine the better state for devices managed by system from the analysed data for tenants. Proposed architecture supports multi-tenancy which means that system is expected to handle data coming from a number of tenants in parallel. To sum it up contribution of this thesis is proposing a big data analytics subsystem model for an energy optimization system with following features. System should be capable of doing above mentioned analyses like baseline energy forecast and HVAC set point regression more frequently than hourly intervals. System should support multi tenancy in a scalable manner. System should meet the timing requirements of a near real time system and also system should provide a reasonable accuracy rate.

Rest of the thesis is organized as follows: In Chapter 2, big data and big data analytics concepts are defined and several tools for handling big data analytics are described in technical details. In Chapter 3, previous researches related with this thesis are presented. What previous researches propose and what they have not achieved is given. The focus of this study is how the missing points can be achieved. Then, in Chapter 4 a reference energy optimization system is described and

requirements of it related with big data analytics are explained, in Chapter 5, the basic architecture of the proposed energy optimization system is described in detail. Also, several variations and parameter changes are mentioned. Then, in Chapter 6, first the testing environment is described and numerous tests are performed. Later, these tests and their results are presented and discussed in detail. Finally, the thesis is summarized and remaining works for the future are given in conclusion Chapter 7.

CHAPTER 2

BIG DATA ANALYTICS

There is not a clear or well-accepted definition for the concept "big data". Many organizations and research institutes use their own definition of big data but in general sense big data is a term used for a data collection which is so large or complex that is formidable to be stored and processed neither adequately nor efficiently by using traditional technology, hardware and software tools. Data sets are growing rapidly because the equipment that generates digital data gets cheaper and data sources get widespread. These data sources include information-sensing mobile devices, software logs, cameras, microphones, RFID readers and wireless sensor networks [6].

Most of the prevalent definitions of the big data in the literature focus on the size of the data but a comprehensive definition is constituted by the three main attributes of big data: volume, variety and velocity also known as the three Vs of big data as it can be seen in Figure 1.

Volume

The primary attribute of big data concept is clearly data volume which is the size of the gathered data. Data that is measured with the terabytes or petabytes is considered as big data. On the other hand, volume of the data can also be quantified by count of the records, transactions, tables, or files depending on the definition. Sensors are coming into use in various industries and also with higher sampling frequencies than before. Also with the social media becoming one of the main aspects of human life billions of messages, photos and tweets are added to data storages daily. By 2010 the global amount of information would rapidly up to 988 billion GB. Experts predict that by 2020 annual data will increase 43 times [7]. It is obvious that regardless of the definition, each quantification of the big data is

growing continuously with the development of technology. It is safer to label the data as big data after the point when storing this data is not efficient with traditional data storage techniques.

Variety

A considerable amount of big data specialists think that the main problem with the big data is not handling the scale of the data, but integrating the poorly structured data. The old traditional structured data was easier to store and analyse with the existing RDBMS (Relational Database Management Systems) even in large volumes. This structured data is now joined by semi-structured data like XML and RSS feeds and unstructured data which is coming from diverse data sources including various different sensors and also web sources like logs, clickstreams and mostly social media and also coming in various data types like texts, photos, videos and audio. Also, the IDC study predicts that unstructured information will account for 90% of all data created over the next decade [8]. Therefore, this big data neither can fit into predefined rectangular tables of RDBMSs nor can be analysed with existing technologies.

Velocity

Velocity refers to the frequency of data generation or the frequency of data delivery. Data like temperature, sound and image is coming continuously from sensors, also data like posts, tweets and messages is generated in huge amounts every single second. Handling big data does not only mean retrieving and storing it, but also processing it. So, velocity also refers to the frequency of analysing the data. In many areas data's value is decreasing as the time passes like disaster prediction. In areas like these data have been requested to carry out in real-time or near-real-time processing. Depending on the design of how to utilize the data, performing batch analysis like daily or weekly is widespread.

There are also other substantial V-word attributes that are associated with Big Data. For example, veracity deals with the accuracy of the data generated. As the data sources increase and generates more data to use a fraction of this data is useless,

corrupted or incompatible. Volatility refers to time period of data considering as valid and storing in the system. As new data comes to the system the older data becomes less meaningful. Value means worth of the extracted information from the data. If data cannot be used for extracting something out of it or assigning a new mean to it, it does not have a value from the business point. Variability refers to data whose meaning is changing with respect to concept it is being used in. For example, in language processing field one specific word may define different values in different concepts. Visualization means the way of representing the big data or results extracted from it in a readable and accessible manner.

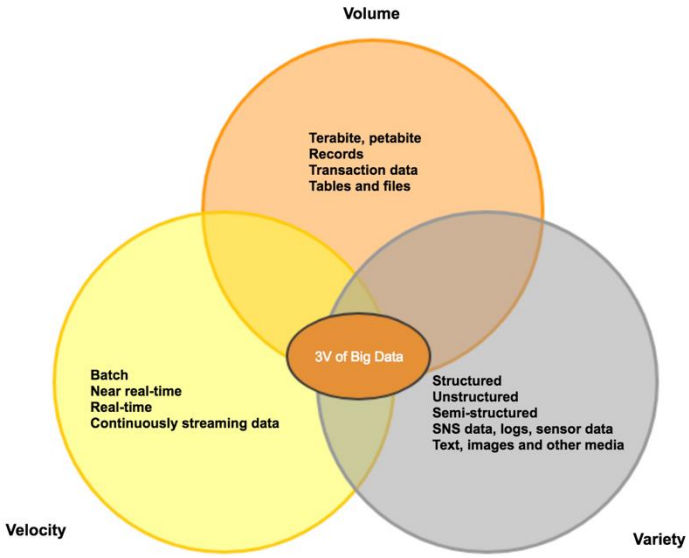

Figure 1: Big Data Features

The bigger problem with big data is how to extract valuable information from it rather than how to store it. With the increase in data being received and stored, in every industry most of the leading companies wonder whether they are getting full value from the massive amounts of information they already have within their organizations [9]. This is the reason that significance of big data analytics improves every day. Big data analytics is about two things, big data and advanced analytics, or simply using advanced data analytics techniques on big data. Advanced analytics is a term used for collection of techniques and tools including data mining, statistical analysis, machine learning, artificial intelligence, natural language processing, data

visualization, complex SQL etc. These techniques are not recently discovered techniques, they are already being used for years with the structured data, but since most of these techniques adjust well with concept of big data too, big data analytics have emerged as one of the most popular concepts. Sensor data is becoming more relevant in all fields of daily life, memory storage capacity and processing power made big data analytics possible to reach out more and more. Big data analytics is common nowadays and clearly going to be used as the primary mechanism for many decisions as it increases accuracy and reduces irrelevant influences [10]. Big data analytics can be classified under two main categories, bulk data analytics and streaming data analytics.

- Bulk Data Analytics

Bulk data analytics is the sub-concept of big data analytics which is specialized on processing batch data. Most of the time bulk data analytics work on masses of structured and semi-structured historical data which are typically stored in database or file system. Bulk data analytics mostly involves several separate processes. First data is collected from the storage point, usually over a period of time. Then the data is processed by the implemented program. Finally, the results are obtained and output is stored. Bulk data analytics is not that time-sensitive at all. In fact, bulk data analytics can take hours, or perhaps even days depending on the volume of the data and the complexity of the desired analysis. Bulk data analytics are advantageous in the applications dealing with payroll and billing activities which occurs on cycles of wide time periods or deep analytics applications which does not require immediate decision making. There are different bulk data analytics tools, most common ones are Apache Hadoop and Apache Spark.

- Streaming Data Analytics

Streaming data analytics is the sub-concept of big data analytics which is specialized on processing streaming data. Most of the time streaming data comes from continuous data sources which means analytics has to be done on flowing data which can be either structured or unstructured. This is different than the traditional store and process systems. In the traditional systems input data is first stored in

database or file system or memory which can be based on the nature of the system. After the storage of the data is completed it is transmitted to units where processing take place and results are obtained, which is not practical when the results of the analytics is needed in real or near-real time. In the applications that deals with network monitoring, fraud detection, market data management and disaster prediction analysis have to be done on-the fly before the incoming data loses value, so streaming analytics is a crucial tool for applications like these. Since the data is not stored before processing, methods to replay the data are not usable when streaming analytics takes place so this kind of applications have to cope with issues such as data loss, corruption, and reordering.

The high-level architecture of a stream processor simply works like this: arriving data streams are clocked through a network of processing nodes, where an action or transformation like aggregation, analytics and filtering is performed on each of the nodes as data flows through it [11]. These nodes can perform either concurrently or parallel depending on the action. After the processing is completed data is stored with the help of this nodes in the desired system (file, database, disk, and memory). There are different stream processing tools most common ones are Apache Storm and Apache Spark.

In the following subsections, we discuss some of the relevant technological advancements that enable handling of the mentioned challenges of big data analytics.

2.1 Map Reduce

Handling big data is not applicable with traditional systems and infrastructures since it is hard to predict the size of the data. An option for acquiring more power is adding more computers, more CPU cores, more memory and more hard disks, which has the disadvantage of having capacity limits. After some point this option fail to stand over the continuous data increase. Besides the cost of adding specialized hardware is lot more than general purpose hardware. Another option is, which is more favourable, using as many resources as required which are relatively powerless in terms of computing but working in parallel. Of course, handling processes working on different machines in a parallel manner is not an easy task from programming point of view. This is the part where MapReduce comes in.

Map reduce is a programming model that is based on two main functions. Map function processes a key/value pair and generates a set of intermediate key/value pairs while reduce function combines all intermediate values associated with the same intermediate key. All of the features like managing the required inter-machine communication, handling machine failures, partitioning the input data and scheduling the program's execution across a set of machines are taken care of by the run-time system in programs written by using map-reduce programming style. Furthermore, these programs are automatically parallelized by the run-time system [12]. The main advantage of the MapReduce programming model is its simplicity. Level of abstraction provided by this programming model ensures user focusing more on the algorithm than spending time with inner details like parallelization, task setup, fault tolerance and concurrent data access, which are all handled behind the scenes. Only task user has to do is specifying an implementation with a pair of map and reduce functionalities, which are conformed to the requirements of the programming model. This abstraction makes MapReduce programming model very advantageous for developing data-analysis algorithms which are highly scalable [13].

2.2 Tools

Big data analytics is one of the most trending topics nowadays as mentioned in previous sections. To successfully handle big data analytics several tools are available. Some of those tools concentrate on bulk data analytics while some of them concentrate on streaming data analytics, of course there are tools available which are both able to work with bulk and streaming data analytics. Tools with ability to work on bulk data analytics are tries to solve problems related with volume feature of big data, meanwhile tools with ability to work on streaming data analytics are tries to solve problems related with velocity feature of big data.

2.2.1 Apache Hadoop

Hadoop is an open source framework for data analytics which uses MapReduce algorithm as primary mechanism of processing. Hadoop is designed for processing very large data sets using distributed storage systems on clusters. Hadoop stores data with the help of Hadoop Distributed File System (HDFS) and processes

data using MapReduce. Hadoop is a batch processing framework and have the mentality "Store first post-processing" while processing the data. This mentality and loading the data from disk while processing makes the velocity a feature Hadoop has problem to handle. HDFS supports various file formats so it is safe to say variety is handled by Hadoop and also volume is handled by Hadoop.

Apache Mahout is a package of implementations which contains the most popular and usable machine learning algorithms. Most of the implementations included in Apache Mahout are designed to work on Apache Hadoop to enable scalable and robust processing of huge datasets. Algorithms that are not available in a parallelizable form due to the nature of the algorithm can also take advantage of HDFS for suitable access to data in the Hadoop processing pipeline. Core algorithms of Apache Mahout are for clustering, classification and collaborative filtering are implemented on top of Apache Hadoop using the map/reduce paradigm.

2.2.2 Apache Storm

Storm is a distributed real-time computation system for processing large volumes of high-velocity data [14]. Storm is designed for massive scalability, offers a strong guarantee that every tuple will be processed and supports fault-tolerance. For determining the flow of process of streaming data coming from the input sources users describe topologies. These topologies directed acyclic graphs. Program acts on how the topology is built when the data comes in. As it can be seen in Figure 2 storm topologies are based on two main components. Spouts are sources of streams, they bring data into the system and hand it to bolts. Bolts, on the other hand, are modules which process the input streams and constructs outputs. Spouts have the ability to transmit the input stream to one or more bolts in a parallel fashion, moreover bolts can handle the processing either concurrently or parallel. Storm ecosystem can receive data from a rich array of types of sources. Large range of spouts are available for reading data from the Twitter streaming API to Apache Kafka to JMS brokers to everything in between. Also, custom spouts for highly specialized applications can be implemented by users. This is an important advantage of the Storm ecosystem [15].

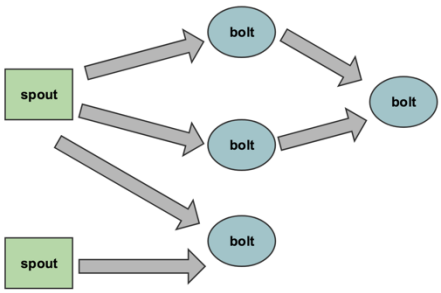

Figure 2: Storm Spot/Bolt Architecture

2.2.3 Apache Spark

Apache Spark is an open source data analytics framework which allows in-memory computation on large distributed clusters with high fault-tolerance [17]. Spark's primary abstraction is a distributed collection of items called a Resilient Distributed Dataset (RDD). An RDD is simply an immutable distributed collection of objects in Spark. RDDs take advantage of two different kinds of operations which are transformations and actions. Transformations convert previous RDDs to a new RDD. On the contrary actions, compute a result using a RDD. This result is either saved to an external storage system or returned to the driver program for further calculations [16]. The main difference between transformations and actions is that transformations don't make Spark compute. Spark computes RDDs only the first time they are used in an action. Spark handles scalability with the feature that RDDs can be split into multiple partitions, which may be computed on different nodes of the cluster.

Spark and Hadoop have their similarities and differences. Spark is similar to Hadoop in basic concepts like using MapReduce algorithm as primary mechanism of processing and also Spark is designed for processing very large data sets using distributed storage systems on clusters. Spark has of course its differences. The prominent difference is that Hadoop stores the data in disk, in the HDFS, while Spark stores the data in memory. The performance boost of Spark over Hadoop while processing data comes from this basic difference. Another important reason of the better performance is after each map task results are written to disk in Hadoop,

while in Spark results are stored in memory if not stated otherwise. Spark also has the streaming implementation which helps Spark with handling velocity. Streaming implementation will be analysed in the ensuing chapter. Spark have the options like using NoSQL databases like Cassandra or using Spark SQL, when working with structured data, for distributed storage alongside with HDFS.

Apache Spark has also streaming implementation. Spark Streaming is also, like Storm, is a distributed real-time computation system for processing large volumes of high-velocity data and also provide massive scalability and fault-tolerance. The main concept Spark Streaming built on is called DStreams or discretized streams. Spark Streaming uses a micro-batch technique which splits the input stream as a sequence of small batched chunks of data, which are DStreams. Each DStream is processed like batch and results of this process is sent to next batch process. Spark Streaming represents each DStream as a sequence of RDDs arriving at each time step which can be created from numerous input sources, such as Flume, Kafka, or HDFS. DStreams provide most of the operations available on RDDs. Besides new operations related to time, such as sliding windows are implemented in Spark Streaming [16].

Apache Spark has also a library focused on machine learning named MLlib which aims to make practical machine learning scalable and easy. MLlib contains a variety of common machine learning algorithms such as classification, regression, clustering and collaborative filtering. It also provides featurization, pipelines and persistence to users at high level. The main factor that differentiates MLlib from most of the other machine learning libraries is the ability to work on distributed datasets in parallel efficiently. Also, the bond with Spark makes MLlib more attractive since it has the key features for big data analytics like high-performance, scalability and fault-tolerance inherited from Spark core. Users can also use MLlib with other Spark libraries like Spark Streaming and SparkSQL which gives them the opportunity to use machine learning with streaming data and data-frame based data.

MLlib and streaming data can be integrated in two different manners. First one is creating a suitable machine learning model from the batch data and then predicting the results with the data from the stream by using the created model. Other one corresponds to online learning phenomena in machine learning world. It is

simply creating a machine learning model of which parameters can be updated with the data from the stream. This is achieved by implementing the same fitting algorithm that is performed offline on each batch of data, so that the model continually updates to reflect the incoming streaming data. Both of these two manners will be used in this thesis and more details about their working principles will be analysed in the oncoming chapters.

CHAPTER 3

LITERATURE REVIEW

There are plenty of both on-going and completed researches in the domain of IoT and sensor data. Most of these researches are focused on frameworks with the ability of gaining meaningful information from the incoming sensor data and mostly by using cloud computing. In this research, we will be focusing on an energy optimization system (EOS) which has the ability to analyse incoming sensor data in near real time and adjust the building's energy system in respect to results to use less energy.

Energy optimization systems and smart grid applications have their similarities and differences. In both of them, there are large number of sensors, large amount of data flows from these sensors and the goal to analyse this big data for adjusting the system for future. However, sensors in energy optimization systems are located on devices running in the building while in smart grids sensors are located over the electric lines. It is safe to say studies focused on smart grid are applicable to energy optimization systems with small modifications. Natasha Balac et al [18] studies a “big data” analytics platform with the ability of advanced forecasting is studied for utilizing smart grid. The goal of this study is to create highly accurate models predicting energy demand for the campus micro grid at multiple levels of granularity. Xiufeng Liu et al [19] also studies an energy optimization solution focused on "smart-grid". This paper proposes an innovative ICT-solution to streamline smart meter data analytics. The proposed solution offers an information integration pipeline for ingesting data from smart meters, a scalable platform for processing and mining big data sets, and a web portal for visualizing analytics results.

There are also studies related with building energy optimization systems rather than smart-grid energy optimization. Most of these researches are based on

analysing data in a batch process mode. Ioan Petri et al [20] studies on a framework based on Comet Cloud. In this paper, the use of cloud computing for efficiently running and deploying Energy Plus simulations with sensor data in order to fulfil a number of energy related objectives for buildings is described. W. Khamphanchai et al [21] also focuses on batch processing. This paper introduces the BEMOSS operating system aiming at improving sensing and control of equipment, reducing energy consumption and enabling demand response of small- and medium-sized commercial buildings. Vaclav Jirkovsky et al [22] is focused on demonstrating the computational power of big data by analysing time series data measured from a passive house with the goal of detecting specific events. Stampfli, J et al [23] is proposing novel alarm verification service which uses Apache Spark for comparing different machine learning algorithms.

Recent studies started focusing on real-time analytics for energy optimization systems as much as batch analytics. Mario L. Ruz et al [24] defines a tool which is used to estimate predicted mean vote (PMV) and predicted percentage dissatisfied (PPD) indices. The tool communicates remotely with the renewable system and proposes corrective control indications to maintain the indoor air conditions inside the optimal comfort range. Kanet Katchasuwanmanee et al [25] aims to create a simulation methodology and to investigate the modelling of thermal and energy management called e-ProMan across manufacturing site. In this article, a simulation-based approach is presented to develop thermal and energy-management systems applied to SME manufacturing environments, supported by real-time ‘big data’, and the corresponding predictive control and optimisation analytics. Liehuo Chen et al [26] designs a new framework for real-time big data analytics and also implemented a prototype system and evaluated its performance using important data analysis benchmarks adapted to model real-time data analytics. Sunil Mamidi et al [27] implement a multi-modal sensor agent that is non-intrusive and low-cost, combining information such as motion detection, CO2 reading, sound level, ambient light, and door state sensing. It is shown that machine learning techniques can be used to estimate room occupancy using a set of simple sensors, and that similar techniques to learn agent models that predict occupant behaviour can be used. By using these agent models to predict room occupancy up to an hour in advance, the BLEMS system can

intelligently control the multi-agent HVAC system to minimize energy usage while maintaining occupant comfort. Mayet et al [28] proposes a programming model and data-parallel system architecture that allows real-time machine learning model training. In this study, incremental K-Means clustering and Markov model training in the context of anomaly detection in smart factories are focused on. Grolinger et al [29] investigates sensor-based forecasting in the context of event-organizing venues, which present an especially difficult scenario due to large variations in consumption caused by the hosted events. Two machine learning approaches were considered, neural networks and support vector regression, and four prediction models were explored with each. Each were considered together with three data granularities: daily, hourly, and 15 minutes.

Real time information derivation from sensors are not only studied on energy optimization field. Mohiuddin Solaimani et al [30] presents a novel, generic real-time distributed anomaly detection framework based on Apache Spark for multi-source stream data. The framework can monitor network traffics periodically and builds the training model. Later it can be used to identify suspicious or anomalous network traffics that potentially indicate Cyber-attacks. Yedu C. Nair et al [31] aims at performing real time big data analytics on vehicular data collected from a network of ECUs (Electronic Control Unit) in cooperated into the different automobiles. The analytics has been performed on big data frameworks like Hadoop and Spark. Batch processing model is developed using Hadoop map reduce and Hive. Real time stream processing has been achieved through Spark streaming and Storm streaming. Daesik Ko et al [32] proposes an application framework for smart cold chain management system based on Hadoop, Spark and IoT (Internet of Things) techniques. To satisfy the requirements to provide multi-tenancy of cold chain application and to develop with common component in cold chain management systems, system is designed based on Hadoop, Spark and Spark Streaming. Domann J. et al [33] present a highly scalable news recommender system optimized for the processing of streams built on Apache Spark framework. The use of Apache Spark enables running the recommender as a distributed system in a cluster ensuring the scalability of the approach. Lekha R. Nair et al [34] aims at developing a real time remote health status prediction system built around open source big data processing engine, the

Apache Spark, deployed in the cloud which focus on applying machine learning model on streaming big data. In this scalable system, the application receives the data with health attributes of user, extracts the attributes and applies machine learning model to predict user's health status.

All of the researches above seek solution to big data storage and analytics problems in a high architectural perspective. Most of them speaks of a solution related with energy optimization systems. Some of this energy optimization systems are implemented for working in micro-grid while some of them are implemented for working in real-buildings. Some of the referred systems are based on batch analysis of the big sensor data and some of them are based on real time analysis of the big sensor data. There are also researches mentioned above which are not related with the building energy area but tries to analyse and process the sensor based big data either on real time or batch. This research aims to introduce a big data analytics architecture for large-scale multi-tenant energy optimization systems which is suitable for working on both historical and real-time sensor data in a multi-tenant manner and will be based on Apache Spark. This framework would be able to work with different machine learning algorithms and optimization models.

CHAPTER 4

BIG DATA ANALYTICS IN AN ENERGY OPTIMIZATION SYSTEM

Energy efficiency of buildings is vital because it helps to preserve finite resources, lowers costs for businesses and users, and can be accomplished relatively quickly. Hence, to move towards a low carbon economy, making “more intelligent” use of energy in buildings will fundamentally contribute to energy and cost savings. Existing buildings are responsible for more than 40% of the world’s total primary energy consumption [35]. Increase in global population and the tendency of modern generation spending more time in buildings leads to increase in number of buildings all around the world, which eventually leads to higher energy consumption overall. As these factors are not expected to change, energy efficiency in buildings represents a prime objective for energy policy at regional, national and international levels.

This is the part where energy optimization systems emerge. To make things clear energy optimization system in a building is handling electric energy. The main goal of an energy optimization system is reducing the energy consumption of the building while not affecting comfort level of people present in the building. It is not an easy task as it sounds, because of the volume and velocity of the data the systems have to handle. Sensor data have to be analysed real or near-real time in most of the cases to take precaution and adjust the system according to results. Also in other cases, which requested batch analysis huge amount of old sensor data have to be analysed. These two cases are issued with two important aspects of big data respectively, velocity and volume, which makes energy domain one of the most popular domains for using big data analytics.

In this chapter, first the overall structure of a subsystem which handles the big data analytics of a sample energy optimization system will be described. Then focused analyses in this subsystem and operations that subsystem will perform are described in detail.

4.1 Overall Structure of Big Data Analytics Handling Subsystem

The subsystem of an energy optimization system which is handling the big data analytics collects data from lots of sensors, mostly energy consumption data of devices running in the target building and the weather sensor data. This data has to be stored efficiently, which is a challenge but there is a bigger challenge for the system. Before this streaming data is migrated to the system's database, it has to be processed by the system. This streaming time-series data can be used in some of the analyses even before getting stored in the database. The subsystem must be able to carry out both batch analyses using the data stored in the database and real-time analyses using the streaming data. The workflow of a subsystem handling with big data analytics of an energy optimization system is shown in the Figure 3.

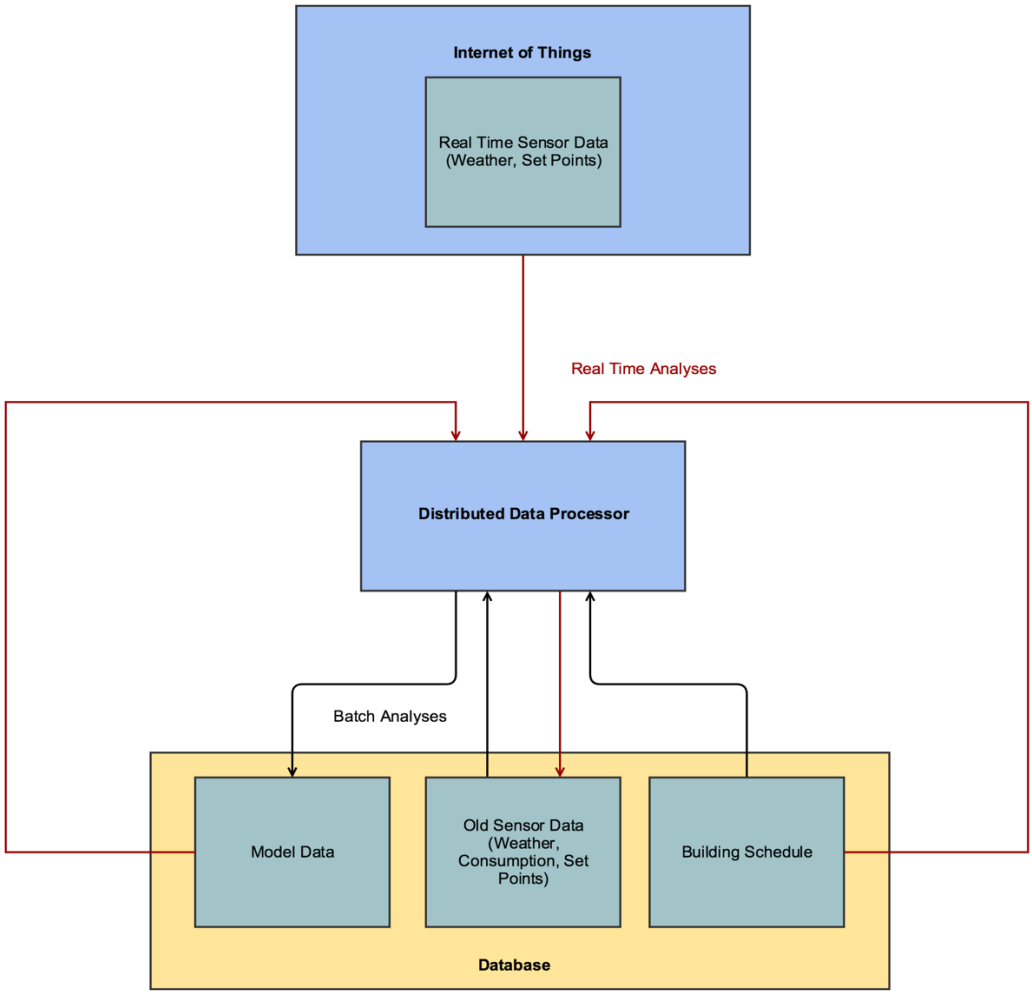

Figure 3: Overall Structure of Big Data Analytics Handling Subsystem

4.2 Big Data Analytics Operations

This section will explain different kinds of operations system is capable to perform without providing technical details.

4.2.1 Model Create

Big data analysis handling subsystem creates machine learning models from batch data using the selected machine learning algorithm. In model creation old consumption, temperature and occupancy data are used. Model creation process is done for only once for every building. For every building in the system a different model is created using the data related with that building.

4.2.2 Model Update

Big data analysis handling subsystem updates the created machine learning models from streaming data using the selected machine learning algorithm. In model update live consumption, temperature and occupancy data are used. Also, the old model data (variable weights) is used for updating the models, however old data is not used again and again for every iteration of model update. For every building in the system the model of the building is updated using the data related with that building.

4.2.3 Consumption Prediction

Big data analysis handling subsystem predicts the consumption values from streaming data using related machine learning model. In consumption prediction current temperature, occupancy data and variable weights of related machine learning model are used. For every building in the system the consumption is predicted using the data and model related with that building.

4.3 Focused Analyses

This section will explain different kinds of analyses implemented in the system in order to fulfil the listed operations above, without providing technical details.

4.3.1 Baseline Energy Forecast

Baseline energy forecasting is used to forecast an ideal energy consumption value for a specific forthcoming time slot based on the historical values of target building. Required input dataset for regression analysis consists of;

- Energy consumption
- Average outdoor temperature
- Building operation schedule

A model is defined based on those historical data. As a result of the regression analysis, coefficients can be obtained for each independent variable. By using these coefficients, ideal energy consumption for desired forthcoming time slots can be generated with already known temperature and occupancy data.

Baseline energy forecast models are updated in hourly, daily, weekly and monthly periods in most of the modern energy optimization systems. Increasing this updating frequency gives the system some serious advantages like improving the accuracy of predictions, making model more sensible to temperature changes in small time periods and making the system more compatible with real time pricing concept. Frequency of updating models for baseline energy forecast should be less than one hour, as a matter of fact it would be better to update the models in 10 or 20 minutes. Predicting consumption values should be completed in at most 5 seconds if it is desired to near-real time property of the system to be meaningful.

4.3.2 HVAC Set-Point Regression

Effect of set point temperature changing on total energy consumption can be determined with HVAC set-point regression analysis. Required input dataset for regression analysis consists of;

- Energy consumption
- Average outdoor temperature (T_{out})
- Indoor heating/cooling set point temperature (T_{set})
- Building operation schedule

Energy consumption and operation days of building are directly used in regression analysis. However, temperature difference between set point temperature

and average outdoor temperature values (T_{set}-T_{out}) is used as an independent variable in regression analysis.

As a result of the regression analysis, coefficients can be obtained for each independent variable. With using these coefficients energy consumption for different set point values can be generated.

HVAC set-point regression models are updated in hourly, daily, weekly and monthly periods in most of the modern energy optimization systems. Increasing this updating frequency gives the system some serious advantages like improving the accuracy of predictions, making model more sensible to temperature changes in small time periods and making the system more compatible with real time pricing concept. Frequency of updating models for HVAC set-point regression should be less than one hour, as a matter of fact it would be better to update the models in 10 or 20 minutes. Predicting consumption values should be completed in at most 5 seconds if it is desired to near-real time property of the system to be meaningful.

4.3.3 Weather Normalization

Weather normalization is used to eliminate the effect of outdoor air temperature on building energy consumption in order to make comparison of building energy consumption for different time periods with different weather conditions. Required input dataset for regression analysis consists of;

- Energy consumption
- Average outdoor temperature
- Cooling system starting temperature
- Heating system closing temperature
- Building operation schedule

After the regression analysis is performed and coefficients are determined, weather normalization can be implemented to data. To eliminate impact of weather, energy consumption for specific temperature is normalized according to the change point temperature.

Weather normalization models are updated in hourly, daily, weekly and monthly periods in most of the modern energy optimization systems. Increasing this updating frequency does not give the system any serious advantages. So, the

maximum model update frequency for weather normalization models is left at 1 hour. For updating the models in periods of at least 1 hour using a real-time updating mechanism is not necessary, so this functionality is not implemented in this subsystem, it is accomplished by batch updates implemented with the help of outer subsystems. Predicting consumption values should be completed in at most 5 seconds if it is desired to near-real time property of the system to be meaningful.

4.4 Data Used for Operations

Energy consumption is not the only required data for the system to operate. The system also uses weather and occupancy data to determine the working parameters for an efficient energy consumption model. Occupancy data is obtained by the help of other services and then stored in the database, and hence when the system makes a real-time analysis energy consumption data and weather data is obtained from the streaming data while occupancy data have to be obtained from the database. Furthermore, data of machine learning model is required to perform each real-time analysis and this data have to be obtained from the database.

CHAPTER 5

PROPOSED SOLUTION

For the described energy optimization system with the previously mentioned functionalities, an example software component architecture will be proposed in this section. Several different variations and parameters will also be proposed. These different variations are based on the basic architecture and use the same components but the functioning inside these components have differences among them. First the basic architecture with details will be explained. All the variation and parameter changes will be explained later.

5.1 Basic Architecture

Apache Spark is used as the data processor in this architecture. Apache Spark runs on a cluster of computers, each of which is a worker. The driver program node does not have to retrieve the whole data from the database and distribute it over the cluster to the workers. Instead, since Spark supports HBase data sources, each worker connects to the database and retrieves the data of its own partition as RDDs and processes them. This functionality has the advantage of preventing the overhead of transmitting the whole data more than one time over the cluster.

Apache Spark also has the streaming and machine learning libraries which are two of the most crucial parts of this architecture. Sensor data is read as RDDs with the integration of Spark Streaming, so worker nodes have the ability to analyse this data with no significant difference from the analyses done by Spark core. Each worker node processes its data and writes the results to database on its own. One of the most important duties of energy optimization system is to predict the consumption outcome and adjust the system according to what was learned from the older batch data. This is accomplished with machine learning models and this is the part where Spark MLlib comes in. Spark MLlib has the advantage of having the

ability to work with both streaming and batch data. In our architecture machine learning models are created from batch data and updated with the streaming data with the help of both Spark Streaming and Spark MLlib.

Sensor data is processed before being written to database so it would bring overhead to the node that retrieves and writes the data and some of the data might have been lost while waiting for node to process previous data. To prevent data loss at the point where incoming data velocity exceeds the pre-processing velocity of the pre-processor node Apache Kafka service is used in this architecture. Kafka is a distributed streaming platform which lets the stream of records to be stored and processed in a fault-tolerant manner.

In this architecture, a column-based NoSQL database is used, more specifically HBase is chosen. Since Apache Spark has integration with HBase over third party libraries, it is a useful choice. The basic architecture of Spark and HBase is about getting an HBase connection object in every Spark Executor as it is shown in Figure 4.

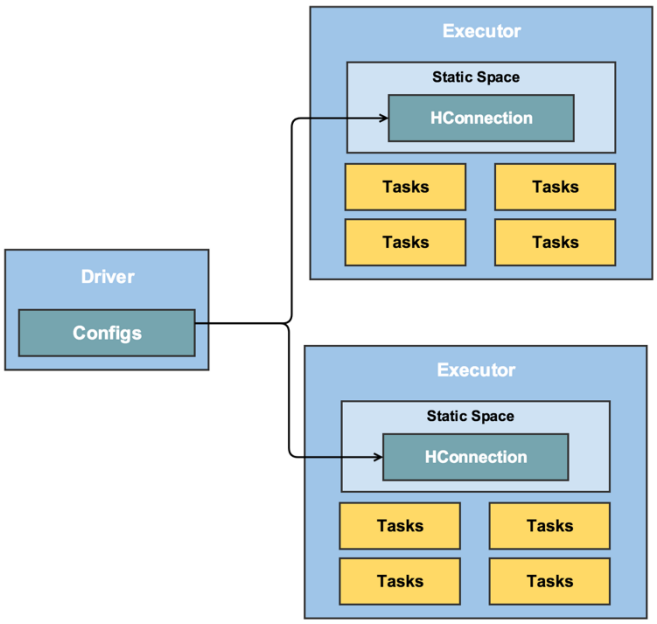

Figure 4: Spark HBase Connection Architecture

At a high-level which is shown in Figure 5, each action is performed in the executors. The driver processes the query, aggregates queries on the region's

metadata, and generates tasks per region. The tasks are sent to the preferred executors co-located with the region server, and are performed in parallel in the executors to achieve better data locality and concurrency. If a region does not hold the data required, that region server is not assigned any task. A task may consist of multiple queries, and the data requests by a task is retrieved from only one region server, and this region server will also be the locality preference for the task [36]. Since the driver is not involved in the real job execution except for scheduling tasks, driver being the bottleneck is avoided.

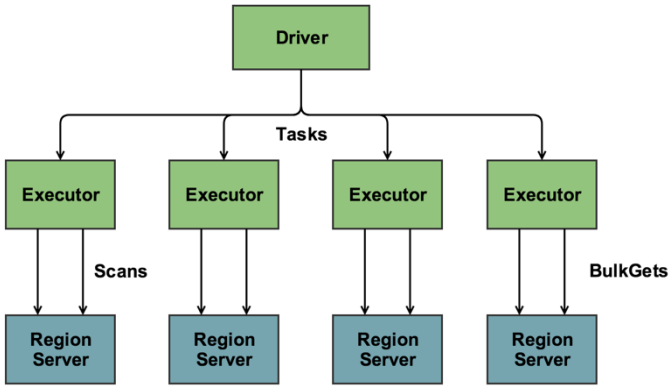

Figure 5: Spark HBase Query Architecture

In HBase two tables are used for this architecture. First one is prediction table. The primary key of prediction table is formed by the union of date of the data and id of the tenant. Date of the data consists of year, month, day, hour and minute columns, defined in this order. These columns will be used the keep data sorted. The data will be sorted by year first. Then will be sorted by month, then by day and goes on like this. The last column is the prediction value. In Figure 6 a high-level design of prediction process is visualized.

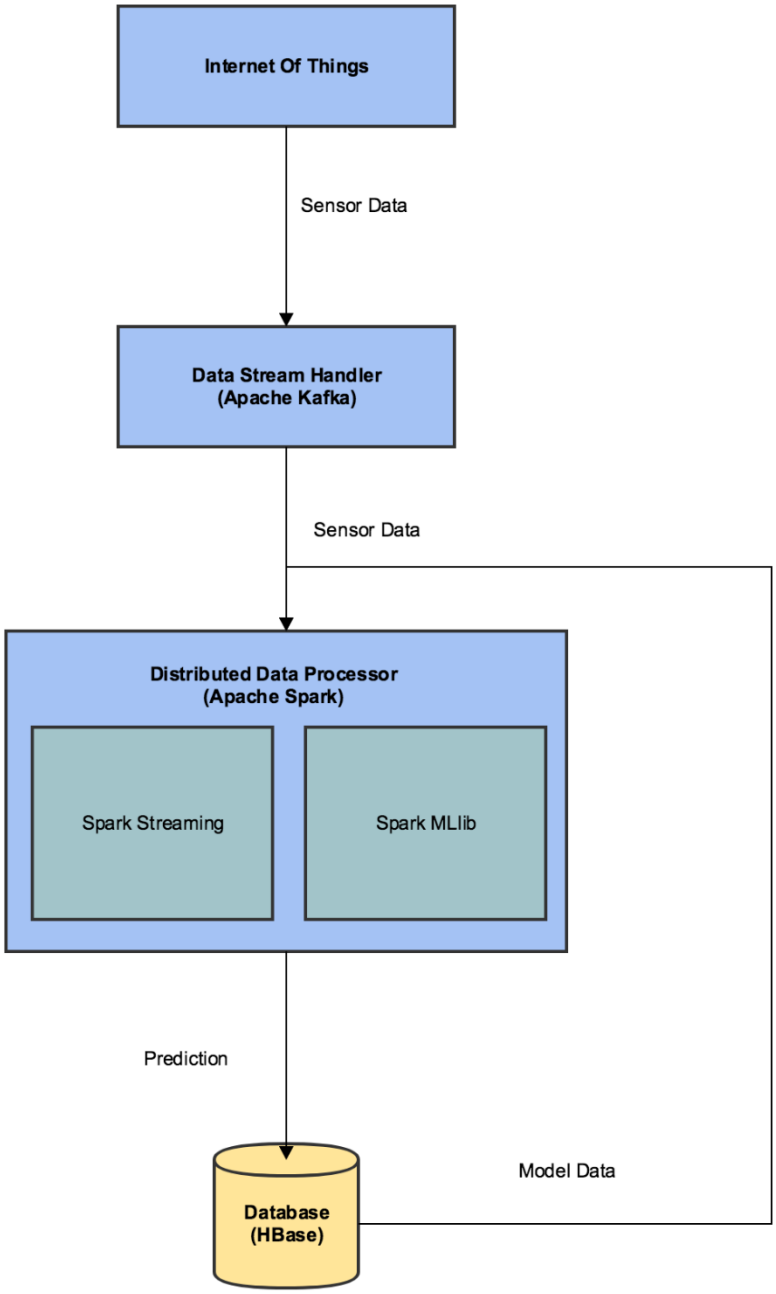

Figure 6: Prediction Process Overview

Second table is model table. The primary key of model data is id of the tenant. Other columns are weights of the machine learning model, interception value of the machine learning model and equal difference value obtained from the

analyses. Each model row in the table is created from batch data. In Figure 7 a high-level design of model creation process is visualized.

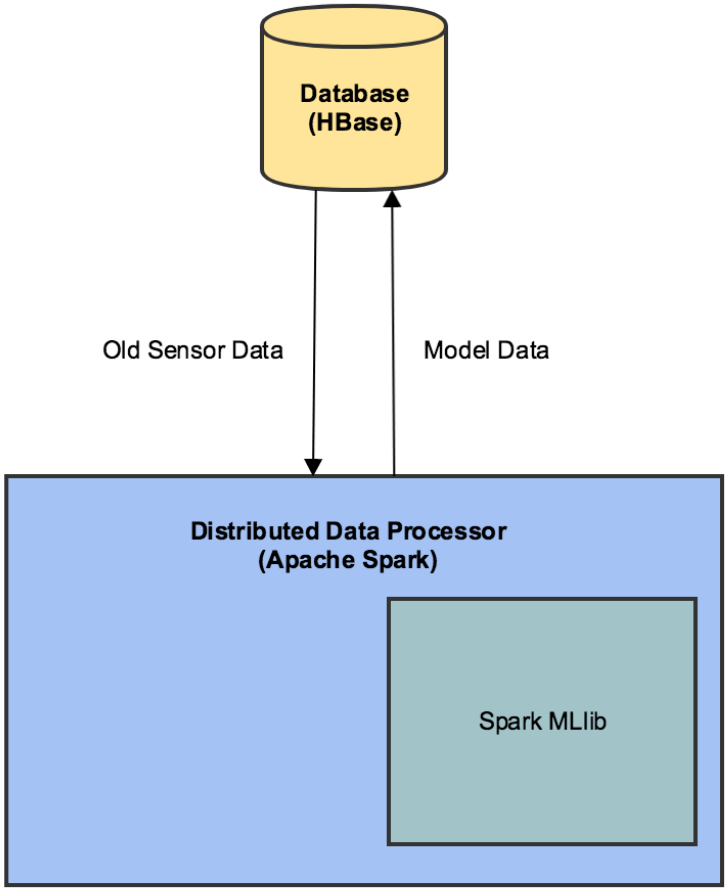

Figure 7: Model Creation Process Overview

Each model row in the table is updated by the analyses and machine learning algorithms from the incoming streaming data. In Figure 8 a high-level design of model update process is visualized.

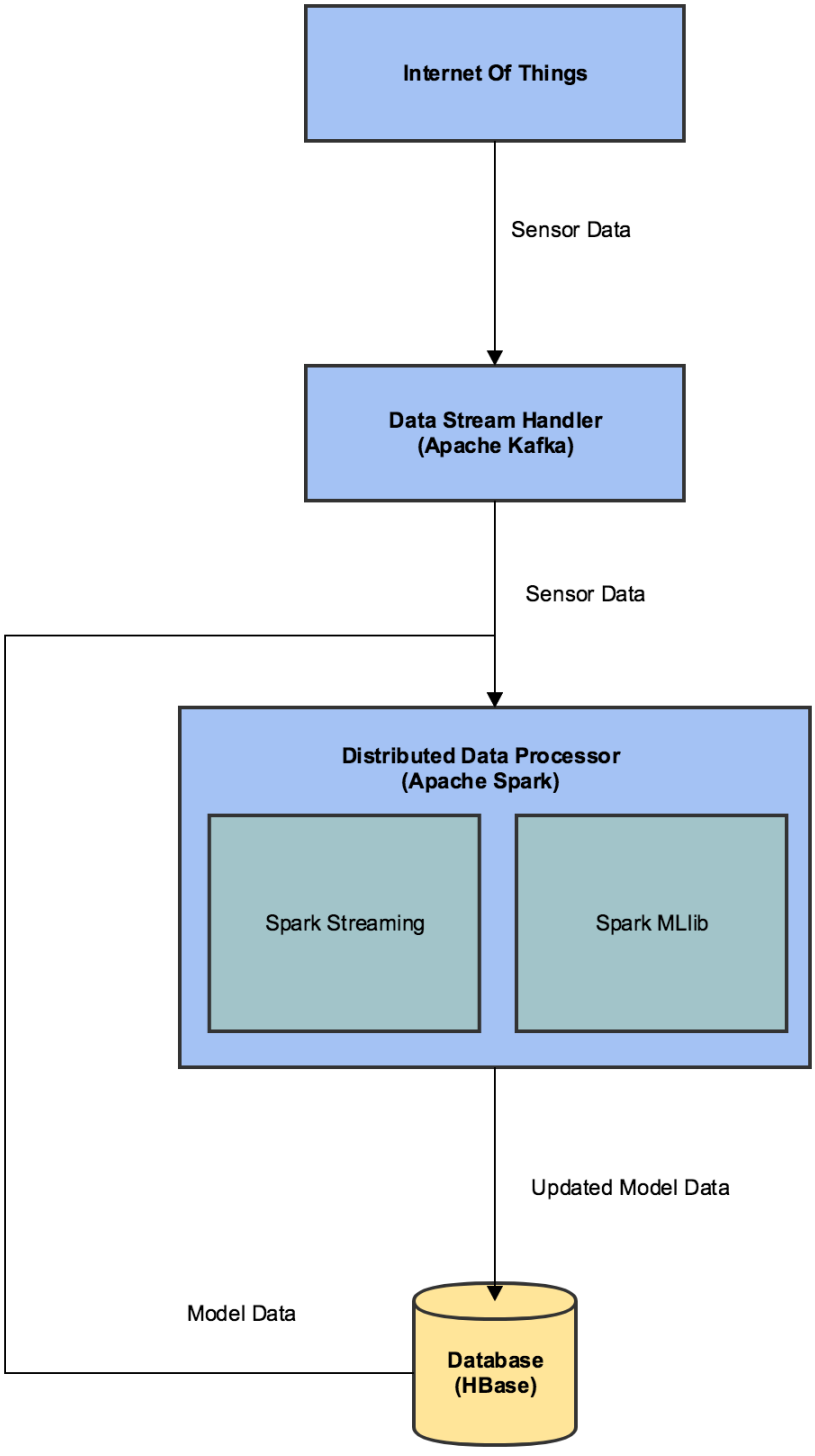

Figure 8: Model Update Process Overview

5.2 Variations and Parameters

Proposed energy optimization system is based on the basic architecture explained in the previous section. To increase the accuracy and performance of the proposed energy optimization system several variations and parameter changes are proposed in this section.

Variations for handling the multi tenancy and parameter changes in partitioning mechanism of the system are related with increasing the performance of the system, while variations for the selected machine learning algorithm and parameter changes in analysis frequency are related with the accuracy of the system.

5.2.1 Handling Multi Tenancy

Two variations for handling multi tenancy will be proposed. Used tools and implemented functionalities are same for each variation. The way system handles multi tenancy is the main distinction between these two variations.

First one is tenant-based variation. In this variation data coming from the sensors are written to separate Kafka topics with the name of tenant id. Spark jobs read data from these topics and analyse the data based on the tenant which means there are as many Spark jobs running on the cluster as the number of active tenants in the ecosystem. When a new tenant is registered to ecosystem a new Spark job is started which reads the incoming sensor data from a new Kafka topic with the name of new tenant id. Both the prediction and the update of the model are achieved with the analysis of incoming sensor data read from this topic. The overview of tenant based variation is shown in Figure 9.

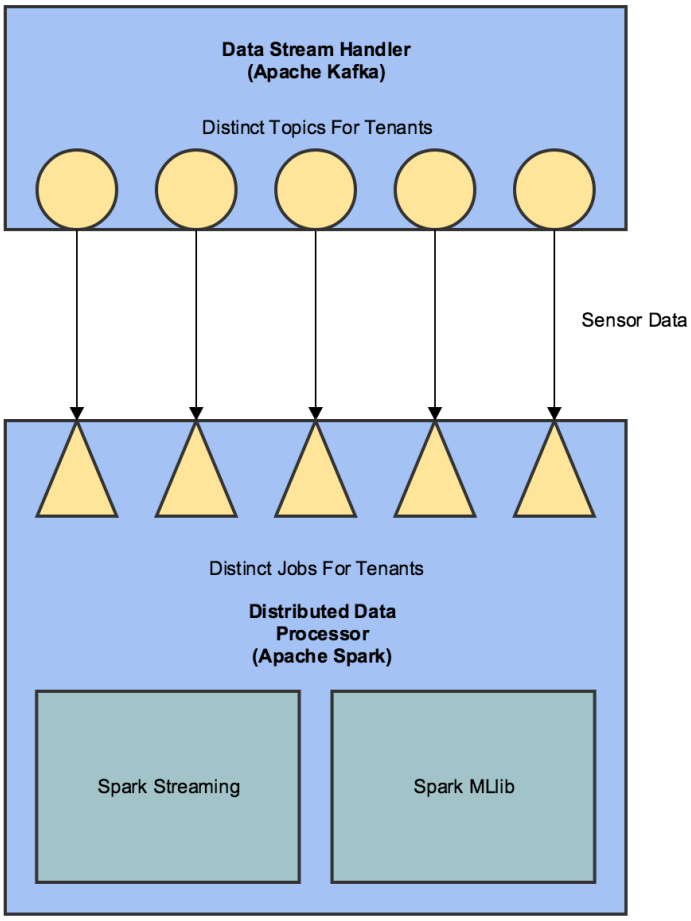

Figure 9: Tenant Based Variation

Second one is pool-based variation. In this variation data coming from the sensors are written to the same Kafka topic. There is only one Spark job for all of the tenants in the ecosystem and that job reads data from this topic. Both the predictions and the update of the models are achieved with the analysis of incoming sensor data read from this topic. In this variation, first tenant info is recognized from the incoming sensor data then the corresponding model is updated and prediction is made with this model. This analysis over data with multi-tenant is achieved by pair wise functionality implemented in Spark. The overview of tenant based variation is shown in Figure 10.

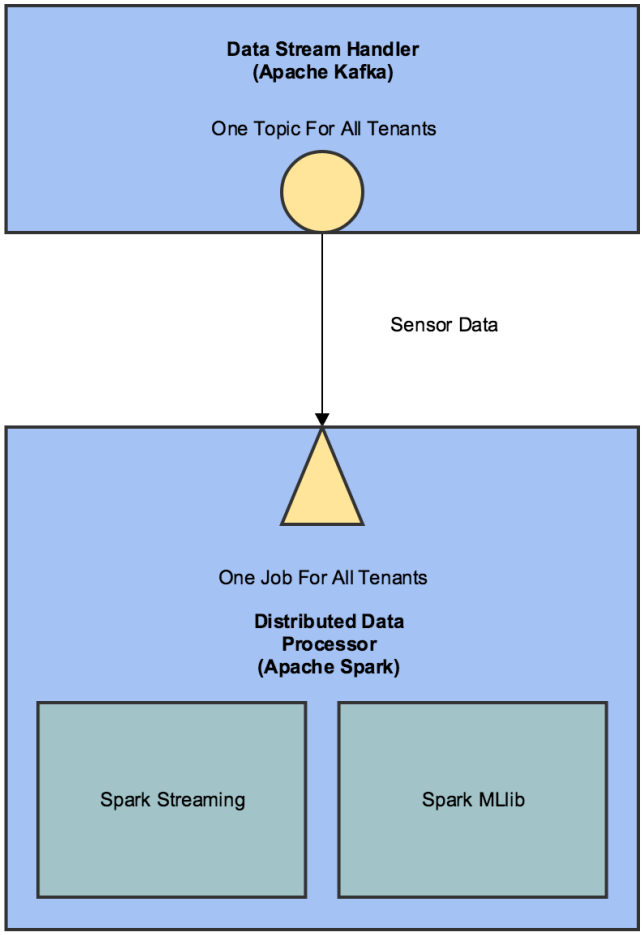

Figure 10: Pool Based Variation

5.2.2 Partitioning

For updating the machine learning models and for predicting the consumption values the sensor data is used by the proposed energy optimization system. Partitioning this sensor data will affect the performance of system. By using too few partitions system cannot utilize all resources available in cluster, on the other hand by using too many partitions system will face excessive overhead in managing many small tasks. Adjusting the parameter of partition size of the sensor data to an optimal value would result in a better system performance.

A partition is a logical chunk of a distributed data set. Spark allows developers to run multiple tasks in parallel across machines in a cluster or across multiple cores on a desktop. The number of tasks will be determined based on

number of partitions [37]. Each partition will be executed in a different core in an executor in parallel as it can be seen in Figure 11, which means the maximum count of tasks running in parallel is equal to core per executor count times executor count.

In Spark partitioning is done automatically without the need for programmer's effort. However, there are times when programmer would like to adjust the partitioning scheme with respect to the needs of the application. Spark allows obtaining the partition count of a RDD, DStream or a DataFrame and adjusting it to desired value in number of ways.

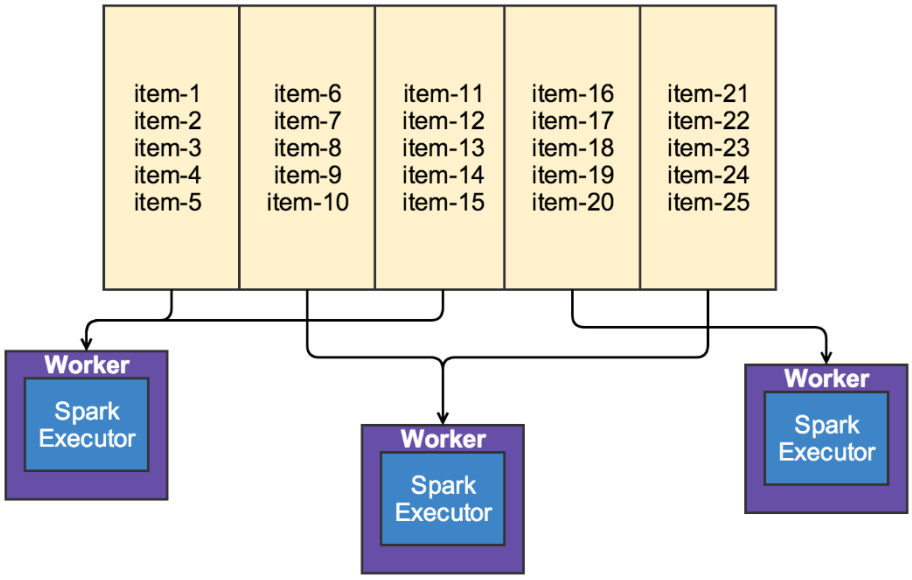

Figure 11: Visualization of RDD Being Partitioned

In the example code which is shown in Figure 12, first the sensor data is read. After the sensor data is read, the number of desired partitions is given as a parameter to the repartition function. Repartition function make partitions get redistributed among executors since it calls shuffle inside. The partition parameter will be applied to all further transformations and actions on this DStream and since remaining flow of the feature is based on this sensor data partition parameter will be applicable in every point of this feature.

```
JavaDStream<String> streamingData = jssc.textFileStream(streamingPath).repartition(partition);
```

Figure 12: Arranging Partition Count in Spark

5.2.3 Prediction Frequency

For creating the machine learning models batch sensor data is used by the proposed energy optimization system. Creating this machine learning model with data with different measurement frequencies results in different machine learning models in terms of variable weights. Using different machine learning models in consumption prediction feature of the basic architecture while using the same streaming sensor data as input leads to different consumption prediction values. Adjusting the parameter of data measurement frequency to an optimal value would result in a better system accuracy. On the other hand, the parameter of data measurement frequency also affects the frequency of prediction and model updating features. If the model is created with data with measurements of N minutes, consumption prediction and model update can be carried out every N minutes at the highest frequency settings. Basic architecture provides carrying out the prediction and model update features every N x K minutes with lower frequency settings. Since carrying out prediction in the most frequent is the sensible choice for an energy optimization system measurement frequency is also equivalent to prediction frequency in the proposed system. Changing the measurement frequency of the data models are created from also effects the performance of the model creation functionality in the proposed system since the volume of the data processed in model creation phase is related with measurement frequency.

```
public static int getHourOfWeek(Date date) {...}

public static int get30MinuteOfWeek(Date date) {...}

public static int get20MinuteOfWeek(Date date) {...}

public static int get10MinuteOfWeek(Date date) {...}

public static int get5MinuteOfWeek(Date date) {
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(date);
    int retVal = calendar.get(Calendar.HOUR_OF_DAY) * 12;
    retVal += calendar.get(Calendar.MINUTE) / 5;
    retVal += 24 * 12 * (calendar.get(Calendar.DAY_OF_WEEK) - 1);
    return retVal;
}
```

Figure 13: Feature Extraction Code for Different Sampling Frequencies

With the change in measurement frequency, the feature extraction functionality also has changes. In the example code which is shown in Figure 13 the change of extraction of one feature is displayed with different measurement frequencies. Feature extraction affects both model update and consumption prediction features of the proposed system.

5.2.4 Model Update Frequency

For updating the machine learning models streaming sensor data is used by the proposed energy optimization system. Updating the model with different frequencies results in different machine learning models in terms of variable weights in certain time intervals. Using different machine learning models in consumption prediction feature of the basic architecture while using the same streaming sensor data as input leads to different consumption prediction values. Adjusting the parameter of model update frequency to an optimal value would result in a better system accuracy. Changing the model update frequency means changing the window size of data that is going to be processed in the current interval. To be clearer if measurement frequency is N minutes and model update frequency is N x K minutes window size is K for model update and in each model update K units of micro-batches will be processed.

```
JavaStreamingContext jssc = new JavaStreamingContext(jsc, Durations.seconds(interval));
```

Figure 14: Arranging Model Update Frequency in Spark

In the example code which is shown in Figure 14, sensor data is read in periods of "interval" seconds. All the newly arrived sensor data since last interval is read in current interval and model is updated with analysis of this data. In this way, old data does not have to be processed in every interval, model is updated with processing of newly arrived data only.

5.2.5 Machine Learning Algorithms

All three main functionalities in the basic architecture are dependent on the selected machine learning model. While changing the machine learning algorithm does not affect the basic architecture, it changes the inside mechanisms like parsing raw sensor data to data which will be used in Spark MLlib functions like model creation, model update and consumption prediction. Changing the machine learning algorithm will affect the proposed energy optimization system's both accuracy and performance. A more accurate machine learning model does not always mean a better fit for the proposed system since it can be worse in terms of performance or vice versa. Another important point for selecting the machine learning algorithm is the compatibility of the algorithm with Apache Spark's streaming and batch functionality. Selected algorithm's effect on model creation performance of the system is also taken into consideration. Three variations for the selected machine learning algorithm are proposed which are all implemented in Apache Spark MLlib.

First tested algorithm is logistic regression. Logistic regression is used to explain the relation between one dependent binary variable and one or more independent variables. Logistic regression is mostly referred as the most popular classification algorithms other than support vector machines. In Spark MLlib linear support vector machines support only binary classification, while logistic regression supports both binary and multiclass classification problems. Spark MLlib also has the support for logistic regression in streaming mode. Second tested algorithm is linear regression. Linear regression is a linear method for modelling the relation

between a dependent variable and one or more independent variables. Linear regression is the most commonly used and most popular regression algorithm. Spark MLlib also has the support for linear regression in streaming mode. The last tested algorithm is regression tree. Regression trees uses a predictive model to arrive at conclusions about the item's target value from observations about this item. Regression trees are widely used since they are easy to interpret, do not require feature scaling, extend to the multiclass classification, handle categorical features setting, and are able to capture non-linearities and feature interactions. Spark MLlib does not have the support for regression trees in streaming mode.

More complicated algorithms like neural networks are not tested because most important priority for our system is performance and more complicated algorithms have their respective performance issues in model creation and model update functionalities.

For comparing the accuracies of selected machine learning algorithms K-Fold Cross Validator implementation in Spark MLlib is used. At a high level comparing tool works as follows which is also shown in Figure 15. Input data is split into training and test datasets. Then, for each training-test pair selected algorithm is iterated through the set of parameters and fitted model is obtained using training dataset. In the last phase model's performance is evaluated with the selected evaluator function using test dataset.

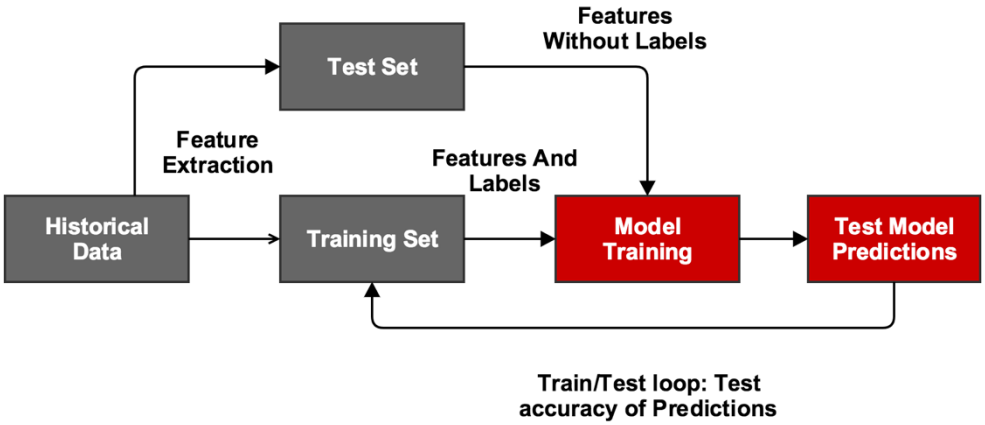

Figure 15: Cross Validation Process in Spark

On the other hand, effect of K-Fold is as follows. Cross Validator implementation splits the dataset into a set of folds which are used as separate training and test datasets. For example, with k folds, implementation will generate 3 pairs of training-test datasets, each of which uses 2/3 of the data for training and 1/3 for testing.

5.3 Alternative Machine Learning Tools

R is a language for statistical component which provides a wide variety of graphical and statistical techniques like linear and nonlinear modelling, time-series analysis, clustering, classical statistical tests, and classification and is greatly extensible. The R language is generally used among data scientists and statisticians for data analysis and developing statistical software. Spark has also an R package called SparkR which provides a distributed data frame implementation that supports operations like selection, filtering and aggregation however on large datasets. Since R is an open source programming language and has lots of statistical libraries to use and since it is popular among statisticians and data scientists, using it as the machine learning tool may have advantages. Another option to use Spark with R language is pipe functionality implemented in Spark. Pipe operator in Spark, allows developer to process RDD data using external applications. External application gets the input from Spark using stdin and returns the output produced to Spark using stdoud. External scripts which written in R language can be accessible from Spark with this pipe functionality.

CHAPTER 6

IMPLEMENTATION & TESTING

A typical IoT system consists of five main components which can be listed as IoT, data storage component, data processor component, visualization component and management system. IoT can be described as a cloud of sensors, which generates data continuously. Data storage component is where this IoT data is stored, it is mostly selected as a NoSQL data store. Data processor uses the stored data to do technical analyses and sends commands or data to visualization component and management system. Data visualization component is used to visualize incoming data in a human-readable format. Management system, manages a sensor network, for example a building management system (BMS). These components and communication and processing mechanisms between them are described in detail in Chapter V.

As described earlier, there are several researches testing reasonable solutions for storage and analysing problems of big data. Some of these previous researches like Natasha Balac et al [18] focus on testing micro-grid solutions, while our tests focus on building based solutions. Previous researches which are focused on building based solutions mostly run tests with batch analysis like Ioan Petri et al [20] and W. Khamphanchai et al [21] while our tests are run with both streaming and batch analysis. Also, some of the previous researches like Liehuo Chen et al [26] and Sunil Mamidi et al [27] focus on building based streaming solutions, but our tests are different from them in terms of our architecture which provides us to test both machine learning, batch and streaming analyses by using same structure. This thesis aims to test an energy optimization framework which is suitable for working on both historical and real-time sensor data in a multi-tenant manner and will be based on Apache Spark. In this chapter, a feasible subset of the proposed tools and

architectures will be tested and results will be presented. Testing and environment details will be described.

6.1 Tested Variations and Parameters

The system has three main features to be tested. First feature is performance. System must have met the timing requirements of a real-near time system. Second feature is accuracy. Although selecting the best machine learning algorithms is not in the scope of this thesis, selecting a machine learning algorithm with reasonable success rate must be provided by the system. Scalability is also a key feature to be tested. Increase in tenant count or frequency of sensor data should be easily coverable with more resources supervened to the proposed system.

6.1.1 Multi Tenancy Handling Tests

In this set of tests model update and consumption prediction functionalities of the system are tested with two variations based on the main architecture with similar structures but different handling mechanisms for multi tenancy. Performance and scalability of system are examined with each variation.

6.1.2 Partitioning Tests

In this set of tests model update and consumption prediction functionalities of the system are tested with data with different partition counts. Performance and scalability of system while using different partition counts are examined. Also, performance and scalability of system are examined while using same partition counts but with different core and executor distribution in the cluster.

6.1.3 Prediction Frequency Tests

In this set of tests model creation functionality is tested with data with different frequencies of data measurement. Accuracy of the system is examined with models created with data with different measurement frequencies. Also, performance of model creation functionality is examined when data with different measurement frequencies is processed. Measurement frequency of the data models are created with is also equivalent to the prediction frequency of the proposed system.

6.1.4 Model Update Frequency Tests

In this set of tests model update functionality is tested with different frequencies. Accuracy of the system is examined with models updated on different frequencies. Moreover, in the system model update frequency is related with the measurement frequency of data which related model is created with.

6.1.5 Machine Learning Algorithm Tests

In this set of tests different kinds of machine learning algorithms are tested. Although selecting the best machine learning algorithms is not in the scope of this thesis, selecting a machine learning algorithm with reasonable success rate must be provided by the system. Accuracy of the system with different kinds of machine learning algorithms is examined. Also, the effect of the selected algorithm on model creation functionality is tested.

6.2 Testing Environment

Testing environment is built on MapR Ecosystem. HBase is chosen as NoSQL service in the ecosystem. Spark Core, Spark Streaming and Spark MLlib is used for doing technical analyses. Tests are deployed on the cluster of Big Data and Innovation Laboratory at METU. Cluster consists of 7 nodes with following properties:

- 2 x Intel Xeon 10-core 2.40GHz CPU
- 8 x 6TB = 48TB NL-SAS Data Disk
- 8 x 16GB = 128GB RAM
- 2 x 240GB = 480GB SSD O/S Disk

Tests are initiated from the web interface of MapR called HUE and displayed there. Therefore, data transmission is only in cluster's own system and performance measurements are performed for internal transmission only. Then only results are returned to the user by the web interface.

The sensor data used in the tests is the sensor data of a real building. There is one year of data from this building and data is written with frequency of 5 minutes. This data consists of consumption, outside temperature, timestamp and tenant id

values. For executing tests in which different frequencies of analysis are tested, this data is modified accordingly, to be clearer when frequency of data measuring is decreased consumption values are summed and outside temperature values are averaged. 7 different features are extracted from this data. First one is related with timestamp and prediction frequency, for example if prediction frequency is 5 minutes the extracted feature is “5 minutes of week” variable, meanwhile if prediction frequency is 10 minutes the extracted feature is “10 minutes of week” variable. While calculating this variable working hours and working days of the relevant building is also taken in consideration. Other 6 features are related with the outside temperature. The difference between the highest and lowest outside temperature value is divided by 6 and the equal difference parameter is calculated. Then according to this equal difference value and the outside temperature value other features are extracted. For creating and updating the machine learning models, data with consumption values are processed. On the other hand, while predicting the consumption, values processed data does not have the consumption values intrinsically. For creating machine learning models %80 of the data is used, meanwhile for consumption prediction and model update features remaining %20 of the data is used. By not using the same data in both model creation and consumption prediction, accuracy of the models is tested more successfully.

For executing the tests in which the accuracy of the system is tested, this real-life data is used for obtaining reasonable accuracy rates. On the other hand, for executing the tests in which the performance of the system is examined, data replicated from this real-life data with random values with respect to examined tenant counts is used.

6.3 Test Results and Comments

Results of all tests are explained in details and comments on all the test results are given in this section.

6.3.1 Multi Tenancy Handling Tests

In this set of tests, two variations of basic architecture with different mechanisms for handling multi tenancy is examined in terms of performance. In both

of these variations two different features are tested, model update and consumption prediction.

6.3.1.1 Model Update

In this feature, model data is read from HBase and sensor data is read from relevant Kafka topic. Model data is read once when the application is first initializing, while sensor data is read at regular intervals determined by system. At each interval, model is updated with the analysis of weights of old model and newly read sensor data. Update of the machine learning model is carried out by selected algorithm from Spark MLlib. At each interval, updated model is written to HBase to let the system use the updated models in prediction test cases.

- Tenant Based Variation Tests

In this set of tests, different Spark jobs are run for each tenant. Also, data coming from sensors are read from different Kafka topics for each tenant.

Table 1: Results of Tenant Based Variation Tests of Model Update Feature

Tenant Count	5	10	20	50
Execution Time(seconds)	2	2	-	-

- Pool Based Variation Tests

In this set of tests, a single Spark job is run for every tenant. Also, data coming from sensors are read from a single Kafka topic for every tenant.

Table 2: Results of Pool Based Variation Tests of Model Update Feature

Tenant Count	5	10	20	50	100	200
Execution Time(seconds)	2	2	2	6	13	25

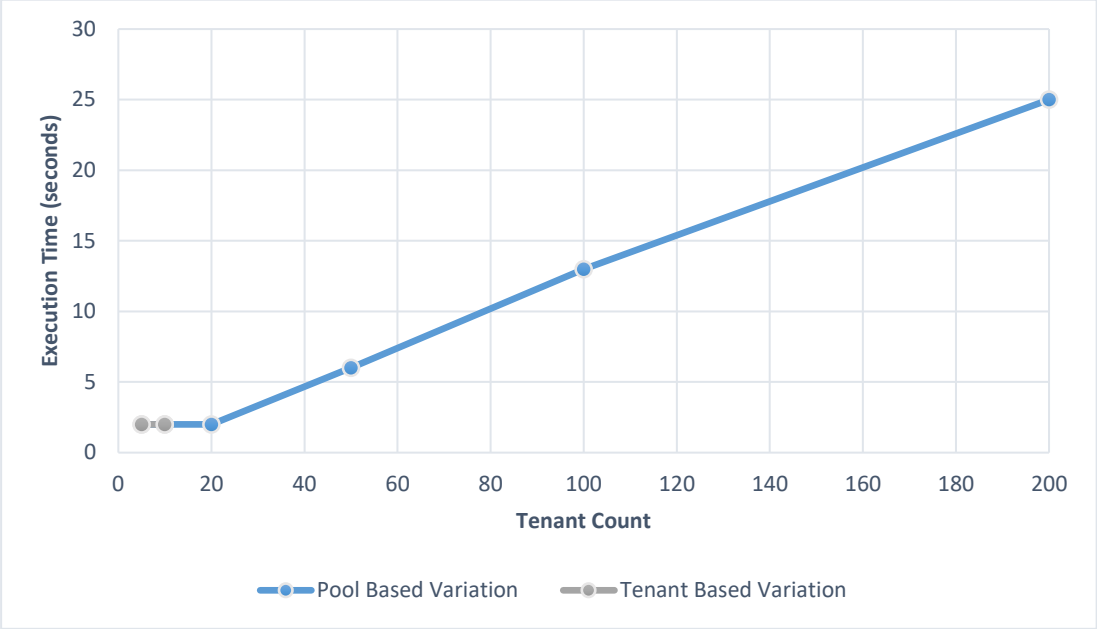

Figure 16: Results of Multi Tenancy Handling Tests of Model Update Feature

In pool based variation one mutual Spark job is running for update of all models. Parallelization of model update when tenant size increases must be managed programmatically. Parsing the raw data to a usable format for model updating is implemented in parallel by the Spark pair functionality. Unfortunately, due to the internal implementations in Spark, parallel update of models cannot be achieved with the proposed basic architecture. This leads to a not constant execution time of model update while tenant count is increasing. However, model update with pool based variation uses small amounts of resource.

In tenant based variation a different Spark job is running for update of every separate model. This leads to a constant execution time of model update while tenant count is increasing, on the other hand this also leads to too much resource usage. Using our cluster at most 15-20 Spark jobs can be activated parallel.

Results in the Figure 16 show that tenant based variation is practicable with small numbers of tenant count. For the systems with tenant count more than 15-20, pool-based variation is the reasonable selection.

6.3.1.2 Consumption Prediction

In this feature, model data is read from HBase and sensor data is read from relevant Kafka topic. Model data and sensor data is read at regular intervals

determined by system. Since model data is read at regular intervals and not only when the application is initializing, updated model data is used for prediction. At each interval, prediction on consumption values is carried out with the analysis of weights of model and newly read sensor data. Prediction of consumption values is carried out by selected algorithm from Spark MLlib.

- Tenant Based Variation Tests

In this set of tests, different Spark jobs are run for each tenant. Also, data coming from sensors are read from different Kafka topics for each tenant.

Table 3: Results of Tenant Based Variation Tests of Prediction Feature

Tenant Count	5	10	20	50
Execution Time(seconds)	1	1	-	-

- Pool Based Variation Tests

In this set of tests, a single Spark job is run for every tenant. Also, data coming from sensors are read from a single Kafka topic for every tenant.

Table 4: Results of Pool Based Variation Tests of Prediction Feature

Tenant Count	5	10	20	50	100	200
Execution Time(seconds)	1	1	1	1	2	2

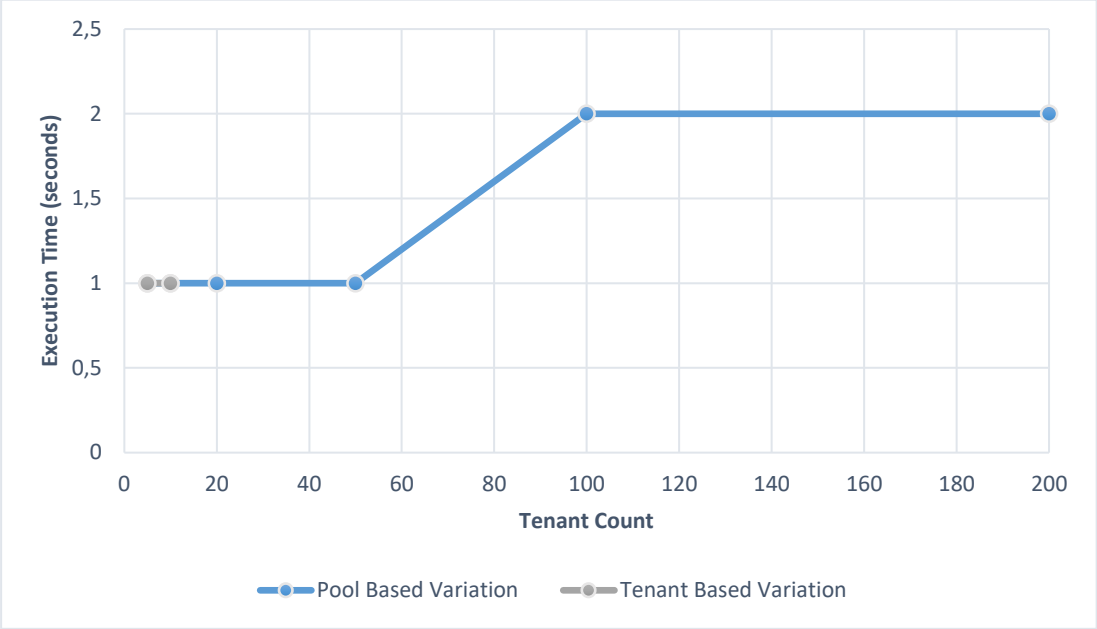

Figure 17: Results of Multi Tenancy Handling Tests of Consumption Prediction Feature

In pool based variation one mutual Spark job is running for predicting consumption of all tenants. Parallelization of consumption prediction when tenant size increases must be managed programmatically. Parsing the raw data to a usable format for consumption prediction and predicting the consumption is implemented in parallel by the Spark pair functionality. This leads to a constant execution time of consumption prediction if resources are adjusted to tenant count while tenant count is increasing. Also, consumption prediction with pool based variation uses small amounts of resource.

In tenant based variation a different Spark job is running for predicting consumption of every separate tenant. This leads to a constant execution time of model update while tenant count is increasing, on the other hand this also leads to too much resource usage. Using our cluster at most 15-20 Spark jobs can be activated parallel.

Results in the Figure 17 show that pool-based variation has a scalable infrastructure thereby has the ability to handle multi-tenancy when resources are extended. It is safe to say that pool-based variation is the reasonable selection among these two variations independent of tenant size.

6.3.2 Partitioning Tests

In this set of tests, effect of partition count over the performance of system is examined. Pool based variation from the above-mentioned variations is used in these tests. In the system two different features are tested, model update and consumption prediction.

6.3.2.1 Model Update

In this set of tests, data with 500 tenants are examined with different partition counts. Streaming data is acquired every 60 seconds. Testing environment has the available resource for supporting the parallelization of each case.

Table 5: Results of Partitioning Tests of Model Update Feature with 500 Tenants

Partition Count	1	2	5	10	50	100
Execution Time(seconds)	40	28	29	91	115	115

Model update functionality of the system is tested using data with 200 tenants. While using too few partitions, benefits of parallelization in Apache Spark is not being used. Increasing the partition count gives the system the ability to handling the functionalities in parallel, but as it can be seen in the results using too many partitions bring excessive overhead to system since it will be managing many small tasks.

When model update is done with data with 200 tenants the optimal partition count is 2-5 according to results. Results show that to stabilize the performance of the system partition count must be increased when tenant count or data arrival frequency is increased.

6.3.2.2 Consumption Prediction

In this set of tests, first data with 1000 tenants are examined with different partition counts. Streaming data is acquired every 30 seconds. Testing environment has the available resource for supporting the parallelization of each case.

Table 6: Results of Partitioning Tests of Prediction Feature with 1000 Tenants

Partition Count	1	2	5	10	50	100
Execution Time(seconds)	2.6	2	2	2.2	5.2	7

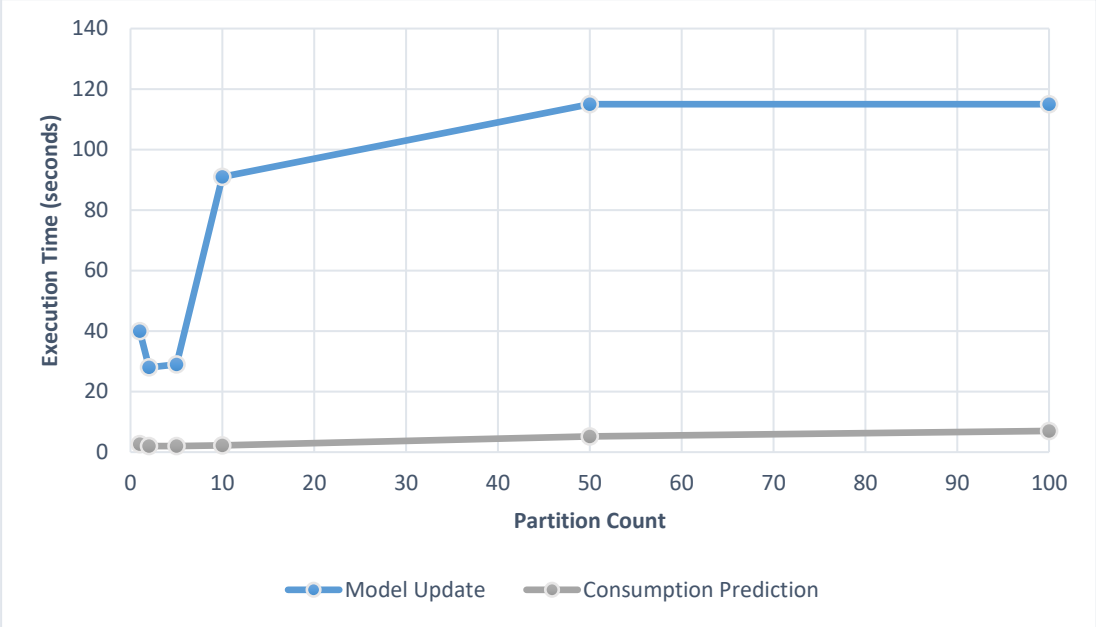

Figure 18: Results of Partitioning Tests with Constant Core/Executor Distribution

Consumption prediction functionality of the system is tested using data with 1000 tenants. When prediction is done with data with 1000 tenants the optimal partition count is 2-5 according to results in Figure 18. While using too few partitions, benefits of parallelization in Apache Spark is not being used. Increasing the partition count gives the system the ability to handling the functionalities in parallel, but as it can be seen in the results using too many partitions bring excessive overhead to system since it will be managing many small tasks.

In the last tests of this case, the effect of different numbers of core and executor distribution with same partition size on the performance is examined. Data with 20000 tenants are used in this set of tests with partition size 10.

Table 7: Results of Partitioning Tests of Prediction Feature with Different Core/Executor Distribution

Executor Count	10	5	4	3	2
Core Per Executor Count	1	2	3	4	5
Execution Time(seconds)	4	4	3.3	3.3	3.3

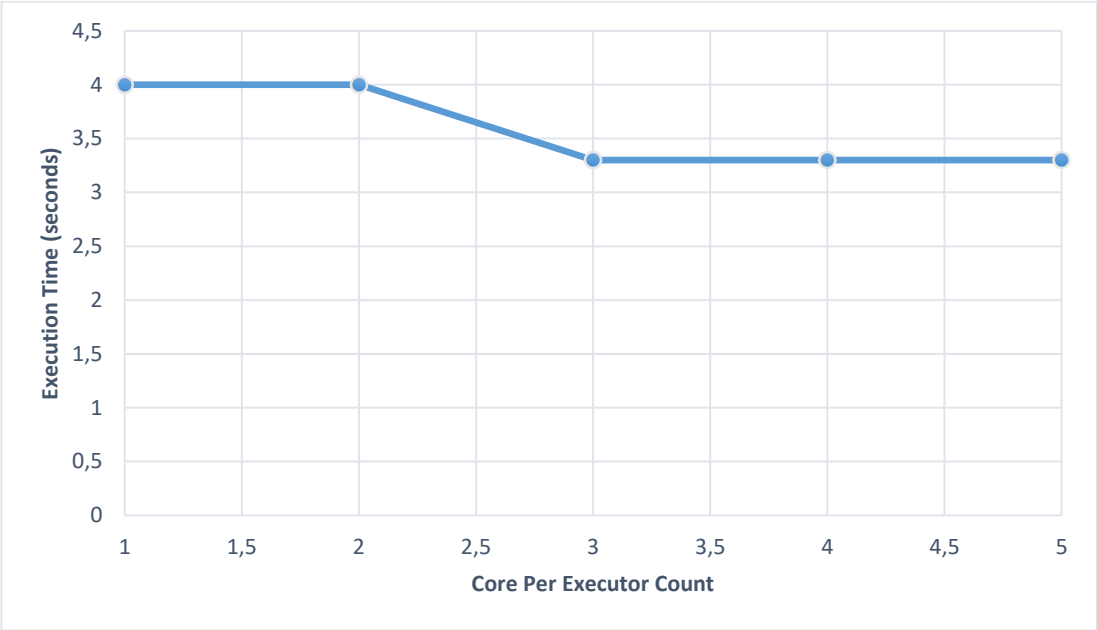

Figure 19: Results of Partitioning Tests of Prediction Feature with Different Core/Executor Distribution

Test results in Figure 19 show that using more than 2 cores per executor improves the performance. After 2 cores per executor the performance of the system is not effected observably. In our tests memory is not a limiting factor, but in a scenario with the need of more memory usage improving the executor count will be reasonable since executors splits the system memory.

6.3.3 Prediction Frequency Tests

In this set of tests, accuracy of system with different analysis frequencies are examined. Pool based variation from the above-mentioned variations is used in these tests. In this set of tests, machine learning models are created from batch sensor data containing one year of information. In each case this one year of data is measured in different frequencies. System uses linear regression algorithm as the selected algorithm in all of the cases.

Table 8: Results of Model Creation Accuracy of Prediction Frequency Tests

Model Sampling Frequency (minutes)	5	10	20	30	60
Mean Squared Error	0.0519	0.1407	0.4031	0.9171	2.6683
Root Mean Squared Error	0.2279	0.3752	0.6349	0.9576	1.6335

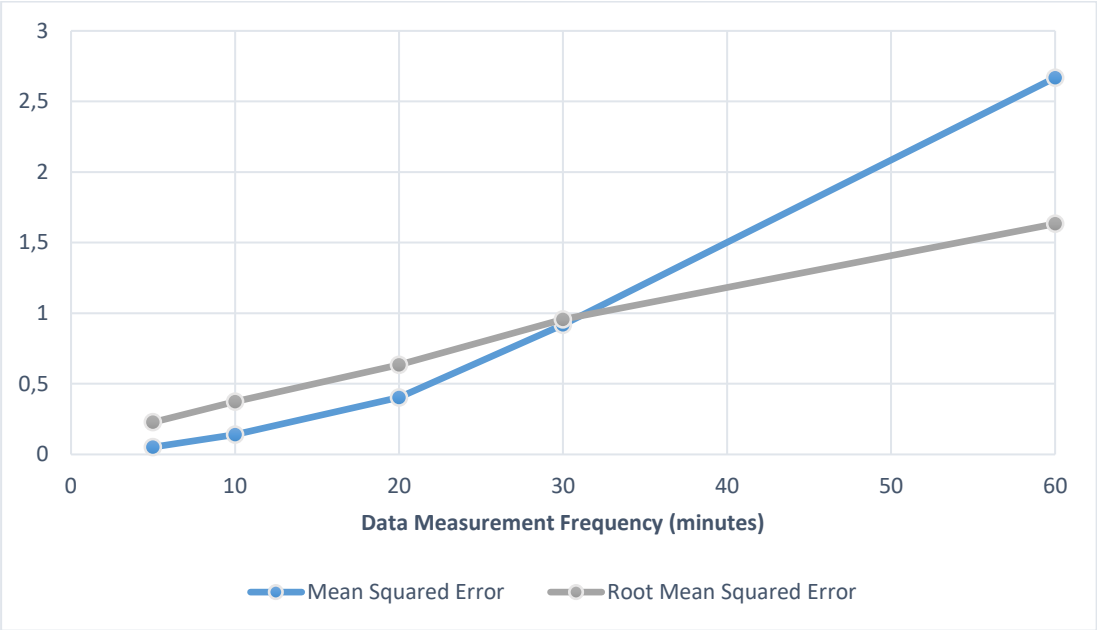

Figure 20: Results of Model Creation Accuracy of Prediction Frequency Tests

As it can be seen in the results in Figure 20 as the measuring gets more frequent all metric values (mean squared error, root mean squared error and r square) gets lower. It is safe to say that with models created from more frequently measured data, our system becomes more accurate, of course this type of analyses provides more flexibility to the temperature changes in small time periods. Another advantage of creating models with data with more frequently measurements is the compatibility of system to real-time pricing which is mentioned in previous chapters.

Creating models from more frequent measuring means using more data to create model, which may mean creating model takes more time but it is ignorable for our system since models are created only once when a new tenant is introduced to the system. Afterwards models are updated using their initial weights. Nevertheless, durations of model creation with different frequencies of measurement are examined.

Table 9: Results of Model Creation Performance of Prediction Frequency Test

Model Sampling Frequency (minutes)	5	10	20	30	60
Model Creation Duration (seconds)	50	46	43	42	41

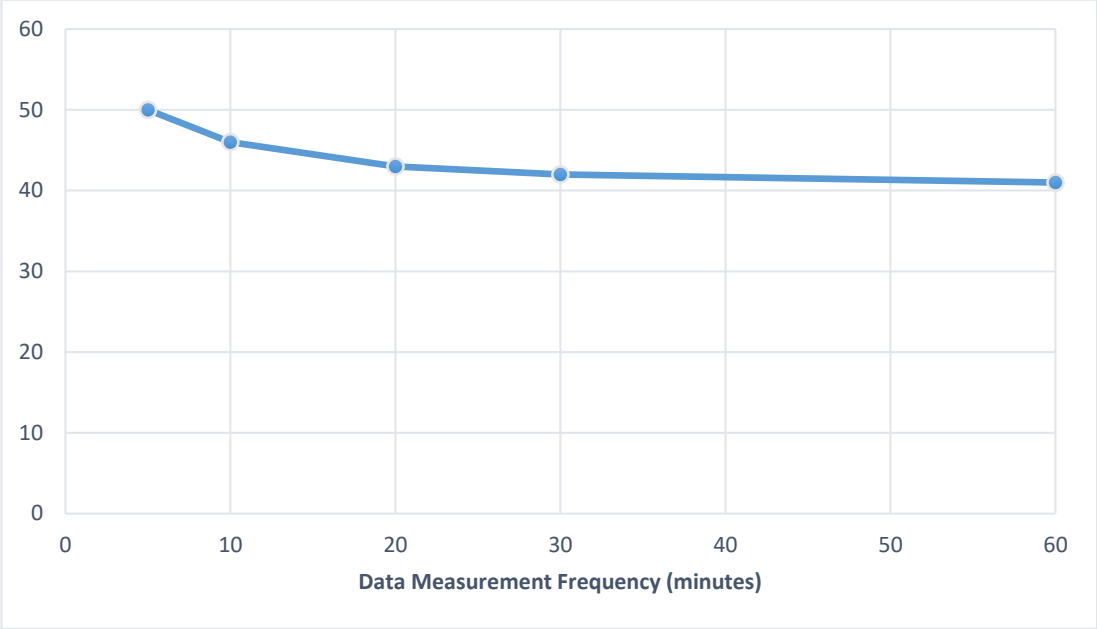

Figure 21: Results of Model Creation Performance of Prediction Frequency Test

As it can be seen in Figure 21 creating models from more frequent measuring and using more data to create models, has a very slight disadvantage in terms of performance for the proposed system and since model creation is executed once for each tenant this slight disadvantage is ignorable.

6.3.4 Model Update Frequency

In this set of tests, same machine learning model is updated with 10 minutes of frequency in one case and with 60 minutes of frequency in other case. Data arrival frequency is constant in this set of tests, which leads to models becoming identical when the frequencies intercept later on. Which means in each case model is identical after 60 minutes. However, within this period it is two different models in terms of variable weights which leads to different consumption prediction values.

Table 10: Results of Model Update Frequency Tests

Time (minutes)	10	20	30	40	50	60
Mean Squared Error	0.17049	0.17049	0.17047	0.17037	0.17031	0.16995
Root Mean Squared Error	0.41291	0.41290	0.41288	0.41276	0.41269	0.41225

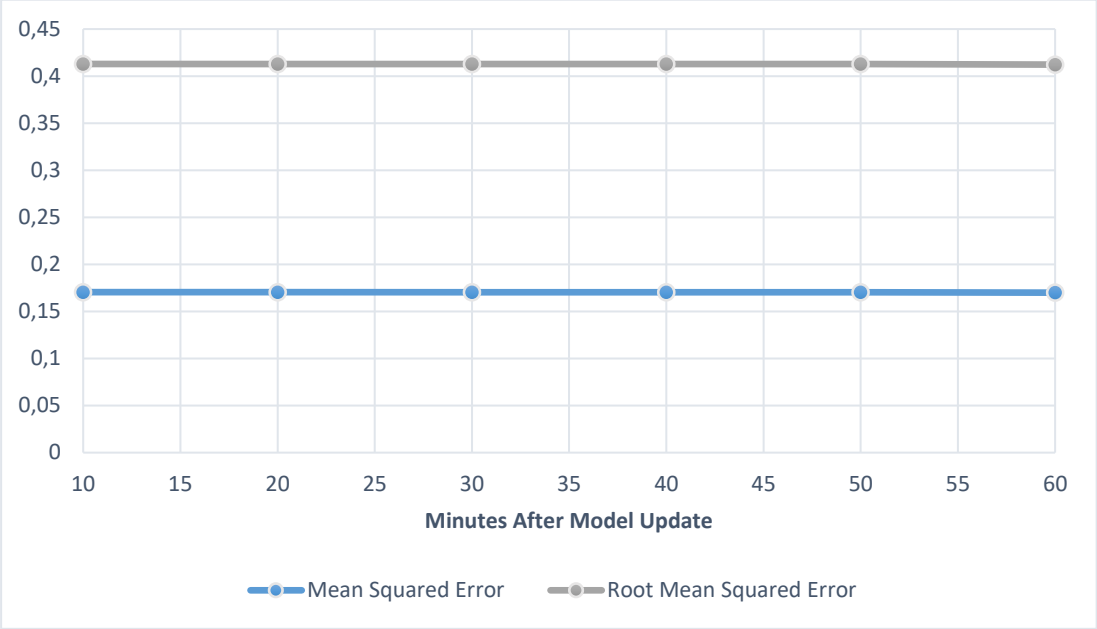

Figure 22: Results of Model Update Frequency Tests

As it can be seen in the results in Figure 22 accuracy of the system is increasing in every time the model is updated. On the other hand, as it can also be seen in the results the improvement in accuracy is very slight within 10 minutes of time intervals. Model updating functionality is the costliest functionality in the proposed energy optimization system since handling the multi-tenancy for this functionality has its' problems. It is not safe to propose that updating the model in the most frequent manner is the optimal solution, since increasing the frequency can prevent the model update functionality from completing in time when the tenant size increases. There is a payoff in the system between the tenant count and the model update frequency. For handling more tenants successfully, the model update frequency can be decreased since the improvement in accuracy is very slight.

6.3.5 Machine Learning Algorithm Tests

In this set of tests, effect of different type of machine learning algorithms are examined on the accuracy and performance of the system. Pool based variation from the above-mentioned variations is used in these tests. In each test case models are created with 5 minutes of sampling frequency.

For accuracy of the system metrics of models created with different machine learning algorithms are examined. For performance of the system two different

functionalities are tested, model update and consumption prediction with 1000 tenants.

Table 11: Results of Accuracy of Machine Learning Algorithm Tests

Machine Learning Algorithm	Linear Regression	Logistic Regression	Regression Tree
Mean Squared Error	0.0519	0.3741	0.0469
Root Mean Squared Error	0.2279	0.6116	0.2166

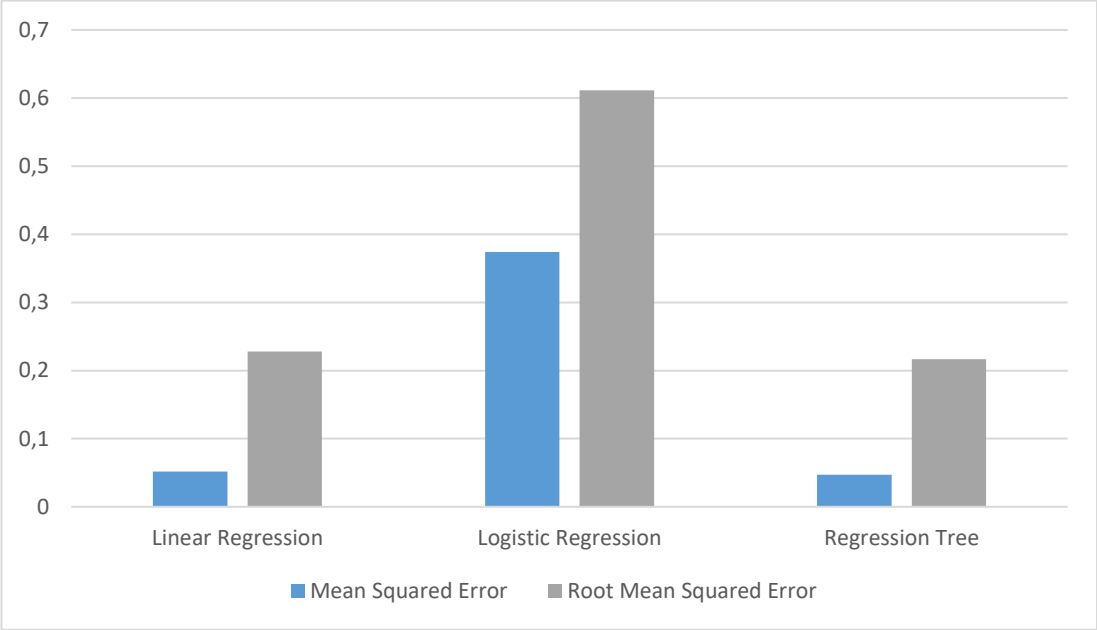

Figure 23: Results of Accuracy of Machine Learning Algorithm Tests

Three different machine learning algorithms that are implemented in Spark MLlib is used in these tests. Each algorithm is run with 5 minutes of input sampling frequency, to make analysis frequency not a factor in this set of tests. Results in the Figure 23 show that the model created with logistic regression algorithm has the highest mean squared and root mean squared error values, which makes it the worst option of these three algorithms for our system. Model created with regression tree algorithm has the lowest mean squared and root mean squared error values, which makes is the best option of these three algorithms for our system in case of accuracy, but since Apache Spark does not have the streaming support for regression tree algorithm streaming model update functionality of our system cannot be provided with this algorithm. Model created with linear regression algorithm has the second

lowest mean squared and root mean squared error values and since Apache Spark has the streaming support for this algorithm, linear regression is the best machine learning algorithm for our system in the selected ones.

Table 12: Results of Performance of Machine Learning Algorithm Tests

Machine Learning Algorithm	Linear Regression	Logistic Regression	Regression Tree
Model Creation Duration (seconds)	50	47	43

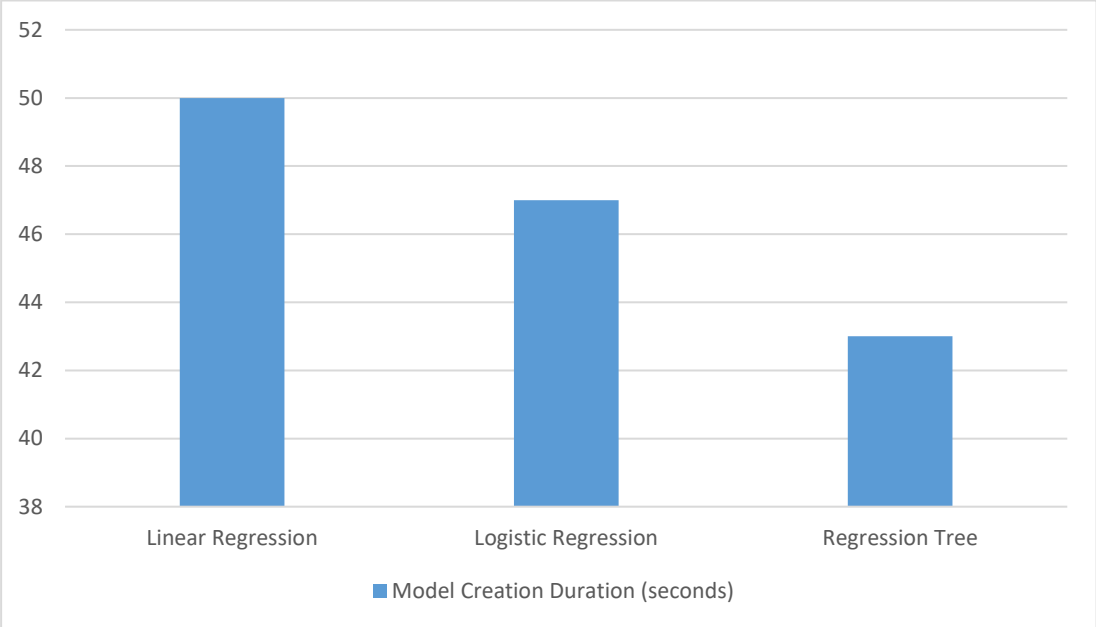

Figure 24: Results of Performance of Machine Learning Algorithm Tests

It can be seen in the results in Figure 24 the model creation performance of the regression tree is the best one amongst the examined algorithms, meanwhile since model creation is executed for all tenants only once this case can be ignored.

6.4 Overall Comments

After performing the tests, the recommended variations and parameters of the basic architecture are explained in this chapter.

Pool based variation is the preferred variation in almost all of the test cases over tenant based variation because of handling the multi-tenancy successfully with

the low resource usage. The only case tenant based variation is preferable is model updating functionality with a system with small number of tenants, because in this case tenant based variation handles parallelization better than pool based variation. Results show that pool-based variation has a scalable infrastructure thereby has the ability to handle multi-tenancy when resources are extended.

Partition count is a key feature for handling the parallelization in Apache Spark and for improving the performance of the system. Increasing the partition count gives the system the ability to handling the functionalities in parallel, but using too many partitions bring excessive overhead to system since it will be managing many small tasks.

Creating the machine learning models with data with more frequent measurements gives the system a better accuracy. The disadvantage of creating models with data with more frequent measurements is the creation taking more time, but since the models are created once it is not a valid concern for the proposed system.

Updating the models more frequently gives the system a slightly better accuracy also, on the other hand updating the models more often increases the load of the system heavily while tenant count is increasing. There is a payoff in the proposed system between the tenant count and the model update frequency.

Linear regression algorithm is the best option from the tested algorithms for the proposed system, because linear regression has the second-best accuracy, slightly worse than decision tree, and also it is supported by Spark MLlib in both batch and streaming modes.

Using Spark MLlib as the machine learning tool of the proposed system is the better fit. SparkR does not fit to our solution since streaming support is not provided in the meantime. Implementing batch analyses using SparkR while implementing streaming analyses not using SparkR means implementing same algorithms twice in different languages. It is not a reasonable solution since it increases the workload and also correlating input/output flow in different environments is not an easy task. Using R scripts from Apache Spark with pipe functionality solves the issue with streaming implementation problem, however this approach does not meet to the scalability requirement of the proposed system. Besides data exchange between the

external library and Apache Spark is handled by stdin and stdoud which makes it open to faults and hard to debug.

CHAPTER 7

CONCLUSION

In this thesis, we have analysed needs of a big data analytics architecture for large-scale multi-tenant energy optimization systems. Proposed system is capable of doing several near-real time analyses on streaming sensor data with the help of machine learning models created from old sensor data with batch analyses. Proposed system handles multi-tenancy in a scalable manner. After describing needs, a basic software architecture with different variations and parameter changes is recommended in order to fulfil the requirements of the system. Then, the basic architecture and different variations of this architecture is tested in a cluster environment. Then, test results are shared and comments on the results are presented. Pros and cons of all variations and parameter changes are explained. Finally, an architecture with best performing variations and parameters is proposed depending on the test results and previous architecture proposals.

According to results, there are several conclusions that can be made:

- A tenant-based architecture for handling multi tenancy has high performance with small tenant sizes. When the tenant size increases tenant-based architecture needs to use unreasonable amount of resources to scale up, so using a pool-based architecture has the advantage when tenant size increases. Results show that pool-based variation has a scalable infrastructure thereby has the ability to handle multi-tenancy when resources are extended.
- Optimal partition count is changing with tenant size and data arrival frequency, but both too few and too many partition counts have their disadvantages respectively. Using too few partitions prevents the benefits of parallelization in Apache Spark, while using too many partitions causes excessive overhead to system since it will be managing many small tasks.

- Creating machine learning models with data with more frequent measurements gives the system a better accuracy. The measurement frequency of data used in model creation affects the consumption prediction and model update frequency of the system. Consumption prediction frequency is reasonable to be the same as the measurement frequency of data used in model creation since consumption prediction performance of the system is sufficient. Model update frequency does not have to be the same as the other two since model update performance of the system is a little troubled with increasing tenant counts. Besides increasing the model update frequency gives the system a slight better accuracy.
- Using linear regression as the selected machine learning algorithm is the best option for the proposed system. Since it is more accurate than most of other algorithms for our case and since it has both batch and streaming implementation in Spark MLlib.

To sum it up, proposed system is capable of doing essential analyses of an energy optimization system like baseline energy forecast and HVAC set point regression more frequently than hourly intervals. System supports multi tenancy in a scalable manner. System meets the timing requirements of a near real time system and also system provides a reasonable accuracy rate.

There are some parts in the need of improving and some future tests to be performed on this thesis. A major deficiency is the handling mechanism of multi-tenant model update feature. Spark handles parallelization by using the inside functions of RDD and/or Data Frame concepts, but since these functions do not support fitting models, model update feature of each tenant in a multi-tenant architecture have to be done simultaneously. Finding a convenient way of overcoming this deficiency is the first task for the future work. In this thesis, basic energy forecast and HVAC set-point regression analyses are implemented. Another improvement for this study could be implementing more analyses to be used in a big data analysis subsystem of an energy optimization system. More analyses can be implemented using the footprints of already implemented analyses for the proposed energy optimization system. Examining the effect of feature selection and sampling of the data are also essential works to be implemented since both of them have the

potential of improving the performance of the system. Performance of the model update feature which is the most critical issue of the proposed system can be improved according to results of this examinations. Feature selection also has the potential of improving the accuracy of the system.

REFERENCES

[1] Gartner Says 6.4 Billion Connected. (n.d.). Retrieved July 13, 2017, from http://www.gartner.com/newsroom/id/3165317

[2] Integrators can Capitalize on Smart Buildings. (n.d.). Retrieved from http://us.allegion.com/en/home/newsroom/2017/integrators-capitalize-on-smart-buildings.html

[3] Laouafi, A., Mordjaoui, M., & Dib, D. (2014, December). Very short-term electricity demand forecasting using adaptive exponential smoothing methods. In Sciences and Techniques of Automatic Control and Computer Engineering (STA), 2014 15th International Conference on (pp. 553-557). IEEE.

[4] Qdr, Q. (2006). Benefits of demand response in electricity markets and recommendations for achieving them. US Dept. Energy, Washington, DC, USA, Tech. Rep.

[5] Spiller, B. (2015, December 21). All Electricity is Not Priced Equally: Time-Variant Pricing 101. Retrieved August 22, 2017, from http://blogs.edf.org/energyexchange/2015/01/27/all-electricity-is-not-priced-equally-time-variant-pricing-101/

[6] Surdak, C. (2014). Data Crush: How the Information Tidal Wave is Driving New Business Opportunities. AMACOM Div American Mgmt Assn.

[7] Zheng, Z., Wang, P., Liu, J., & Sun, S. (2015). Real-time big data processing framework: challenges and solutions. Applied Mathematics & Information Sciences, 9(6), 3169.

[8] S. Singh and N. Singh. Big Data analytics. In 2012 International Conference on Communication, Information & Computing Technology (ICCICT), pages 1–4, 2012.

[9] LaValle, S., Lesser, E., Shockley, R., Hopkins, M. S., & Kruschwitz, N. (2011). Big data, analytics and the path from insights to value. MIT sloan management review, 52(2), 21.

[10] Mayer-Schnberger, V. (2013). Big Data: A Revolution That Will Transform How We Live, Work and Think, ed. Viktor Mayer-Schonberger and Kenneth Cukier.

[11] Akerkar, R., & Sajja, P. S. (2016). Analytics and Big Data. In Intelligent Techniques for Data Science (pp. 211-236). Springer International Publishing.

[12] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

[13] Kumar, V., Andrade, H., Gedik, B., & Wu, K. L. (2010, March). DEDUCE: at the intersection of MapReduce and stream processing. In Proceedings of the 13th International Conference on Extending Database Technology (pp. 657-662). ACM.

[14] Solaimani, M., Iftekhar, M., Khan, L., Thuraisingham, B., Ingram, J., & Seker, S. E. (2016). Online anomaly detection for multi-source VMware using a distributed streaming framework. Software: Practice and Experience, 46(11), 1479-1497.

[15] Oliver, A. C. (2014, December 04). Storm or Spark: Choose your real-time weapon. Retrieved August 22, 2017, from http://www.infoworld.com/article/2854894/application-development/spark-and-storm-for-real-time-computation.html

[16] Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). Learning spark: lightning-fast big data analysis. " O'Reilly Media, Inc.".

[17] Namiot, D. (2015). On big data stream processing. International Journal of Open Information Technologies, 3(8), 48-51.

[18] Balac, N., Sipes, T., Wolter, N., Nunes, K., Sinkovits, B., & Karimabadi, H. (2013, October). Large scale predictive analytics for real-time energy management. In Big Data, 2013 IEEE International Conference on (pp. 657-664). IEEE.

[19] Liu, X., & Nielsen, P. S. (2016). A hybrid ICT-solution for smart meter data analytics. Energy.

[20] Petri, I., Rana, O., Rezgui, Y., Li, H., Beach, T., Zou, M., ... & Parashar, M. (2014, May). Cloud supported building data analytics. In Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on(pp. 641-650). IEEE.

[21] Khamphanchai, W., Saha, A., Rathinavel, K., Kuzlu, M., Pipattanasomporn, M., Rahman, S., ... & Haack, J. (2014, October). Conceptual architecture of building energy management open source software (BEMOSS). In IEEE PES Innovative Smart Grid Technologies, Europe (pp. 1-6). IEEE.

[22] Jirkovský, V., Obitko, M., Novák, P., & Kadera, P. (2014, September). Big Data analysis for sensor time-series in automation. In Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA) (pp. 1-8). IEEE.

[23] Stampfli, J., & Stockinger, K. (2016). Applied Data Science: Using Machine Learning for Alarm Verification. ERCIM NEWS, (107), 35-36.

[24] Ruz, M. L., Fragoso, S., Rodríguez, D., & Vázquez, F. (2015, June). Real-time estimation of thermal comfort indices in an office building with a solar powered HVAC system. In Control and Automation (MED), 2015 23th Mediterranean Conference on (pp. 803-808). IEEE.

[25] Katchasuwanmanee, K., Bateman, R., & Cheng, K. (2016). Development of the Energy-smart Production Management system (e-ProMan): A Big Data driven approach, analysis and optimisation. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 230(5), 972-978.

[26] Chen, L., & Kang, K. D. (2015, August). A Framework for Real-Time Information Derivation from Big Sensor Data. In High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on (pp. 1020-1026). IEEE.

[27] Mamidi, S., Chang, Y. H., & Maheswaran, R. (2012, June). Improving building energy efficiency with a network of sensing, learning and prediction agents. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1 (pp. 45-52). International Foundation for Autonomous Agents and Multiagent Systems.

[28] Mayer, C., Mayer, R., & Abdo, M. (2017, June). Stream Learner: Distributed Incremental Machine Learning on Event Streams: Grand Challenge. In Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems (pp. 298-303). ACM.

[29] Grolinger, K., L'Heureux, A., Capretz, M. A., & Seewald, L. (2016). Energy forecasting for event venues: Big data and prediction accuracy. Energy and Buildings, 112, 222-233.

[30] Solaimani, M., Iftekhar, M., Khan, L., Thuraisingham, B., Ingram, J., & Seker, S. E. (2016). Online anomaly detection for multi-source VMware using a distributed streaming framework. Software: Practice and Experience.

[31] Nair, Y. C., Neethu, P. V., Menon, V. K., & Soman, K. P. (2016). Real Time Vehicular Data Analytics Utilising Big data Platforms and Cost Effective ECU Networks. Indian Journal of Science and Technology, 9(30).

[32] Ko, D., Kwak, Y., Choi, D., & Song, S. (2015). Design of Smart Cold Chain Application Framework Based on Hadoop and Spark. International Journal of Software Engineering and Its Applications, 9(12), 99-106.

[33] Domann, J., Meiners, J., Helmers, L., & Lommatzsch, A. (2016). Real-time News Recommendations using Apache Spark. In CLEF (Working Notes) (pp. 628-641).

[34] Nair, L. R., Shetty, S. D., & Shetty, S. D. (2017). Applying spark based machine learning model on streaming big data for health status prediction. Computers & Electrical Engineering.

[35] Nizamic, F., Nguyen, T. A., Lazovik, A., & Aiello, M. (2014, August). GreenMind-An Architecture and Realization for Energy Smart Buildings. In ICT4S.

[36] Spark-on-HBase: DataFrame based HBase connector. (2017, May 25). Retrieved August 22, 2017, from https://hortonworks.com/blog/spark-hbase-dataframe-based-hbase-connector/

[37] Apache Spark Performance Tuning – Degree of Parallelism | Treselle Systems | Big Data, Technology & Integration, Quality Assurance. (n.d.). Retrieved from http://www.treselle.com/blog/apache-spark-performance-tuning-degree-of-parallelism/