ISSN: 0976-3104
SPECIAL ISSUE: Emerging Technologies in Networking and Security (ETNS)
Mahesh and Manimegalai

ARTICLE OPEN ACCESS

AN EFFICIENT DATA PROCESSING ARCHITECTURE FOR SMART ENVIRONMENTS USING LARGE SCALE MACHINE LEARNING

A Mahesh*, P Manimegalai
Karpagam University, Coimbatore, Tamil Nadu. INDIA

ABSTRACT

The evolution of semiconductor devices used in embedded systems are requires an Internet connection for data processing and analytics in an automation system. The increasing number of everyday embedded devices that are interconnected over the Internet leads to the need of new software solutions for managing them in a proficient, scalable, and elegant way. In addition, these devices may produce a large volume of data, this can lead to classic Big Data problem that need to be stored and processed. Huge volume of information are being produced by the Social networking users and this data can be high in velocity and the traditional database fails to store the data which is generated, hence this raises two main issues, storing the information generated by the users/IoTs and processing this big data for analytics. In this project we address the above issues by implementing an efficient, fault tolerant and data immutability architecture for big data processing in addition to that applied large scale machine learning techniques such as pattern recognition for the Intelligent Reports

Published on: 2nd -December-2016

KEY WORDS

IoT, Big Data, Machine Learning Algorithms, Decision Making

***Corresponding author: Email:** legacymahesh@gmail.com, manimegalai.vairavan@gmail.com

INTRODUCTION

The Internet of Things (IoT) is the network of physical objects or "things" embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data. Processing the wide variety of data which the sensors may produce during the data acquisition on the smart environments in high velocity, these data provides the classic Big Data problem, Big Data is a technology where the processing or querying can't be done with traditional Relational Database Management System (RDBMS), the characteristics of the Big Data are: Volume, Velocity and Variety. In which Volume describes the size of the data sets which are produced in an embedded systems usually these size of the datasets varies for few millions of Bytes to PB (Peta Bytes). Velocity describes the speed in which the data arrives to the processing environment this varies from 10 messages per second to 1000 messages per second. Variety talks about the different format of data such as structured, semi-structured and un-structured data.

Acquiring the data from the sensors are collected through an API (Application Programming Interface), in the present decade, however the data from an automation system is in high velocity, however the smart device or an electronic sensor were connected through Wifi - WiMax, Zig-bee and many, such technology usually provided an associated IP address in IPV6. As we know that the address space of the IPV6 is 128 bits long, so we can add the device to IPV6 network by providing the address to that device being connected to the internet this is what we called Internet of things (IoT). To process these data to fast-decision making we need an architecture to process the same, Lambda Architecture [1], is a solution for the processing the big data for pattern extraction. In this proposed work, we implement the smart-home as a case study with appropriate machine learning algorithm is mapped from Spark MLlib for decision making or prediction or classification in the smart-home [2]. For example, to close the windows curtains automatically, based on the intensity of the light. To sense the intensity of the light is done using an LDR (Light Dependent Resistor). The value of the LDR is applied to the System that is built using Raspberry Pi. Raspberry Pi has an associated motor controller to open the curtain if the intensity inside a Smart Home is lesser than a threshold value.

The main contribution in this paper is, to develop an architecture/framework that suitably parallel processing of sensor data for quick decision making.

The remainder of this paper is organized as follows, Section II describes the Background of related technologies used in this paper. Section 3 describes the architecture for parallel processing of the sensor data. Section 4 describes the implementation details of our work. Section 5 describes the related work and finally Section 6 concludes.

BACKGROUND

RASPBERRY PI 2

The Raspberry Pi is a series of credit card-sized single-board computers developed in the United Kingdom by the Raspberry Pi Foundation with the intent to promote the teaching of basic computer science in schools and developing countries. The original Raspberry Pi and Raspberry Pi 2 are manufactured in several board configurations through licensed manufacturing agreements with Newark element, RS.

Components and Egoman. The hardware is the same across all manufacturers. The firmware is closed-source. Several generations of Raspberry Pi's have been released. The first generation (Pi 1) was released in February 2012 in basic model A and a higher specification model B. A+ and B+ models were released a year later. Raspberry Pi 2 model B was released in February 2015 and Raspberry Pi 3 model B in February 2016. These boards are priced between 20 and 35 US$. A cut down "compute" model was released in April 2014 and a Pi Zero with smaller footprint and limited IO (GPIO) capabilities released in November 2015 for 5 US$.

All models feature a Broadcom system on a chip (SOC), which includes an ARM compatible CPU and an on chip graphics processing unit GPU (a VideoCore IV). CPU speed ranges from 700 MHz to 1.2 GHz for the Pi 3 and on board memory range from 256 MB to 1 GB RAM. Secure Digital SD cards are used to store the operating system and program memory in either the SDHC or MicroSDHC sizes. Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phono jack for audio. Lower level output is provided by a number of GPIO pins which support common protocols like I2C. Some models have an 8P8C Ethernet port and the Pi 3 has on board WiFi 802.11n and Bluetooth.

[Figure- 1] The Foundation provides Debian and Arch Linux ARM distributions for download, and promotes Python as the main programming language, with support for BBC BASIC (via the RISC OS image or the Brandy Basic clone for Linux), C, C++, PHP, Java, Perl, Ruby, Squeak Smalltalk and more also available.
[Figure- 1] shows the sample Rasperry Pi Board and layout[1]

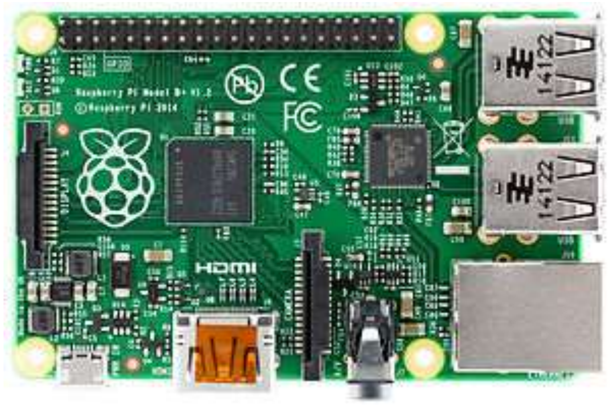


Fig: 1. Sample Raspberry Pi 2 Board

……………………………………………………………………………………………………..

SMART HOME

Home automation is the residential extension of building automation and involves the control and automation of lighting, heating, ventilation, air conditioning (HVAC), appliances, and security. Modern systems generally consist of switches and sensors connected to a central hub sometimes called a "gateway" from which the system is controlled with a user interface that is interacted either with a wall mounted terminal, mobile phone software, tablet computer or a web interface.

While there are many competing vendors, there are very few world-wide accepted industry standards and the smart home space is heavily fragmented. Popular suites of products include X10, Ethernet, RS-485, ZigBee and Z-Wave, or other proprietary

[1] https://en.wikipedia.org/wiki/Raspberry_Pi

protocols all of which are incompatible with each other. Manufacturers often prevent independent implementations by withholding documentation and by suing people.

SPARK MLLIB

MLlib is Spark's machine learning (ML) library [3]. Its goal is to make practical machine learning scalable and easy. It consists of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as lower-level optimization primitives and higher-level pipeline APIs.

- Summary statistics, correlations, stratified sampling, hypothesis testing, random data generation
- Classification and regression: state vector machines, logistic regression, linear regression, decision trees, naive Bayes classification
- Collaborative filtering techniques including alternating least squares (ALS)
- Cluster analysis methods including k-means, and Latent Dirichlet Allocation (LDA)
- Dimensionality reduction techniques such as singular value decomposition (SVD), and principal component analysis (PCA)
- Feature extraction and transformation functions
- Optimization primitives such as stochastic gradient descent, limited-memory BFGS (L-BFGS). It divides into two packages:
    - spark.mllib contains the original API built on top of RDDs.
    - spark.ml provides higher-level API built on top of DataFrames for constructing ML pipelines.

Using spark.ml is recommended because with DataFrames the API is more versatile and flexible. But we will keep supporting spark.mllib along with the development of spark.ml. Users should be comfortable using spark.mllib features and expect more features coming. Developers should contribute new algorithms to spark.ml if they fit the ML pipeline concept well, e.g., feature extractors and transformers.

The main difference between Hadoop MR and Spark MLlib will came from underlying frameworks. In case of Mahout it is Hadoop MapReduce and in case of MLib it is Spark. To be more specific - from the difference in per job overhead

If Your ML algorithm mapped to the single MR job - main difference will be only startup overhead, which is dozens of seconds for Hadoop MR, and let say 1 second for Spark. So in case of model training it is not that important. Things will be different if the algorithm is mapped to many jobs. In this case we will have the same difference on overhead per iteration and it can be game changer.

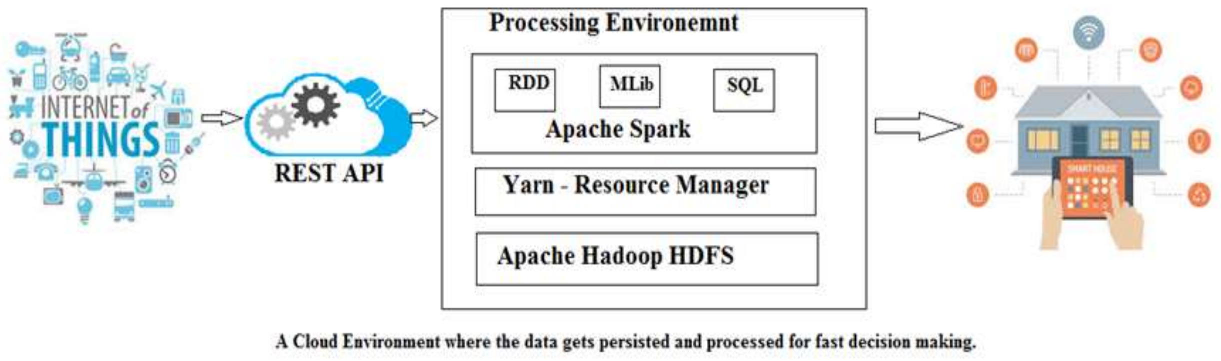


Fig: 2. An Architecture for Smart Home

………..

SYSTEM ARCHITECTURE

[Figure-2] shows the complete System Architecture for processing the IoT data. This architecture receives the data that are emitted by IoT. A RESTful API is designed with MQTT protocol and send to the processing environment via the API. In the processing environment Apache Spark Streaming receives the data from MQTT Streaming Source, Apache Spark Streaming receives the data emitted by IoT, and MQTT Streaming and creates a mini batch of data is send to Apache Spark Core for Parallel Processing on top of Apache Hadoop Yarn. The data flow diagram of this process is shown in **[Figure- 6]**.

Apache Spark Streaming creates many mini batch of dataset that can be manually configured in the Apache Spark Streaming Context. This allows to process the real-time data seamlessly. For each mini batch of data, Apache Spark Core applies the feature extraction on the mini batch of data, the extracted features are stored in the HDFS in the *libsvm* format. Followed by, Apache Spark MLlib takes the pre-processed data from the HDFS. The required ML algorithm will be applied on the datasets which are complex and in real-time.

The data for the analytics is extracted from HDFS, then feature extraction is applied for the pattern recognition and finally the training model is built based on the extracted features using some Machine Learning tools. There are several technology alternatives available to build a training model on the real-time data for activity detection on the Smart Home application.

IMPLEMENTATION

In this section, the most suitable analytics tool, for analyzing, extracting the patterns and reduce the usage of energy in a Smart Home. In this paper, a rule based activity detection has been explored for the IoT dataset [4] and Random Forest [5] classification technique has been applied using Apache Spark Streaming to evaluate the time taken to build the training model
DataSet

A batch of time-series datasets has been stored in the HDFS. This data is being used as a training data for building the classification model. In this paper, this has been processed by Apache Spark Core. To test a new data against the model to calculate the accuracy of the classification and also to predict the unknown classes for certain data, Apache Spark Streaming is used. The predicted values of the test data are retrieved from the HDFS. Two different smart houses were deployed and data over the months has been collected. The duration of the deployment for "House A" was 82 days with around 22.5 million collected data points. For "House B", [4] had deployed the system for 62 days with about 20 million collected data points.
[Figure-3] shows the sample pre-processed data in *libsvm* format.

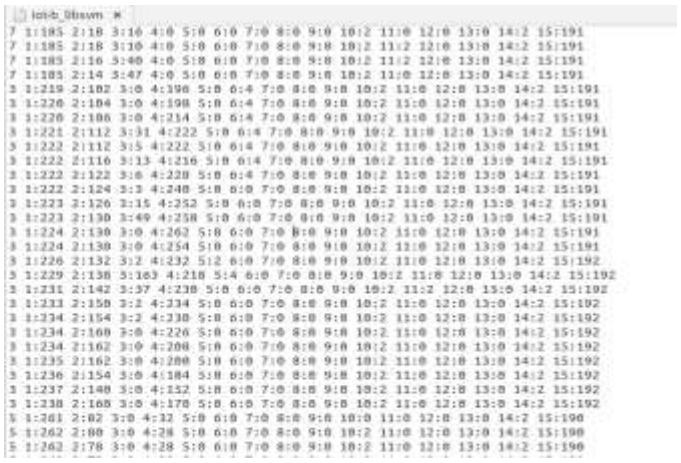

```
iot-b_libsvm
7 1:185 2:18 3:16 4:0 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
7 1:185 2:18 3:10 4:0 5:0 6:0 7:0 8:0 9:0 10:2 11:2 12:0 13:0 14:2 15:191
7 1:185 2:16 3:40 4:0 5:0 6:0 7:0 8:0 9:0 10:2 11:2 12:0 13:0 14:2 15:191
7 1:185 2:14 3:47 4:0 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:219 2:102 3:0 4:196 5:0 6:4 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:220 2:104 3:0 4:198 5:0 6:4 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:220 2:106 3:0 4:214 5:0 6:4 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:221 2:112 3:31 4:222 5:0 6:4 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:222 2:112 3:5 4:222 5:0 6:4 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:222 2:116 3:13 4:216 5:0 6:4 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:222 2:122 3:6 4:220 5:0 6:4 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:222 2:124 3:3 4:240 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:223 2:126 3:15 4:252 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:223 2:130 3:49 4:258 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:224 2:130 3:0 4:262 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:224 2:130 3:0 4:254 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:191
3 1:226 2:132 3:2 4:232 5:2 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:192
3 1:229 2:138 3:163 4:218 5:4 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:192
3 1:231 2:142 3:37 4:238 5:0 6:0 7:0 8:0 9:0 10:2 11:2 12:0 13:0 14:2 15:192
3 1:233 2:150 3:2 4:234 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:192
3 1:234 2:154 3:2 4:230 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:192
3 1:234 2:160 3:0 4:226 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:192
3 1:234 2:162 3:0 4:208 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:192
3 1:235 2:162 3:0 4:200 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:192
3 1:236 2:154 3:0 4:184 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:192
3 1:237 2:148 3:0 4:152 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:192
3 1:238 2:168 3:0 4:170 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:192
5 1:261 2:82 3:0 4:32 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:2 15:190
5 1:262 2:80 3:0 4:28 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:190
5 1:262 2:78 3:0 4:28 5:0 6:0 7:0 8:0 9:0 10:2 11:0 12:0 13:0 14:2 15:190
```

Fig: 3. IoT libsvm Pre-processed data

FEATURE EXTRACTION

The time series data consists of a sequential set of activities accompanied with another sequential set of sensor readings. In order to train the machine learner, it is essential to build a training set which includes training instances that match the current activity with the features extracted from the sensor data. Therefore, the sequential sensor data needs to be divided into time windows so that each window can be provided as an input to the learning algorithm with the activity during this window as an output.

[Figure-4] The time series data is divided into equally sized time slots. For each time slot, the features extracted from sensor data are taken as an input and the current activity as an output. The time slots define the boundaries of the feature extraction algorithm. The algorithm runs for each time slot and extracts the features required for the machine learner. For each sensor, a feature is defined that represents the arithmetic mean of the readings related to this sensor during the time slot as shown in the following equation.

$$F_x = \frac{\sum_{i=1}^{n} r_i}{n}$$

Where F_x is the feature that represents sensor x, n is the number of sensor readings during the time slot, and r_i is the sensor reading i.

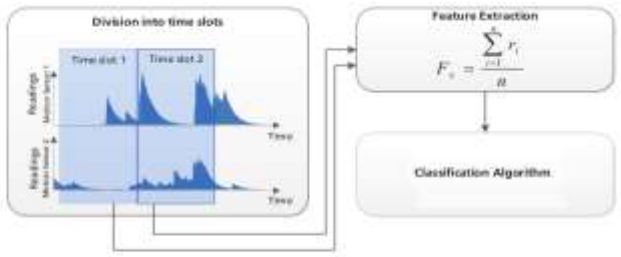


Fig: 4. Feature Extraction Dataset

RESULTS

Spark MLlib supports random forests for binary and multiclass classification and for regression, using both continuous and categorical features. spark.mllib implements random forests using the existing decision tree implementation.

Random Forest Classification [5] on the real-time IoT dataset [4] using Apache Spark MLlib is executed as follows. Feature Extraction Algorithm is applied on the time-series data as mentioned in Section B, and data cleaning is performed to execute Random Forest in Spark MLlib. Spark accepts input training files in *libsvm* format.

[Figure-5] and **[Figure-6]** 90% of the real-time IoT data has been given for training and the remaining 10 % of the real-time IoT data has been given for testing the unknown instances and the accuracy of the model built is evaluated. The depth of the tree would be 9 level, and the number of trees are constructed is varied from 100 to 500.
The performance of the proposed method as follows.

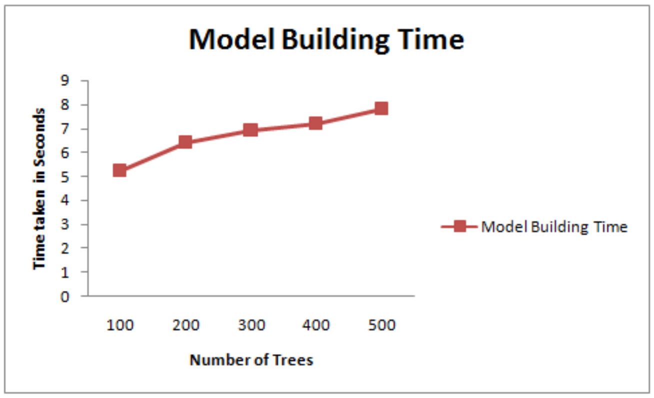


Fig: 5. Time taken for building the Model

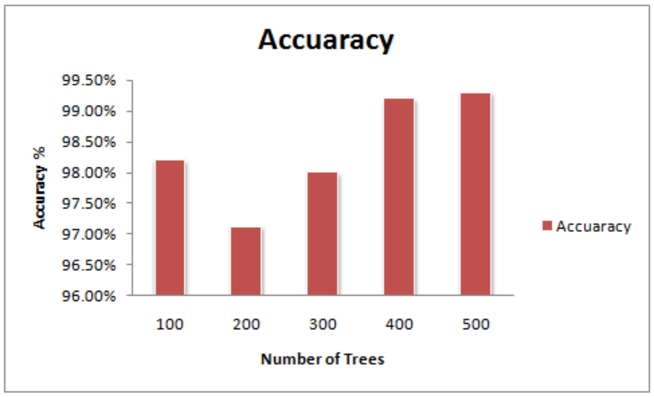


Fig: 6. Accuracy of the Proposed Approach

RELATED WORKS

[Figure-7] AllJoyn, is a system that allows devices to communicate with other devices around them. A simple example would be a motion sensor letting a light bulb know that no one is in the room it is lighting, so it can shut itself off. AllJoyn is an open source software framework that makes it easy for devices and Mobile Applications to discover and communicate with each other. Developers can write applications for interoperability regardless of transport layer, manufacturer, and without the need for Internet access. The software is openly available for developers to download, and runs on popular platforms such as Linux and Linux-based Android, iOS, andWindows, including many other lightweight real-time operating systems.

Real, digital and virtual worlds are converging to create environments that make cities, energy, transport and many other services smart. In this context, the Internet of Things (IoT) is a very challenging paradigm that enables things, i.e., embedded devices, to be ideally available anytime and anywhere to anyone. Integrating heterogeneous systems is the main goal of IoT. In this work, Lambda Architecture is used for processing and recognizing patterns. Batch Layer of LA is implemented using Apache Hadoop, Similarly, Speed Layer of LA is implemented using Apache Storm.

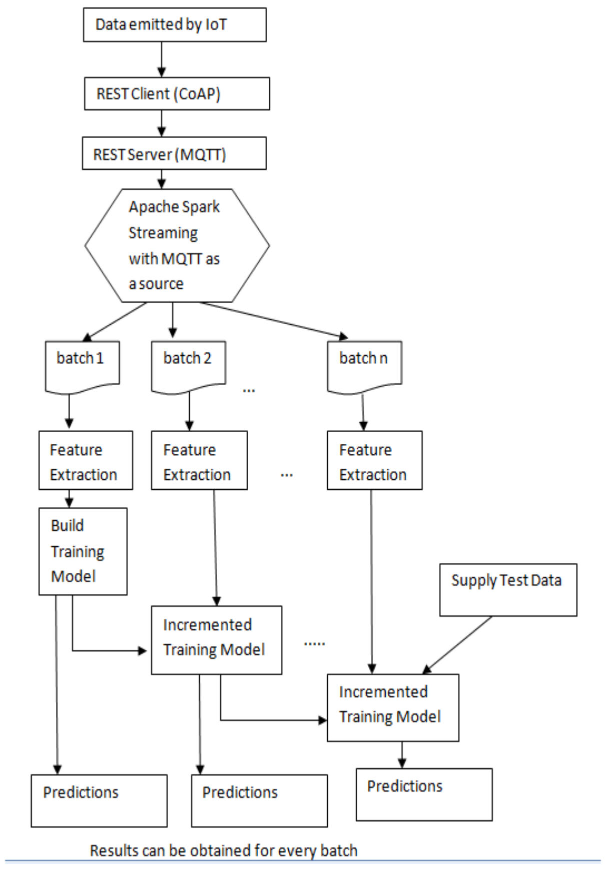


Fig: 7. Data Flow

The idea of energy efficient Smart Homes or Home Automation System has attracted many researchers in academic as well as in industrial domains [6-8]. The main challenge in this domain is to capture and process the data which are emitted by each sensors in a smart home. Energy efficient smart homes are similar applications in which helps to reduce the amount of energy consumed by a particular home. An activity based energy advisor will work in non-intrusive way. Detecting the serious of activities on a particular home will helps to monitor where the energy is getting wasted through a alert message received by the smart phone send by the central server.

[Figure- 8] Alhamoud et al. developed a SMARTENERGY.KOM an activity based intelligent system for energy saving in smart home. The system architecture of SMARTENERGY.KOM is shown.

The major sensors to sense the information in a home such as temperature, light and humidity can be sensed by this system and sent to the central server through a Raspberri Pi. Central server is responsible for collecting these data from the Raspberri Pi which acts as a gateway of the sensors and to process these data to reduce the energy usage in smart home. This work uses Traditional MySQL as a database for storing the sensor data, However, Relational Database will becomes to process if the data grows and also the number of smart homes increases [9-10].

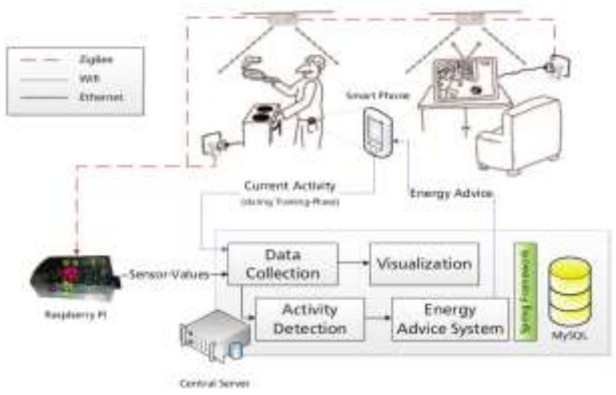
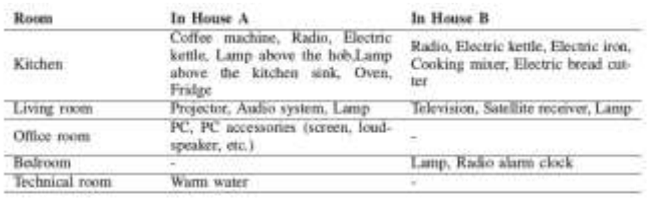


Fig: 8. Smart Home Application

[Figure- 9] Alhamoud et al. have used a standard Random Forest classification for activity detection in the smart home.This algorithm have been implemented in a WEKA. However, This algorithm can be distribute it multiple servers for quick processing and decision making. A. Alhamoud, et. al. had deployed 2 smart homes HomeA and HomeB where the researcher Alhamoud was the user of HomeA and for the HomeB a non-technical user activities been recorded. The sensors connected in both the homes are as follows.



Room	In House A	In House B
Kitchen	Coffee machine, Radio, Electric kettle, Lamp above the hob,Lamp above the kitchen sink, Oven, Fridge	Radio, Electric kettle, Electric iron, Cooking mixer, Electric bread cutter
Living room	Projector, Audio system, Lamp	Television, Satellite receiver, Lamp
Office room	PC, PC accessories (screen, loudspeaker, etc.)	-
Bedroom	-	Lamp, Radio alarm clock
Technical room	Warm water	-

Fig: 9. Appliances at Each House

CONCLUSION

This paper has presented a novel architecture for efficiently process the large scale data using machine learning algorithms. MQTT streaming has been implemented in Apache Spark Streaming for collecting the real-time IoT data. The various mini batch of datasets are processed in parallel with the cluster. A cluster consisting of 3 machines had been setup. The performances of the proposed approach has been analyzed for incrementally increasing the number of trees constructed in the Random Forest for activity classification. The proposed approach proves that it provides better performance as compared to the existing implementation.

As a future of this work, extending the data processing and analyzing using Apache Spark H20 (A Deep Learning tool). Deep Learning enables the training model and feature extraction can be selected from the large scale data and constructs the model as its own.

CONFLICT OF INTEREST
The authors declare no conflict of interests.

ACKNOWLEDGEMENT
None

FINANCIAL DISCLOSURE
The authors report no financial interests or potential conflicts of interest.

REFERENCES

ISSN: 0976-3104

[1] Villari M, Celesti A, Fazio M, Puliafito A. [2014]. Alljoyn lambda: An architecture for the management of smart environments in iot. In*Smart Computing Workshops (SMARTCOMP Workshops), 2014 International Conference on* (pp. 9-14). IEEE.

[2] Dixit A, Naik A. [2014] Use of Prediction Algorithms in Smart Homes.*International Journal of Machine Learning and Computing*, *4*(2): 157.

[3] Kabir, M. H, Hoque, M. R, Seo, H, & Yang, S. H. [2015] Machine Learning Based Adaptive Context-Aware System for Smart Home Environment.*International Journal of Smart Home 9*(11):55-62.

[4] Alhamoud A, Ruettiger F, Reinhardt A, Englert F, Burgstahler D, Böhnstedt D, Steinmetz R. [2014]. Smartenergy. kom: An intelligent system for energy saving in smart home. In *Local Computer Networks Workshops (LCN Workshops) 2014 IEEE 39th Conference on* (pp. 685-692). *IEEE*.

[5] https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

[6] Choi S, Kim E, Oh S. [2013]. Human behavior prediction for smart homes using deep learning. In *2013 IEEE RO-MAN* (pp. 173-179). IEEE.

[7] Barker S, Mishra A, Irwin D, Cecchet, E, Shenoy, P, & Albrecht, J. [2012]. Smart*: An open data set and tools for enabling research in sustainable homes. *SustKDD, August*, 111-112.

[8] https://www.raspberrypi.org/documentation/

[9] http://spark.apache.org/docs/latest/mllib-guide.html

[10] https://www.smarthome.com/