Received February 28, 2019, accepted March 21, 2019, date of publication March 27, 2019, date of current version April 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2907808

Distributed Decomposed Data Analytics in Fog Enabled IoT Deployments

MOHIT TANEJA, (Student Member, IEEE), NIKITA JALODIA, (Student Member, IEEE), AND ALAN DAVY, (Member, IEEE)
Emerging Networks Laboratory, Telecommunications Software and Systems Group, Department of Computing and Mathematics, School of Science and Computing, Waterford Institute of Technology, Waterford X91N2FP, Ireland
CONNECT—Centre for Future Networks and Communications, Dublin 2, Ireland

Corresponding author: Mohit Taneja (mtaneja@tssg.org)

This work has emanated from research conducted with the financial support of the Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant 13/RC/2077, and in part by the CISCO Research Gift Fund.

ABSTRACT The edge of the network plays a vital role in an Internet of Things (IoT) system, serving as an optimal site to perform an operation on data before transmitting it over the network. We present the fog-specific decomposition of multivariate linear regression as the predictive analytic model in our work using the statistical query model and summation form. The decomposition method used is not the contribution, but applying the decomposition method to the analytics model to run in a distributed manner in the fog-enabled IoT deployments is the contribution. What is novel is the decomposition made on a fog-based distributed setting. To test the performance, our proposed approach has been applied to a real-world dataset and evaluated using a fog computing testbed. The proposed method avoids sending raw data to the cloud and offers balanced computation in the infrastructure. The results show an 80% reduction in the amount of data transferred to the cloud using the proposed fog-based distributed data analytics approach compared with the conventional cloud-based approach. Furthermore, by adopting the proposed distributed approach, we observed a 98% drop in the time taken to arrive at the final result compared with the cloud-centric approach. We also present the results on the quality of analytics solution obtained in both approaches, and they suggest that the fog-based distributed analytics approach can serve as equally as the traditional cloud-centric approach.

INDEX TERMS Fog computing, cloud computing, Internet of Things (IoT), data analytics, decomposition, distributed.

I. INTRODUCTION

With the exponential growth rate of technology, the future of all activities involves an omnipresence of widely connected devices, or as we better know it, the Internet of Things (IoT). IoT is set to reform the future of connectivity and reachability. With the emerging IoT adoption, it is estimated [1] that there will be 1 trillion interconnected IoT devices by 2025. A recent major survey [2] revealed that IoT will be the center of the upcoming technologies, and would drive significant impacts in machine-aided commerce applications in the next five years. Another recent publication [3] by Cisco in June 2017 indicates that we have already reached the Zettabyte Era, and the number of devices connected to the Internet is growing exponentially.

A thing,[1] in the IoT, can be a physical device such as temperature sensor, a person with body wearable or implanted device such as heart monitor implant, a farm animal with a wearable device, an automobile that has built-in sensors or any other man-made or natural object that can be assigned a unique identifier such as an IP address and provided with the ability to transfer data over a network. The simple idea is that anything that can be connected will be connected.

IoT aims to bring every object online, hence generating massive amounts of data that can overwhelm the cloud centric application systems. Fog computing is a relatively new networking paradigm that aims to provide traditionally

The associate editor coordinating the review of this manuscript and approving it for publication was Jad Nasreddine.

[1]It is noteworthy that there are other terms such as ''objects'', ''Inter Connected Objects (ICOs)'' [4] that have the same meaning as things here and are frequently used in IoT and fog related documentation. Some other terms used by the research community are ''smart objects'', ''devices'', and ''nodes'' [5]

centralized data-center operations at the edge of the network. It has recently emerged as a potential architecture for scaling IoT network applications. Several interpretations [6] have been proposed for the implementation of fog nodes and their configuration—via servers, networking devices, cloudlets, base stations, or vehicles. Fog computing utilizes the available in-network computing resources and shows the capability of reducing the dependency on the cloud by facilitating data analytics on the network edge [7]. This improves the responsiveness of the system, reduces resource requirements on the remote cloud infrastructure, and in turn increases the efficiency of the system in terms of energy consumption and network usage [8].

The capabilities of the computing gateways [9] comes as an important aspect of fog computing architecture for IoT based applications. With the increasing demand of providing smart solutions, the shift towards utilization of gateway devices as fog nodes for edge analytics is being supported by an increasing number of industrial [10]–[12] IoT platform developers and solution providers, including IBM, Intel, and Microsoft.

The increasing range of real-world IoT deployments essentially increases the sources of data generation, thereby globally strengthening the challenges already being faced in the Big Data space [13], particularly regarding moving data from one end (i.e. from data sources such as sensor/IoT devices at the edge level of infrastructure) to the other extreme end (i.e. centralized data centers at the cloud) in the network infrastructure. Therefore, building architectures that allow executing services in multiple points have recently gained attention from industry players [14], [15].

Sending the entire data set across the extreme ends in the infrastructure becomes an unrealistic solution, specifically in scenarios with constrained network bandwidth and scarce internet connectivity. Instead, approaches that collect data and perform computational processing near the data source itself present a more practical alternative in such scenarios. This generates a vast array of benefits across use cases such as those with video oriented applications, where the transport of video across infrastructure can claim considerable network resources such as requirement for storage at each node from source to destination. While IoT deployments vary across use cases, the most prominently common underlying aim is to analyze the data generated from the devices to achieve a specific set objective. Analyzing data at early stages of infrastructure pipeline presents additional benefits of data and communication security in the overall system, owing to the fact that raw data is now processed closer to the data source, and only processed data is sent further. The efficient use of available computational resources realized by means of fog computing, in turn, promotes the idea of green computing [16], [17].

Contrary to the cloud which can be thought of as resource rich, the fog devices are resource constrained in nature whereby resource scaling (up/down and horizontal/vertical) cannot be done dynamically. The fog devices are already performing their fundamental computing/network operation (for e.g. in case of gateway acting as a fog device, it is already forwarding the packets to the set destination), so these operations are already utilizing the available resources (CPU, RAM and bandwidth) on it. An additional deployment of a complete data analytics computing module on the said resource might lead to full utilization of resources on device as the workload or data input increases, and may also affect its fundamental network operation. Hence, a careful placement of computing operations is sought for efficient overall system performance, and thus, the approach of decomposed computing units seems ideal in an IoT environment with fog assistance.

For e.g. consider a case where the additional deployment of physical devices is not possible and we are required to predict temperature based on given readings of limited environmental parameters such as humidity and air pressure for a large physical area. In such scenario sending all the raw sensing data to cloud for analyzing and sending the result back to user is neither feasible nor scalable [18]. This becomes even more challenging with limited Internet connectivity and growing volume of data. In such scenarios it's ideal to leverage fog computing architecture where fog focuses on local knowledge in different local areas, while cloud can have a global view of the environment by combining all the information received from fog. These kinds of cases exists in industrial systems such as monitoring complex hydraulic systems [19] to be working in order, and to its desired capacity. Another related example can be to know the temperature of different areas of a large stadium during an event to get actionable insights for regulating air conditioning of those specific areas.

Our contribution is summarized as follows:

- We present the fog based distributed data analytics solution for IoT deployments.
- The decomposition method used for distributing the analytics/intelligence in the infrastructure is based on Statistical Query Model and Summation Form, which makes it closed form in nature.
- To the best of our knowledge this is first attempt to decompose an analytics model (in closed form) to make it run in distributed manner in a fog based setting.
- The solution and methodology is generic is nature and is applicable to a wide variety of IoT based use case scenarios.
- The proposed approach has been applied to a real-world data set in a fog based testbed, and metrics related to resource utilization and quality of analytics solution have been presented.

The paper has been further structured as follows: §II presents the literature review, background, related work and motivation, §III outlines the mathematical model of the system and the problem, §IV presents the analytics model (Multivariate Linear Regression) used in the work, §V presents the decomposition methodology, §VI presents the experimental setup and data sets used, §VII presents results and discussion, and finally §VIII presents the conclusion and future work.

II. LITERATURE REVIEW: BACKGROUND, RELATED WORK AND MOTIVATION

IoT can be seen as a network of interconnected physical or virtual 'things' that are capable of using intelligent interfaces to be integrated as an information network in order to communicate with one another, with other devices and with services over the Internet to accomplish some objective [20]. Authors in [21] identified and envisioned the adoption of IoT to affect both domestic and industry fronts. Some examples of domestic impact include application scenarios in assisted living, e-health, etc. On the industrial front, the areas of most visible impact include the likes of automation, industrial manufacturing, logistics, and intelligent transportation.

In traditional centralized approaches [22], [23] for data analytics in IoT, all collected data are transferred to centralized location such as server(s) in data center (i.e. cloud) and is then subjected to the desired data analytics model, thus such approaches suffer from the bottleneck of data transfer. The centralized approach becomes redundant when data itself is being generated and stored in a distributed manner, and the entire data set has to be transported across the infrastructure for analysis. In some use cases such as healthcare, network nodes might not want to share data because of privacy issues. Our work decomposes the desired data analytics model to run on the edge of the network coping with the above mentioned constraints.

There are approaches [24], [25] that have been proposed for Wireless Sensor Networks (WSNs) based on selective forwarding that take into account the constraint of bandwidth, latency and energy. The issue with such approaches is that they only focus on communication efficiency without being aware of analytical task being performed at the destination. Further advanced methods based on selective forwarding [26]–[28] work on dynamic optimal decision making to find best time to deliver data for communication efficiency and to minimize reconstruction error at the destination. However, such methods are limited to communication overhead and have not be developed and applied to the network edge i.e. in a fog based setting.

Authors in [29] present the survey of a subset of methods that can be modified to run in a distributed manner to solve the problem of linear least-squares.

Distributed approaches [30]–[32] specific to regression work on the constraint that gathering data centrally is either expensive or impossible, and focus on distributing estimation of global model parameters over nodes with the aim to achieve the same prediction performance that would have been achieved by the corresponding centralized model. The issue with such approaches is that they need additional techniques for parameters update and synchronization, which restricts their use for a wide set of IoT based applications.

Recent work [32], [33] in edge-analytics exploits the computational power of devices such as Raspberry Pi and BeagleBone to design and launch lightweight algorithms directly at the data sources. Authors in [33] presents edge stochastic gradient descent (EdgeSGD) algorithm for solving linear regression problem with the objective of estimating the feature vector on the edge node.

They show that such approaches can converge faster to the optimal values as compared to the centralized approach. Their approach is different from ours as their approach is iterative in nature and needs to converge to find the solution, while we present the closed form solution of the problem that fits to the parameter without the need to use an iterative algorithm. We elaborate more on this in the next sections.

Authors in [34] and [35] present methods of data suppression based on local forecasting models on sensors in view of re-constructing data at the sink node. These methods exclusively focus on reducing data communication by means of data suppression using forecasting models.

Data in IoT deployments moves from things to cloud, and along this continuum passes through a number of network devices such as routers, gateways, etc. Each of these devices can be a potential candidate to host partial computing analytics capability to analyze the data, and further sending the calculated partial results instead of sending the raw data to cloud [36].

Initial exploratory work by authors in [37] shows that such decompositions can reduce bandwidth consumption and can significantly decrease the associated costs. But further research and developments on pointers like decomposition methods, system performance and quality of analytics need to be carefully studied to design efficient distributed algorithm solutions for fog enabled IoT settings, and that is where we position our work.

Our approach, as devised from [38] and explained further in the next sections does not modify the algorithm in use, rather remains an exact implementation of it albeit capable of running in a distributed manner in fog enabled IoT deployments.

III. MATHEMATICAL MODEL OF THE SYSTEM AND PROBLEM FORMULATION

We consider the network architecture with fog nodes forming a layer between IoT devices and the cloud. A graphical representation of the same is shown in Fig. 1. We examine scenarios where local IoT devices and the remote cloud services carry out data sensing, collection and analytics. With fog layer in the middle of IoT devices and cloud, the analytics computation can be distributed among fog nodes and can be collectively solved by either fog nodes alone or in a combined manner by fog nodes and cloud.

We consider a tree like network architecture in which an IoT device i is connected with its unique fog node j which is further connected to the cloud. We have sensing devices that are transmitting their data to fog nodes and they are sending their data towards cloud or another central location. The tree topology is a hierarchy of nodes with single root node at the highest level of hierarchy, which is connected to one or many nodes in the level below. The communication, computing and storage capabilities (we collectively call them as CCS capabilities) in node(s) increases as one moves from

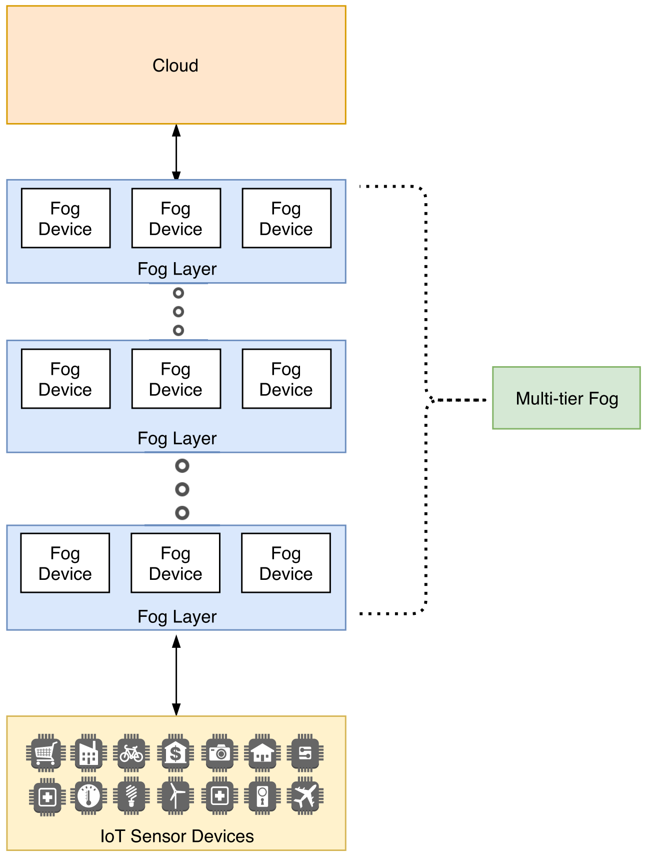


FIGURE 1. **Three tier IoT-fog-cloud architecture (with multi-tier fog).**

branches of the tree towards the root node. In this tree-like topology, the root represents cloud, intermediate nodes represent fog nodes and the leaf nodes represent IoT devices. Data in IoT based deployments moves from ‘ things ’ to cloud or in terms of tree representation from branches of the tree to cloud via fog nodes, allowing data to be processed closer to where its generated. As presented by the authors in [39], the benefits of tree topology include the fact that it’s scalable in nature, and has a simple structure that makes it easier to identify and isolate faults. The graphical representation of an end-to-end tree[2] like network architecture is as shown in Fig. 2.

The neighborhood of fog node j is denoted as $\mathcal{N}_j$ which is the set of IoT devices connected to it and is written as $\mathcal{N}_j = \{n_j\}$ such that $i \in \mathcal{N}_j$. The range of communication of a fog node is defined in terms of its communication distance capability i.e. any IoT device that can communicate with the fog node within its range can become a part of the fog node’s neighborhood.

We consider a discrete time domain $t \in \mathbb{T} = \{1, 2, \ldots\}$ such that an IoT device i at every time instance $t \in \mathbb{T}$ senses a d-dimensional vector $\mathbf{x}_t \in \mathbb{R}^d$ termed as sensed or context vector[3] containing contextual features/parameters for

[2]It should be noted that there can be topologies inside each layer/level but the main structure and abstract reduced topology is tree-like.

[3]Note that any vector by defualt is assumed as column vector, the defualt convention is to write given vector as column vector.

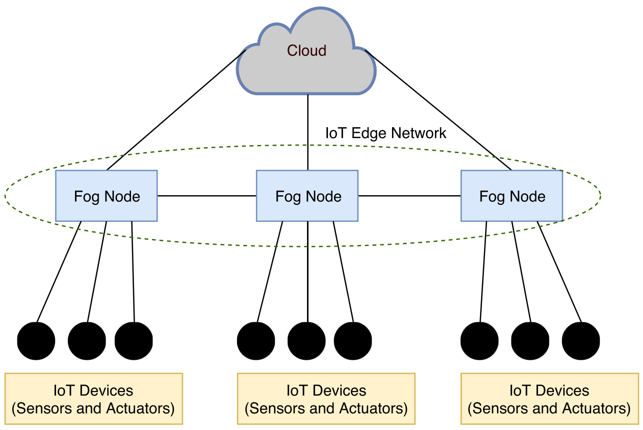

Cloud
IoT Edge Network
Fog Node
Fog Node
Fog Node
IoT Devices (Sensors and Actuators)
IoT Devices (Sensors and Actuators)
IoT Devices (Sensors and Actuators)

FIGURE 2. **Tree topology in IoT deployments.**

e.g. accelaration, g-force values, temperature, humidity, feature counter etc. The IoT device i communicates with its fog node j by sending sensed vectors at a a set frequency f. The transmission frequency f of IoT devices depends on a number of factors such as sampling rate and can range from seconds to minutes to hour(s) depending upon the use case. So at any time instance $t \in \mathbb{T}$ fog node j receives a set of sensed vector from an IoT device and overall sets of sensed vectors from its neighbourhood $\mathcal{N}_j$. A set frequency (or to say time-window or time-frame) is defined on fog node to perform the computing operation over data received till that point.

Depending on specification, data retention capacity and set calibration of IoT devices it can also store the transmitted sensed vectors locally and keep on appending the sensed vectors, and then purge the local storage after the specified period, let’s say for example 12 hours. Usually in sensing devices a sliding window is specified using certain parameters which keeps on appending the new sensed vectors and discarding the older ones based on their appearance without having the specific need to save them in local storage, but it is dependent on the IoT device(s) in use i.e. the sensing infrastructure, and varies from one to another.

IV. ANALYTICS MODEL-MULTIVARIATE LINEAR REGRESSION

One of the most widely used and well-understood predictive analytics model is the multivariate linear regression [40], and thus we choose it as the analytics model in this work for the desired fog specific decomposition. Given a data set $\mathcal{D} = \langle \mathbf{x}_k^{in}, y_k^{out} \rangle_{k=1}^{m}$ with m training examples where $\mathbf{x}_k^{in} \in \mathbb{R}^d$ and $y_k^{out} \in \mathbb{R}$ represents input-output pairs, the linear regression estimates the current coefficient $\mathbf{w} \in \mathbb{R}^d$ which interprets the dependency between $\mathbf{x}_k^{in}$ and y_k^{out}:

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}\in\mathbb{R}^d} \sum_{k=1}^{m} \left(y_k^{out} - \mathbf{w}^{\mathbf{T}}(\mathbf{x}_k^{in})^2 \right) \tag{1}$$

The predicted output by the linear regression model is $\hat{y}_k^{out} = \mathbf{w}^{\mathbf{T}}(\mathbf{x}_k^{in}), k = \{1, 2 \ldots m\}$. The Root Mean Square

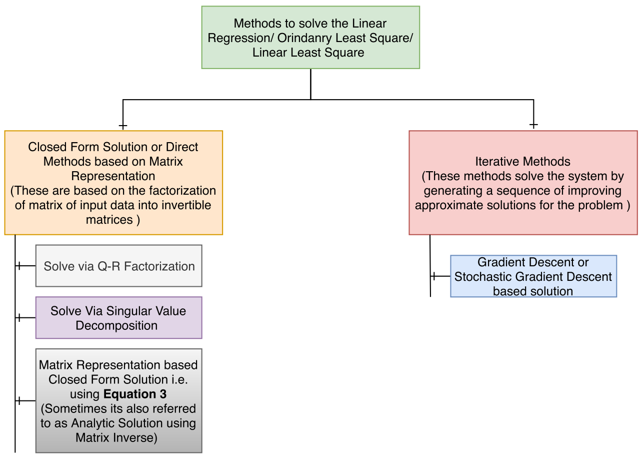

Methods to solve the Linear Regression/ Orindanry Least Square/ Linear Least Square
Closed Form Solution or Direct Methods based on Matrix Representation (These are based on the factorization of matrix of input data into invertible matrices)
Iterative Methods (These methods solve the system by generating a sequence of improving approximate solutions for the problem)
Solve via Q-R Factorization
Solve Via Singular Value Decomposition
Matrix Representation based Closed Form Solution i.e. using Equation 3 (Sometimes its also referred to as Analytic Solution using Matrix Inverse)
Gradient Descent or Stochastic Gradient Descent based solution

FIGURE 3. Brief categorical representation of methods available to solve the problem of linear regression.

Error (RMSE) over q predictions is defined as:

$$\epsilon = \sqrt{\left(\frac{1}{q}\sum_{k=1}^{q}(y_k^{out} - \hat{y}_k^{out})^2\right)} \tag{2}$$

Linear regression is also referred as Ordinary Least Squares (OLS) and Linear Least Squares (LLS). The methods to solve Linear Regression or so as to say the problem of Linear Least Square i.e. equation 1 are typically classified in two categories, Closed Form Solution or Direct Methods and Iterative Methods. The same has been illustrated in Fig. 3.

Each of these methods are equally used in different domains of study depending on number of factors including the type of problem being solved using Linear Regression and the objective at hand such reducing time to get the solution, to get numerical stability [41] in the solution etc. The main difference in the available approaches is that each of them uses a different numerical method to solve the problem. There is extensive research in numeric linear algebra which looks at and aims for parallelizing and to certain extent distributing these numerical operations [41], [42] with each having its own set objective. What is different and novel here is the utilization of fog computing paradigm to achieve that and trying to get an efficient solution for the upcoming IoT and fog based applications and use case scenarios.

Our objective is to have the fog specific decomposition of the linear regression and focus on the arising trade-offs in latency (computing and communication), quality (analytical result obtained without decomposition and with decomposition) as the metrics for evaluation. We choose matrix representation based closed form solution for linear regression in this work and present its fog specific decomposition in the next section. The major difference and reason behind choosing closed form solution is that Gradient Descent and Stochastic Gradient Descent are iterative in nature and their convergence towards solution is time consuming whereas matrix representation based closed form solution gives us a way for solving the least squares problem fit to the parameter without needing to use an iterative algorithm. Other reasons of preferring direct methods over iterative include their robustness and predictable behaviour in terms of resources required for their execution [43], [44]. The selection of the method depends on the context of the problem being solved [29]. So given m training examples: $(\mathbf{x}_1^{in}, y_1^{out}), (\mathbf{x}_2^{in}, y_2^{out}), \ldots\ldots, (\mathbf{x}_m^{in}, y_m^{out})$, we write a matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$ with $\mathbf{x}_1^{in}, \mathbf{x}_2^{in}, \ldots\ldots, \mathbf{x}_m^{in}$ as rows, and column vector $y \in \mathbb{R}^{m\times 1}$, $y = [y_1^{out}, y_2^{out}, \ldots\ldots, y_m^{out}]^{m\times 1}$ as the targeted value corresponding to each row, then the solution for parameter $\mathbf{w}$ is:

$$\mathbf{w}^* = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}y \tag{3}$$

V. DECOMPOSING DATA ANALYTICS MODEL

In this section we present the decomposition method for fog enabled IoT systems. We present the theoretical fundamentals that are used in this work for the decomposition of the multivariate linear regression model. The following mathematical notation is used in the sub-section-A below:

- Data set $\mathcal{D} = \left\langle \mathbf{x}_l^{in}, y_l^{out} \right\rangle_{l=1}^{m}$
- X is input to the learning model — $X = \langle (\mathbf{x}_l^{in}) \rangle_{l=1}^{m}$
- Y is a function of X that we want to learn — $Y = F(\mathbf{x}) = \langle y_l^{out} \rangle_{l=1}^{m}$

A. STATISTICAL QUERY MODEL AND SUMMATION FORM

The Statistical Query Model [45], [46] is often presented as a restriction on the Valiant model [47]. In Valiant model, the learning algorithm uses randomly drawn training example $\langle \mathbf{x}_l^{in}, y_l^{out} \rangle$ to learn the target function f where as in Statistical Query model the learning algorithm uses some aggregates over the examples and not the individual examples. Given a function $f(\mathbf{x}, y)$ over instances (data points $\mathbf{x}$ and labels y), the statistical computing operation returns an estimate of the expectation of $f(\mathbf{x}, y)$(averaged over training/test distribution). Any learning model/algorithm that calculates sufficient statistics or gradients fits this model, and since these calculations may be batched, they are expressible as a sum over data points [38]. The class of such models/algorithms is large,[4] [38] for example Linear Regression, Naive Bayes, Logistic Regression, Support Vector Machine (SVM) are to name few among-st many others.

Authors in [38] show that any algorithm that fits the Statistical Query Model may be written in a certain "summation form". This form does not change the underlying algorithm and so is not an approximations, but is instead an exact implementation. They show that the summation form can be expressed in a map-reduce framework and the technique can achieve a linear speed up with the number of cores on a multicore machine. Their approach is a novel contribution to achieve parallelization for a large class of machine learning methods on a multicore machine. Their main objective was to develop a general and exact technique for parallel

[4] Although we present the fog specific decomposition only for Linear Regression, but in the same manner the methodology can be extended and modified for decomposition of other predictive analytics models as well.

programming of a large class of ML (Machine Learning) algorithms for multicore processors. The authors show that when an algorithm does sum over data then its calculation/computation can be distributed over multiple cores by dividing the data set into as many pieces as there are cores, give each core its share of data to sum the equations over, and aggregate the results at the end. They call this form of the algorithm as the "summation form". We use and extend their technique for the fog specific decomposition in our work to achieve distributed data analytics in fog enabled IoT deployments.

B. SUMMATION FORM OF LINEAR REGRESSION

In this section we present that how to put the algorithm into "summation form" as worked by authors in [38]. The solution for the parameter **w** as shown in equation 3 is:

$$\mathbf{w}^* = (\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}\mathbf{X}^\mathbf{T}y \tag{4}$$

To put the above computation into "summation form", it is reformulated into a two phase algorithm where first "sufficient statistics" are computed by summing over the data, and those statistics are aggregated to get and solve:

$$\mathbf{w}^* = \mathbf{A}^{-1}\mathbf{b} \tag{5}$$

where $\mathbf{A} = \mathbf{X}^\mathbf{T}\mathbf{X}$ and $\mathbf{b} = \mathbf{X}^\mathbf{T}y$, and is computed as follows:

$$\mathbf{A} = \sum_{\alpha=1}^{m} \left(\mathbf{x}_\alpha^{in}(\mathbf{x}_\alpha^{in})^\mathbf{T}\right) \quad \text{and} \quad \mathbf{b} = \sum_{\alpha=1}^{m} (\mathbf{x}_\alpha^{in} y_\alpha^{out})$$

m is the total number of training examples in the dataset. The computation of **A** and **b** can now be divided into equal size pieces and distributed amongst the cores as presented by authors [38] to achieve parallelization for ML programs on multicore machines. In terms of their proposed Map-reduce framework for ML the above computation is divided among mappers. As in this case, one set of mappers is used to compute $\sum_{subgroup} \left(\mathbf{x}_\alpha^{in}(\mathbf{x}_\alpha^{in})^\mathbf{T}\right)$ and another set to compute $\sum_{subgroup}(\mathbf{x}_\alpha^{in} y_\alpha^{out})$. A set of reducers then respectively sum up the partial values for **A** and **b**, and finally computes the solution $\mathbf{w}^* = \mathbf{A}^{-1}\mathbf{b}$ (i.e. equation 5).

In our work instead of distributing the computing operations over cores, the computing operation of summation is put on each fog node in the infrastructure. So each of the fog node performs the computing operation of summing over the collected data and then sending the summarized outputs to either fog node or cloud (depending on the implementation and use-case) rather than sending raw data over the network.

In many industrial settings and IoT deployments, the data is collected and stored in a decentralized manner. When the data generation/ storage is itself distributed, then it appears more desirable to also process/analyse it in a distributed fashion to avoid the bottleneck of data transfer to the centralized cloud.

The pseudo code of the implementation discussed in this section and as used in the experiments has been presented in Algorithm 1 and Algorithm 2. Algorithm 1 represents the Linear Regression component running on fog nodes and

Algorithm 1 FLRC (Fog Linear Regression Component) - Decomposed Computing Program Running on Each Fog Node VM

```
Initialize : $\mathbf{A_t} = 0$, $\mathbf{b_t}$ [] = 0
▷ $\mathbf{A_t}$ will be a real value and $\mathbf{b_t}$ will be a vector with same dimension as $[\mathbf{x}_\alpha^{in}]_t$. $\mathbf{A_t}$ and $\mathbf{b_t}$ will contain the partial calculated values after the execution of the program.
Input : $X_t$ [] = $[\mathbf{x}_\alpha^{in}]_t$, $Y_t$ [] = $[y_\alpha^{out}]_t$
▷ $[\mathbf{x}_\alpha^{in}]_t$ and $[y_\alpha^{out}]_t$ represent the data received in set processing frequency $t$
Output : Processed outputs calculated in the set processing frequency
1: function FLRC( $X_t$ [], $Y_t$ [])
2:     for index = 0 to (size of $X_t$ []) − 1 do
3:         $\mathbf{A_t}$+ = dotproduct ($X_t$ [index], Transpose ($X_t$ [index]))
4:         $\mathbf{b_t}$ [] + = $X_t$ [index] * $Y_t$[index]
5:         index + = 1
6:     end for
7:     return ($\mathbf{A_t}$, $\mathbf{b_t}$ [])
8: end function
```

Algorithm 2 CLRC (Cloud Linear Regression Component) - Program Running on Cloud to Combine the Partial Results Obtained From Fog Nodes

```
Initialize : A = 0, b [] = 0, w* [] = 0
Input : Processed outputs obtained from Fog Nodes i.e. Different $\mathbf{A_t}$'s and $\mathbf{b_t}$'s [] received during the whole duration of the experiment
Output : Regression Coefficients i.e. Linear Regression Model in the Distributed Approach
1: function CLRC($\mathbf{A_t}$'s, $\mathbf{b_t}$'s [])
2:     A = SUM of all $\mathbf{A_t}$'s received
3:     b [] = SUM of all $\mathbf{b_t}$'s received
4:     w* [] = $\frac{\mathbf{b}}{\mathbf{A}}$
5:     return (w* [])
6: end function
```

Algorithm 2 represents the component running in cloud for the fog based distributed analytics approach. In cloud centric approach, both Algorithm 1 and 2 runs on the cloud as the whole processing happens there and fog node acts as normal gateways without the analytics component running on them.

VI. EXPERIMENTAL SETUP AND DATA-SETS

The experiment was performed on the OpenStack VM (Virtual Machine) instances. The setup consists of a total of 5 VMs as shown in figure 4. The configuration details of the VMs are presented in table 1. All VMs have Ubuntu 18.04.2 LTS as their operating system and Intel Xeon processors (@2.60 GHz).

The configuration of VMs acting as fog nodes were kept inline to the commercially available IoT Gateways by Dell and Intel [48], [49]. The computing capacity of the

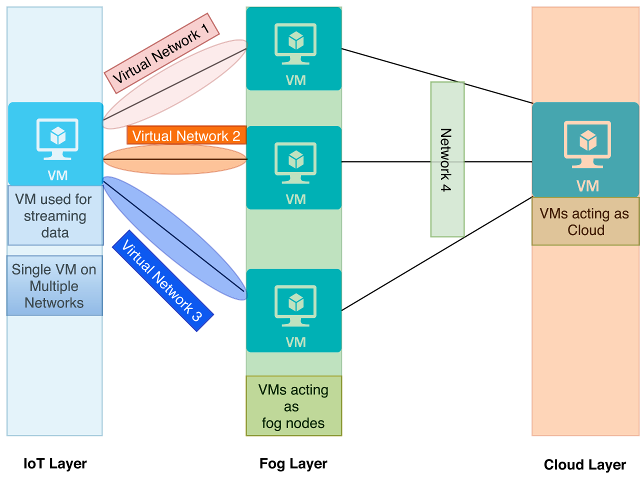
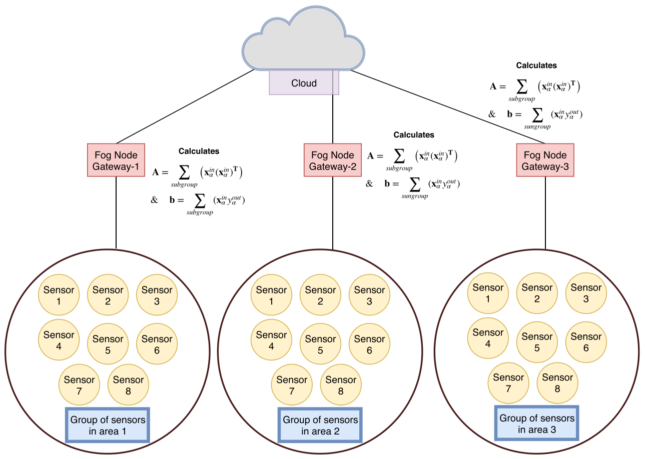


FIGURE 4. Experimental setup deployed on OpenStack.



Cloud
Fog Node Gateway-1
Fog Node Gateway-2
Fog Node Gateway-3
Calculates
$\mathbf{A} = \sum_{subgroup} \left(\mathbf{x}_\alpha^{in}(\mathbf{x}_\alpha^{in})^\mathbf{T}\right)$
$\& \quad \mathbf{b} = \sum_{subgroup} (\mathbf{x}_\alpha^{in} y_\alpha^{out})$
Calculates
$\mathbf{A} = \sum_{subgroup} \left(\mathbf{x}_\alpha^{in}(\mathbf{x}_\alpha^{in})^\mathbf{T}\right)$
$\& \quad \mathbf{b} = \sum_{sungroup} (\mathbf{x}_\alpha^{in} y_\alpha^{out})$
Calculates
$\mathbf{A} = \sum_{subgroup} \left(\mathbf{x}_\alpha^{in}(\mathbf{x}_\alpha^{in})^\mathbf{T}\right)$
$\& \quad \mathbf{b} = \sum_{sungroup} (\mathbf{x}_\alpha^{in} y_\alpha^{out})$
Sensor 1
Sensor 2
Sensor 3
Sensor 4
Sensor 5
Sensor 6
Sensor 7
Sensor 8
Group of sensors in area 1
Group of sensors in area 2
Group of sensors in area 3

FIGURE 5. Real-world equivalent representation of the experimental setup deployed on OpenStack.

TABLE 1. Experimental configurations.

Use and type of VM	VCPUs	RAM (in GB)	DISK (in GB)	Number of VMs
VM used for streaming data	2	4	10	1
VMs acting as Fog Nodes	4	8	20	3
VM acting as Cloud	8	16	30	1

VMs increases as we move up in the hierarchy. We used real-world dataset from UCI repository [50] in our experiments to evaluate the performance of the proposed mechanism. The dataset [51] contains 9358 instances of hourly averaged responses of chemical compounds and environmental parameters from air quality sensor. In the dataset, the parameters used to measure the air pollution of a specific area include—CO (ground truth values), PT08.S1 (CO), NMHC (Non Metanic HydroCarbons, ground truth values), Benzene (C6H6, ground truth values), PT08.S2 (NMHC), NOx (ground truth values), PT08.S3 (NOx), NO2 (ground truth values), PT08.S4 (NO2), PT08.S5 (O3), temperature, relative humidity, absolute humidity. These parameters are used to measure the air pollution of a specific area. It consists of a total of 13 features out of which 5 represent the ground truth values for the same type. The authors in [51] have used this dataset for benzene estimation in an urban environment pollution scenario. We also use this dataset for benzene estimation in our linear regression task. We remove the ground truth values from the dataset and thus remain with 8 features as input ($\mathbf{x}_i^{in}$) to the regression task with Benzene (C6H6, ground truth values) as the targeted variable y_i^{out}.

The data is column-standardized (mean centering and scaling) and normalized i.e. each vector $\mathbf{x}_i^{in}$ is mapped to $\frac{\mathbf{x}_i^{in}-\mu}{\sigma}$ with mean value μ and variance σ^2, and scaled in [0, 1].

The 8 features correspond to 8 sensors in the real-word setting. The data split is made as 70-30 i.e. 70% (approximately 6540 instances) data is used for model training and 30% (approximately 2818 instances) is used for testing. We then randomly divide the train data into three equal parts each containing approximately 2180 instances and each part is streamed on row by row basis to corresponding VM acting as fog node gateway. This corresponds to a real-world setup where a group of 8 sensors is present in 3 different locations (so 24 sensors in all) which are streaming their sensed values to their corresponding gateways. The graphical representation of such an equivalent real-world setup is as shown in Fig. 5

We have chosen Message Queue Telemetry Transport (MQTT) [52] as the streaming protocoal in our setup. MQTT is an open-source protocol originally invented and developed by IBM [53]. It is a lightweight publish-subscriber model based protocol designed on top of the TCP/IP stack. It is specifically targeted for remote location connectivity with characteristically unreliable network environments such as high delays and low bandwidth [54]. The MQTT architecture comprises of two components, namely MQTT clients (such as publishers and subscribers) and MQTT broker (for mediating messages between publishers and subscribers). In our setup these components are as follows:

- **MQTT Publisher:** Script running on streaming VM acts as the MQTT publisher client. It streams the data on row by row basis every second (i.e 1 row per second) to each of three VMs acting as fog node gateway.
- **MQTT Broker:** The VMs acting as fog node gateway act as broker between VM acting as cloud and VM streaming data as sensor. The VMs acting as fog node gateway subscribe to the streaming VM acting as publisher.
- **MQTT Subscriber:** Another script running on VM acting as cloud subscribes to the VMs acting as fog node gateways.

The processing frequency for data received at fog node VMs (i.e. MQTT Broker) was set to 5 seconds. We used Paho [55] as MQTT client library and Mosquitto [56] as MQTT broker library in our implementation. The VMs acting

TABLE 2. Accuracy of generated models.

Approach	Root Mean Square Error (RMSE)	Variance Score/ Accuracy
Distributed Approach	0.0384	0.9673
Cloud Centric Approach (Summation Form)	0.0384	0.9673

as fog node gateway receive data, performs the computing operation as described earlier and presented in Algorithm 1, and sends the output to the VM acting as cloud.

The experiment was performed in two scenarios, one where the VMs acting fog node gateways perform the computing operation of calculating subgroups of **A** and **b** as per Algorithm 1, and send the processed output to the cloud VM. Second scenario is the traditional centralized setup where fog gateway VMs receive the streaming data and forward it as it is to the cloud, and the whole data analytics operation takes place in the cloud VM. The results from the first scenario have been labelled as *Distributed Approach* and from second scenario as *Cloud Centric Approach*.

VII. RESULTS AND DISCUSSION

The system utilization metrics such as CPU, memory and bandwidth utilization were noted in both the scenarios and have been presented in this section. Along with that, metrics to measure the quality of analytics solution in both scenarios were evaluated and have also been presented here. The centralized solution acts as a comparative measure against the distributed approach.

The following notations have been used in this section:

- **Fog Layer**: Average value of metrics obtained from the 3 VMs acting as fog nodes
- **Cloud Layer**: VM acting as Cloud

The experiment runs for approximately around 35 minutes. We used Python Resmon [57] and Glances [58] to measure the resource utilization metrics of VMs in the experiment.

A. ACCURACY AND DISTRIBUTION PLOTS

The Linear Regression model generated in both the approaches was tested on the same test data, and results of the same have been presented in table 2. The values presented for both RMSE (Root Mean Square Error) and variance score/accuracy are up-to 4 significant digits after the decimal.

The lower the RMSE the better the model. The results suggest that both the approaches generate the same model, as the RMSE and variance score values obtained are exactly the same. So distributed approach can be used to obtained the same results as one would have obtained from the traditional centralized approach.

The error distribution in both the approaches has been presented in fig. 6. The X-axis here represents $\Delta Y = Y_{test} - Y_{predicted}$ and Y-axis represents the probability density of it. The first thing to notice here is that the errors are centered at zero in both approaches i.e. the most often found error is zero, which is good. The distribution plot from both the approaches look exactly the same, which suggests that distributed approach can be used in fog enabled

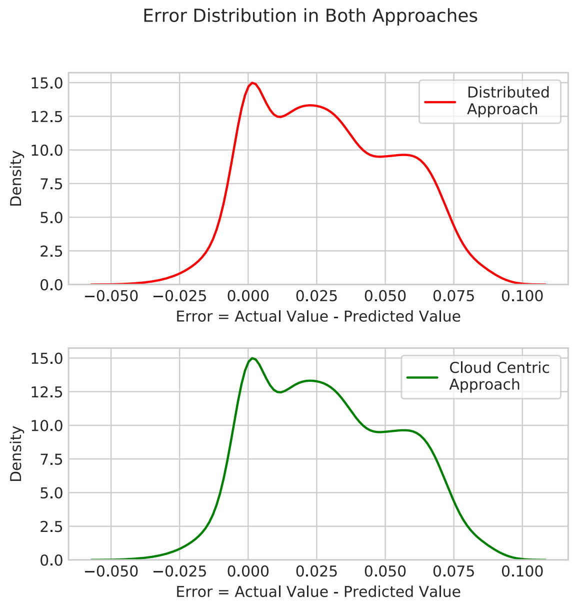


FIGURE 6. Error distribution plot in both approaches.

IoT deployments. In both the approaches the errors are more on the positive side as we can see from the heights of the distribution on the +ve side and there are fewer error on the −ve side. The errors are fairly small in both approaches which suggests that its a fairly decent solution.

Fig. 7 presents the scatter plot of actual and predicted values in both the approaches. Fig. 7a represents the plot on shared axis and fig. 7b represents the plot on same axis. As visible from the plot, the distributed approach and cloud centric approach trace out each other.

B. CPU UTILIZATION

The CPU utilization of various entities involved in the experimental setup has been presented as box plots in Fig. 8. The values shown in the figure represents the median values. The results of CPU utilization have been discussed below:

1) FOG LAYER CPU UTILIZATION

CPU utilization of fog node VMs increases in the distributed approach as now they are also performing the analytics operation rather than just forwarding the data to cloud for analysis. This also adds to efficient resource utilization of these devices.

2) CLOUD LAYER CPU UTILIZATION

CPU utilization of Cloud VM is less in distributed approach and more in the traditional cloud-centric approach. The reason for this is that in cloud centric approach all the processing happens in cloud, while in distributed approach cloud only sums up the partial results obtained from fog nodes, thus leading to less CPU utilization in the latter.

This is beneficial for the user as now the monetary cost will be less for the cloud service utilization under the ‘pay as you go’ model. In most IoT based deployments the gateways are

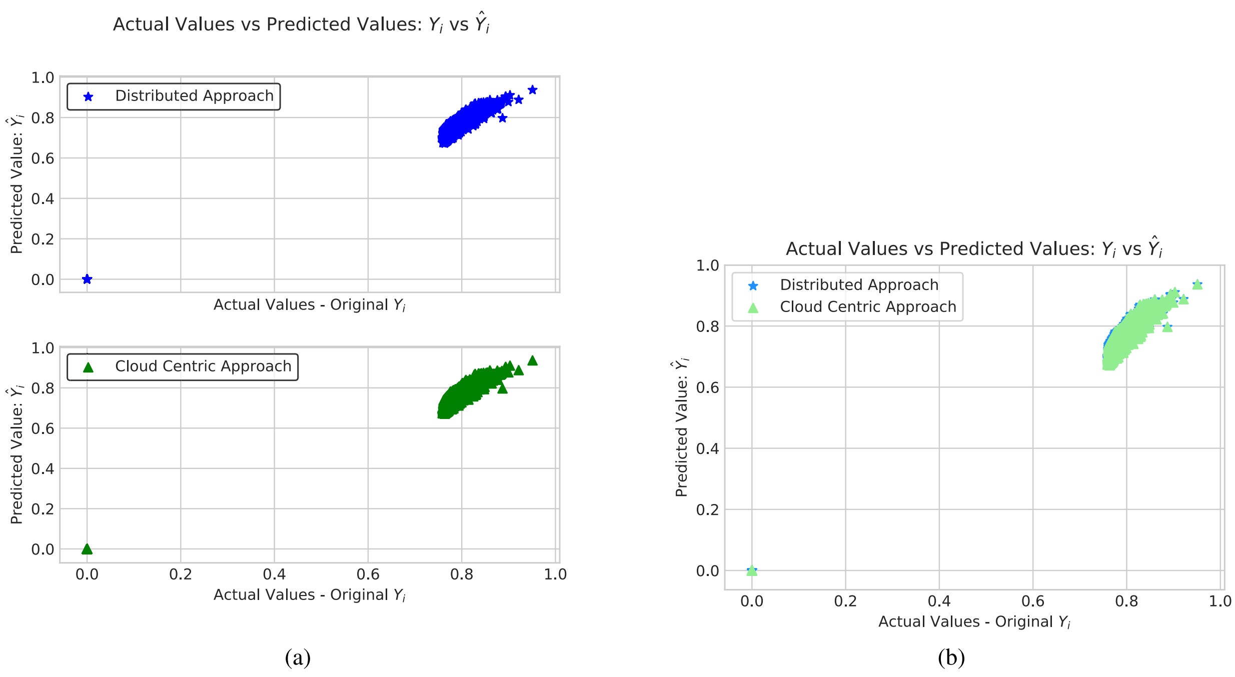


FIGURE 7. Scatter plot that shows that the distributed and cloud centric approach both trace out each other. (a) Scatter Plot of actual and predicted values in both approaches. (b) Scatter Plot of actual and predicted values in both approaches plotted together.

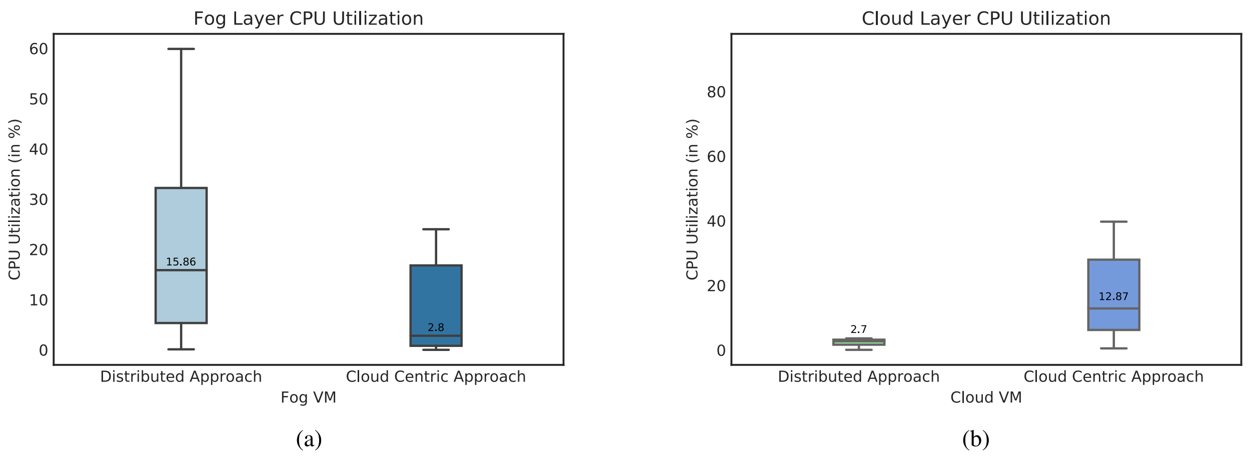


FIGURE 8. CPU utilization visualized for both the distributed and cloud centric approach. (a) Fog layer CPU utilization. (b) Cloud layer CPU utilization.

usually owned by the user so effectively the user will have to pay significantly less amount in the distributed approach.

C. MEMORY UTILIZATION

The memory utilization has been presented as bar plots in Fig. 9. The values shown in the plot represent the median values. This follows the same behavior as for CPU utilization as discussed above.

D. DATA REDUCTION

Bar plot in Fig. 10 represents the reduction in amount of data being streamed from fog VMs to cloud VMs in both approaches. There is 80% reduction in the amount of data being streamed in distributed approach as compared to the cloud centric approach. The reason for this reduction is the Fog Linear Regression Component (Algorithm 1) running on fog VMs in distributed analytics approach while in cloud centric approach they just act as normal gateways to forward the data received to cloud for analytics and the whole analytics operation happens there.

E. TIME

We also measured the time required to calculate the regression coefficients in both the approaches i.e. to generate the final linear regression model. In both the approaches, the final model is calculated at the cloud. The values have been presented in table 3.

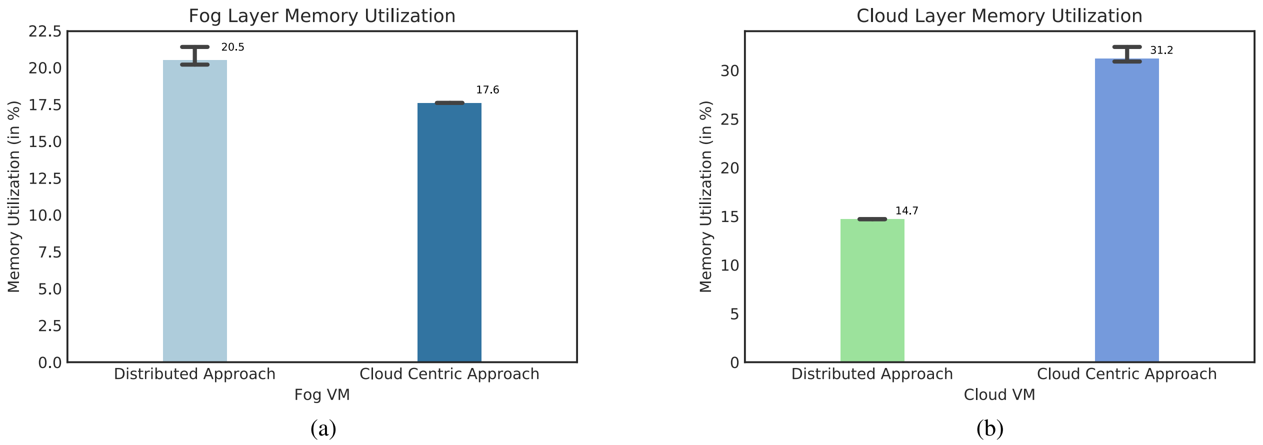


FIGURE 9. Memory utilization visualized for both the distributed and cloud centric approach. (a) Fog layer memory utilization. (b) Cloud layer memory utilization.

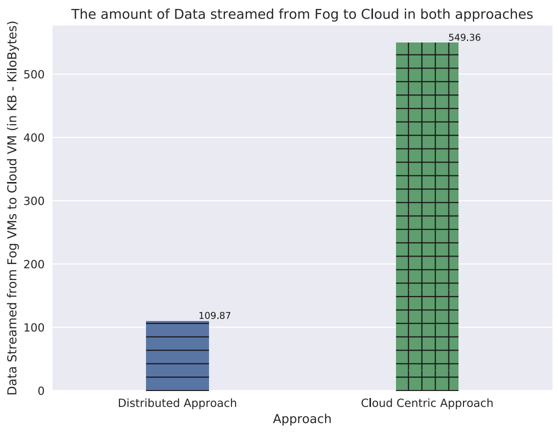


FIGURE 10. Reduction in amount of data being streamed from fog VMs to cloud VM in fog based distributed approach.

TABLE 3. Time taken to calculate regression coefficients in both approaches.

	Distributed Approach	Cloud Centric Approach
Time Taken (in milliseconds)	2.87 ms	138.62 ms

There is significant reduction in the amount of time required to calculate the final model in distributed approach compared to the cloud centric approach. The reason behind this is that in distributed approach cloud sums (Algorithm 2) the partial outputs obtained from fog nodes and then calculates the final model, while in cloud centric approach, the entire end-to-end processing happens in cloud (i.e. both Algorithm 1 and Algorithm 2 run on cloud).

F. FURTHER DISCUSSION

1) USE CASE AND CONSTRAINTS

Given that the streaming rate of data from sensor/base node to the fog device is fixed, there are two kinds of use cases that define the role of the fog node in the distributed computing approach:

1) The first use case is where the fog node uses its resources for pre-processing the data acquired from sensors. This serves two purposes:
    a) **Data reduction:** Lesser data is sent from fog node to the cloud
    b) **Decreased computation time (in the cloud):** Since the data is already pre-processed with some initial operations, the time taken for the cloud component to process the entire data set into the desired output is lesser than the regular cloud-centric case. This is in addition to no overheads in terms of the total time required in the complete end-to-end process.
2) The second use case is where the fog node acts as both the data processing and decision making entity. This is particularly for latency sensitive use cases, where the data streaming and processing scenarios require latency critical decision making.

The three constraints that drive the above scenarios include:

1) Rate of streaming of data from end node to fog layer
2) Resource capacity of fog layer
3) Nature of use case:
    a) latency sensitive / time critical decision making
    b) regular computation required from data set

2) IMPACT OF CHANGING PROCESSING FREQUENCY AT FOG NODE AND DATA REDUCTION

With the above understanding we also measured the effect of varying processing frequency for the data received at fog node and corresponding reduction in data transfer from Fog Layer to Cloud Layer. The corresponding plots has been presented in Fig. 11a and 11b.

As visible from the plot, if we keep the data processing frequency at fog nodes same as the data receiving rate then

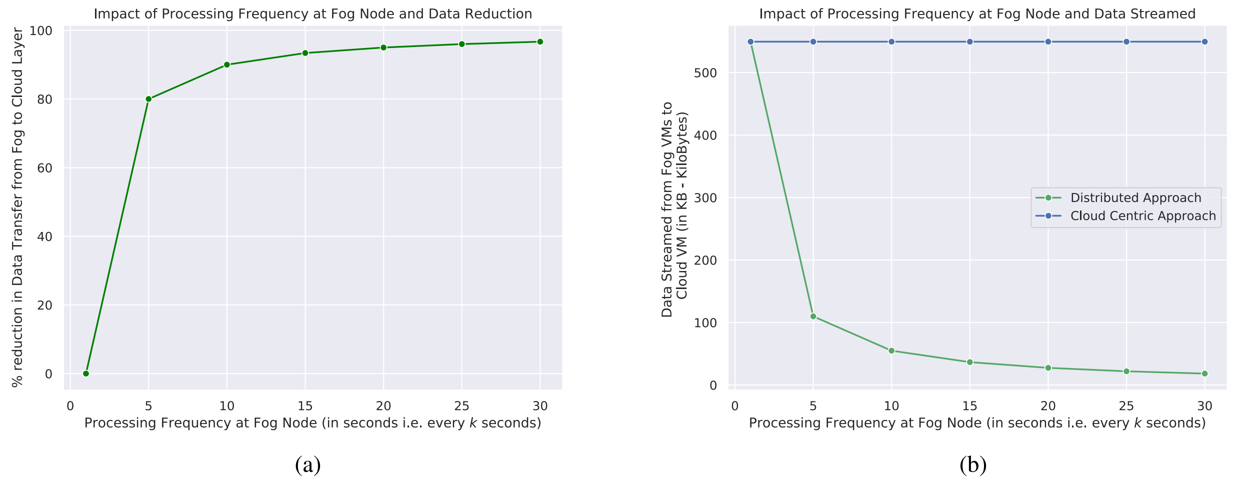


FIGURE 11. **Impact of changing data processing frequency at fog node and data reduction. (a) Effect of changing data processing frequency at fog node VMs and corresponing % reduction in data transfer to cloud using fog based distributed data analytics approach compared to traditional cloud centric approach. (b) Effect of changing data processing frequency at fog node VMs and corresponing reduction in data transfer to cloud using fog based distributed data analytics approach compared to traditional cloud centric approach.**

there is no reduction in the data transfer, but as we gradually increase the processing frequency the data reduction becomes significant. The decrease in data reduction with increasing processing frequency almost saturates after some point and % reduction in data transfer becomes smaller.

In cloud centric approach even if we increase the data processing frequency at fog nodes, they still send all the received data, just that now they send it collectively, while in distributed approach, only the processed data is sent further to cloud and hence gain in data reduction. With higher processing frequency at fog node, which in turn means bigger buffer size to process the received data, the approach in turn effectively becomes centralized in nature, and thus the gain is reduction does not increase much after a certain point.

VIII. CONCLUSION AND FUTURE WORK

Carrying forward our work [59] here we present the method, approach and results for adopting distributed decomposed data analytics in fog enabled IoT deployments. The benefit of using fog computing for all IoT based applications may not be obvious since benefits gained may not be significant to motivate the use of the edge of the network. It might also be argued that it is more desirable to develop cloud centric solutions with sufficiently large number of resources available on hand, rather than designing fully distributed computing programs/algorithms which might bring along additional complexities. However, the number of data centres is less likely to grow at the same rate as the number of devices at the network edge (and thus the generated amount of data) being connected to the Internet, since traditional data centres consume a lot of power and global network bandwidth, and have begun to raise the impending concern of increased carbon footprint.

Overall, keeping in mind the challenges, the decomposition of analytics programs in fog assisted IoT environments does look promising towards the effort to design efficient distributed data analytics solutions and making the edge of network smarter, and in line with the vision of distributed computing towards future networks. Our future work involves building a framework for efficient placement of decomposed computing units and dynamic task sharing between fog and cloud.

REFERENCES

[1] J. Manyika *et al.*, *Unlocking the Potential of the Internet of Things*. New York, NY, USA: McKinsey, Jun. 2015.
[2] M. Evans. *IoT Will Have the Most Impact on Business in the Next Five Years, Survey Says*. [Online]. Available: https://www.forbes.com/sites/michelleevans1/2018/05/31/iot-will-have-the-most-impact-on-business-in-the-next-five-years-survey-says/#1303d3035a3d
[3] *The Zettabyte Era–Trends and Analysis*, Mobile Vis. Netw. Index, Cisco, San Jose, CA, USA, Jun. 2017.
[4] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog computing for sustainable smart cities: A survey," *ACM Comput. Surv.*, vol. 50, no. 3, p. 32, Oct. 2017.
[5] L. Belli, S. Cirani, G. Ferrari, L. Melegari, and M. Picone, "A graph-based cloud architecture for big stream real-time applications in the Internet of Things," in *Advances in Service-Oriented and Cloud Computing*, G. Ortiz, and C. Tran, Eds. Cham, Switzerland: Springer, 2015, pp. 91–105.
[6] R. Mahmud, R. Kotagiri, and R. Buyya. (2016). "Fog computing: A taxonomy, survey and future directions." [Online]. Available: https://arxiv.org/abs/1611.05539
[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*, New York, NY, USA, 2012, pp. 13–16.
[8] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 1222–1228.
[9] T. Zachariah, N. Klugman, B. Campbell, J. Adkins, N. Jackson, and P. Dutta, "The Internet of Things has a gateway problem," in *Proc. 16th Int. Workshop Mobile Comput. Syst. Appl. (HotMobile)*, New York, NY, USA, 2015, pp. 27–32.
[10] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog computing for sustainable smart cities: A survey," *ACM Comput. Surv.*, vol. 50, pp. 32-1–32-43, Jun. 2017.
[11] *Foghorn Systems: Edge Intelligence Software for IIoT*. Accessed: Dec. 4, 2018. [Online]. Available: https://www.foghorn.io/
[12] *IBM Watson—IBM and Cisco Cloud Analytics*. Accessed: Sep. 2, 2018. [Online]. Available: https://www.ibm.com/internet-of-things/partners/ibm-cisco/
[13] B. Tang *et al.*, "Incorporating intelligence in fog computing for big data analysis in smart cities," *IEEE Trans. Ind. Informat.*, vol. 13, no. 5, pp. 2140–2150, Oct. 2017.
[14] Amazon. *AWS IoT Greengrass—Amazon Web Services*. [Online]. Available: https://aws.amazon.com/greengrass/

[15] Microsoft. *Azure IoT Edge*. Accessed: Jan. 5, 2019. [Online]. Available: https://github.com/Azure/iot-edge-v1
[16] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: A green computing paradigm to support IoT applications," *IET Netw.*, vol. 5, no. 2, pp. 23–29, 2016.
[17] M. Taneja and A. Davy, "Resource aware placement of data analytics platform in fog computing," *Procedia Comput. Sci.*, vol. 97, pp. 153–156, Jan. 2016.
[18] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.
[19] N. Helwig, E. Pignanelli, and A. Schütze, "Condition monitoring of a complex hydraulic system using multivariate statistics," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, May 2015, pp. 210–215.
[20] A. Whitmore, A. Agarwal, and L. D. Xu, "The Internet of Things—A survey of topics and trends," *Inf. Syst. Frontiers*, vol. 17, no. 2, pp. 261–274, Apr. 2015.
[21] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.
[22] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proc. Adv. Neural Inf. Process. Syst.*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. New York, NY, USA: Curran Associates, Inc., 2008, pp. 161–168.
[23] E. Siow, T. Tiropanis, and W. Hall, "Analytics for the Internet of Things: A survey," *ACM Comput. Surv.*, vol. 51, pp. 74-1–74-36, Jul. 2018.
[24] H. Jiang, S. Jin, and C. Wang, "Prediction or not? An energy-efficient framework for clustering-based data collection in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 1064–1071, Jun. 2011.
[25] R. V. Kulkarni, A. Forster, and G. K. Venayagamoorthy, "Computational intelligence in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 1, pp. 68–96, 1st Quart., 2011.
[26] N. Cheng, N. Lu, N. Zhang, T. Yang, X. Shen, and J. W. Mark, "Vehicle-assisted device-to-device data delivery for smart grid," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2325–2340, Apr. 2016.
[27] C. Anagnostopoulos, "Time-optimized contextual information forwarding in mobile sensor networks," *J. Parallel Distrib. Comput.*, vol. 74, no. 5, pp. 2317–2332, 2014.
[28] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context-aware computing, learning, and big data in Internet of Things: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 1–27, Feb. 2018.
[29] L. Shi, L. Zhao, W.-Z. Song, G. Kamath, Y. Wu, and X. Liu. (2017). "Distributed least-squares iterative methods in networks: A survey." [Online]. Available: https://arxiv.org/abs/1706.07098
[30] A. Lazerson, D. Keren, and A. Schuster, "Lightweight monitoring of distributed streams," *ACM Trans. Database Syst.*, vol. 43, pp. 9-1–9-37, Jul. 2018.
[31] H. Wang and C. Li, "Distributed quantile regression over sensor networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 2, pp. 338–348, Jun. 2018.
[32] M. Gabel, D. Keren, and A. Schuster, "Monitoring least squares models of distributed streams," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2015, pp. 319–328.
[33] G. Kamath, P. Agnihotri, M. Valero, K. Sarker, and W.-Z. Song, "Pushing analytics to the edge," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[34] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, "Practical data prediction for real-world wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 8, pp. 2231–2244, Aug. 2015.
[35] N. Harth and C. Anagnostopoulos, "Quality-aware aggregation & predictive analytics at the edge," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 17–26.
[36] M. Taneja and A. Davy, "Poster abstract: Resource aware placement of data stream analytics operators on fog infrastructure for Internet of Things applications," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2016, pp. 113–114.
[37] T.-C. Chang, L. Zheng, M. Gorlatova, C. Gitau, C.-Y. Huang, and M. Chiang, "Decomposing data analytics in fog networks," in *Proc. 15th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, New York, NY, USA, 2017, pp. 35-1–35-2.
[38] C.-T. Chu *et al.*, "Map-Reducefor machine learning on multicore," in *Proc. 19th Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Cambridge, MA, USA: MIT Press, 2006, pp. 281–288.
[39] M. J. McGrath and C. N. Scanaill, "Sensor network topologies and design considerations," in *Sensor Technologies*. Berkeley, CA, USA: Apress, 2013, pp. 79–95.
[40] M. Kuhn and K. Johnson, *Applied Predictive Modeling*, vol. 26. New York, NY, USA: Springer-Verlag, 2013.
[41] A. R. Benson, D. F. Gleich, and J. Demmel. (2013). "Direct QR factorizations for tall-and-skinny matrices in MapReduce architectures." [Online]. Available: https://arxiv.org/abs/1301.1071
[42] C. Misra, S. Haldar, S. Bhattacharya, and S. K. Ghosh, "SPIN: A fast and scalable matrix inversion method in apache spark," in *Proc. ACM 19th Int. Conf. Distrib. Comput. Netw. (ICDCN)*, New York, NY, USA, 2018, pp. 16-1–16-10.
[43] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *J. Comput. Phys.*, vol. 182, no. 2, pp. 418–477, 2002.
[44] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA, USA: SIAM, 2003.
[45] M. Kearns, "Efficient noise-tolerant learning from statistical queries," *J. ACM*, vol. 45, no. 6, pp. 983–1006, 1998.
[46] M. Kearns, "Efficient noise-tolerant learning from statistical queries," in *Proc. 25th Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, 1993, pp. 392–401.
[47] L. G. Valiant, "A theory of the learnable," *Commun. ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.
[48] Intel. *Intel IoT Gateway*. Accessed: Dec. 24, 2018. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/gateway-solutions-iot-brief.pdf
[49] Dell. *Edge Gateway 5100 | Dell Ireland*. Accessed: Dec. 24, 2018. [Online]. Available: https://goo.gl/vWcdSW
[50] *UCI Machine Learning Repository: Air Quality Data Set*. Accessed: Dec. 24, 2018. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/air+quality
[51] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. D. Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario," *Sens. Actuators B, Chem.*, vol. 129, no. 2, pp. 750–757, 2008.
[52] *MQTT*. Accessed: Aug. 3, 2017. [Online]. Available: http://mqtt.org/
[53] *Getting to Know MQTT*. Accessed: Aug. 3, 2017. [Online]. Available: https://www.ibm.com/developerworks/library/iot-mqtt-why-good-for-iot/index.html
[54] S. Lee, H. Kim, D.-K. Hong, and H. Ju, "Correlation analysis of MQTT loss and delay according to QoS level," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2013, pp. 714–717.
[55] *Eclipse Paho—MQTT*. Accessed: Dec. 28, 2018. [Online]. Available: https://www.eclipse.org/paho/
[56] *Eclipse Mosquitto-MQTT*. Accessed: Dec. 28, 2018. [Online]. Available: https://mosquitto.org/
[57] *GitHub—Xybu/Python-Resmon*. Accessed: Jan. 2, 2019. [Online]. Available: https://github.com/xybu/python-resmon
[58] *Glances—An Eye on Your System*. Accessed: Jan. 2, 2019. [Online]. Available: https://nicolargo.github.io/glances/
[59] M. Taneja, N. Jalodia, and A. Davy. (Sep. 2018). Towards Decomposed Data Analytics in Fog Enabled IoT Deployments. IEEE Internet of Things Newsletter. [Online]. Available: https://iot.ieee.org/newsletter/september-2018/towards-decomposed-data-analytics-in-fog-enabled-iot-deployments.html

MOHIT TANEJA received the bachelor's degree in computer science and engineering from The LNM Institute of Information Technology, Jaipur, India, in 2015, and the master's degree. He is currently pursuing the Ph.D. degree with the Emerging Networks Laboratory, Telecommunications Software and Systems Group, Department of Computing and Mathematics, Waterford Institute of Technology, Ireland. He has been working as a part of the Science Foundation Ireland funded CONNECT Research Centre, since 2015. His current research interests include fog and cloud computing, the Internet of Things (IoT), distributed systems, and distributed data analytics. His research focuses on decomposing data analytics and machine learning programs for the fog-enabled IoT systems toward effective resource and service management to support and meet the requirements for the real-time IoT analytics.

NIKITA JALODIA received the bachelor's degree in computer science and engineering from The LNM Institute of Information Technology, Jaipur, India, in 2017, with a specialization in big data and analytics at IBM. She is currently pursuing the Ph.D. degree with the Emerging Networks Laboratory, Telecommunications Software and Systems Group, Department of Computing and Mathematics, Waterford Institute of Technology, Ireland. She was a Developer at Sapient Global Markets, India. She has been working as a part of the Science Foundation Ireland funded CONNECT Research Centre, since 2017. Her current research interests include the Internet of Things (IoT), fog and cloud computing, machine learning, virtualized telecom networks, and network function virtualization (NFV).

ALAN DAVY received the B.Sc. degree (Hons.) in applied computing and the Ph.D. degree from the Waterford Institute of Technology, Waterford, Ireland, in 2002 and 2008, respectively. Since 2002, he has been with the Telecommunications Software and Systems Group, originally as a student and then, since 2008, as a Postdoctoral Researcher. In 2010, he was an Assistant Professor with IIT Madras, India, lecturing in network management systems. He is currently a Senior Research Fellow and the Research Unit Manager of the Emerging Networks Laboratory, Telecommunications Software and Systems Group. He is also the Coordinator of the EU H2020 FETOpen Project CIRCLE: Coordinating European Research on Molecular Communications. His current research interests include virtualized telecom networks, fog and cloud computing, molecular communication, and terahertz communication. He was a recipient of the Marie Curie International Mobility Fellowship, in 2010, which brought him to work at the Universitat Politècnica de Catalunya for two years.