Scalable and Flexible IoT data analytics: when Machine Learning meets SDN and Virtualization

Jordi Serra, Luis Sanabria-Russo, David Pubill and Christos Verikoukis, Senior Member, IEEE
Telecommunications Technological Center of Catalonia (CTTC), 08860 Castelldefels, Spain
E-mails: {jserra,lsanabria,dpubill,cveri}@cttc.es

***Abstract*—This paper deals with Internet of Things (IoT) data analytics in a collaborative platform where computing resources are available both at the network edge and at the backend cloud. Thereby, the requirements of both low-latency and delay-tolerant IoT applications can be met. Moreover, this platform faces the challenging heterogeneous features of IoT data, i.e. its high dimensionality or its geo-distributed and streaming data nature. The proposed approach relies on two pillars. On the one hand, recent advances of machine learning (ML) techniques are leveraged to describe how the IoT data analytics can be performed in our platform. On the other hand, the virtualization, centralized management, global view and programmability of the computing and network resources is considered to fulfill the requirements of the ML methods. Unlike the related work, herein the interplay and synergies between those two pillars is explained. Also the ML methods for this collaborative platform are described in more detail.**

***Index Terms*—IoT, SDN, Large-Scale Machine Learning, online learning, MANO, NFV, Edge-Cloud computing.**

I. Introduction

IoT is a technological revolution, as it will connect to the Internet a huge amount of heterogeneous devices that are equipped with computing, communications, sensing and actuating capabilities, see e.g. [1]. Thereby, IoT will provoke a data deluge and it will be one of the main contributors to the Big Data paradigm. Groundbreaking applications are expected by connecting the IoT devices globally to the Internet, though to obtain economic benefits the huge amount of generated data must be analyzed. Thus, IoT applications must be equipped with a brain that permits to infer, learn and extract information from the IoT data [2]. This capacity is provided by ML tools.

The conventional approach, for IoT data analytics, has been to send all the data to the cloud, where ML tools are applied to extract valuable information for the end-user applications. The rationale is that third-party companies provide ubiquitous access to a large pool of computing and storage resources. Thereby, at the cloud, the ML tools have access to all the IoT devices' measurements, which leads to obtain deeper insights about the problem than performing local analytics at computing nodes that are close to the IoT devices. E.g. one can take time-variable measurements throughout the smart energy grid and send them to the cloud to detect anomalies in any part of the network, thanks to the global view about the problem. However, this cloud-based data analytics has several potential drawbacks. First, the high-dimensionality of the IoT data can impair the ML algorithms, as the computational burden to implement them can be high [3], which leads to performance loss of ML methods. Also, the third-party company can charge a high economic cost for the use of its computational and storage resources. Second, the network between the IoT devices and the cloud may support a large traffic load, as IoT will generate a huge amount of data. In that case, the network offers a poor service and leads to a performance loss of the ML algorithms. These issues are exacerbated in IoT applications requiring low latency, such as in predictive fault diagnosis in Industry 4.0 [4], self-driving vehicles [5] or anomaly detection in smart energy grids [6]. The communication and computational delays incurred by the cloud-based ML methods may severely impact the performance of those applications. In order to circumvent the drawbacks of the cloud-based approach, several works have proposed to perform the data analytics at the network edge, see [1], [7], [8]. This approach is so-called edge-computing (EC), as the pool of resources is closer to the data sources, i.e. the IoT devices, which permits to reduce the communication latency and to alleviate the traffic network load. Moreover, the data dimensionality at each EC node is smaller than in the cloud as the analytics are performed on the available local data, which reduces the computational burden of the ML methods. The price to pay is that the ML algorithms are suboptimal compared to their counterparts in the cloud, as the latter have a global view of the data set and the former a local view.

The previous paragraphs highlight the pros and cons to carry out the IoT data analytics at the cloud or at the network edge. Thereby, herein to obtain the advantages of those two options, a collaborative edge-cloud IoT data analytics architecture is proposed. Namely, the proposed solution relies on application virtualization [9] to deploy the ML service in a flexible manner at virtual computing resources available at the network edge or at the cloud, which are managed by the Management and Orchestration (MANO) and the Virtual Infrastructure Manager (VIM) controllers. Moreover, to meet the Quality of Service (QoS) requirements of the IoT application and deal with the dynamic network state, the proposed architecture will rely on the use of a Software Defined Network (SDN) framework.

The rest of the paper is organized as follows. In section II the related work and contributions are described. Section III presents the collaborative edge-cloud platform and relevant scenarios for IoT data analytics. In section IV, ML methods, the SDN controller and the MANO are presented as the key tools to face the scenarios of section III. Finally, section V concludes the paper.

II. Related work and Contributions

Several works have considered a collaborative edge-cloud network architecture for IoT data analytics, though this research topic is in its infancy. Namely, the authors in [1] provide the features, challenges and enablers for big IoT data analytics. They highlight the need of a collaborative edge-cloud architecture to process IoT data with different requirements, e.g. low latency versus delay-tolerant needs. The former is analyzed at the edge and the latter at the cloud. Moreover, the cloud guides the operation of the EC thanks to its network-wide view and massive amount of historical information about the network. SDN is proposed as an enabler for a collaborative edge-cloud computing framework, as it controls the network under different dynamic network states. The work in [10] is rather generic and conceptual, as in [1]. Namely, the pros and cons of EC and Cloud computing (CC) are explained to propose a layered and hierarchical architecture integrating both of them. Also they highlight the need of a coordinated edge-cloud management to address challenges such as the set up and release of cloud resources dynamically or the QoS guarantees of the tasks to be computed in such a dynamic and hierarchical system. In [11] the authors deal with the computation offloading of IoT tasks in a collaborative edge-cloud computing framework. They consider a hybrid fiber-wireless edge network, i.e. fiber links between EC servers and access points (AP) and wireless links between APs and IoT devices. Then, they propose a game-theoretic approach to deal with the computation offloading problem among IoT devices, EC and CC servers. On the other hand, [12] proposes an SDN enabled architecture for the cloud and edge interplay in the context of 5G networks. They consider the virtualization of storage and computing resources of the edge, assuming a C-RAN. This SDN approach enables the global view of the net resources state and a programmable control plane. Thereby, [12] claims that users' quality of experience and network resources usage are improved, as their SDN approach leads to assign dynamically the tasks to the proper EC and CC nodes.

Herein, as in the related work, a collaborative edge-cloud computing platform is considered for IoT data analytics. However, unlike in those works, herein we describe the ML algorithms that permit to analyze big IoT data with different latency requirements at the collaborative edge-cloud platform. The features and requirements of these ML algorithms definitely determine the requirements of the collaborative edge-cloud computing architecture and their key actors, i.e. SDN and resource virtualization controllers. Also, we highlight how SDN and the virtualization of computing and storage resources at the edge and the cloud are fundamental for the proper performance of the ML algorithms. That is, to circumvent the impairments posed by the dynamic state of the network and to adapt to the available computing and storage resources both at the CC and the EC platforms. Moreover, leveraging the virtualization of the computing and storage resources, the proposed architecture provides the flexibility to deploy the ML algorithms both at the network edge and at the cloud.

III. Collaborative Edge-Cloud computing: system model and scenarios

The aim of this section is twofold. First, the system model of the proposed collaborative edge-cloud computing platform is described. Second, several scenarios for IoT data analytics are introduced, within the context of that platform.

In Fig. 1 we display the system model of the collaborative Edge-Cloud Computing platform. The system model is composed of three layers. The top layer is called backend cloud and it is composed of data centers or CC clusters with large computing and storage resources, whose main aim is to perform data analytics of large volumes of data in delay-tolerant applications. We consider that each of those data centers are geographically distributed. Moreover, the backend cloud contains the SDN controller, the VIM and the MANO controller. They have a global view about the state of the virtual computing, storage and communications resources of the overall system. Thereby, they are the cornerstone for the efficient allocation of resources required by the IoT applications at hand, see section IV for further details.

The middle layer in Fig. 1 transports all the data between the IoT devices and the backend cloud and is so-called the core network. It consists of networking nodes such as metro aggregation gateways, routers, switches and firewalls. It is assumed that all of them allow the VIM manager to virtualize their resources and a logical control of their functions through software interfaces, which is driven by the SDN controller.

The bottom layer is called edge network. Its main components are the IoT devices and a set of EC nodes to analyze the IoT data closer to their origin, which leads to reduce the latency and the backhaul network load. The EC nodes are virtual machines (VM) either within the EC server or other access network node, as it is described below in more detail. The IoT devices have sensing, computing and communication capabilities and they are the sources of the IoT data. Due to the heterogeneity of IoT devices' communication interfaces, we consider that the edge network corresponds to a set of heterogeneous access networks. For instance, the IoT devices can be within a C-RAN in 5G networks. In this case, they are within the coverage area of a Remote Radio Head (RRH), which communicates via a fronthaul link with the base band unit (BBU). The BBU is a pool of computing and storage resources that centralizes and implements all or part of the physical and Medium Acces Control (MAC) layers of the RRHs. Moreover, an EC server with computing and storage resources is available for the IoT data analytics. This EC server can be deployed either within the BBU or close to it, as suggested in [13]. Thereby, the EC nodes will be mainly VM within the EC server, but they could be VM within the BBU or even within the RRH if they are equipped with enough computing and storage resources.

Another possibility is that the IoT devices are within the coverage of a small cell or macro-cell in 4G-LTE or 5G systems. In these cases the EC server can reside at the macro base station sites [13]. The IoT devices can be also within

the LAN controlled by a WiFi AP or within a low power Wireless Personal Area Network (LPWAN) based on the IEEE 802.15.4 technology. In those cases the EC server will be an entity between the AP and the core net, which is properly connected via fiber optical links and can be thought as a powerful gateway deployed to give data analytics services within malls, sport centers or museums [13] [11].

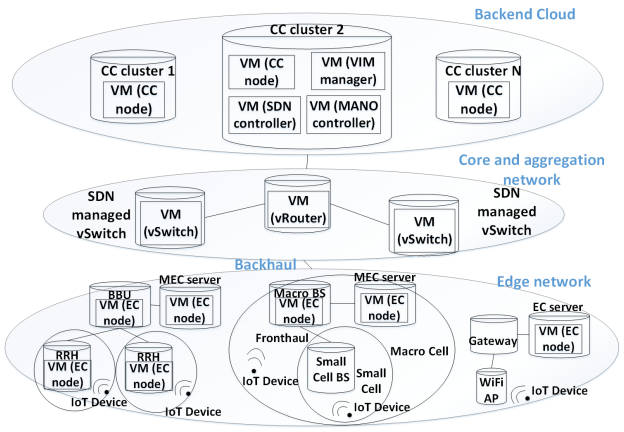

Fig. 1. System model of the collaborative Edge-Cloud Computing platform.

Next, we present three different scenarios for the IoT data analytics in the collaborative edge-cloud of Fig. 1. Then, section IV-C will specify how IoT data analytics in those scenarios can be properly carried out by leveraging the ML, SDN and virtualization tools exposed in sections IV-A and IV-B, respectively.

- In the first scenario, the IoT data analytics are performed in the backend cloud. This situation applies to delay-tolerant applications. The dimensionality of the IoT data is very high both in terms of the observation dimension and the number of samples in the training set. Thereby, the IoT data analytics requires large computing and storage resources.
- In the second scenario, the IoT data analytics are carried out at the network edge. This applies to IoT applications requiring low-latency. In this regard, performing the processing at the backend cloud is not a scalable or viable solution, as the backhaul and in general the network supports a huge load due to the aggregation of geo-distributed Big IoT data. Moreover, the delay for processing all the data at the backend cloud may be too high or inefficient compared to analyzing smaller data sets at the geo-distributed EC servers.
- In the third scenario. Both the backend cloud and the edge network analyze IoT data. This applies in delay-tolerant application, in situations where the aim is to reduce the computational load of the backend cloud or the reduce the load of the backhaul or the core network. In those cases a computational offloading strategy is applied to analyze part of the data at the network edge.

IV. Using ML and SDN/MANO synergies for IoT data analytics in a Collaborative Edge-Cloud

A. Machine Learning in IoT

In this section we present ML tools that address the IoT data analytics features. They will be used in section IV-C, jointly with the techniques of section IV-B, to explain how to analyze the IoT data in the scenarios depicted in section III. First, it is important to highlight that many ML problems can be expressed in the form of the optimization problem stated in (1), e.g. linear regression, logistic regression, support vector machines (SVM), the sparse PCA or neural nets, see [14] [3].

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \sum_{i=1}^{n} f_i(\mathbf{w}) + h(\mathbf{w}) \tag{1}$$

where $h : \mathbb{R}^d \to \mathbb{R}$ is usually a convex, non-smooth function used to impose constraints or enforce desired structures on the solution, e.g. sparsity; $f(\mathbf{w}) \triangleq \sum_{i=1}^{n} f_i(\mathbf{w})$, $f_i : \mathbb{R}^d \to \mathbb{R}$ are rather general cost functions, i.e. they can be convex or non-convex, though it is usually assumed that they are smooth [14] [3]. Moreover, d is the number of features and n the number of training samples. In the context of IoT n is related to the number of samples that each IoT device takes, whereas d is related to the number of IoT devices.

High dimensionality is one of the most remarkable features of IoT data, i.e. both n and d in (1) can be large. This leads to a large-scale optimization, which poses serious challenges to the conventional algorithms that solve convex optimization problems, i.e. interior point methods. In the high dimensional setting, these methods incur in a high computational delay or slow convergence and even they may not converge, see e.g. [15, Table 1] for the LASSO. Thereby, several approaches have been proposed recently to address the high dimensional challenge in ML. One of them are randomization methods.

Randomization algorithms

These techniques rely on an iterative procedure to solve (1). At each iteration they sample randomly the gradient or subgradient involved in the optimization procedure, either along the feature dimension or the training samples dimension to simplify the computational cost. Namely, for the sake of simplicity let us assume that all the functions in (1) are smooth, i.e. h is not present. Then, the traditional Gradient Descent (GD) method solves (1) by doing the next iterations $\mathbf{w}_{t+1} = \mathbf{w}_t - h_t \nabla f(\mathbf{w}_t)$, where $h_t > 0$ is a stepsize parameter. However, computing $\nabla f(\mathbf{w}_t)$ is very costly in large d or n scenarios. Thereby, to simplify this cost, at each iteration the randomization algorithms just compute $\nabla f(\mathbf{w}_t)$ at a random subset of the elements. Thereby, the type of randomized coordinate descent (RCD) algorithms, at each iteration generates a uniformly random index i_t within the set $\{1, \ldots, d\}$ and then computes the iteration

$$\mathbf{w}_{t+1} = \mathbf{w}_t - h_t \nabla_{i_t} f(\mathbf{w}_t) \mathbf{e}_{i_t}. \tag{2}$$

Where $\nabla_i f(\mathbf{w}_t)$ is the $i-th$ partial derivative of f and $\mathbf{e}_{i_t}$ the $i-th$ unit standard basis vector in $\mathbb{R}^d$. On the other

hand, the Stochastic Gradient Descent (SGD) method, at each iteration, generates a uniformly random index i_t within the set $\{1, \ldots, n\}$ and then computes the iteration

$$\mathbf{w}_{t+1} = \mathbf{w}_t - h_t \nabla f_{i_t}(\mathbf{w}_t). \tag{3}$$

That is, SGD just picks one of the functions f_{i_t} within the summation $f(\mathbf{w}) = \sum_{i=1}^{n} f_i(\mathbf{w})$ in (1). Both RCD and SGD have been the benchmark of randomization algorithms, and variants of them have been proposed to improve their performance. Focusing on the SGD, it is known that SGD performs fast iterations but it needs more iterations than the GD to converge. If the application at hand requires low accuracy, then SGD is a good option. Otherwise, variants of the SGD improve its performance at each iteration by reducing the variance in the estimation of the gradient, e.g. the Stochastic Variance Reduced Gradient (SVRG), see [3]. The SVRG consists of two nested loops. In the outer loop the gradient of the entire function is computed using the current value of the parameter vector $\mathbf{w}_t$, i.e. $\nabla f(\mathbf{w}_t) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\mathbf{w}_t)$. In the inner loop, m fast stochastic iterations are done to estimate the gradient and update $\mathbf{w}_t$, by picking at each iteration a random index $i \in \{1, \ldots, n\}$ and performing the operation $\mathbf{w} = \mathbf{w} - h(\nabla f_i(\mathbf{w}) - \nabla f_i(\mathbf{w}_t) + \nabla f(\mathbf{w}_t))$.

Distributed ML algorithms

The randomization methods presented above assume that all the data set is available at a single computing node. However, in big data settings and when the data source is geo-distributed, e.g. in IoT, storing and computing all the data in a single node can be inefficient or even infeasible. Thereby, a more scalable and efficient solution is to have the data set distributed among a set of storage and computing nodes. This leads to distributed ML algorithms, where the computational burden at each node is reduced drastically. They require iterations based on local computation, communication among nodes and an update step. The communication among computing nodes can take more time than the communication among the processor and the memory in a single node. Thereby, designing distributed ML methods that require few and low-latency communication rounds among nodes is a desirable feature to guarantee the convergence of the algorithm in a reasonable time. Distributed versions of the randomization methods introduced above have been presented in the State-of-the-Art (SoA), e.g. distributed SVRG algorithms see [3]. Namely, [3] presents a distributed SVRG method, called Federated (FSVRG), that faces the challenge of requiring few communications rounds among nodes. It also takes into account that the size of the data set can be different at each computing node and that data can be drawn from a different probability distribution. All these features make this algorithm appealing in the IoT context, though it was developed for a rather general framework. As in the centralized SVRG, the FSVRG relies on two nested loops. In the inner loop each computing node do in parallel the fast stochastic iterations mentioned above in the SVRG, by taking into account the available local data set and a weighting matrix accounting for the sparsity structure of the data at each node. The outer loop computes the full gradient, i.e. it is the same than the SVRG, though first it requires the aggregation of the parameter vectors $\mathbf{w}$ obtained at each node in the inner loop. This aggregation incorporates a weight to account for the data set size at each node and the sparsity pattern.

An important feature of distributed ML methods is whether they are synchronous or asynchronous. As an exemplification let us consider the popular Alternating Direction Method of Multipliers (ADMM) algorithm. Synchronous and asynchronous versions of the ADMM have been presented in the SoA for the distributed setting [14]. For simplicity let us assume a star topology of computing nodes. Then, distributed methods rely on a local computation at the worker nodes followed by a communication of the results to a master node, which updates a global parameter with this information. Finally, the master node broadcasts the update to the nodes. For instance, let us assume N workers and the reformulation of (1) with a consensus constraint

$$\min_{\mathbf{w}_0, \mathbf{w}_i \in \mathbb{R}^d, \mathbf{w}_i = \mathbf{w}_0} \quad \sum_{i=1}^{N} f_i(\mathbf{w}_i) + h(\mathbf{w}_0) \tag{4}$$

Then, in the synchronous distributed ADMM (SD-ADMM) the workers optimize the augmented Lagrangian function $\mathcal{L}$ related to (4) respect the pairs $(\mathbf{w}_i, \boldsymbol{\lambda}_i)$, where $\boldsymbol{\lambda}_i$ is a Lagrange multiplier. After receiving $(\mathbf{w}_i, \boldsymbol{\lambda}_i)$ the master optimizes $\mathcal{L}$ respect $(\mathbf{w}_0$. Then, the master broadcasts $\mathbf{w}_0$. Thereby (1) is solved in a distributed way by the SD-ADMM. Therefore, in synchronous methods the update step at the master can not start until all the worker nodes finish their local computations and the master receives all of them. This constraint seems reasonable to guarantee the optimal performance of the algorithm and it has been widely assumed [3] [14]. However, when scaling up the distributed ML methods, the speed of the algorithms is limited by the slowest nodes, i.e. the ones with worse computation and communication delays, which leads to inefficient use of computation resources as some nodes are idle. To circumvent this issue asynchronous algorithms allow the master to start the update operation with the computations of only a subset of workers. The flip side is that the parameters controlling the asynchrony must be handled with care, as slight modifications lead to different convergence conditions or to destroy the convergence [14]. For instance, in the asynchronous distributed ADMM (AD-ADMM) of [14] the update at the master can start only with the information received from some nodes, though they consider that the information of all the nodes must arrive before a maximum number of iterations τ. Actually, they provide a relation between the design parameter of the algorithm and τ that guarantees the convergence of the AD-ADMM to the Karush-Kuhn-Tucker (KKT) points.

Online ML algorithms

In the previous sections the ML algorithms assume that all the training set is available, i.e. a batch processing mode. However, when the underlying scenario is dynamic or low latency is required, the ML algorithm has only access to

new data points in a sequential manner. In this case, the ML methods learn sequentially as new samples arrive and also using the past observations. Thereby, they are so-called online learning methods. To assess their performance they are compared, through a metric called regret, to a clairvoyant offline method that has access beforehand to the learning information for all the operation times $t = 1, \ldots, T$. Namely, let $\mathbf{x}_t$ be the variable to be learnt, $\mathbf{f}_t$ the training set observations. The function $l_t(\mathbf{x}_t, \mathbf{f}_t)$ measures the loss in choosing $\mathbf{x}_t$ given the training set $\mathbf{f}_t$. In the offline methods $\mathbf{f}_t$ is known beforehand and thus they obtain an optimal vector for all t, i.e. $\mathbf{x}^* = \text{argmin}_x \sum_{t=1}^{T} l_t(\mathbf{x}, \mathbf{f}_t)$. On the contrary, in online learning to decide $\mathbf{x}_t$ only $\mathbf{f}_t, \forall u < t$ is available and $l_t(\mathbf{x}_t, \mathbf{f}_t)$ will implicitly account for a prediction error. Thereby, the regret $\mathcal{R}$ measures over time the difference between the losses obtained by the online learner and the optimal offline learner, i.e. $\mathcal{R} = \sum_{t=1}^{T} l_t(\mathbf{x}_t, \mathbf{f}_t) - \sum_{t=1}^{T} l_t(\mathbf{x}^*, \mathbf{f}_t)$. For a centralized setting, several online learning methods have been proposed in the SoA to control the regret growth. They are special cases of a strategy known as follow the regularized leader see [16]. Also for the distributed setting online versions of the distributed ML have been proposed to control the regret growth rate, e.g. the saddle point method in [16].

B. SDN, MANO and VIM

SDN and Network Functions Virtualization (NFV) belong to the core set of technologies proposed for enabling the 5G vision [17]. The former achieves centralized control of the network through a SDN controller, which takes care of the control plane of networking devices. The latter detaches the functionality from the hardware, realizing virtual network functions (VNF), e.g. routing or switching, via software. VNFs, as EC nodes, run on top Compute Nodes (CN) at data centers controlled by a VIM.

In an IoT setup, the network is expected to be flooded by great amounts of uncategorized, often redundant, sensor data. Usually, traffic generated by a geographically distributed layer of sensors is aggregated and preprocessed at IoT Gateways, which then forward all the resulting data to a cloud server that attempts to extract meaningful information applying ML. Despite yielding optimal results, this *cloud* approach can be considered too costly for delay-sensitive or real-time IoT applications. This is partly due to the overhead imposed onto the network backhaul. A viable alternative for extracting meaningful information from massive IoT data involves the decomposition of the problem into smaller sub-problems, and the parallelization of the processing across the edge. One of the advantages of this strategy relates to the proximity of CN to the client, potentially yielding lower delay. To realize this strategy, low-latency, secure, and reliable communication should exist among nodes composing the edge cloud, which is provided by the SDN controller via virtual tenant networks and NFV [18]. Moreover, ML services in the form of virtual applications are deployed throughout the edge cloud.

In order to guarantee computing and communications requirements demanded by ML services, a centralized MANO controller is fed from metrics exposed by the SDN Controller and VIMs via northbound interfaces (NBI), like RESTful Application Programming Interfaces (API). Networking metrics are gathered by the SDN Controller via southbound interfaces (SBI) using protocols such as OpenFlow [19]. Similarly VIMs such as OpenStack gather and expose details about CN in the platform. This information is combined at the MANO controller to describe the current state of the complete platform, and then to orchestrate the spin off of VMs at network locations complying with the respective ML services' network requirements. Widely used MANO controllers such as OSM [20], provide ample support for a variety of standard APIs, allowing a single MANO to realize ML services throughout data centers managed by different VIMs.

C. IoT data analytics leveraging ML, SDN, MANO and VIM

This section deals with the IoT data analytics for the three scenarios defined in section III within the context of Fig. 1.

Scenario 1: IoT data analytics at the Backend cloud

This case is envisaged for delay-tolerant applications and all the data is analyzed at the backend cloud. The aggregation of geo-distributed IoT data leads to a high dimensional setting, which requires large computational and storage resources at the backend cloud, and increases notably the backhaul and core network loads. To face those issues, the MANO controller makes use of VIM's APIs to determine the availability of compute resources for the selected ML service and schedule the spin off of the corresponding ML worker. To face the huge network load, the SDN Controller is leveraged, which modifies network devices' forwarding decisions in a way that only the less congested paths towards the ML worker at the backend cloud are selected. Depending on the available computing and storage resources, and the dimensionality of the data, the MANO controller decides whether the ML service would comprise a single, or several computing nodes. In the single computing node case, randomization is the most convenient among the ML techniques proposed in section IV-A to face high dimensional data analytics in a centralized setting. The requirement for the SDN controller is to guarantee a reliable communication path between the edge and the cloud to avoid a performance degradation due to missing samples in the ML training set. Distributed ML techniques are the most convenient methods when the MANO controller decides to analyze the data using several computing nodes. To allow the convergence of the distributed ML methods in a feasible operation time, the SDN controller must ensure a low latency and reliable communication among the computing nodes (see section IV-A). Such communication paths are achieved thanks to the global view of network and compute resources exposed by the SDN controller and the VIM NBIs, respectively. Compute nodes can be within the same cloud computing cluster or geo-graphically distributed in several cloud computing clusters, see Fig. 1.

Scenario 2: IoT data analytics at the Network Edge

In this scenario the data analytics service is deployed at the network edge to address the requirements of low-latency

applications. Thereby, online ML is the perfect fit for this scenario, as in those algorithms the learning process is done sequentially as new data samples arrive to address a dynamic scenario with low latency requirements, see section IV-A. In turn, to address the geo-distributed nature of IoT data, distributed online learning algorithms are proposed. To this end, the centralized view of computing resources allows the VIM and MANO to deploy EC nodes as Virtual Machines (VM) close to the IoT devices, see Fig. 1. For instance, N VM can be deployed at N different EC servers of N RANs. The orchestration of multiple geographically distributed EC nodes requires fast and reliable communication among them. This constraint is not only imposed by delay-sensitive applications, but also by distributed online ML services, as low latency iterations among EC nodes are needed to guarantee the convergence and to avoid the performance degradation in terms of the regret metric see section IV-A. To this end, the SDN Controller enforces priority for the distributed ML service traffic between EC nodes. In cases where the data is localized at a given RAN or when lower latency is required, one can apply an online learning method at a single EC node.

Scenario 3: IoT data analytics at the Backend and the network Edge

In this scenario, in delay-tolerant applications, part of the data is analyzed at the network edge and the rest at the cloud. The aim is to alleviate the communication load of the backhaul and the computational load at the backend cloud. The ML algorithms for the backend cloud are the same than in scenario 1, i.e. either distributed ML or randomization methods to tackle the high dimensional data, which recall that require low latency and reliable communication paths among computing nodes. For the network edge, we propose to use distributed ML algorithms to tackle the geo-distributed nature of IoT data. The VIM and SDN controller play a central role in this scenario. First, they ensure the low latency and reliable communication required by the distributed ML methods. Second, they provide a global view of the communication and computing resources to external applications (such as the MANO controller) via NBIs. Thereby, they lead to identify whether the backhaul load or backend computational load are too high, and to create EC nodes at the network edge to reduce this burden. The offloading of ML services tasks from the backend to the edge not only alleviates computation and network loads, but also could be part of a greater energy or OPEX saving strategies. In this case, the offloading could be based on estimated energy consumption or estimated monetary cost when using a third-party backend cloud infrastructure.

V. Conclusions

This paper has proposed a collaborative edge-cloud computing architecture to face the challenging requirements of IoT data analytics. The fundamental tools for this platform have been presented. Firstly, ML frameworks which fulfill the needs of IoT data analytics. The second tool is the centralized and global management of virtual networking and computing resources. Three different scenarios for the collaborative edge-cloud platform have been defined, which highlight the synergies an interplay of the tools mentioned above.

Acknowledgment

This work has been funded by the following projects: SEMIOTICS (780315), SPOT5G (TEC2017-87456-P) and by the Generalitat de Catalunya under grant 2017 SGR 891.

References

[1] S. K. Sharma and X. Wang, "Live data analytics with collaborative edge and cloud processing in wireless iot networks," *IEEE Access*, vol. 5, pp. 4621–4635, April 2017.

[2] Q. W. et al., "Cognitive internet of things: A new paradigm beyond connection," *IEEE Internet of Things*, vol. 1, pp. 129–143, April 2014.

[3] J. Konecny, B. H. McMahan, D. Ramage, and P. Richtrik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv:1610.02527*, 2016.

[4] J. W. et al., "Cloud robotics: Current status and open issues," *IEEE Access*, pp. 2797–2807, June 2016.

[5] M. O. et al., "Coalition games for spatio-temporal big data in internet of vehicles environment: A comparative analysis," *IEEE Internet of Things Journal*, vol. 2, pp. 310–320, August 2015.

[6] J. Hu and A. Vasilakos, "Energy big data analytics and security: Challenges and opportunities," *IEEE Trans. Smart Grids*, vol. 7, pp. 2423–2436, September 2016.

[7] C. A. et al., "A cloud to the ground: The new frontier of intelligent and autonomous networks of things," *IEEE Communications Magazine*, vol. 54, pp. 14–20, December 2016.

[8] F. Bonomi and R. M. et al., "Fog computing and its role in the internet of things," in *in Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, 2012.

[9] M. Raho, A. Spyridakis, M. Paolino, and D. Raho, "Kvm, xen and docker: A performance analysis for arm-based nfv and cloud computing," in *IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), 2015*, pp. 1–8, IEEE, 2015.

[10] X. Masip-Bruin, E. Marin-Tordera, G. Tashakor, A. Jukan, and R. Guang-Jie, "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Comm.*, pp. 120–128, October 2016.

[11] H. Guo, J. Liu, and H. Qin, "Collaborative mobile edge computation offloading for IoT over fiber-wireless networks," *IEEE Network*, pp. 66–71, February 2018.

[12] Y. Peng, Z. Ning, B. Yuanguo, Y. Li, and S. Xuemin, "Catalyzing cloud-fog interoperation in 5G wireless networks: an SDN approach," *IEEE Network*, pp. 14–20, October 2017.

[13] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Comm. surveys and tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[14] T. Chang, M. Hong, W. Liao, and X. Wang, "Asynchronous distributed ADMM for large-scale optimization Part I: algorithm and convergence analysis," *IEEE Trans. on Signal Proc.*, vol. 64, pp. 3118–3130, June 2016.

[15] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data," *IEEE Signal Proc. Mag.*, pp. 32–43, September 2014.

[16] A. Koppel, F. Jakubiec, and A. Ribeiro, "A saddle point algorithm for networked online convex optimization," *IEEE Trans. Signal Proc.*, vol. 63, pp. 5149–5164, October 2015.

[17] Bell Canada et al., "Network functions virtualisation (nfv): Network operator perspectives on nfv priorities for 5g."

[18] A. Mayoral, R. Vilalta, R. Casellas, R. Martinez, and R. Munoz, "Multi-tenant 5g network slicing architecture with dynamic deployment of virtualized tenant management and orchestration (mano) instances," in *Proc. 42nd European Conference on Optical Communication ECOC 2016*, pp. 1–3, VDE, 2016.

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[20] ETSI, OSM, "Open Source MANO (OSM)," 2018.