Collaborative Execution of Distributed Mobile and IoT Applications Running at the Edge

Mohammad M. Shurman[1]and Maha K.Aljarah[2]
[1]Jordan University of Science and Technology/Network Engineering and Security Department, Irbid, Jordan
[2]Jordan University of Science and Technology/Computer Engineering Department, Irbid, Jordan
alshurman@just.edu.jo, mkaljarah16@cit.just.edu.jo

Abstract–**The use of mobile and IoT applications is growing rapidly and gaining huge interests in research and commercial investment to meet future visions, these applications have common demands to support their needs of low latency, geographically distributed services and location based awareness. These demands contrast with services provided by the centralized distant cloud datacenters. Edge computing paradigm has emerged as a more applicable solution for these applications providing an extension of cloud computing storage and network resources placed in a geographically distributed manner at the edge of the network closer to mobiles and IoT devices. In this paper, we focus on exploiting edge resources to the maximum using a collaborative approach of distributing modules of pre-partitioned applications between edge resources, aiming to reduce amount of traffic travelling through the core network to the cloud and reduce amount of delays in order to provide an efficient services delivery and a better user experience. We tested our approach using iFogSim toolkit, the obtained results show reduction in network usage and total delays compared to the baseline approach of placement, yielding in a better throughput and better utilization of edge resources.**

Keywords: Edge Computing, Fog Computing, Mobile Edge Computing (MEC), IoT, Resource management.

I. INTRODUCTION

Edge computing paradigm was introduced as a response to the rapid increase in mobile and IoT applications. The need of a new platform to support such applications demands of computational and storage resource in a close proximity to end devices in a geographically distributed manner was an urge. Unlike the current cloud with centralized infrastructure that suits regular pc's application rather than mobile and IoT applications.

Edge computing (the term we use in this paper) or similar platforms also known as fog computing, cloudlet, Mobile Edge Computing (MEC) or Micro Data Center (MDC). Fog computing refer to the cloud closer to the ground, the paradigm was introduced by Cisco [1] to meet the critical needs of IoT services for a platform that support mobility, highly geographically distributed, location based and real time applications. These applications along with mobile applications produce tremendous amount of traffic, which if directed all the way through the network to the cloud, will result in network congestion and unbearable delays. Cloudlet was defined in [2] as the computing resources located closer to mobile devices to support their need of intensive computing power. Mobile Edge Computing was considered by [3] as a key technology enabling for 5G technology along with Software Defined Network (SDN) and Network Function Virtualization (NFV).

Edge computing paradigm was inspired from the idea of the Content Delivery Networks (CDNs) that are used to cache and pre-fetch heavily accessed web content to edge nodes closer to the end devices. On the other hand, Edge computing extends this concept for the cloud infrastructure, where an edge node or a cloudlet can cache data or run code instead of the cloud, it could also be used as a hierarchy multilevel bridging the distance between the cloud and end devices [4]. Predicting and caching users' desirable content, based on their location in a proactive manner before requesting are considered in [5] as key characteristic for Fog servers to predict location based services mostly needed for mobile users, and yield in a better utilization of fog resources and better user experience.

Edge computing, clearly provides many solutions and benefits for mobile and IoT applications that need powerful computing resources and low latency responses, such as gaming, video streaming, augmented reality, speech and face recognition applications.On the other hand, Edge computing is still a hot area for research especially for issues like edge networking, security, privacy, provisioning, and resource management.

In this paper, we introduce a collaborative approach of executing modules resulted from pre-partitioned applications in a distributed manner between edge resources. Our approach aims to increase the utilization of edge resources, by allowing collaborative modules execution among neighboring edge nodes. On one hand, reduces the delay compared to executing modules on the cloud and reduce amount of traffic travelling through the network to the cloud. On the other hand, provides a better experience to the user through a faster service delivery and traffic reduction to the core network.

The paper organization is as follows, in the following section, a comprehensive overview on the management and provisioning of edge resources is provided. Section 3 details the proposed protocol for collaborative edge computing. Section 4 summarizes the simulation environment and results in addition to the significance of the proposed protocol. Our conclusion is presented in Section 5.

978-1-5386-0872-2/17/$31.00 ©2017 IEEE

II. Related Work

Management and provisioning of edge resources have been considered by many researchers. Wenlu Hu et al. [6] examine the impact of offloading computation from mobile devices to edge node compared to the current mainstream of offloading to the cloud, they also quantified the improvement in response time and battery life of mobile devices that use diverse interactive applications with intensive computation and memory need.They examined offloading pre-partitioned and dynamically partitioned applications through Wi-Fi and 4G LTE networks. Results show that offloading greatly reduces energy consumption on mobile devices regardless to the site of offloading.On the other hand, response time is greatly affected by offloading site degrading with offloading site distance increase.They also shows that offloading to edge node is less affected with application interactivity level, compared to cloud offloading.

Azam and Huh [7] present framework for resource management, they predict user behavior against resource utilization to prevent resource wastage and guarantee fair resources allocation. They classify users in three categories: new users, users with high probability of resource relinquish and users with low probability of resources relinquish, taking into account type and price of services.

Resource provisioning model proposed by [8] aims to maximize utilization of fog resources. They introduced a hierarchal fog framework with two levels of resources control and orchestration in the cloud and fog. The main idea in their framework is that fog colony consists of multiple fog cells, IoT devices, and one fog control node. So, the first layer is multiple fog colonies, the second layer is another fog control node orchestrating and connecting all fog colonies in the first layer, and the third layer is the fog-cloud control middleware responsible for the services running on the cloud. Control nodes perform provisioning plan to manage underlying resources and achieve the maximum possible utilization, they also insure delivery of requested services or propagate it to a higher control level.

Kapsalis et al. [9] proposed fog platform architecture with 4 layers: bottom layer which is the device layer containing IoT devices connected with above layer of gateways named hub layer, after that the fog layer with fog servers and a fog broker entity responsible for resource management, and the last layer is the cloud layer. They also proposed a workload balancer procedure executed by the fog broker to insure balanced utilization of fog resources based on three characteristics: current utilization, latency and battery life of mobile devices. Message Queuing Telemetry Transport (MQTT) protocol [10] is adapted for communication between devices, were messages exchanged contain the scripted code need to be executed along with data section and metadata section specifying some characteristics of the task.

Another work adapted MQTT protocol to communicate in their proposed Software Defined Networks (SDN)-based fog-computing platform [11].They customized edge switches named fog nodesto be integrated with SDN controller and an MQTT broker.Fog node works as a collecting point receiving MQTT massages from IoT devices, publishing their data with a certain topic (type), and then the fog node forwards the message to devices asking for the same topic. They also suggest performing analytics inside the SDN controller as it is the central node that all data pass through.

III. Proposed Approach

Since our objective is to increase resource utilization through a collaborative approach, we propose edge computing architecture platform that facilitates communication between edge resources.We also proposed an approach to distribute pre-partitioned application modules between edge resources.

A. Architecture

1) Edge computing platform

Figure.1 illustrates the structure of overall Edge computing platform, consisting of three layers End layer, Edge layer and Cloud layer. End layer is the lowest layer in this platform consists of Mobile and IoT devices with poor resources unable to execute tasks with intensive computation. Edge layer is the middle layer contains a geographically distributed edge nodes with storage, computation and communication resources in a close proximity of the End layer devices, Edge layer is independently capable of performing tasks, running predefined applications on behave of end layer devices with low latency and efficient streaming without assistance of the cloud layer. Cloud layer is the last and the farthest layer in the platform that we need to minimize communication with other layers to prevent going through the core network where severe delays will be imposed. Hence, traffic amount reduction to core network saves bandwidth and reduces congestion. Cloud resources could be used in some cases like long-term storage of data and in-depth data analysis or in case edge nodes resources are not enough to execute some tasks, edge node can forward tasks with heavy processing demands to the cloud as its final choice.

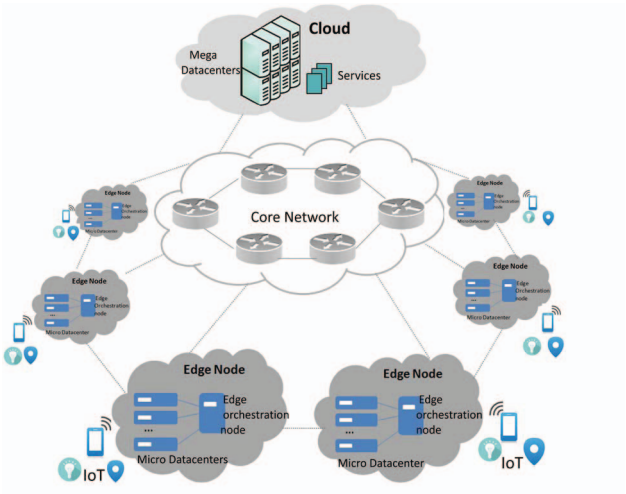

Fig. 1: Edge computing platform

2) Edge Node Structure

Each edge node has its own computation, storage, and communication resources. Edge resources could be heter-

ogynous and distributed in a hierarchy approach, bridging the distance to the cloud at some extent, where less powerful resources are placed closer to end devices. The farthest we go from end devices, the more powerful the resources become. In our proposed approach, we distribute the processing of tasks such that tasks with less computation requirement are going to be executed close to end devices and tasks with more complex computation requirements are going to be executed in an upper level of resources.

Edge nodes might vary in the resources they have, based on type of services they intended to provide. Edge nodes that intend to provide computational intensive services like gaming and augmented reality should be provided with richer resources than nodes that provide light services. This design increases the potential need for our collaborative approach, for example if a node with light resources get overwhelmed with task requests and has a neighboring node with rich resources and low utilization ratio, then some tasks could be forwarded to it rather than forwarding them to the cloud. In addition, systems that need to be deployed in wide regions like a smart driving assistance can benefit from our proposed approach. Several edge nodes provide the same service might be distributed to cover a wide region, where amount of traffic varies at different areas and different times. So, using our collaborative approach between neighboring nodes we can increase utilization of edge nodes resources and prevent interacting with the cloud, as we will prove later in section 4.

Each edge node also has an Edge orchestration node, which is the main element in the edge node, it maintains information about each task running on which host, the current state of computation and storage resources, and it has information about the connection state to the neighboring edge nodes and the cloud.Orchestration node is responsible for resource management and taking decisions whether to execute tasks on the node resources, on the cloud, or on a neighboring node. Our proposed approach will be discussed with details later on.

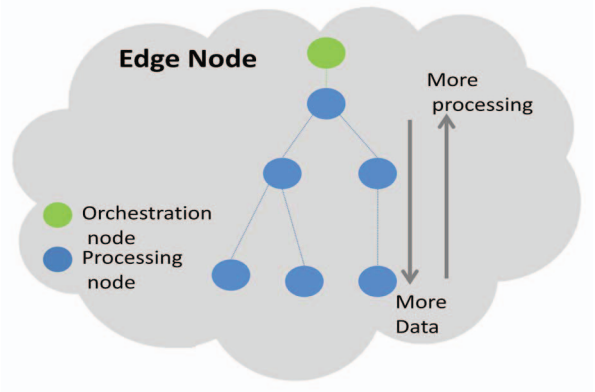

Fig. 2: Edge Node Structure

B. Approach Description

To increase utilization of edge resources, we used an approach that allows collaborative execution of application modules between neighboring edge nodes.

1) Application

An application that is needed to be executed on the edge platform should be pre-partitioned into modules; each module performs certain functionality in the application. Distributed data flow model is used to represent the application in a directed graph form, where modules are represented with vertices and directed edges represent dependencies and flow of data between modules.

2) Module placement

Edge orchestration node receives the pre-partitioned application as a set of modules. Next, it should distribute these modules on the available computing resources based on their need of computational and storage resources.

The baseline module placement approach used in iFogSim depends on spanning the landscape from bottom to top in the same edge node looking for suitable resource to place each module. If no resource found fitting to module requirements, it will be placed on the cloud. In [8], a system model was presented that consists of utilization and latency metrics for resources inside the same node but does not take into account utilization and latency of neighboring nodes. In our proposed approach, we span the platform horizontally looking for suitable resources in neighboring nodes, leaving the option of placing the module on the cloud as the final choice.

Our proposed approach for module placement has the following steps:

1. Orchestration node maintains leaf to root paths of hosts inside the edge node along with the current CPU utilization for each host. Orchestration node represents hosts inside the edge node by a tree structure, based on their computing power, where the least powerful hosts are the leaves and the most powerful host is the root. A leaf to root path contains the possible hosts for executing the module, orchestration node traverses each path from leaf to root as it needs to try running the module on less powerful hosts closer to end devices before moving to upper level of resources.
2. Orchestration node sorts the sequence of executing modules inside the edge node based on dependencies between them.
3. For each module, the orchestration node traverses leaf to root paths of hosts checking the CPU load results from executing the module on this host and compares it with CPU load that this host can sustain.
4. If the host is capable of executing this module, then the orchestration node updates its information of the current CPU load of this host and its information about the module.
5. If hosts in the lowest level of resources in the path were unable to execute the module, then the orchestration node checks the next level of hosts.
6. Orchestration node will check all levels of resources to execute the module inside the node before considering executing it in neighbor node or in the cloud. The choice between neighbor nodes and the cloud

depends on the estimated delay to reach each of them.

7. If the orchestration node was not able to execute the module inside the same node due to module's excessive need of computing power unavailable at the current state of hosts inside the node then the orchestration node will sort its neighboring nodes and the cloud according to the estimated latency to reach each of them.
8. Orchestration node starts with the neighbor node that has the minimum estimated delay and sends it a request for executing the module.
9. In our approach, we use one level of cooperation between nodes that if the neighboring orchestration node receives a module from other node then it will check the ability of executing it only on the highest level of resources it has, and based on that it will respond to the requesting node.
10. When the original orchestration node receives a positive response from a neighboring node it will start forwarding all data needed to be processed by this module to this neighboring node.
11. The original orchestration node will check all neighboring nodes that could be reached with less latency compared to the cloud, if it doesn't get a positive response for any neighboring then it will finally forward the module to the cloud.

IV. Simulation and Results

To evaluate the proposed platform along with our collaborative placement approach against baseline approach we used iFogSim [12] toolkit, which is a new powerful simulator build on the top of CloudSim [13] to provide functionalities that support fog computing environments.

We used the first case study provided by the simulator itself [12] for our experiments; it is an implementation for a game application named EEG Tractor Beam Game an online game where each player wears an EEG headset with a sensor on it sending signals about the user's brain state to an application on the user's smart phone.

The application shows all players surrounding an object, each player will try to pull the object toward him but the movement of the object depends on the concentration level of each player; the player with highest concentration level will succeed to pull the object in his direction. The application is partitioned to three modules: client, concentration calculator, and coordinator. Client module receives the EEG sensor readings, performs simple preprocessing on the readings, and then forwards it to the concentration calculator module, which calculates the concentration level of the player, and then it sends it back to the client module. Next, the client module sends it to the display actuator.The concentration calculator module also sends its result periodically to the coordinator module which responsible for sharing the global game state with other clients in order to update their displayed global state of the game. Fig.3 shows dependencies and data flow between the three modules in the application [12].

Task specifications and data transfered between modules to be processed on the receiving module are called tuples, each tuple has a type for example in our case tuples that sensors send to client module containing EEG sensor readings have the type EEG, tuples sent from client to concentration module after performing preprocessing operations on sensors readings have the type sensor and so on as seen in fig.3. Each tuple type has it's characteristics like the size of the task to be executed specified in unit Million Instruction (MI), size of data before execution in byte and other characteristics needed to perform the task.

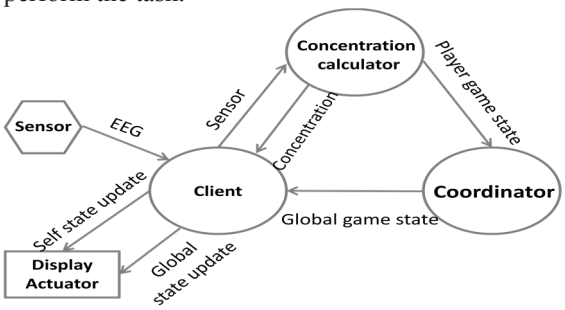

Fig.3: EEG Tractor Beam Game modules [12]

A) Simulation configuration

The platform we built and used through the experiments consists of two edge nodes, each edge node has one orchestration node, one edge server and number of mobile devices varies through experiments from 10 devices to 100 devices, each mobile is connected with a sensor and an actuator. Fig.4-a shows our proposed Edge computing platform with the orchestration nodes and delays between each component implemented using iFogSim, Fig.4-b the platform used without orchestration node, to be compared with our platform.

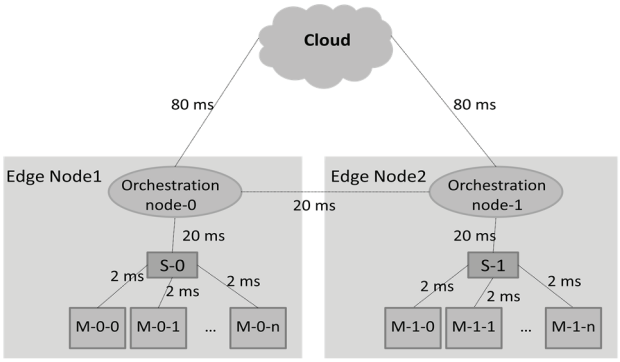

Fig.4-a: Edge computing platform withOrchestration nodes

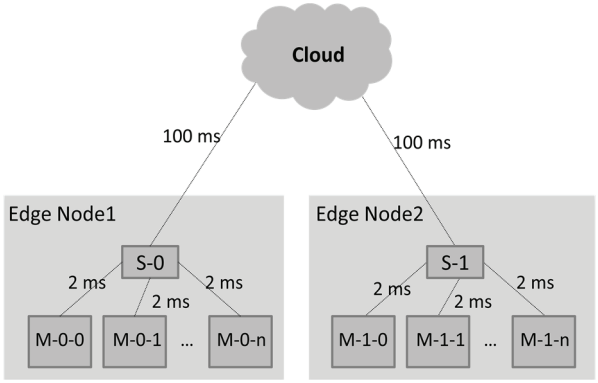

Fig.4-b: Edge computing platform withoutOrchestration nodes

To compare our placement approach and the placement approach of iFogSim we unified configurations and characteristics for devices in both platforms.

Metrics used for evaluation:

1- Total delay: We measured the total delay needed to execute all tuples, this is include network delay as these tuples need to travel from module to module on different hosts and different nodes. The simulator measurement of time is in abstract time unit.
2- Throughput: Total number of tuples executed per time unit.
3- Network usage: amount of data carried by tuples that travel through the network and occupy it for certain amount of time.

To be consistent we measure these metrics for the same number of tuples (tasks) executed in both our implementations and the baseline implementation, then we measure on larger number of tuples when increasing number of devices in edge nodes, as the more devices involved will produce more tuples to be executed through the application.

Collaborative Scenario: all mobile devices have the same capability and could only execute client module, edge servers have different capabilities so one could execute the concentration module and the other cannot. In the baseline placement approach, all entities of this module will be placed on the cloud while in our placement approach the server with sufficient computational power will execute all entities of this module.

Fig.5 shows the improvements that our approach achieved compared to the baseline approach in term of throughput, delay and network usage with variable number of mobile devices in edge nodes.

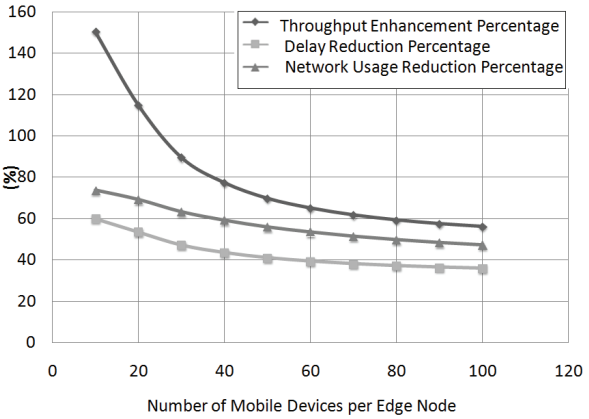

Fig.5: Scenario 1 results

In the configuration for our edge platform, we set the latency from the original node to neighbor node to be less than the latency from the original node to the cloud by 40% as shown in fig.4, and the results give an average reduction in latency around 43.3%.We gain even more reduction in latency because when the traffic is directed to the core of the network it get imposed to the network congestion, as we see in fig.5 reduction in delay is around 60% when there is only 10 devices emanating data to be processed and this ratio degrade when more devices are involved because congestion will increase. In general we can say that reduction in delay when using the neighboring node relatively reflects how much closer the neighbor node is to the original node compared to the cloud.

V. CONCLUSION

Edge computing plays a vital role as a key enabler for the Mobile and IoT-based applications that are gaining increasing interest in research and commercial, edge computing support applications with needs of intensive computational and real time response it also has key role in serving applications with location based awareness.

Our work focus on improving utilization of edge resources using collaborative execution of application between neighboring edge nodes with an architecture that facilitate communication between edge nodes, simulation results shows that adopting our approach results in a better performance for the whole platform, reducing latencies and network congestion.

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, Finland, 2012.

[2] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, 2009.

[3] Y.-C. Hu, M. Patel, D. Sabella, N. Sprecher and V. Young, "Mobile-edge computing-introductory technical white paper,"*White Paper, Mobile-edge Computing (MEC) industry initiative*, 2014.

[4] S.-K. Sharma and X. Wang, "Live Data Analytics With Collaborative Edge and Cloud Processing in Wireless IoT Networks,"*IEEE Access*, vol. 5, 2017.

[5] T.-H. Luan, L. Gao, Z. Li, Y. Xiang, G. We, and L. Sun, "Fog Computing: Focusing on Mobile Users at the Edge," *arXiv preprint technical report,*2015.

[6] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen,P. Pillai and M.Satyanarayanan,"Quantifying the Impact of Edge Computing on Mobile Applications,"*Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems*,Hong Kong,2016.

[7] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," *IEEE International Conference onPervasive Computing and Communication Workshops (PerCom Workshops), USA*,2015.

[8] O. Skarlat, S. Schulte, M.Borkowski and P.Leitner, "Resource provisioning for iot services in the fog,"*IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), 2016*, China,2016.

[9] A. Kapsalis, P. Kasnesis, I.-S. Venieris, D.-I. Kaklamani and Ch.-Z. Patrikakis, "A Cooperative Fog Approach for Effective Workload Balancing,"*IEEE Cloud Computing,* vol. 4, no. 2, 2017.

[10] U. Hunkeler, H.-L. Truong and A.-S. Clark, "MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks,"*3rd international conference on Communication systems software and middleware and workshops, COMSWARE*2008,India, 2008.

[11] Y. Xu, V. Mahendran and Sr. Radhakrishnan,"Towards SDN-based fog computing: MQTTbroker virtualization for effective and reliable delivery,"*IEEE 8th International Conference on Communication Systems and Networks (COMSNETS), India, 2016*.

[12] H.Gupta, A.Dastjerdi, S. Ghosh and R. Buyya, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments," *Software: Practice and Experience*, vol. 47, no. 9, 2017.

[13] R.-N. Calheiros, R. Ranjan, A.Beloglazov, C.-ADe Rose and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,"*Journal of Software: Practice and experience,* vol. 41, issue 1, 2011.