A Cloud-Fog based Architecture for IoT Applications Dedicated to Healthcare

Randa M.Abdelmoneem*†, Abderrahim Benslimane†, Eman Shaaban*, Sherin Abdelhamid‡ and Salma Ghoneim§
* Faculty of Computer Sciences, Ainshams University, Cairo, Egypt, (randa_aboelfatoh,eman.shaaban)@cis.asu.edu.eg
† LIA/CERI, University of Avignon, Avignon, France, abderrahim.benslimane@univ-avignon.fr
‡ School of Computing, Queens University Kingston, Canada,sherin@cs.queensu.ca
§ Department of Informatics Engineering, French University in Egypt, Cairo, Egypt,salma.ghoneim@ufe.edu.eg

***Abstract*—Cloud-Fog computing architectures are new paradigms designed to add advantages to the existing architectures for Internet of Things (IoT). This paper proposes an inter-operable cloud-fog based IoT architecture for healthcare. It describes its architecture, environmental context, and user context. The proposed architecture supports the mobility of the patients as well as the diversity of the medical cases. The features of the individual modules are discussed and the interconnection between the different underlying modules and tiers is explained. Task scheduling and allocation approach is proposed to effectively balance healthcare tasks distribution. The performance evaluation of the proposed approach is presented with different number of tasks and cloud nodes. The simulation results show acceptable results in terms of miss ratio, cost, and latency.**

I. Introduction

IoT dedicated for healthcare aims at improving the quality of life for elder and unwell people by facilitating their movement, helping them in their daily activity routines, besides supporting them in their healthcare treatment plan. Applications of IoT dedicated for healthcare include but are not limited to user's activity monitoring, person's fall detection, abnormality detection and diagnosis, and classification of diseases [1][2][3].

For the worthy implementation of the aforementioned applications and other more, several wearable and environmental sensors are used. This heterogeneity of IoT devices in healthcare domain in accordance with the expected growth in the number of IoT devices poses the challenge for storing and processing this big volume of data [4]. Consequently, this brings up the challenge for the design and implementation of powerful architectures that support these systems. Traditional IoT architectures that are based on cloud computing adds several limitations to the current healthcare systems despite their huge capabilities. Limitations include huge processing and data transmission costs and high overhead degrading network performance. Moreover, extra time connecting to the cloud adds a high latency limitation which is unacceptable for the criticality and time sensitivity nature of the healthcare applications. On the other side, fog computing extends the cloud computing infrastructure to the edge of the environment. This is done by adding an intermediate layer between the IoT devices and the cloud [5][6][7]. This layer is composed of distributed fog nodes such as switches, wireless routers and laptops capable of performing the processing, forwarding the data to other fog nodes or even to the cloud layer. The concept of sharing the computations between the fog and the cloud nodes achieves the QoS trade-offs.

In this paper, we propose an IoT architecture for healthcare applications based on the interoperability between cloud and fog computing so that delay and cost can be reduced and time constraints can be met. The proposed architecture is generic to healthcare applications without being restricted to a specific medical case. The use of fog computing helps achieve better quality of service for the healthcare applications. Critical and time-sensitive tasks are distinguished from delay-tolerant tasks through several factors. Current status of the patients, emergency situations and decisions are best handled. The main contributions of the paper can be summarized as follows: 1) Introducing a generic cloud-fog based architecture for IoT healthcare applications, 2) Classifying different medical and contextual tasks that the architecture can support, and 3) Presenting an implementation of the architecture and describing the scheduling and allocation algorithms that constitute the core working unit of the architecture.

The remainder of the paper is organized as follows. Section II provides an overview of IoT architectures for healthcare. The proposed architecture is introduced in Section III. Task scheduling and allocation approach is proposed in Section IV. The implementation and performance evaluation are presented in Section V. Finally, Section VI concludes the paper.

II. Overview of IoT Architectures for Healthcare

Several IoT cloud-based architectures for healthcare have been proposed. Traditional infrastructures involve a set of cloud resources accessed remotely by the users. The authors in [8] propose a smart healthcare system called Health-CPS which is intended for aiding in healthcare application services such as statistical monitoring, knowledge and prediction. In [9], a system is introduced to monitor patients with chronic diseases using a mobile device and a server. In [10], the authors propose a remote monitoring and processing system based on bio-potential signals from the patient. The architecture is composed of IoT body sensing devices, the gateway and the cloud. The work in [11] proposes the Ubiquitous Healthcare System (UHS) architecture intended for the elderly. The proposed system works on several factors aiming to help the elderly in terms of providing home care services, emergency assistance, and remote medical services. In [12], a cloud-based healthcare system is built to monitor the status of the patient through the measurement of ECG signals. A feedback medication modification is returned to

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

the patient through a mobile device. Some authors proposed fog computing based healthcare architectures. A fog based system for e-health is proposed in [13]. The system aims for detecting abnormal events such as patient falls and gas leaking in a home environment. This system is practically designed and implemented only for two static scenarios. In [14], authors propose using fog computing devices (represented by mobile device held by the patient) along with cloud computing servers for a pervasive fall detection health system. Despite that the aforementioned two fog-cloud based papers are only limited to simple processing that can be processed on the mobile devices, other proposals define the fog nodes and their processes as being more complex such as [15][16][17]. In [15], the authors propose a fog computing architecture for the aim of clinical speech processing for patients with Parkinson's disease. Processing is performed on a fog computer where the cloud is only used for the long term storage. Another work is proposed in [16] for Chronic Obstructive Pulmonary Disease patients. The authors focus on the idea of the existence of real-time processing provided by the fog to cloud (F2C) concept. The work lacks the details about the processing nodes, place, capabilities, and how tasks are handled between the fog and cloud. Generally speaking, the existing work is very specific to certain medical use cases. In addition, all complex processing is performed on the cloud layer even if it is emergent and greatly affects the latency. Comparing to our architecture, we take advantage of the flexibility of allocating the tasks on both fog and cloud resources based on other factors than the complexity of the tasks. Moreover, our architecture is generic and not restricted to a single medical case. Medical scenarios can be easily added with simple configuration changes in the hardware setup and software.

III. Proposed Architecture

Our architecture assumes an environmental context of a hospital or a retirement home which encompasses a user context of multiple mobile patients who are able to perform diverse set of activities. Our architecture is not restricted to specific medical cases as presented in the previous section. It can support patients with different diseases for example patients with Parkinson disease who need speech analysis process in addition to patients with heart problems who need ECG signal processing, etc. An extension of the scripts describing tasks to be processed is added in task triggering modules as will be described next. Furthermore, based on the algorithms implemented, our architecture enables dynamic distribution of health tasks between the fog and the cloud nodes in real time in contrast to static off-line approaches in the literature. The proposed architecture is shown in Fig. 1 and consists of four layers as follows:

1) **Things layer (IoT devices)** The things layer contains all sources of data needed for the healthcare system. We describe two sets of sensors in this healthcare architecture; set A and set B. Set A contains the environmental sensors connected through wireless sensor networks (WSNs). These sensors are static such as Infrared (IR), cameras, light and temperature. Set B involves wearable sensors that generate two types of sensing data: activity monitoring and biomedical (BIO). The activity sensors are physical body sensors that are used to infer the motion such as accelerometer and gyroscope. Examples of BIO sensors are the heart beat, ECG, and glucose sensors.

2) **Sink layer** The sink layer contains the data aggregating nodes such as the micro-controllers, smart watches, and mobile devices. These nodes are responsible for: *1) Data acquisition;* the sink node acquires and collects the raw data generated from the data sources in the things layer, *2) Data forwarding;* the sink node forwards the collected data to the fog layer for further processing, *3) Inquiries forwarding;* the sink node forwards the inquiry requests to the fog layer. Inquiries are on-demand requests issued by the end users of the system including the care givers and the patients themselves to obtain health information. The inquiries are issued using mobile devices connected to the fog layer - mainly the fog broker. Examples include when a patient wants to know his updated pills timetable assuming the timetable is updated periodically according to the patient's health status during the last 24 hours. Another example is when a doctor wants to get data log on the last 100 heartbeat readings of a patient, *4) Limited processing;* the sink node helps in the processing of some tasks according to their processing capacity and if this will lead to the global objective of the allocation. The sink nodes are connected to the fog layer where the data is received and stored in the sensors database.

3) **Fog layer** The fog nodes are devices with high computation, communication, and storage capabilities, and are capable of performing diverse processing operations. Each fog node contains its internal database used for the computation and storage of its allocated tasks. Beside the internal databases, three global fog databases are maintained in this layer: *1) Resources database* where the processing capacity, utilization, and storage of each processing node are reported, *2) Sensor database* where all the sensing data reported from the things layer is stored and updated, and *3) Patients record database* for storing the results of the analysis operations. The fog nodes are responsible for performing the computations of the following four sets of tasks: *data analysis, critical analysis, critical control* and *context management* [18]. Critical health analysis and critical control tasks have relatively higher priorities. A fog broker is responsible for querying information from the *Resources database* to manage the allocation of described tasks on those resources. An extension can be made to include more than one fog broker node if needed and if there are cases that allow for that.

Fog broker It includes the following modules: *a) Periodic Task Trigger.* This module is responsible for releasing periodic tasks. We assume a set of periodic health tasks with different frequencies and deadlines which are varied based on the current configurations and requirements of the environment (number of medical cases, current status of the patients health, etc). *b) Emergency Monitoring.* This module continuously monitors the patients' status by accessing the *patient record database*. Through the continuous checking of the results of the analysis operations performed, this module can detect the emergency cases. Next, proper emergency actions are handled whether notifying the care givers or triggering critical control tasks which are passed to the task collector module to be performed. *c) Task Collector.* This module collects all the tasks needed to be processed from the periodic task trigger and emergency monitoring modules, and inquiries from the patients and the caregivers.

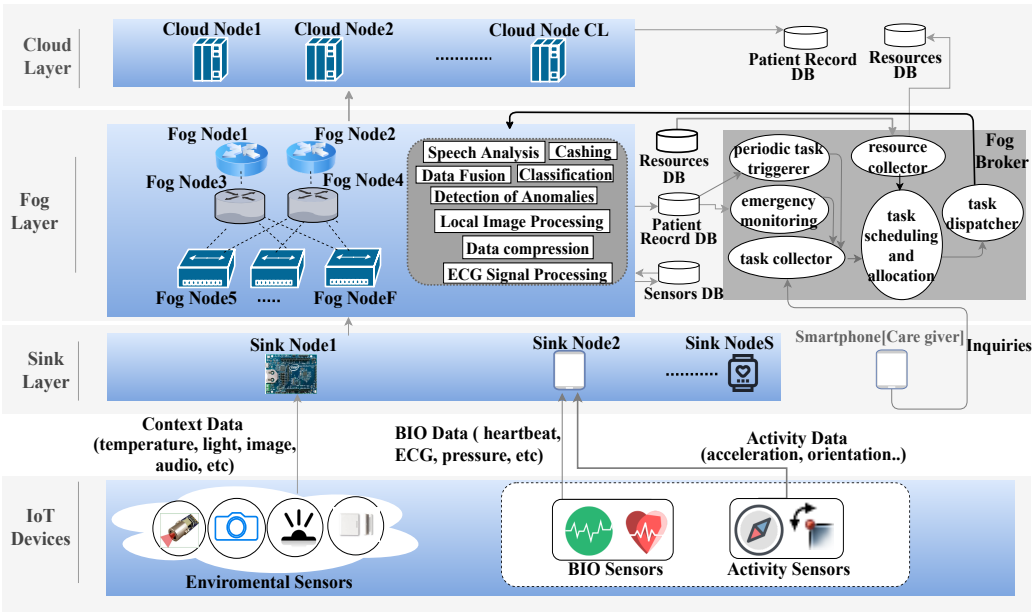


Fig. 1: Proposed IoT architecture for healthcare.

It establishes a set of structures containing information and features of the tasks. *d) Resource Collector.* This is the module that receives information about the available resources on the fog layer. It maintains the utilization of the fog nodes as well as the updated capacities of these nodes. *f) Task Scheduling and allocation.* This module is responsible for the formulation and setup of the task allocation process for the fog and cloud nodes. This module mainly implements a software routine for an optimization technique to achieve an optimal solution for minimizing the latency of performing the health tasks. Heuristic approaches can reach approximate solutions in a faster way than optimization techniques that search for optimal solutions [19][20]. The output of this module is a scheduled mapping between the tasks to be performed and their allocated fog or cloud nodes. *e) Task Dispatcher.* This module is responsible for contacting the fog and cloud nodes, assigning them their allocated tasks.

4) **Cloud layer** The cloud layer contains resource nodes on the cloud for performing part of the tasks that are not performed in the fog layer; usually time unconstrained tasks. This layer contains a permanent database for keeping the patient's health analysis results to be used by the care givers.

IV. Task Scheduling and Allocation

This section describes an approach for task scheduling and allocation that is implemented in the fog broker. The approach achieves a balanced and effective task scheduling and allocation to healthcare tasks. Our objective is to minimize the latency of healthcare tasks and ensure that critical tasks meet their deadlines.

A. System Model

The input to the task scheduler and allocation module includes a set of independent input tasks ($\mathcal{U}$) and a set of fog nodes ($\mathcal{V}$), which are represented as a bipartite graph, and a set of cloud nodes ($\mathcal{W}$), as shown in Fig. 2. The bipartite graph links, through an edge, each task represented as a vertex in the set $\mathcal{U}$ with one or more fog nodes represented by a vertex in the set $\mathcal{V}$, where $\mathcal{U}$ and $\mathcal{V}$ are two disjoint and independent sets. The existence of the edge means that the data related and needed by that task to be executed is resided in the internal database of that fog node. So, an edge describes the locality of the data in a fog device. This locality comes from the layered architecture where the sensors are connected to their upper layer sink and gateway nodes. Hence, the data of these sensors resides initially on that sink and gateway nodes before they are transmitted to the global database in the fog layer or in the cloud layer. Each task is defined by a set of six features: *1) TaskID* which is a number that uniquely identifies the task, *2) Class* which is a number that classifies the task type into one of three categories according to their priorities; 3 for being a critical analysis or critical control task, 2 for being a context management class or 1 for being a data analysis class, *3) Size of the measurement data* that the task will work on and which depends on the segment collected from the corresponding sensor(s), *4) MaxResponse* which defines the maximum time that the task needs to be executed through, *5) Task complexity* which is defined in terms of MIPS, and *6) Set of Sensors* related to this collected

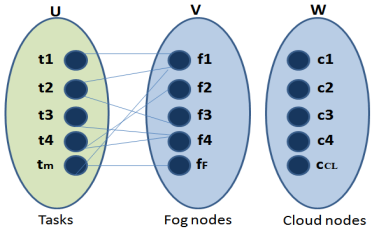


Fig. 2: Input to the task scheduler and allocation module.

task. The objective is to choose the best fog or cloud node to allocate a specific task to, and to schedule all the tasks allocated so that latency can be minimized. The proposed approach is divided into two phases. The first phase adapts and implements a multi-criteria decision making algorithm called Weighted Sum Method (WSM) [21] for ranking the tasks. The second phase proposes and implements a task scheduling and allocation algorithm, Modified BAR, based on the BAlance-Reduced (BAR) algorithm [22].

B. Ranking

The WSM algorithm is implemented to evaluate and rank the tasks to be executed in terms of task classification and maximum response time according to Eq. 1:

$$T_i^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij} \qquad for\ i = 1, 2, 3, ..., m \quad (1)$$

where $T_i^{WSM-score}$ is the total importance of task T_i, w_j is the relative weight of importance of the criterion c_j, a_{ij} is the performance value of task T_i when it is evaluated in terms of criterion c_j and m is the total number of tasks. Two significant weighting criteria ($n = 2$) are selected: *1) task classification*, and *2) 1/MaxResponse* which is the inverse of *MaxResponse* of the task. The weight is set to 0.5 for each criteria. The ranked list is scanned and the tasks are scheduled and allocated according to the second phase.

C. MBAR

The first step of MBAR is balancing the load where we allocate a task t_i on a processing node considering task data locality. We select a processing node P which has the minimum processing load to allocate a task t_i. The load represents the finish time after the execution of that task and can be calculated following Eq. 2:

$$L_P = L_P^{init} + L_P(t_i) \quad (2)$$

where L_P^{init} is the initial load of a processing node P at the start time and updated continuously, and $L_P(t_i)$ is the total time to process task t_i on that processing node and calculated according to Eq. 3.

$$L_P(t_i) = e(t_i, P) + t(t_i, P) \quad (3)$$

where $e(t_i, P)$ is the execution time of task t_i on processing node P and $t(t_i, P)$ is the transmission time for transferring the data needed by task t_i to the processing node P. In the balanced step, $t(t_{i,P}) = 0$ due to task data locality. The execution time $e(t_i, f)$ of processing a task t_i on processing node P is calculated according to Eq. 4

$$e(t_i, P) = \frac{Complexity\ of\ task\ t_i}{Processing\ capacity\ of\ node\ P} \quad (4)$$

The complexity of task t_i is obtained from the task collector module, and the processing capacity of the processing node P is obtained from the resource collector module. The processing capacity is represented in terms of MIPS (million instruction per second). In the reduce step, we iteratively reallocate the tasks on different fog and cloud nodes to enhance the schedule length. We choose the fog node with the maximum load to reallocate one of its tasks. We implement three different strategies to choose the task to be reallocated; The first strategy selects a random task, whereas the second and third strategies are based on the nature of the application. The second strategy selects the task with the highest class. The third strategy chooses the task with the maximum MaxResponse. To reallocate a task on a new processing node P, whether fog or cloud, we calculate the processing load using Eqs. 2, 3 and 4. The transmission time is the time to transfer the data blocks for task t_i from the database in fog layer to the processing nodes whether fog through a LAN connection or a cloud through an Internet connection. The transmission time is calculated according to Eq. 5.

$$t(t_i, P) = DataSize_{t_i}/BW_P \quad (5)$$

where $DataSize_{t_i}$ is the size of the measurement data obtained from the task collector, and BW_P is the bandwidth of the link connecting the sensor database in the fog layer with the processing node P.

V. Performance Evaluation

This section discusses the experimental setup and evaluation metrics. The configuration details of the fog nodes, cloud nodes, and tasks are shown in tables I, II and III, respectively, where two sink nodes are connected to each gateway. The simulations are carried out using iFogSim [23] which is a simulator for modeling customized fog computing environments with large number of fog nodes and IoT devices such as sensors, actuators, routers, etc. All the simulation results are averaged over 10 runs.

TABLE I: Configurations details for fog devices

Parameter	Value
Number of fog nodes	22
Processing capacity	[10, 100] MIPS for sink devices [100, 500] MIPS for others
RAM	[512, 1] GB
Storage	4 GB
Bandwidth	1024 Mbps

TABLE II: Configurations details for cloud nodes

Parameter	Value
Number of cloud nodes	25
Processing capacity	[250, 1500] MIPS
RAM	40 GB
Storage	11 TB
Bandwidth	10 ,100, 512 Mbps
Cost per time unit	3 $

TABLE III: Configurations details for tasks

Parameter	Value
Number of tasks	300
Complexity	[1, 3000] MIPS
Data size	[100, 500] MB
Max response time	[15 sec, 30 min]
Sampling rate	[20, 5000] Hz

A. Evaluation Metrics

We evaluate the proposed architecture and algorithms according to four metrics: Makespan, MissRatio, AvgDelay,

and Cost per execution.

Makespan which defines the total length of the schedule and it is calculated using Eq. 6.

$$\max_{\forall P \in N} (L_f(P)) \tag{6}$$

where $L_f(P)$ is the final processing load of processing node P and N is the total number of fog and cloud processing nodes.

MissRatio is the percentage of tasks that misses the maximum response time and it is calculated as in Eq. 7.

$$MissRatio = \frac{No.\,of\ tasks\ missing\ MaxResponse(n)}{Total\ number\ of\ tasks} \tag{7}$$

A task t misses max response time if the condition in Eq. 8 is true.

$$Trig(t) + MaxResponse(t) < CompletionTime(t) \tag{8}$$

where $Trig(t)$ is the time when the task is triggered by periodic triggerer, emergency monitoring modules, or inquired by the users.

AvgDelay is the average of the times that exceeded the max response time for all the tasks and it is calculated as in Eq. 9.

$$AvgDelay = \frac{\sum_{j=1}^{n} Delay(t_j)}{No.\,of\ tasks\ missing\ MaxResponse(n)} \tag{9}$$

where t_j is a task that misses its maximum response time.

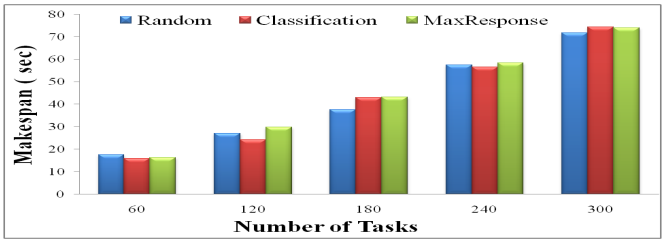
Cost per execution which is the total execution cost for using cloud resources and it is calculated using Eq. 10.

$$Cost = cPt \times \sum_{\forall C \in CL} L_f(C) \tag{10}$$

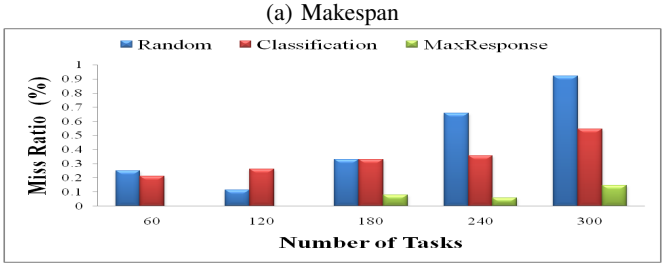
where cPt is the processing cost per time unit, $L_f(C)$ is the final processing load of a processing cloud node C, and CL is the total number of cloud nodes. We investigate the performance of the aforementioned parameters with varying the number of tasks and the number of cloud nodes with respect to three different task selection strategies implemented.

B. Varying number of tasks

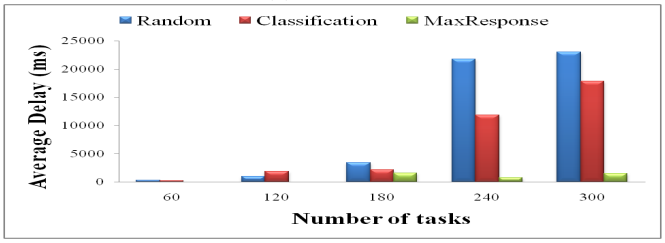
Fig. 3 illustrates the impact of varying the number of tasks on the Makespan, MissRatio and AvgDelay metrics for the different task selection strategies. Makespan, MissRatio and Avgdelay are increasing with increasing the number of tasks. Applying different strategies has no significant effect on Makespan, as shown in Fig. 3(a). However, Figs. 3(b) and 3(c) show great differences on MissRatio and Avgdelay. Classification and MaxResponse strategies show less MissRatio and Avgdelay compared to Random strategy, since they qualify the choice of the candidate task to be reallocated. Also, MaxResponse strategy shows less MissRatio and AvgDelay compared to the classification strategy since it reallocates the tasks that have maximum MaxResponse time with extra transmission time cost. Hence, the time-constrained tasks remain for processing with a better chance of finishing earlier. It is worth noting that the MissRatio when using the MaxResponse strategy is not exceeding 0.2% when the total number of tasks is 300. Also, the AvgDelay does not exceed 5 seconds in the MaxResponse strategy, whereas it reaches about 25 seconds in the Random strategy.



(a) Makespan

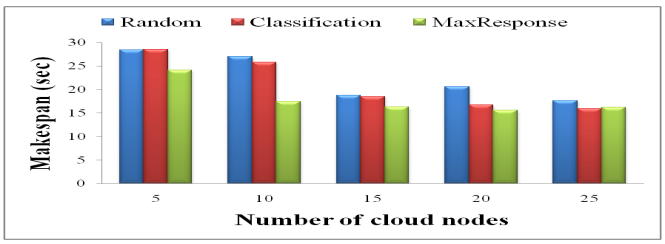


(b) MissRatio

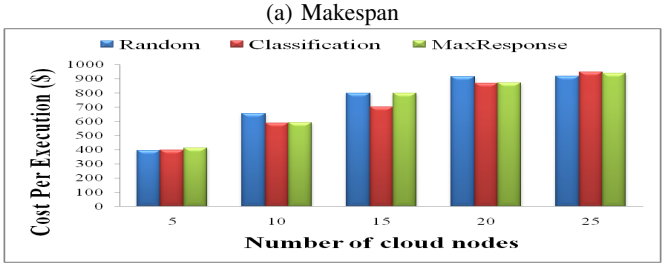


(c) Average Delay

Fig. 3: Varying number of tasks



(a) Makespan

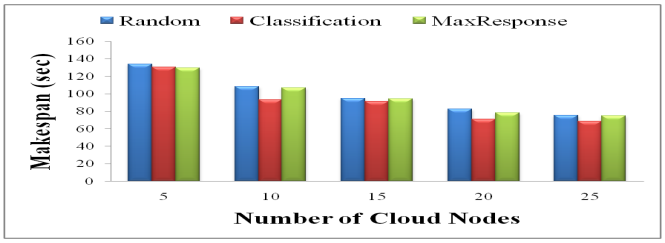


(b) Cost per execution

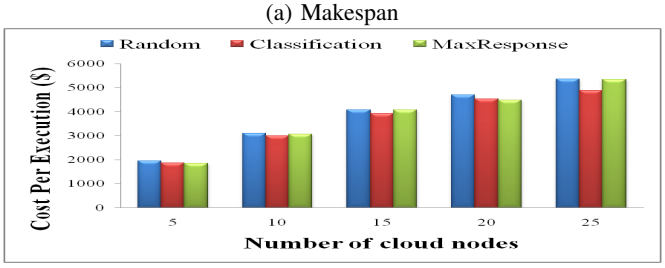
Fig. 4: Varying number of cloud nodes for 60 tasks

C. Varying number of cloud nodes

Fig. 4 demonstrates how varying the number of cloud nodes affects Makespan and the Cost with number of tasks is set to 60. It is worth noting that despite increasing the cloud resources raises the costs, no significant improvement in Makespan is observed. The impact of varying the number of cloud nodes when the number of tasks is 300 is shown in Fig. 5. Increasing the number of cloud nodes results in increasing the cost and decreasing the Makespan. So, based on our configurations and the capabilities of the fog devices, and the complexities of the tasks, it is not preferred to



(a) Makespan



(b) Cost per execution

Fig. 5: Varying number of cloud nodes for 300 tasks

increase the cloud nodes before the number of tasks reaches at least 300 tasks. This leads to monetary cost savings.

VI. Conclusion and Future Work

A generic Cloud-Fog based IoT architecture dedicated for healthcare is proposed. The purpose is to eliminate the latency problems associated with using cloud services only for performing the processing needed to provide healthcare services. The core component in the proposed architecture is the fog broker which includes multiple modules for ensuring the accurate and beneficial distribution of the system tasks between the fog and cloud nodes. The proposed architecture is implemented under the iFogSim simulator and evaluated under different evaluation criteria. The proposed scheduling and allocation algorithm shows reasonable Makespan values when varying the number of tasks and the number of cloud nodes. Furthermore, different strategies for performing the reallocation on fog and cloud nodes are compared and analyzed. Classification and MaxResponse strategies outperforms Random strategy through their reduction in MissRatio and AvgDelay of meeting the deadline for time-constrained tasks. In addition, the results show that we can eliminate the unworthy increasing cost for deploying more cloud resources in some cases without affecting the latency.

References

[1] A.-M. Rahmani, N. K. Thanigaivelan, T. N. Gia, J. Granados, B. Negash, P. Liljeberg, and H. Tenhunen, "Smart e-health gateway: Bringing intelligence to internet-of-things based ubiquitous healthcare systems," in *12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, 2015, pp. 826–834.

[2] S. Chernbumroong, S. Cang, A. Atkins, and H. Yu, "Elderly activities recognition and classification for applications in assisted living," *Expert Systems with Applications*, vol. 40, no. 5, pp. 1662–1674, 2013.

[3] L. Catarinucci, D. De Donno, L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and L. Tarricone, "An iot-aware architecture for smart healthcare systems," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 515–526, 2015.

[4] "Iot platforms: enabling the internet of things," http://cdn.ihs.com/www/pdf/enabling-IOT.pdf, accessed: 2018-10-13.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[6] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: a fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[7] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[8] Y. Zhang, M. Qiu, C.-W. Tsai, M. M. Hassan, and A. Alamri, "Health-cps: Healthcare cyber-physical system assisted by cloud and big data," *IEEE Systems Journal*, vol. 11, no. 1, pp. 88–95, 2017.

[9] J. Gomez, B. Oviedo, and E. Zhuma, "Patient monitoring system based on internet of things," *Procedia Computer Science*, vol. 83, pp. 90–97, 2016.

[10] M. Jiang, T. N. Gia, A. Anzanpour, A.-M. Rahmani, T. Westerlund, S. Salanterä, P. Liljeberg, and H. Tenhunen, "Iot-based remote facial expression monitoring system with semg signal," in *IEEE Sensors Applications Symposium (SAS)*, 2016, pp. 1–6.

[11] Y.-Y. Ou, P.-Y. Shih, Y.-H. Chin, T.-W. Kuan, J.-F. Wang, and S.-H. Shih, "Framework of ubiquitous healthcare system based on cloud computing for elderly living," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*. IEEE, 2013, pp. 1–4.

[12] E.-M. Fong and W.-Y. Chung, "Mobile cloud-computing-based healthcare service by noncontact ecg monitoring," *Sensors*, vol. 13, no. 12, pp. 16 451–16 473, 2013.

[13] R. Craciunescu, A. Mihovska, M. Mihaylov, S. Kyriazakos, R. Prasad, and S. Halunga, "Implementation of fog computing for reliable e-health applications," in *49th IEEE Asilomar Conference on Signals, Systems and Computers*, 2015, pp. 459–463.

[14] Y. Cao, S. Chen, P. Hou, and D. Brown, "Fast: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation," in *IEEE International Conference on Networking, Architecture and Storage (NAS)*, 2015, pp. 2–11.

[15] A. Monteiro, H. Dubey, L. Mahler, Q. Yang, and K. Mankodiya, "Fit a fog computing device for speech teletreatments," *arXiv preprint arXiv:1605.06236*, 2016.

[16] X. Masip-Bruin, E. Marín-Tordera, A. Alonso, and J. Garcia, "Fog-to-cloud computing (f2c): the key technology enabler for dependable e-health services deployment," in *IEEE Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2016, pp. 1–5.

[17] H. Dubey, J. Yang, N. Constant, A. M. Amiri, Q. Yang, and K. Makodiya, "Fog data: Enhancing telehealth big data through fog computing," in *Proceedings of the ASE BigData & SocialInformatics 2015*. ACM, 2015, p. 14.

[18] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, "Fog computing in healthcare—a review and discussion," *IEEE Access*, vol. 5, pp. 9206–9222, 2017.

[19] M. Kafil and I. Ahmad, "Optimal task assignment in heterogeneous distributed computing systems," *IEEE concurrency*, vol. 6, no. 3, pp. 42–50, 1998.

[20] S. A. Cook, "An overview of computational complexity," *Communications of the ACM*, vol. 26, no. 6, pp. 400–408, 1983.

[21] E. Triantaphyllou, "Multi-criteria decision making methods," in *Multi-criteria decision making methods: A comparative study*. Springer, 2000, pp. 5–21.

[22] J. Jin, J. Luo, A. Song, F. Dong, and R. Xiong, "Bar: An efficient data locality driven task scheduling algorithm for cloud computing," in *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2011, pp. 295–304.

[23] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.