Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 07, No. 01, March 2021.

LIVE BIG DATA ANALYTICS RESOURCE MANAGEMENT TECHNIQUES IN FOG COMPUTING FOR TELE-HEALTH APPLICATIONS

Ragaa Shehab, Mohamed Taher and Hoda K. Mohamed

(Received: 20-Nov.-2020, Revised: 11-Feb.-2021, Accepted: 23-Feb.-2021)

ABSTRACT

Enhancing the IoT health monitoring systems used in various environments, such as smart homes and smart hospitals, imply lively analyzing the patients' critical streams (e.g. ECG stream). Conducting these tele-health applications over the traditional cloud violates the deadline constrains of the stream analytics applications, which results not only in performance degradation, but also in inaccurate analytics results due to patient's stream loss. Fog computing can take place within the patient's vicinity and is considered as the best candidate for critically analyzed stream applications. Fog nodes are geo-distributed and are poor in resources, thus a scalable and fault-tolerant resource management platform for stream analytics in fog computing is a must. Current Stream Processing (SP) resource managers are designed for massive resource nodes, deploying them over the poor resource edge fog nodes greatly decreasing the fog infrastructure utilization. Innovative SP resource managers that cope with the fog nature are needed. We propose Fog Assisted Resource Management (FARM) platform based on Apache Hadoop2 resource manager (YARN) for compatible stream/batch analytics. Static FARM (S-FARM) represents two YARN schedulers; per-user and per-module. Results indicate that per-user scheduler overcomes the lack of resources issues of the edge fog nodes, fully utilizes the fog infrastructure and allows the system to expand safely up to its double size. In addition, Differentiated S-FARM scheduler is proposed to support per-user control to the analytic results' accuracy and speed. Stream CardioVascular Disease (S-CVD) application for patient's ECG analytics is simulated in iFogSim to judge the proposed YARN schedulers. The research is pioneer in enhancing the poor resource edge fog node utilization, supporting per-user control to live big data analytics IoT applications and utilizing iFogSim to implement and evaluate the resource manager performance of a stream analytics platform.

KEYWORDS

Fog/Edge computing, Big data analytics, Stream analytics, Apache Hadoop2 YARN schedulers, Per-user control, Analytics accuracy, Fog infrastructural management, Patient monitoring, Smart hospital.

1. INTRODUCTION

According to the National Council on Aging, up to 80% of older adults have at least one chronic health condition that requires continual treatment management. This fact increases the burden on the world's health care systems. The evolution of the IoT technologies and health informatics systems aims to realize the remote patient's monitoring (tele-health) with high quality of care and to make the management of these populations more cost-effective [1]. Tele-health applications allow patients to live more independently and improve their quality of life while reducing the cost of medical care and hospital re-admissions. In addition, online patients' data aids caregivers in early patient state classification, emergency situation management and following up patient adherence to the given treatments.

Data analytics plays an important role in tele-health ecosystems, especially for smarter decision-making within the time constrains; i.e., patient's critical state detection. For this fully distributed data sources, cloud data processing fails to meet the requirement of delay sensitive applications, which results not only in performance degradation, but also in inaccurate analytics results due to patient's stream loss [2]. Fog computing, also known as edge computing [3]-[4], is a distributed computing paradigm that aims to tackle the issue by offloading data analytics and sensitive delay tasks to the edge of the network closer to the data sources, leaving the delay tolerant highly computational tasks to be performed at the cloud. Resource management in fog computing is a challenging issue [5]-[6]. This is because fog allows application modules to be distributed along the fog tiers to provide an enhanced

R. Shehab, M. Taher and H. K. Mohamed are with Computer and Systems Engineering Department, Ain Shams University, Cairo, Egypt.
Emails: rashehab@yahoo.com, mohamed.taher@eng.asu.edu.eg and Hoda.Korashy@eng.asu.edu.eg

application delay and network usage [2], [7]. However, deployment of critical tele-health stream applications in such manner degrades the application performance, because edge fog nodes are geo-distributed, poor in resources and sustain to failure that will affect the patient's experience and prevent achieving the main purpose of tele-health applications [2]. A scalable and fault tolerant Stream Processing (SP) platform in fog computing overcomes these issues. The on-market SP resource managers are designed for massive resource nodes. Deploying these resource managers over poor resource edge fog nodes degrades the fog infrastructure utilization. Innovative SP resource managers in fog computing are needed. All the reviewed literature depends on real cluster implementation. To the best of our knowledge, the research is pioneer in utilizing iFogSim simulator [8] to implement and evaluate the performance of SP platform resource manager. This contribution may guide researchers in implementing and judging the resource management performance of other various on-market SP platforms; i.e., Apache (Samza, Flink, Spark,...).

This work proposes a Fog Assisted Resource Management (FARM) platform based on YARN for compatible short-term and long-term big data analytics. Static FARM (S-FARM) represents YARN schedulers. Two schedulers are proposed to control the fog nodes CPU load: per-user and per-module. Per-user scheduler is a YARN scheduler that copes with the edge fog nodes lack of resources. In addition, Differentiated S-FARM scheduler is proposed to allow per-user control to the analytics results QoS. Analytics results are controlled by the analytics tuples' Million Instructions Per Second (MIPS) to represent accurate *versus* fast results. Stream CardioVascular Disease (S-CVD) application is modelled. It lively analyzes the patient's ECG streams to conduct the patient's state using a linear classifier machine learning tool. IFogSim is used to judge the application and the fog infrastructure performance under the proposed YARN schedulers.

The paper is organized as follows: Section 2 introduces a brief background about live big data analytics. Section 3 provides a literature review. Section 4 presents the research methodology. Section 5 presents S-CVD application and system model. Section 6 presents FARM platform and YARN to fog mapping. Section 7 presents the YARN schedulers (S-FARM algorithms). Section 8 analyzes the application and infrastructure performance. Finally, Section 9 concludes the paper and presents suggestions for future work.

2. LIVE BIG DATA ANALYTICS

Big data is characterized by its volume, variety, veracity, velocity and value. Big data could be analyzed either in stream or batch mode [11]-[12], see Table 1.

Table 1. Comparison between stream and batch data analytics Modes.

Differences	**Stream Mode**	**Batch Mode**
Mode	Short-term (live) analytics	Long-term analytics
Management target	Transient streams	Persistent data
Amount of data	Unknown in advance	Finite
Processing model	On the fly	Store then process
Query model	Continuous	One-time query
Access model	Sequential access	Random access
Result repeatability	Nearly impossible	Easy
Result update	Incremental update	Global update
Focus of processing	Low latency	High accuracy
Platforms	Storm, Spark, Samza, Flink,…	Apache Hadoop,…

In the domain of healthcare, IoT big data has several challenges, including high data rate with variable volume, semi-structured or unstructured format (i.e., echo image, voices), correlation across several dimensions (i.e., time, location) and its social relations among related healthcare devices [1]. Analyzing the healthcare IoT streams at the fog network has a set of advantages, including real-time handling, user-centric processing, user’s mobility support and geo-distribution, location and context awareness applications support [10].

Stream Analytics for Critical Healthcare Decision-making: A stream is defined as a sequence of data elements ordered by time. Each data element has a time stamp that measures the data order. Stream processing SP is a one-pass data processing that aims to achieve low processing latency by keeping data in motion. The complete stream analytics data life cycle includes [12]:

1. Data generation stage.
2. Data collection and aggregation stage: from different distributed sources.
3. Messaging and buffering stage: IoT streams are gathered into a centralized buffer.
4. Continuous Logic Processing (CLP) stage: processes data according to the designed continuous logic. Current CLP Systems (CLPS) are scalable and fault-tolerant.
5. Presentation and storage stages: deliver the insights to end users and store them.

Table 2 presents a comparison between Apache's Stream Processing (SP) platforms that represent the third-generation CLPS. SP platforms are characterized by [12]-[13]:

- Programming components of the CLPS: graph name, nodes and edges.
- Type of process: client (graph builder), task scheduler and task executer.
- CLPS capability of accurate recovery of the same processing results when system failures occur.
- State consistency of all participating components during processing, which is related to the fault recovery methods implemented by the system.

Apache Hadoop2 YARN (Yet Another Resource Management Negotiator) [18]-[19] can serve as the core of various Apache's SP platforms. YARN can serve as the core of various Apache's SP platforms. Thus, the proposed FARM platform is based on YARN for compatible stream/batch analytics.

Table 2. Comparison between open-source stream analytics platforms.

Apache's SP Platform	**Storm [14]**	**Spark [15]**	**Samza [16]**	**Flink [17]**
Processing Type	Stream	Stream-Batch	Stream-Batch	Stream-Batch
Type of processes: 1-Client 2-Task scheduler 3-Task executer	1-Topology builder 2-Nimbus and Zookeeper 3-Workers	1-Spark DAG 2-YARN scheduler or standalone 3-Workers	1-User-defined 2-YARN scheduler or Zookeeper 3-Workers	1-Graph builder 2-YARN scheduler or standalone 3-operators executer
Accurate Recovery	Yes	Yes	Yes	Yes
State Consistency	No	Yes	Yes	Yes
Adopted by	Twitter, Yahoo	eBay Inc.	LinkedIn	Research gate

3. Literature Review: Stream Analytics in Fog Computing

Stream analytics research in fog computing could be classified as:

- **Stream Analytics Platform Deployed Stream Applications:** [20]-[24]. See Table 3. In this literature, performance was measured for general-purpose applications. It did not consider healthcare stream applications with high sensor rates and critical reading that concerns patient's safety and security. In addition, no study considered a single platform for both short/long-term analytics that is required for accurate remote patient monitoring.

- **Healthcare Stream Analytics Platforms:** proposed to deploy healthcare stream applications only. The healthcare application modules are placed on the fog network according to the type of the analytics task; i.e., [25]-[27] for healthcare data critical analysis task and [28]-[29] for healthcare data critical control task. See Table 4. In this literature, three tiers of IoT data network are used for permanent task allocation regardless of the encountered application performance: smart watch or smart phone tier was used for data collection tasks, fog/cloud tier was used for data computations tasks and the cloud tier was used for data storage and long-term analytics tasks.

- **Stream Analytics Platform Deployed Healthcare Stream Applications**: [30]-[32]. See Table 5. In this literature, the evaluation method depends on real cluster implementation only. In addition, the lack of resources of the edge fog nodes has not been addressed.

To the best of our knowledge, no research addressed the edge fog node lack of resources, provided per-user control to the accuracy and speed of the analytic results or utilized iFogSim as an innovative tool to implement and judge the performance of a stream analytics platform resource manager.

Table 3. Stream analytics platform deployed general-purpose application in fog computing.

REF.	Platform	Perf. Metric	Scheduler	Imp.
[20]	Storm	Utilization, latency, inter-node traffic	distributed Storm scheduler that adapts to fog network's changes	R
[21]	Storm	Comm. latency to external IoT actuators or databases	modified Storm with a decision module that decides whether to place selected tasks on edge devices at run time	R
[22]	Storm, Nimbus	Latency, average inter-node traffic	modified Nimbus by adding offline and online schedulers that analyze the topology and monitor the effectiveness of the schedule at run time	R
[23]	Spring cloud dataflow	Optimization problem	resource elasticity mechanism to deal with changing rates of streaming data	R
[24]	Spark	Job completion time, scalability, power consumption	Modified Spark to utilize the whole processing capacity of all the available edge devices	R

R: Real Implementation.

Table 4. Healthcare stream analytics platform in fog computing.

REF.	Perf. Metric	Per. Task Allocation	Application	Imp.
[25]	Data size, processing and transmitting time	At LAN level	ECG monitoring	R
[26]	Processing time, system reliability	At PAN level for transport scenario	ECG feature extraction	R
[27]	End-to-end data delay, mobile battery life time	At PAN level for user mobile scenario	Patient monitoring	R
[28]	Patient deterioration and re-admission incidence rate	User mobile scenario	Oxygen level control	R
[29]	Stable patient heart beat	At LAN level for hospital scenario	Pacemaker monitoring	R

Per.: Permanent Task Allocation.

Table 5. Stream analytics platform deployed healthcare stream applications in fog computing.

REF	Platform	Perf. Metric	Scheduler	Application	Imp.
[30], [31]	Storm, Kafka	-	Kafka and Storm cluster architecture for S/B analytics	i.e., pervasive health	-
[32]	Flink, Kafka	CPU, memory usage, average data loss	Stream computing at Kafka's broker and Flink's cluster processing layer	Anomaly detection REALDISP dataset	R

4. Research Methodology

In this research, we aim to develop a FARM platform for critical tele-health stream analytics applications. Figure 1 presents our research methodology.

5. S-CVD Application Model

Stream CardioVascular Disease (S-CVD) application analyzes an ordered Electrocardiogram (ECG) stream with a sensor transmission rate of (25ms: 1000ms). The application is modelled as a Directed Acyclic Graph (DAG), where modules are represented as vertices and inter-module communications are represented as edges, see Figure 2. Client module accepts the sensor ECG stream, adding any

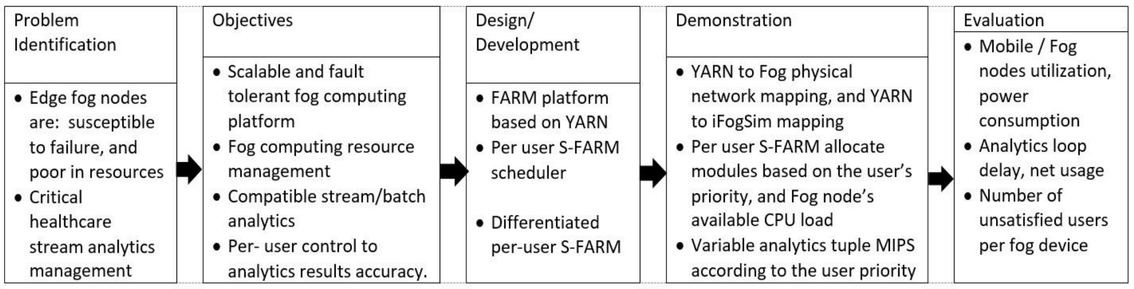
Problem Identification
• Edge fog nodes are: susceptible to failure, and poor in resources
• Critical healthcare stream analytics management
Objectives
• Scalable and fault tolerant fog computing platform
• Fog computing resource management
• Compatible stream/batch analytics
• Per- user control to analytics results accuracy.
Design/ Development
• FARM platform based on YARN
• Per user S-FARM scheduler
• Differentiated per-user S-FARM
Demonstration
• YARN to Fog physical network mapping, and YARN to iFogSim mapping
• Per user S-FARM allocate modules based on the user's priority, and Fog node's available CPU load
• Variable analytics tuple MIPS according to the user priority
Evaluation
• Mobile / Fog nodes utilization, power consumption
• Analytics loop delay, net usage
• Number of unsatisfied users per fog device

Figure 1. Research methodology.

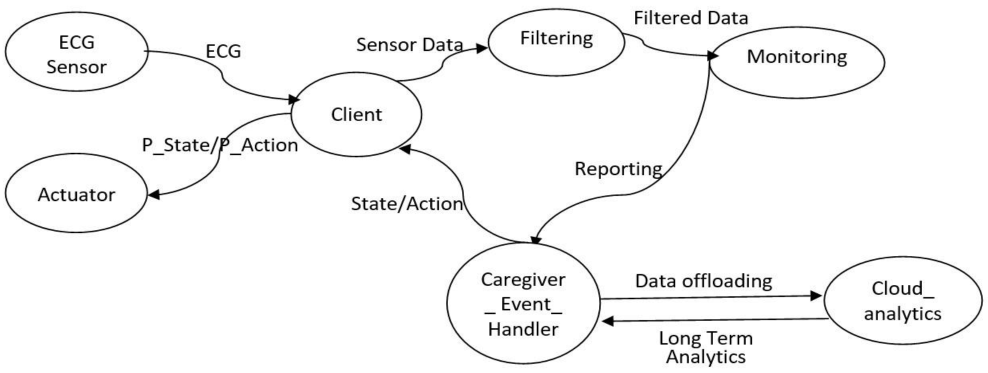

Figure 2. S-CVD application model.

related information and securing the packets. Filtering module cleans the data, eliminates inaccurate and out-of-range readings. Monitoring module is a linear classifier machine learning tool that continuously classifies the patient's state. Caregiver module gets the analytics results and reports the online consultant about the current patient state. Online consultant has a complete access to the patient's electronic health record along with a knowledge base that aids him/her in decision-making. In emergency situations, the consultant can send an ambulance or contact the patient's family for help. After analyzing the data, the patient is informed by his state or with the ongoing actions. Cloud analytics module is located at the remote cloud and is responsible for long-term analytics and managing the analytical operators supervised machine learning for the patient state classifiers.

System Model

- Body Area Network of ECG sensors are connected to the patient and through WiFi or Bluetooth to the smart e-health gateway 1.
- Smart e-health gateway 1 (Mobile) is the patient's smart phone that carries out the data collection task; i.e., S-CVD (Client module).
- Smart e-health gateway 2 (Dept) is located at the patient's vicinity (smart home, smart vehicle or smart hospital ward) and carries out the data analysis tasks of the tele-health applications. It is connected to the healthcare center.
- Remote healthcare center (Proxy-Server) is located at the smart hospital and carries out the decision-making and permanent data storage tasks; i.e., S-CVD (Caregiver module). Data management tasks; i.e., S-CVD (Filtering, Monitoring modules) could be utilized at smart gateway or at the proxy-server according to the required application and fog infrastructure performance.
- Remote cloud is responsible for long-term data analytic; i.e., S-CVD (Cloud analytics module).

Deploying YARN over this system model enables a scalable and fault-tolerant platform for the tele-health application. Also, it preserves the patient's security and privacy, because his/her sensory data is processed locally at his/her vicinity or at his/her smart hospital.

6. FOG ASSISTED RESOURCE MANAGEMENT (FARM) PLATFORM

Based on the complete life cycle of the IoT stream analytics systems [12] and the Apache Hadoop2 YARN architecture [18]-[19], Figure 3 is proposed to represent the FARM platform based on YARN for compatible stream/batch analytics in the fog/cloud system. For a smart hospital system model, the fog nodes (i.e., smart e-health gateway 2 and the remote healthcare center) represent the YARN nodes that carry out the stream analytics. A remote cloud represents YARN nodes that carry out batch analytics tasks for multiple hospital branches that belong to the same owner. For the S-CVD application, patients' streams are queued in messaging system (i.e., Kafka) before turning into the hospital's stream analytics platform. The Hadoop Distributed File System (HDFS) can comprise a large number of directly connected individual fog nodes at the smart hospital. For decision-making, caregiver queries are sent to HDFS in fog network and cloud data center.

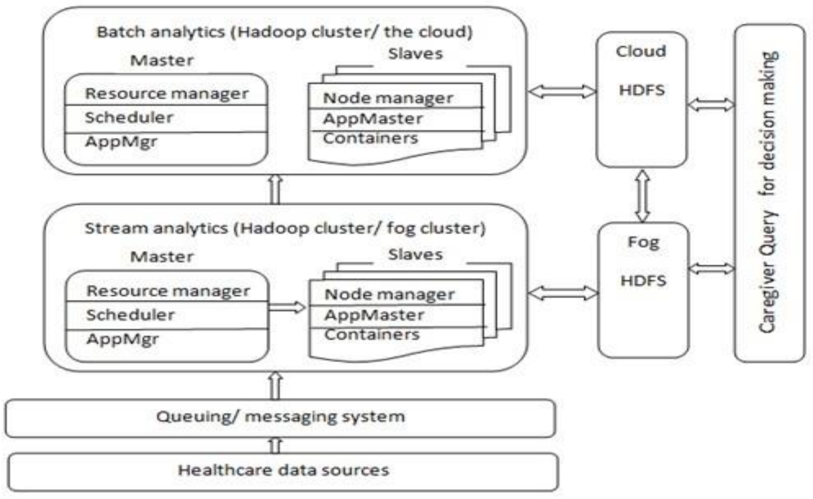

Figure 3. FARM platform based on YARN for compatible stream and batch analytics.

YARN to Fog Mapping: Table 6 proposes YARN to fog physical entity mapping, where Fog Manager Server and Fog Domain Servers are presented to carry out the main YARN responsibilities. Also, it presents YARN to iFogSim mapping.

Table 6. YARN concept to fog physical entity components and iFogSim mapping.

YARN Concept	Fog Physical Entity	iFogSim Mapping
ResourceManager (Master): **Scheduler:** global resource scheduler **Application Manager:** follows up the progress for executing the application's specific ApplicationMaster	**Fog Manager Server (Master):** Scheduler: determines how application's modules are placed across fog devices upon submission of the application Application Manager: monitors the performance and reschedule resources of each host fog device to the application modules **Fog Master Server:** Console for user interface and modules	**Scheduler:** represented by: controller class, per-user-Basic-MP class, per-module-MP class, per-user-Diff-MP class, StreamOperatorScheduler, TupleScheduler class **ApplicationManager**: CheckDelay-Enh-Diff method (FogDevice class), MY-updateAllocatedMips method (FogDevice class)
NodeManager (slaves): Host: tracks the running Virtual Machines (VMs) Container: VMs ApplicationMaster: global monitoring daemon, negotiates resources for VMs.	**Fog Domain Server:** one or multiple tiers **Wireless End Fog Nodes:** light-weight cluster slaves ApplicationMaster: allocated over certain host or centralized	Host: FogDevice class. Container: AppModule class ApplicationMaster: main class

7. FARM ALGORITHMS

This work proposes Static FARM (S-FARM) representing YARN ResourceManager(Scheduler). Other YARN components will be studied as future work, see Table 6.

7.1 Basic S-FARM Algorithm

S-FARM schedulers determine how the application's modules are placed across the fog nodes upon the application submission. Two modes are used:

- Per-user Mode, where each user has his own Virtual Machines VMs carrying out his own workload. Resources are managed by placing the user's modules (VMs) individually and one by one according to the user's priority, until the required RM objectives are reached. This mode allows explicit user differentiation according to the user's Service Level Agreement (SLA).
- Per-module Mode, where all users' workloads are placed together and carried out over one VM. Each VM represents one application module. Resources are managed by placing the whole module according to the application DAG until the required RM objectives are reached.

Per-user S-FARM scheduler, Algorithm 1, places the high-priority user modules at its closest fog nodes first; if there is a shortage in the closest fog node computational resources, the remaining users' (the lower priorities) modules are shifted up to the next fog node in the fog node path hierarchy. Per-module S-FARM scheduler, Algorithm 2, places and shifts up the modules that carry the whole user instances according to the application's DAG.

UsrMIPS in algorithm 1 and ModMIPS in algorithm 2 represent the analytics tuple's CPU length. They are permanently assigned by caregivers based on the requested analytics accuracy, where heavy-weight tuples represent accurate results and light-weight tuples represent fast results. They are used across all the module's edges to calculate the required CPU load for this module $CPULoad_{Mod}^{\mathrm{Re}q}$.

Algorithm 1: S-FARM Per-user Scheduler Algorithm

```
1: procedure SFARM-PERUSER(PATHS,App,Usrprio,UsrMIPS)
2:     Arrange all FogDevices within PATHS (leaf to root traversal)
3:     Arrange PATHS according to the Usrprio (high to low priority users).
4:       for (P ∈ PATHS) do
5:           for (endFogDev ∈ P) do
6:               for (UsrModule ∈ App) do
7:                   Assign the UsrMIPS (analytics tuple's CPUlength).
8:                     if $CPULoad_{Dev}^{Curr} + CPULoad_{Mod}^{req} . \le CPULoad_{Dev}^{\max av}$. then
9:                     Place the UsrModule on the endFogDev
10:                else
11:                    Shift up the UsrModule to ParentFogDev
```

Algorithm 2: S-FARM Per-module Scheduler Algorithm

```
1: procedure SFARM-PERMODULE(PATHS,App,ModMIPS)
2: Arrange all FogDevices within PATHS (leaf to root traversal)
3: for (P ∈ PATHS) do
4:     for (endFogDev ∈ P) do
5:         for (Module ∈ App) do
6:         Assign the ModMIPS (analytics tuple CPUlength).
7:         if (Module is Placed on this endFogDev )
8:             if ($CPULoad_{Dev}^{Curr} + CPULoad_{Mod}^{req} . \le CPULoad_{Dev}^{\max av}$.) then
9:                 Place this Module instance on this endFogDev
10:            else
11:                Shift up the whole Module to ParentFogDev
12:        else
13:            if $CPULoad_{Dev}^{Curr} + CPULoad_{Mod}^{req} . \le CPULoad_{Dev}^{\max av}$.then
14:                Place the Module's first instance on this endFogDev.
```

In addition, in both algorithms, the modules are placed at a fog device until the device's maximum available $CPULoad_{Dev}^{\max av}$ is reached. In this study, dept and proxy fog devices are loaded up to 100% of their maximum CPU load, while the user's mobile is loaded to <= 0.15% of its maximum CPU load. Increasing the allowable mobile's CPU load permits more application modules to be placed at the user's mobile, which is not a desired performance.

Both schedulers shift up the modules to the next level on the fog devices' path hierarchy if: $CPULoad_{Dev}^{Curr} + CPULoad_{Mod}^{req} \geq CPULoad_{Dev}^{\max av}$.

7.2 Differentiated S-FARM Algorithm

Basic S-FARM assigns a constant analytics tuples' MIPS to all users. Differentiated S-FARM assigns a variable analytics tuples' MIPS to each user, according to the user's requested analytics accuracy and speed.

8. RESULTS

Simulation parameters are the same used in [2]. S-CVD are tested for heavy-weight processing modules (10MB RAM, 2000 MIPS) and tuples (2000 MIPS and 500 Byte network length). Both modules (VMs) and tuples (Tasks) are time-shared scheduled, with device scheduling interval (10 ms) and application scheduling interval (300 ms). Fog devices' CPU processing is: Mobile (1000 MIPS), Dept (2800 MIPS) and ProxyServer (16800 MIPS) to express the edge fog node lack of resources. Simulation time is up to 600000ms. The system is configured by the number of depts and the number of mobiles per each dept. The simulated (Depts/Mobile) are: 1D/3M, 2D/4M, 2D/6M, 2D/8M and 3D/10M. Performance Metrics include:

- Stream Analytics Loop Delay (ALD-Mean): the average delay for all tuples within the analytic loop for all users. The analytic loop executes the modules: ECG, client, filtering, monitoring and caregiver, in order.
- Stream Analytics Loop Delay User (ALD-User): the average delay for all tuples within the analytics loop for a single user.
- Standard Deviation of the Analytics Loop Delay (ALD-SD): for N users, the standard deviation is the root of variance:

$$\sigma\hat{}2 = \frac{1}{N}\sum_{i=1}^{N}(ALD_{Useri} - ALD_{Mean})^2$$

- Percentage of unsatisfied users per fog device: percentage of users with analytics loop delay greater than the delay threshold at this device. For SP, the max. allowed analytics delay for all devices should be less than or equal to the sensor's transmission rate.
- Device's Power Consumption (Watt/hour): measured by the device's utilization percentage over the simulation period.
- Total Network Usage (kByte).

8.1 Basic S-FARM Performance

8.1.1 At Various System Configurations

Figure 4 shows the analytics loop delay for ECG with a sensors' transmission rate of 50ms, at two user's Mobile CPU load percents (0.1% and 0.15%) of its max. CPU load. Results indicate that at 1D3M, per-user and per-module modes have the same analytics delay; where the modules are placed similarly in both algorithms. Analytics delay is acceptable (<50ms) under all system configurations for per-user and per-module modes with Mobile CPU load of 0.15%. Expanding the system to 3D8M, both per-user and per-module modes with Mobile CPU load of 0.1% encounter unacceptable analytics delay (>50ms), but per-user mode delay is within the reasonable limit.

Figure 5 shows the energy consumption of dept and proxy fog devices. Starting from 2D/6M configuration, per-module dept device flushes all its load to the proxy device and works by its idle power. At 3D/8M per-module mode, proxy device reaches its maximum allowable CPU load, flushes

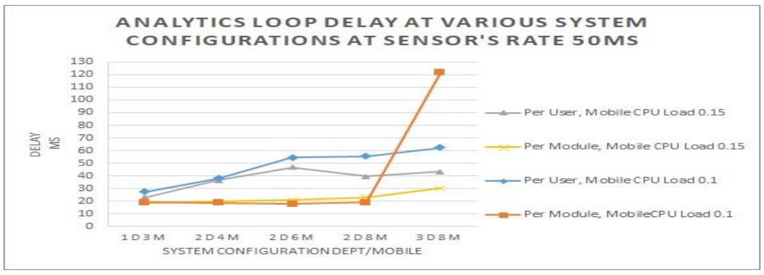

Figure 4. Analytics loop delay under various system configurations.

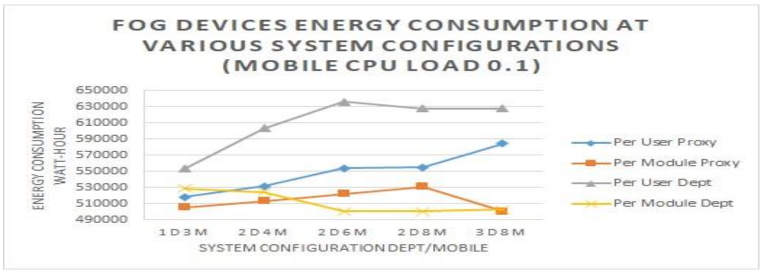

Figure 5. Fog devices energy consumption under various system configurations.

all its workload to the remote cloud and works also by its idle power, leaving the smart hospital fog system completely un-utilized. Per-user mode dept and proxy devices work under all system configurations; even if the device's maximum CPU load is reached, the device shifts up selected user's modules only to the higher fog device. Thus, per-user mode permits for safer system expansion than per-module mode. The same information is deduced from Figure 6 that shows the dept and proxy fog devices CPU utilization at 3D/8M configuration. Per-user devices have 100% CPU utilization, while pe- module devices have 0% CPU utilization. Cloud datacenter carries out the heavy cloud analytics module for batch analytics, consumes higher energy in per-user mode than in per-module mode. Thus, cloud datacenter by its massive resources is not preferable to work in per-user mode. We suggest a hybrid mode of operation, where the limited resource fog nodes can work in per-user mode to save the system expandability, where the massive resource nodes like the cloud datacenter should work in the pe- module mode to save its power consumption. Also, within the same fog node, the hybrid mode could be studied to optimize the fog infrastructure energy consumption while allowing for safe system expansion.

Figure 7 shows Mobile energy consumption. Results indicate that Mobile devices consume lower energy with per-user mode under all system configurations. Also, more energy is consumed when the mobile CPU load is 0.15% of its max. available CPU load.

8.1.2 At Various Sensor Transmission Rates

Figure 8 shows analytics loop delay under various sensor transmission rates, at 2D/4M and 3D/8M configurations. AT 2D/4M, all sensor transmission rates >=25ms are acceptable for per-module mode, while sensor transmission rates >= 40 ms are acceptable for pe- user mode. At 3D/8M, per-module analytics delay is acceptable for sensor rates >=500ms, while per-user analytics delay is acceptable for rates >= 100ms.

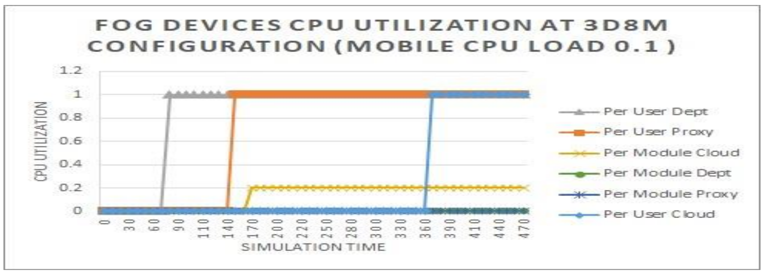

Figure 6. Fog devices CPU utilization at 3D/8M configurations.

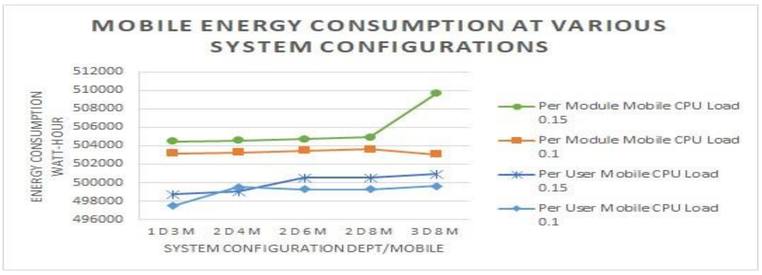

Figure 7. User mobile energy consumption under various system configurations.

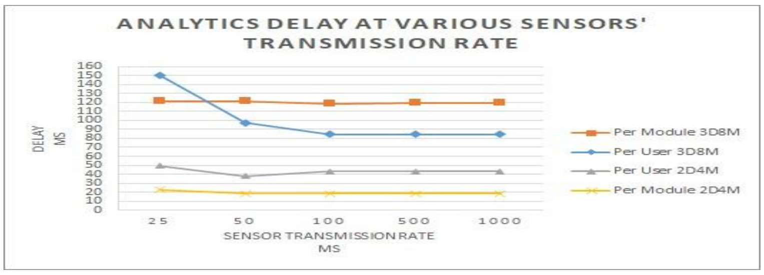

Figure 8. Analytics delay at various sensor transmission rates.

Figure 9 shows that at 2D/4M, per-user network usage is lower than that of per-module mode under all the sensor transmission rates. Same result is obtained under various system configurations and tuple MIPS.

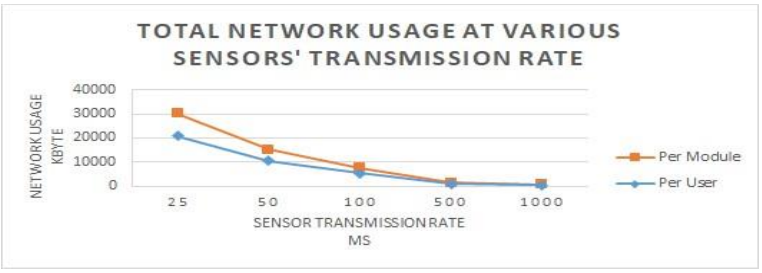

Figure 9. Total network usage at various sensor transmission rates.

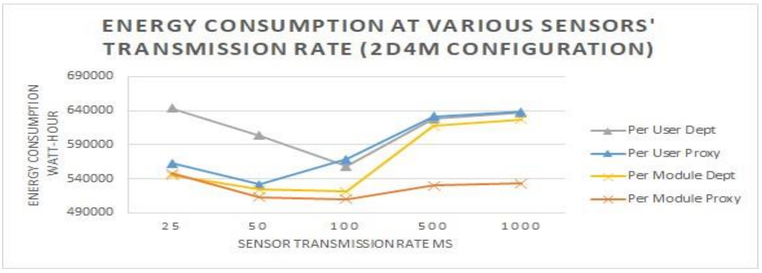

Figure 10. Fog devices energy consumption at various sensor transmission rates.

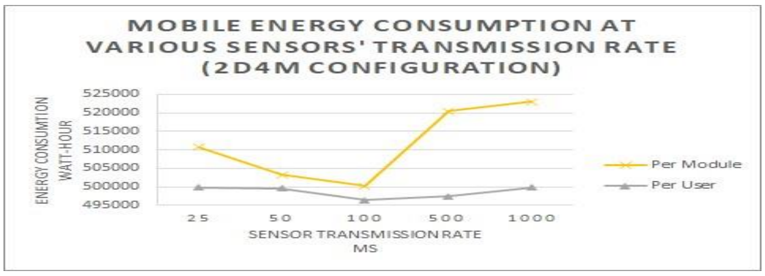

Figure 11. User mobile energy consumption at various sensor transmission rates.

Figure 10 shows the energy consumption of dept and proxy devices. Sensor transmission rate is not linearly affecting fog device energy consumption. Sensor rates 100ms gives the lowest devices' energy consumption under both per-user and per-module modes. The same information is deduced from Figure 11, where the minimum Mobile energy consumption is obtained at a sensor transmission rate of 100ms. Under all rates, per-user mode saves Mobile energy more than per-module mode.

8.1.3 At Various Analytics Tuples' MIPS

Varying the analytics modules (Filtering, Monitoring and Caregiver) and their analytics tuples between 500 and 4000 MIPS, while keeping the remaining modules constant to their original MIPS, Figure 12 shows delay under two situations. Situation1: transmission rate 50ms and 2D4M configuration, we found that delay is acceptable (<50ms) under all analytics tuples' MIPS for per -ser and per-module modes. Situation2: transmission rate 100ms and 3D8M configuration. In per-user mode, the variation in analytics tuples' MIPS doesn't linearly affect the delay and the delay is acceptable (<100ms) under all analytics tuples' MIPS. In per-module mode, delay is acceptable only for analytics tuples' MIPS <= 1000 MIPS.

Figure 13 shows the analytics tuples' MIPS limit that affects the dept utilization. Situation1, analytics tuples' MIPS >= 3000, makes per-module dept CPU utilization=0. Situation2, analytics tuples' MIPS >= 1000, makes per-module dept CPU utilization=0. All tuples' MIPS make per-user mode dept work by its full utilization.

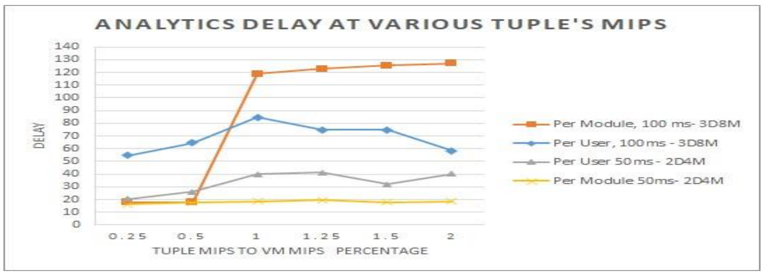

Figure 12. Analytics delay under various Tuple MIPS.

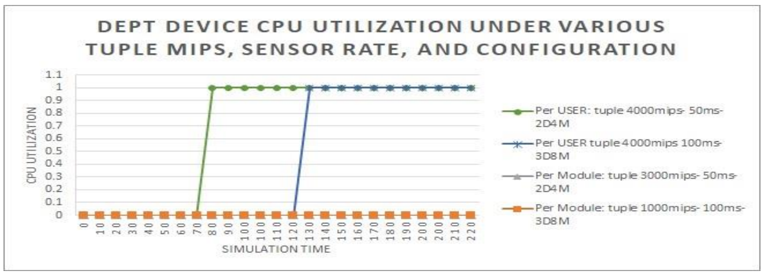

Figure 13. Dept CPU utilization limits under various Tuple MIPS.

Safe Stream Analytics under S-FARM at 3D8M Configuration

Tracing the analytics tuples' MIPS limit that is necessary to obtain an acceptable analytics delay while keeping the dept device working, we found that:

- Rate 50ms: per-user tuple MIPS <= 400; per-module tuple MIPS <= 1000.
- Rate 100ms: per-user tuple MIPS <= 4000; per-module tuple MIPS <= 1000.

8.2 Differentiated S-FARM Performance

Differentiated S-FARM is studied by considering the safe analytics delay limit at 3D8M for per-user basic S-FARM. Two situations have been simulated. Situation1: a random analytics tuples' MIPS between (500:4000) with sensor rate 100ms. Situation2: a random analytics tuples' MIPS between (50:400) with sensor rate 50ms. The average of 10 simulation runs is figured out at each case to test the dept device safe capacity under four configurations: 1D4M, 1D6M, 1D8M, 1D10M.

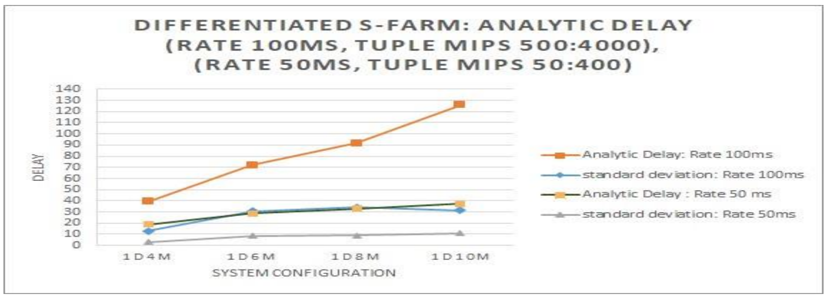

Figure 14. Differentiated S-FARM: analytics delay.

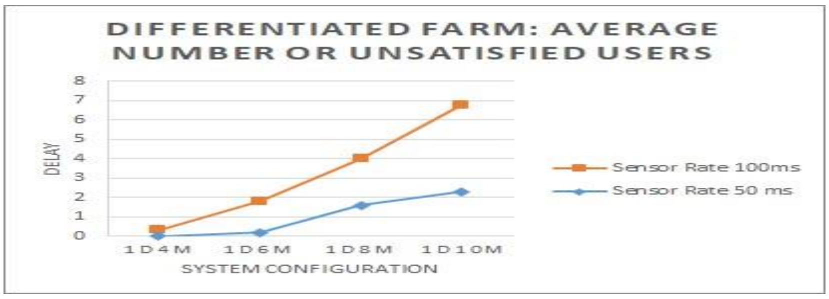

Figure 15. Differentiated S-FARM: average number or unsatisfied users.

Figure 14 shows that average analytics delay is acceptable up to 8 Mobiles per dept for 100 ms sensor rate, but the standard deviation is very high (30 ms) due to the variation in the users' analytics tuples MIPS and the delay is not acceptable for 4 users (50%) on average, as seen in Figure 15, while the average analytics delay is acceptable up to 10 Mobiles per dept for 50 ms sensor rate, but the standard deviation is 10 ms and the delay is not acceptable for only 2 users (20%) on average, as seen in Figure 15.

WORK LIMITATION

To minimize the number of unsatisfied users, performance monitoring should be done on the application run. Analytics delay should be calculated at each fog device to discover the risky users and the risky devices that may cause a problem. If a user's delay at a device exceeds his allowable analytics delay at this device, the device resources should be managed by reallocating more resources to that user (enhanced time-shared scheduling). This will be implemented by the Dynamic FARM (D-FARM) algorithm that represents the YARN ResourceManager (ApplicationManager). It monitors the performance and reschedules resources either locally or by migration to the risky user VMs.

9. CONCLUSIONS AND FUTURE WORK

Fog Assisted Resource Management FARM platform based on YARN for compatible short-term/long-term data analytics is presented. S-FARM is proposed using per-user and per-module modes; it represents the YARN ResourceManager (Schedulers). S-FARM schedulers are simulated over iFogSim. Results indicate that although per-module scheduler minimizes the analytics delay and energy consumption, it is a risky scheduler. It leaves fog devices un-utilized in case that it encounters a shortage in its CPU resources. Per-module scheduler shifts up the whole module to a higher fog device, if there is an increase in: the number of users, the sensor transmission rate or the analytics tuple MIPS. In addition, per-module consumes the user's mobile energy and the network usage more than the per-user scheduler under all the simulated scenarios.

Conducting stream analytics over the poor resources fog nodes, per-user scheduler allows for safer system expansion. If there is a shortage in the device's CPU resources, selected users' modules only could be shifted up to the higher fog device in the path hierarchy. Although being better for the fog infrastructure utilization, per-user scheduler has an average analytics loop delay higher than in per-module mode. Per-user analytics tuples' MIPS should be adjusted carefully under the variable system configurations and transmission rates to allow for a safe stream analytics platform that avoids losing any sensor reading or violating the stream analytics restrictions of the continuous query result.

Managing the fog network resources, differentiated S-FARM scheduler is the best methodology for the per-user control to the live analytic results, as it allows users to request their specified analytics tuple MIPS and thus their analytics QoS. Heavy-weight analytics tuples allow accurate analytics results, while light-weight analytics tuples allow fast analytics results and allows accommodating more users per fog device.

The application’s mean analytics delay and the standard deviation are not sufficient parameters to judge the resource management algorithms’ performance. Maximum users’ analytics delay should also be figured out at each simulation run, in order to figure out whether there is any loss in the user’s stream.

The future work is to minimize the number of unsatisfied users of the differentiated S-FARM algorithm by monitoring the stream analytics application and fog infrastructure performance, as well as to propose Dynamic FARM (D-FARM) that represents the YARN ResourceManager (ApplicationManager) with enhanced time-shared scheduling algorithm to support per-user differentiation.

REFERENCES

[1] 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.

[2] R. A. Shehab, M. Taher and H. K. Mohamed, "Fog Enabled Health Informatics System for Critically Controlled Cardiovascular Disease Applications," Proceedings of the International Conference on Health Informatics & Medical Systems, pp. 35-41, ISBN: 1-60132-500-2, Copyright ' 2019 CSREA Press, United States of America, 2019.

[3] K. H. Abdulkareem, M. A. Mohammed, S. S. Gunasekaran, M. N. Al-Mhiqani, A. A. Mutlag, S. A. Mostafa, N. S. Ali and D. A. Ibrahim, "A Review of Fog Computing and Machine Learning: Concepts, Applications, Challenges and Open Issues, " IEEE Access, vol. 7, pp. 153123–153140, 2019.

[4] S. Parveen, P. Singh and D. Arora, "Fog Computing Research Opportunities and Challenges: A Comprehensive Survey," Proceedings of the 1st International Conference on Computing, Communications and Cyber-Security (IC4S 2019), Part of the Lecture Notes in Networks and Systems Book Series LNNS, vol. 121, pp. 171–181, DOI: 10.1007/978-981-15-3369-3_13, Springer Singapore, 2020.

[5] R. A. Shehab, M. Taher and H. K. Mohamed, "Resource Management Challenges in the Next Generation Cloud Based Systems: A Survey and Research Directions," Proc. of the 13th International Conference on Computer Engineering and Systems (ICCES), pp. 139–144, Cairo, Egypt, Dec. 2018.

[6] A. A. Mutlag, M. K. Abd Ghani, N. Arunkumar, M. A. Mohammed and O. Mohd, "Enabling Technologies for Fog Computing in Healthcare IoT Systems," Future Generation Computer Systems, vol. 90, pp. 62 – 78, 2019.

[7] A. A. Mutlag, M. Khanapi Abd Ghani, M. A. Mohammed, M. S. Maashi, O. Mohd, S. A. Mostafa, K. H. Abdulkareem, G. Marques and I. de la Torre D´ıez, "MAFC: Multi-agent Fog Computing Model for Healthcare Critical Tasks Management," Sensors, vol. 20, no. 7, Article no. 1853, DOI: 10.3390/s20071853, 2020.

[8] H. Gupta, A. Vahid, A. 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.

[9] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa and I. Yaqoob, "Big IoT Data Analytics: Architecture, Opportunities and Open Research Challenges," IEEE Access, vol. 5, pp. 5247–5261, 2017.

[10] H. Dubey, J. Yang, N. Constant, A. M. Amiri, Q. Yang and K. Makodiya, "Fog Data: Enhancing Tele-health Big Data through Fog Computing," Proceedings of the ASE Big Data &amp; Social Informatics, DOI: 10.1145/2818869.2818889, New York, USA, Association for Computing Machinery, 2015.

[11] 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, 2017.

[12] X. Liu, A. Dastjerdi and R. Buyya, "Chapter 8 - Stream Processing in IoT: Foundations, State-of-the-art and Future Directions," Internet of Things By: R. Buyya and A. V. Dastjerdi, Eds., pp. 145 – 161, Morgan Kaufmann, 2016.

[13] S. Kamburugamuve and G. C. Fox, "Survey of Distributed Stream Processing for Large Stream Sources," Proc. of SPIDAL: CIF21 DIBBs: Middleware and High Performance Analytics Libraries for Scalable Data Science, DOI: 10.13140/RG.2.1.3856.2968, 2016.

[14] Apache Storm, "Storm Apache," [Online], Available: http://storm.apache.org.

[15] Apache Spark, "Spark Apache," [Online], Available: https://spark.apache.org.

[16] Apache Samza, "Samza Apache," [Online], Available: https://samza.apache.org.

[17] Apache Flink, "Flink Apache," [Online], Available: https://ink.apache.org.

[18] Apache Hadoop, "Apache Hadoop," [Online], Available: http://hadoop.apache.org.

[19] Edureka, "Hadoop Tutorials," [Online], Available: https://www.edureka.co/blog/hadoop-tutorial.

[20] V. Cardellini, V. Grassi, F. L. Presti and M. Nardelli, "On QoS-aware Scheduling of Data Stream Applications over Fog Computing Infrastructures," Proc. of the IEEE Symposium on Computers and Communication (ISCC), pp. 271–276, Larnaca, Cyprus, July 2015.

[21] A. Papageorgiou, E. Poormohammady and B. Cheng, "Edge-Computing-Aware Deployment of Stream Processing Tasks Based on Topology-external Information: Model, Algorithms and A Storm-based Prototype," Proc. of the IEEE International Congress on Big Data (BigData Congress), pp. 259–266, San Francisco, USA, June 2016.

[22] L. Aniello, R. Baldoni and L. Querzoni, "Adaptive Online Scheduling in Storm," Proceedings of the 7th ACM International Conference on Distributed Event based Systems, (DEBS 13), pp. 207–218, New York, USA, ACM, 2013.

[23] C. Hochreiner, M. Vogler, S. Schulte and S. Dustdar, "Elastic Stream Processing for the Internet of Things," Proceedings of the IEEE 9th International Conference on Cloud Computing (CLOUD), pp. 100–107, San Francisco, USA, June 2016.

[24] N. Maleki, M. Loni, M. Daneshtalab, M. Conti and H. Fotouhi, "SoFA: A Spark-oriented Fog Architecture," Proc. of the 45th Annual Conference of the IEEE Industrial Electronics Society (IECON 2019), vol. 1, pp. 2792–2799, Lisbon, Portugal, 2019.

[25] T. N. Gia, M. Jiang, A. Rahmani, T. Westerlund, P. Liljeberg and H. Tenhunen, "Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction," Proc. of the IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pp. 356–363, Liverpool, UK, Oct. 2015.

[26] H. Chen and H. Liu, "A Remote Electrocardiogram Monitoring System with Good Swiftness and High Reliability," Computers & Electrical Engineering, vol. 53, pp. 191 – 202, 2016.

[27] K. Wac, M. S. Bargh, B. F. V. Beijnum, R. G. A. Bults, P. Pawar and A. Peddemors, "Power -and delay- Awareness of Health Telemonitoring Services: The Mobihealth System Case Study," IEEE Journal on Selected Areas in Communications, vol. 27, pp. 525–536, May 2009.

[28] X. Masip-Bruin, E. Mar´ın-Tordera, A. Alonso and J. Garcia, "Fog-to-cloud Computing (F2C): The Key Technology Enabler for Dependable e-Health Services Deployment," Proc. of the Mediterranean *Ad Hoc* Networking Workshop (Med-Hoc-Net), pp. 1–5, Vilanova i la Geltru, Spain, June 2016.

[29] C. Rotariu, V. Manta and H. Costin, "Wireless Remote Monitoring System for Patients with Cardiac Pacemakers," Proc. of the International Conference and Exposition on Electrical and Power Engineering, pp. 845–848, Iasi, Romania, Oct. 2012.

[30] G. W. Nkabinde, "Big Data Stream Computing in Healthcare Real-time Analytics," Proc. of the IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), pp. 37–42, Chengdu, China, July 2016.

[31] E. Badidi and K. Moumane, "Enhancing the Processing of Healthcare Data Streams Using Fog Computing," Proc. of the IEEE Symposium on Computers and Communications (ISCC), pp. 1113–1118, Barcelona, Spain, 2019.

[32] L. Greco, P. Ritrovato and F. Xhafa, "An Edge-stream Computing Infrastructure for Real-time Analysis of Wearable Sensors Data," Future Generation Computer Systems, vol. 93, pp. 515 – 528, 2019.

ملخص البحث:

يتضمّن تحسين أنظمة المراقبة الصحية القائمة على إنترنت الأشياء والمستخدمة في بيئات متنوعة -مثل المنازل الذّكيّة والمستشفيات الذّكيّة- التّحليل الحيّ لِسَيْل البيانات الحرج (مثل سَيْل بيانات مخططات القلب الكهربائية). والجدير بالذّكر أن إجراء هذه التطبيقات الصحية عن بُعد بواسطة السّحابة التّقليدية أمْرٌ يخرق المحدِّدات النهائية للتّطبيقات القائمة على تحليل سِيُول البيانات، الأمر الذي ليس من شأنه أنْ يؤدي الى تدهور الأداء فحسب، بل أيضاً الى نتائج غير دقيقة لعمليات التّحليل نظراً لفقدان بيانات المريض. ويمكن للحوسبة الضبابيّة أن تتم على مقربة من المريض، وهي أبرز التقنيات المفضلّة لهذا النوع من التّطبيقات.

يتم توزيع العُقَد الضّبابيّة جغرافياً وتكون فقيرةً في الموارد؛ لذا فإنّ الحاجة الى منصّةٍ لإدارة الموارد قابلة للتّوسعة وقادرة على التّعامل مع الأخطاء تُصبح أمراً واجباً. وفي الوقت الرّاهن، يتم تصميم أنظمة إدارة لمعالجة سُيول البيانات للتّعامل مع أعداد ضخمة من عُقد الموارد وتوظيفها بدلاً من العُقد الضّبابيّة فقيرة الموارد؛ ممّا يعالج النّقص في استغلال بنية النّظام. وهناك حاجة ماسّة الى تصميم منصات إدارة موارد قادرة على مواكبة طبيعة الحوسبة الضّبابيّة.

نقترح في هذه الورقة منصّة إدارة موارد مبتكرة تستند الى الحوسبة الضّبابيّة بناءً على إدارة الموارد باستخدام نظام يارن (YARN) من (Apache Hadoop2) للتّحليلات المتوافقة مع سُيول/حُزم البيانات وتمثل منصّة (S-FARM) مُجَدْوِلين من نوع (YARN)؛ أحدهما يعمل وفْق المستخدِمين والثاني يعمل وفْق الوحدات.

وتُظهر النّتائج أنّ المجَدْوِل الذي يعمل وفْق المستخدِمين يتغلّب على مشكلة نقص الموارد التي تعاني منها العُقَد الضّبابيّة، ويستغلّ بشكلٍ كامل البنية التّحتيّة الضّبابيّة، ويسمح للنّظام بالتّمدّد بأمانٍ الى مِثْلَي حجمه. إضافة الى ذلك، يتم اقتراح مُجَدْوِل YARN من طراز (S-FARM) التفاضليّ لدعم التحكّم في نمط الاستخدام وفْق المستخدِمين من حيث دقّة نتائج التّحليل وسرعة الحصول عليها. ويستخدم تطبيق (S-CVD) لتحليل مخطّطات القلْب الكهربائية للمرضى، وقد تم إجراء محاكاةٍ له في (iFogSim) للتّحقق من أداء مُجَدْوِلي (YARN) المقترحين.

ويُعدّ هذا البحث رائداً في تحسين الاستغلال الفقير للموارد للعُقد الضّبابيّة، ودعم التّحكّم في نمط الاستخدام وفْق المستخدِمين في تطبيقات تحليل البيانات الضخمة استناداً الى إنترنت الأشياء، والاستفادة من (iFogSim) في تقييم أداء منصّات إدارة الموارد في أنظمة تحليل سيول البيانات.