Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 155 (2019) 234–241

Procedia Computer Science

www.elsevier.com/locate/procedia

The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC)
August 19-21, 2019, Halifax, Canada

Container Based Resource Management for Data Processing on IoT Gateways

Bali Ahmed[a,*], Billel Seghir[a], Mahmud Al-Osta[a], Gherbi Abdelouahed[a]

[a]*Department of Software and IT Engineering, École de technologie supérieure, Montreal, H3C 1K3, Canada*

Abstract

IoT is featured by its diversity and heterogeneity in terms of hardware and software components shaping its systems. This in turn burdens the process of deploying and updating hardware components and their corresponding software elements. Moreover, these devices are constantly generating data and sending them to central cloud platforms for advanced processing, with billions of devices are anticipated to be connected in the coming years, this will lead to impact the network performance. To alleviate this concern, a recent tendency is to process these data locally close to their sources at the network edge. However, edge devices are likely to be resource-constrained, which in some cases limits their abilities to perform processing tasks. This work falls within the efforts attempt to provide a solution to enable an effective management of resources at the edge devices, as well as, facilitating the deployment and update of heterogeneous software modules. The proposed solution relies on containers, a lightweight virtualization technology. It takes advantage of some containers concepts, especially the orchestration concept, to enable task forwarding and resource sharing between IoT gateway devices within the same cluster. Our experimentation has shown promising results and revealed the potential of existing virtualization platforms.

Keywords: Internet of Things (IoT), Containers, Docker, Docker Swarm, Resource Management;

1. Introduction

Nowadays, IoT networks have become ubiquitous and an essential part of our life, the growth of connected objects is growing more and more. According to [6], by 2020 IoT networks will connect tens of billions of devices all around the glob. These networks are deployed to serve variety of applications including healthcare, smart homes and industry automation. IoT applications rely on an infrastructure, which is featured by its heterogeneity in terms of the technology stack employed in different layers of such type of systems. Moreover, IoT networks are composed of spectrum of devices ranges from low-cost tiny devices with very limited computing resources to central cloud servers with plenty of resources. With billions of devices are expected to be connected to the Internet, resulting in generating a massive amount of data, which in conventional IoT architectures are mostly to be transferred to the cloud for processing; while

* Corresponding author. Tel.: +1-438-990-2401 ; fax: +1-514-840-5514.
E-mail address: ahmed.bali.1@ens.etsmtl.ca

1877-0509
10.1016/j.procs.2019.08.034

results, in some cases, are required to be sent back to sensor nodes for actuation purposes. This process leads to impose a remarkable pressure on the network resources due to the increasing traffic volumes. In addition, this is likely to raise the data transformation cost and degenerate the network performance.

In order to reduce the load on IoT networks, a recent tendency is to process data close to their resources at the network edge. Edge computing is deemed to improve the overall performance of IoT networks by delegating some cloud tasks to distributed edge devices. In the context of this research study, our approach targets the level of gateway devices as part of the IoT edge networks. Recently, a significant efforts have been dedicated to improve their computing and communication capabilities; thus enabling them to perform some reasonable data processing tasks close to their resources at the network edge. This in turn, is likely to lighten the network traffic, which minimizes latency and enhances the overall network performance [3]. However, in some cases where gateway devices receive data packets that might not be compatible with their limited resources could lead to resources overhead, which in turn leads to network failures. This could be critical especially in some scenarios such as healthcare and surveillance applications. Furthermore, the diversity of sensors connected to gateways hinders the process of deploying new services or updating current ones.

These issues have been the subject of several research projects to propose a solution for an efficient management of the resources offered by IoT gateway devices, and to develop approaches to facilitate the deployment of services on these heterogeneous devices independently. Our main objective in this paper is to propose a solution to address both resource management and deployment. Thus, this work aims to provide an efficient resource management mechanism by enabling resource sharing between IoT devices, as well as, simplifying and accelerating the deployment process of services on IoT devices. The solution proposed in this work is based on lightweight virtualization technologies, such as Docker containers to overcome the challenges of resource limitations and services deployment. In terms of deployment, this solution takes the advantages of the containers technology that are reusable, easy to duplicate, and simple to migrate. As for the resources management, containers are lightweight, reasonable in terms of resource consumption. Furthermore, container-based solutions can significantly contribute to alleviate heterogeneity issue in IoT environments since they enable deployment and communication between different software modules regardless of the framework underneath. Also, containers can be customized to cope with resources available on the target node. Moreover, container orchestration techniques such as Swarm enable resource sharing between IoT devices and provide means of communication between containers across virtual networks.

The reminder of the paper is organized as follows. First, Section 2 discusses the related work with respect to the approach proposed in this work. Then, the architecture of the proposed solution will be presented in Section 3. While Section 4 discusses the case study considered for the evaluation purpose. Followed by the evaluation and experimental results discussion in Section 5. Finally, section 6 concludes the paper and highlights directions for future work.

2. Related Work

The use of containers has been recently an attractive subject of research for both industry and academia. In this context, some research initiatives have been proposed to evaluate the feasibility of employing containers on the edge of IoT networks taking into consideration their limited resources. For instance, [7], [8], and [11] have carried out research studies to explore container virtualization technologies and their rule as an effective means to facilitate the development and to scale-up IoT applications. To assess the impact of containers on the IoT devices resources, IoT applications were developed and deployed in the form of Docker containers, then Benchmark tools were used to measure their performance. Some evaluation criteria have been considered such as service deployment and management, resource consumption, and fault tolerance. The results show that the use of Docker containers offers good performance. However no clear approach is described for effective resource management. Although Docker brings advantages for resource management in an IoT context, it is only a working tool. Thus, it is necessary to describe an effective approach for this purpose. In the following, we investigate some recent work in comparison with the approach presented in this study.

A design of data processing approach is proposed in [9], it is edge oriented and its functional components are customized as reusable Docker containers. This facilities building versatile environment for IoT applications, and enables elastic provisioning of distinctive device and data management, as well as, orchestration capabilities. While their implementation is determined on the evaluation of the CPU, Memory, and Network usage, it lacks a concrete assessment of the orchestration and data processing modules, which is extensively considered in our work.

A container-based resource allocation model was proposed in [10] to increase the use of resources offered by IoT devices and reduce the generated network traffic. The proposed approach allows different applications and users to dynamically allocate the resources offered by IoT devices and thus maximize data processing at the source level instead of sending them to the cloud. The feasibility of this approach has been evaluated in terms of performance since IoT devices are limited in terms of resources. However, the case study considered in this work is focused on presenting the feasibility of the approach and not on improving the performance of resources such as CPU and memory.

A resource management and orchestration approach based on containers is proposed in [5] for supporting autonomic data stream processing applications on the Fog layer. Fog nodes (FNs) are equipped by three main components namely Fog Node Controllers (FNCs), Autonomic Applications (Apps), and Application Controllers (ACs). Apps are encapsulated in the form of docker containers; each of them runs an AC. The AC interacts with the FNC of the corresponding FN to facilitate scaling up/down the set of resources (e.g., number of cores, CPU time, and bandwidth) assigned to the Docker container. Unlike our work, their implementation relies on rich-resource infrastructure and does not take into consideration scenarios where the Fog node is resource-limited, which is the case in a considerable portion of IoT applications.

3. The Proposed Approach

In this work, a solution based on container virtualization technology has been proposed to overcome device limitation and deployment issues. This approach relies on some container clusters concepts to promote resource sharing between devices and enable easy and independent deployment of the hosting infrastructure. Typically, IoT topology is designed on the top of three essential layers namely sensor nodes, gateways and the cloud. The sensor nodes continually send the data to the gateways that store and process it if necessary, and then send it to the cloud for use as needed. Mostly, gateway receives data from multiple sensor nodes, and because of IoT device resource limitations, some gateways become overloaded and can no longer process data. While within the same cluster, it is likely to find some gateways that are not overloaded gateways and have some available resources. As a result, the work on resource management targets the workload distribution between the gateways that represent the essential element in our approach.

3.1. Overall Architecture

In order to fully take benefits of resources available on IoT gateway devices, they will be grouped into clusters. Each cluster is composed of a number of gateways with one as manager while the rest will be workers. The manager node, as indicated by its label, will take care of administering the other worker nodes, as for the other gateways, they will just employ their resources to deploy the services. Each gateway will be equipped with a container framework and its orchestration mechanisms that are responsible on services deployment and communication within the same cluster. This system decomposition will allow the gateways belonging to the same group to share the resources between them and communicate through a virtual network. Usually each gateway receives data from one or more sensor nodes, but in the case of an overload, depending on our approach the data processing could be delegated to another gateway within the same cluster in order to fairly distribute the load. Each gateway of the system will contain three services namely read service, processing service and sending service, these constitute the workflow of data processing coming from the nodes of sensors.

In order to facilitate the deployment of services on the gateways, their images are created with all their dependencies (libraries, version, etc.) and stored in a registry, the latter is a centralized service directory shared between all the gateways. So to deploy a service in a gateway, we need just to access the registry and draw the image corresponding to the service. With such a procedure, it would be possible to overcome the problem of heterogeneity of the gateways, since the services are created with all their dependencies and ready to be executed in containers regardless of the infrastructure of the host. Having a centralized repository will also simplify the update, because users just need to make a new build of the image subject to an update, instead of going through all the gateways and make an update.

Figure 1 shows the high-level architecture of our system, the details of the essential components will be detailed in the following sections. As illustrated, gateways are grouped into a set of clusters. In each cluster, there is a manager gateway (in green) that manages the other gateways. The sensor nodes (in red) communicate directly with the

gateways. The gateways of the same cluster communicate with each other through an overlay network. While they communicate directly with the registry to instantiate services and execute them in containers from existing images.

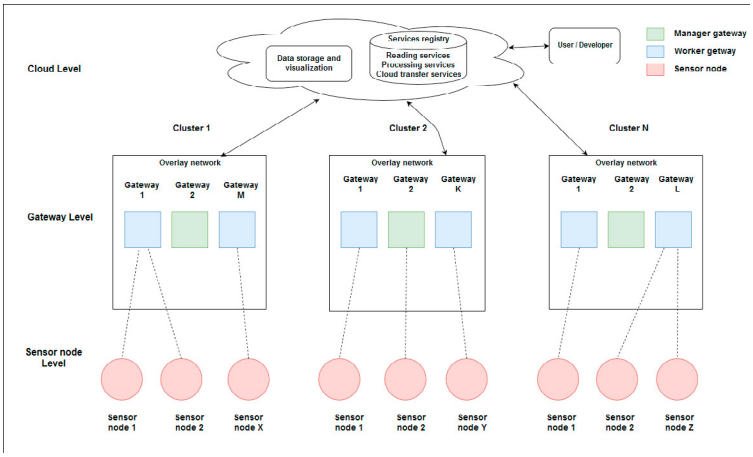

Fig. 1. The Overall Architecture

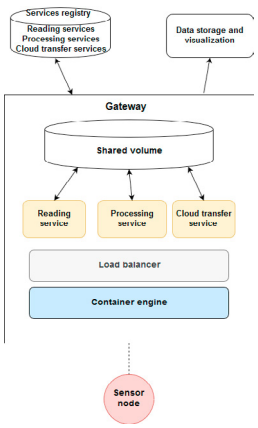

Fig. 2. The Gateway level Architecture

3.2. Gateway level

In this work, some importance is given to the gateway as it represents the core element of our solution. A gateway could be either Manager or Worker, but they both share the same architecture. The sole difference lays on the advantage assigned to the manger gateway; so it can run commands. Figure 2 shows the main elements of the gateway architecture. Each gateway is equipped with a container engine to enable services to run as packaged containers. The chosen engine must be able to support the cluster composition. Also, each gateway is equipped with a load balancer, the latter will allow to distribute the load on the same service instances located in the gateways of the same cluster.

A gateway contains three services that run continuously as containers and constitute the data processing workflow from the sensor nodes. These three services are:

- Data Read Service: This service receives data packets from sensor nodes on predefined interval time; then it will store them in a given directory of the volume shared between the services deployed on the gateway. Each data packet received includes SensorID, measured value, and Timestamp.
- Data Processing Service: This service communicates with the volume directory to accesses collected data and to apply data processing tasks required by the user. In our case, we consider the semantic annotation as the processing task as presented in Section 4.
- Cloud data transfer service: The main objective of this service is to facilitate communication between the gateway level and the cloud level. This communication aims at enabling messages exchange among the two levels. It includes tasks such as sending processed data to the cloud, and receiving deployment and updating requests from the cloud.

As shown in Figure 2, the three aforementioned services use a shared volume to communicate. The latter contains directories for storing the collected data and also for those that are processed. So each service uses the proper directory to accomplish its tasks. While in the case of a gateway manager, it communicates directly with the registry to deploy the services on the gateways, save new images or update existing ones.

3.3. Cluster Functionality

Figure 3 illustrates the distribution of requests in a cluster consisting of three gateways. Each gateway is identified by an IP address. As we explained in the gateway architecture, there are three services shaping the data handling workflow. Since the data reading service is the first component of the data treatment services chain, so it is implicitly realizable that the distribution of reading tasks would automatically lead to decreasing the amount of data to be treated

in the following services on each gateway individually. With the launch of the system, the gateways begin to receive data from the sensor nodes. The load balancer on each gateway receives requests from the sensor nodes, then transfers them to the data read service of the same gateway, or redirects them to another read service instance of another gateway. This division of tasks is based on an overlay network. The data processing service of a gateway accesses the shared volume to retrieve the data delivered by the read service of the same gateway, then it will process them and insert them again in the shared volume. Afterwards, the cloud data transfer service accesses the shared volume to retrieve the data processed by the data processing service of the same gateway, and then transfers it to the cloud. Thus, in the case of IoT system composed of clusters of gateway devices, each cluster will work independently following the aforementioned steps. The goal behind this division of tasks is to have a balanced system; thus avoiding situations where some gateways are overloaded; while others are relatively unburdened. The next part will show us the impact of this solution on gateway resources.

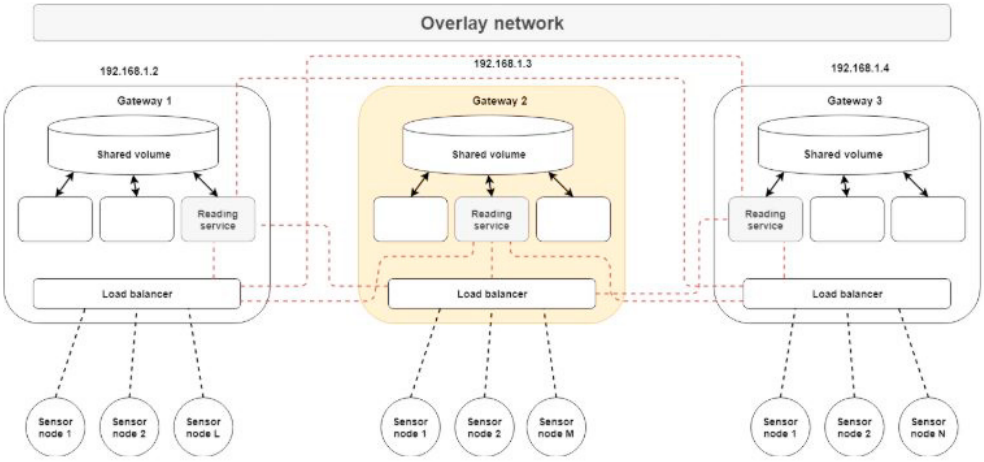

Fig. 3. Cluster operation.

4. Case study

In order to evaluate the feasibility and effectiveness of our approach to resource management, we chose a case study in the context of IoT where data processing is very resource intensive. In this work, the semantic annotation is considered as the data processing task. It involves adding some meaningful description to raw data captured by different sensor nodes; thus, they become easily consumable by different machines and applications. Moreover, the annotation process promotes the interoperability between different IoT systems that rely on set of heterogeneous sensors to collect the data. This significantly contributes to the development of new services regardless of the technology underneath leading to mitigating the vertical IoT data silos issues.

In our previous works [1] and [3], we have addressed issues related to the data annotation process in the context of IoT. The main issue that has been investigated is the annotation process which is too resource demanding; this is not appropriate for IoT devices characterized by their limited resources. The first cause of this high resource consumption is the use of Semantic Web technologies that have been widely adopted to model and integrate data from different sources on the Web [4]. These technologies are considered heavy for IoT purposes, as they often require a significant amount of computing capacity. This situation may become worse in scenarios where there are a large number of sensors (data sources) that should be involved, which is often the case with IoT applications such as the smart city [2].

The main objective of our previous work falls within the attempts to minimize the resource consumption of the annotation process. However, this minimization is applied independently to each gateway. As a result, some gateways could receive larger amounts of data than others, which implies it consumes more resource for data processing tasks. Therefore, in such a situation, over time some gateways will no longer be able to process data from the sensors. Having other gateways available in terms of resources could be an alternative to solve this problem by exploiting

their resources. This then shows the need to apply a load balancing approach to distribute the annotation task equally between the different gateways within the same cluster.

In the following section, we will show and evaluate the application of our container-based resource allocation (i.e. resource management) approach.

5. Implementation and evaluation

To evaluate the impact of our approach on the resources offered by IoT devices, we used Docker technology and its Docker Swarm concept to implement the case study described in Section 4. Docker is used to create the containers, while Docker swarm allows orchestrating the devices containing Docker by forming Swarms. Table 1 represents the mapping between the architecture concepts (presented in figures 1 and 2) and the technical concepts of the implementation:

Table 1. Mapping between architectural and technical concepts

Architecture concept	Technical concept
Container	Docker container
Cluster	Docker swarm
Manager	Manager node
Worker	Worker node
Service registry	Docker Hub
Overlay network	Ingress network
Container engine	Docker container engine
Load balancer	Docker swarm load balancer
Service	Python program with its dependencies in a container Docker.

5.1. Implementation and configuration

To test our approach, we have carried out experimentations using five virtual machines distributed over two physical machines, which are configured as follows:

- On each virtual machine, a Linux operating system Ubuntu 18.04 and Docker Engine 18.01 have been installed. A Docker Swarm is used to create a cluster of five virtual machines; one of them is deemed as a manager node (VM5), while the rest are workers.
- An overlay network has been created between the Swarm machines to enable them to communicate with each other in the same cluster. On each machine, three services were deployed, a data reading service, a data annotation service and a service for sending data to the cloud. To enable data sharing among services, a volume has been created on each node. A Python program has been developed to simulate the process of data generation and transmission from sensor nodes.
- The experimentations have been conducted in two scenarios. First, the simulation programs have been configured in a way to send data only to VM1 and VM2 with a given data load and without the use of the approach. While in the second scenario, the same data load was applied on VM1 and VM2, but with the use of our approach.
- Other services have also been deployed to collect and store performance data of each (i.e. used metrics are CPU, RAM, and Network). Some configurations have been made to initialize the system and others to allow the export of performance data of different machines.

Therefore, during the first scenario of the experiment (i.e. without the use of the approach), all data are processed by VM1 and VM2, then in the second scenario (with the use of the approach), the data are distributed over all Swarm machines (i.e., VM1, VM2, VM3, VM4 and VM5). During both scenarios, CPU and memory usage and inbound/outbound network traffic data are collected every 30 seconds.

5.2. Results Discussion

Due to page limit, we will only present the results of VM1 and VM3, since they should have a different behaviour in both experimentation scenarios. The results presentation and discussion are organized to subsections; each of them represents an evaluation metric.

5.2.1. CPU

As the data load is only applied on VM1 and VM2, so before using the approach these two machines process all the data without distributing them to other machines. After using our approach, VM1's CPU consumption decreased, since in this case the data is distributed over all machines, therefore VM1 processes less data (Figure 4).
Before using the approach, the data are processed only by the VM1 and VM2, so we note that there is only a minimal CPU consumption that returns to the monitoring services. After using the approach, the CPU consumption of VM3 increased, since in this case, it participates in data processing (Figure 5).

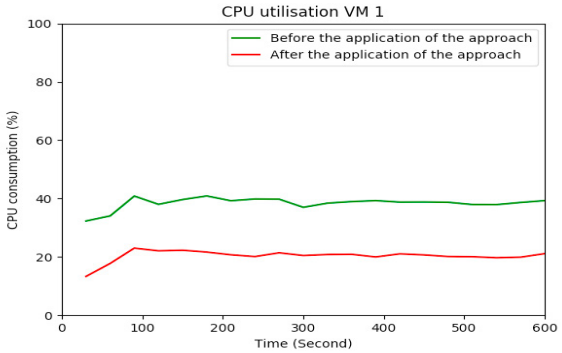

Fig. 4. CPU utilization of VM1

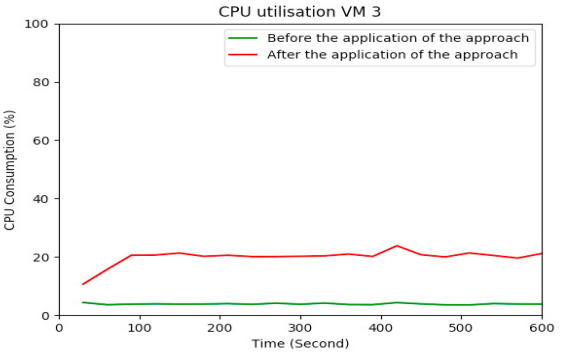

Fig. 5. CPU utilization of VM3

5.2.2. Memory

The largest consumption of memory is produced by monitoring services, which are resource-intensive. These services are used only for the sake of the evaluation to collect performance metrics and will not be part of the final system.The memory consumption of VM1 decreased slightly after the approach using the approach, as data processing is distributed across all machines in the cluster (Figure 6). Conversely, the memory consumption of VM3 increased shortly after using the approach, as it is involved in processing data from sensor nodes. Before using the approach, data processing is performed only on VM1 and VM2 (Figure 7).

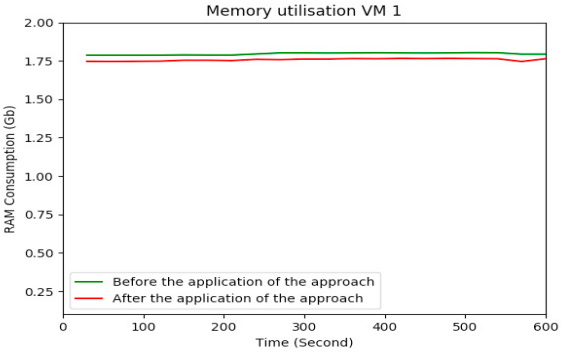

Fig. 6. Memory utilization of VM1

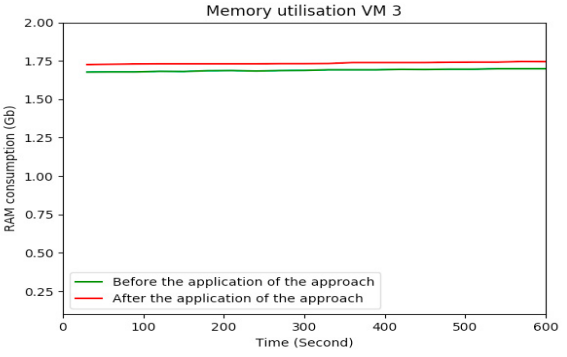

Fig. 7. Memory utilization of VM3

5.2.3. Network

There is a constant network traffic resulted from the continuously data sent by the monitoring services.The network traffic of VM1 containers is higher in the first scenario, because in this period all data load was applied only on VM1 and VM2, while in the second scenario, traffic decreased, since the load was distributed on all Swarm machines. During the first scenario of the experiment, VM3 network traffic remained constant, since before using the approach, all data from the sensor nodes are processed only by VM1 and VM2. During the second scenario of the experiment, VM3 network traffic increased, since this time, after using the approach, this machine also participates in the data processing. Due to the page limit, figures that show the network evaluation metric have not been listed.

6. Conclusion

Edge devices have conveyed significant advantages to IoT networks by taking the charge of performing some cloud tasks, by which alleviating the workload imposed on the network. However, their nature in terms of resources limitation and diversity of technologies employed have imposed new challenges regarding resource management and service deployment. Our studies in this area of research have led us to propose a solution based on lightweight virtualization technologies based on containers to facilitate efficient resource management and service deployment on IoT gateway devices. The deployment using containers makes this operation independent of the host's infrastructure, as well as, containers facilitate the update process since they enable performing modifications on their corresponding images stored in centralized directory. The architecture of the proposed solution, as well as, its main components and their functionalities was presented. To validate the effectiveness of the approach in managing device resources, results of an experiment were presented showing the values of a few metrics with and without the use of the approach. The implementation of the approach is based on the concepts of Docker framework and the Swarm mechanism for containers creation and orchestration. The obtained results show a decrease in CPU, memory and network traffic usage by the gateways when requests from sensors are distributed among all the gateways making up the cluster, resulting in a balanced system capable of processing as much data as possible. However, the cluster's network traffic is increased, due to the fact that the division of tasks between the gateways of the cluster creates new communications.To investigate the real effectiveness of the solution, it would be interesting to apply it in the real case of the industry on a large scale. It would also be interesting to enrich the solution with request distribution rules to minimize this operation and thus reduce the cluster's network traffic.

Acknowledgements

This work is supported by the Nature Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Al-Osta, M., Ahmed, B., Abdelouahed, G., 2017. A lightweight semantic web-based approach for data annotation on iot gateways. Procedia computer science 113, 186–193.

[2] Al-Osta, M., Bali, A., Gherbi, A., 2018. Event driven and semantic based approach for data processing on iot gateway devices. Journal of Ambient Intelligence and Humanized Computing , 1–16.

[3] Bali, A., Al-Osta, M., Abdelouahed, G., 2017. An ontology-based approach for iot data processing using semantic rules, in: International SDL Forum, Springer. pp. 61–79.

[4] Barnaghi, P., Wang, W., Henson, C., Taylor, K., 2012. Semantics for the internet of things: early progress and back to the future. International Journal on Semantic Web and Information Systems (IJSWIS) 8, 1–21.

[5] Brogi, A., Mencagli, G., Neri, D., Soldani, J., Torquati, M., 2017. Container-based support for autonomic data stream processing through the fog, in: European Conference on Parallel Processing, Springer. pp. 17–28.

[6] Evans, D., 2011. The internet of things: How the next evolution of the internet is changing everything. CISCO. White Paper 1, 1–11.

[7] Ismail, B.I., Goortani, E.M., Ab Karim, M.B., Tat, W.M., Setapa, S., Luke, J.Y., Hoe, O.H., 2015. Evaluation of docker as edge computing platform, in: 2015 IEEE Conference on Open Systems (ICOS), IEEE. pp. 130–135.

[8] Morabito, R., 2017. Virtualization on internet of things edge devices with container technologies: a performance evaluation. IEEE Access 5, 8835–8850.

[9] Morabito, R., Beijar, N., 2016. Enabling data processing at the network edge through lightweight virtualization technologies, in: 2016 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops), IEEE. pp. 1–6.

[10] Renner, T., Meldau, M., Kliem, A., 2016. Towards container-based resource management for the internet of things, in: 2016 International Conference on Software Networking (ICSN), IEEE. pp. 1–5.

[11] Ruchika, V., 2016. Evaluation of docker for iot application. International Journal on Recent and Innovation Trends in Computing and Communication 4, 624–628.