Received 2 September 2022, accepted 13 September 2022, date of publication 20 September 2022, date of current version 28 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3208118

APPLIED RESEARCH

Automating the Deployment of Artificial Intelligence Services in Multiaccess Edge Computing Scenarios

DALTON CÉZANE GOMES VALADARES[1,2], TARCISO BRAZ DE OLIVEIRA FILHO[1], THIAGO FONSECA MENESES[1], DANILO F. S. SANTOS[1], (Member, IEEE), AND ANGELO PERKUSICH[1], (Member, IEEE)
[1]Virtus RDI Center, Federal University of Campina Grande, Campina Grande, Paraíba 58429-900, Brazil
[2]Federal Institute of Pernambuco, Caruaru, Pernambuco 50670-901, Brazil

Corresponding author: Dalton Cézane Gomes Valadares (dalton.valadares@embedded.ufcg.edu.br)

This work was supported by the Ministry of Science, Technology and Innovation of Brazil with Resources from Law No. 8,248, in October 1991 (Resources from the 3rd Softex/UFCG Partnership Agreement).

ABSTRACT With the increasing adoption of the edge computing paradigm, including multi-access edge computing (MEC) in telecommunication scenarios, many works have explored the benefits of adopting it. Since MEC, in general, presents a reduction in latency and energy consumption compared to cloud computing, it has been applied to deploy artificial intelligence services. This kind of service can have distinct requirements, which involve different computational resource capabilities as well different data formats or communication protocols to collect data. In this sense, we propose the VEF Edge Framework, which aims at helping the development and deployment of artificial intelligence services for MEC scenarios considering requirements as low-latency and CPU/memory consumption. We explain the VEF architecture and present experimental results obtained with a base case's implementation: an object detection inference service deployed with VEF. The experiments measured CPU and memory usage for the VEF's main components and the processing time for two procedures (inference and video stream handling).

INDEX TERMS Future connected systems, fog computing, edge computing, intelligent services, services deployment.

I. INTRODUCTION

Currently, in industrial scenarios, we have distributed and heterogeneous applications involving, for instance, different Industrial Internet of Things (IIoT) devices with their known constraints (e.g., processing and storage capabilities). These scenarios include integrating those devices with other services and equipment [1], such as integrating video cameras with computer vision services or using model-predictive control models to remotely control an automated guided vehicle [2]. In general, to achieve these applications' requirements, we employ external servers to provide the necessary resources. Besides cloud computing, which brings the required resources properly, we can deploy the applications using two other paradigms: edge computing and fog computing.

Although sometimes the edge and fog computing terms can be used interchangeably by authors, due to the lack of a standardized accepted definition, we can describe them as follows [3], [4]:

- edge computing performs the processing near the source of data;
- fog computing performs the processing in intermediate nodes between the edge and the cloud.

Depending on the application requirements, sending large amounts of data to be processed in the cloud can become inviable, as this process may consume all the network bandwidth and increase the energy and financial costs. Cloud computing generally demands high latency, connectivity dependency,

The associate editor coordinating the review of this manuscript and approving it for publication was Shadi Alawneh.

and network bandwidth, which can be considered a waste of resources (energy and bandwidth, for instance) [5]. In this sense, we can list some edge and fog computing improvements compared to cloud computing: decreased response times (80 to 200 ms), overall service latency (50%), and energy consumption (30 to 40%) [4]. A comparison among edge and cloud, measuring the communication and processing latency, can show that the edge can be better thanks to the faster communication latency, even considering that the cloud servers can have faster processing with more robust servers [6].

Edge computing is one of the main enablers for 5G networks, providing high bandwidth, high performance, low latency, and enabling real-time decision-making and location-based awareness [6]. When deploying the edge servers to provide Internet services and mobile access networks, we have Mobile Edge Computing (MEC) [7], [8], also called Multi-access Edge Computing [9], [10]. MEC places processing and storage resources at the network edge, working as a small data center to provide different application services [6]. Applying MEC in industries, we can optimize service scheduling and routing, decrease machine downtime alerts, improve machine uptime, and decrease replenishment costs and time [11]. The MEC advantages can enhance the power of artificial intelligence (AI) techniques. For instance, AI can help IIoT systems to predict fault classes, maintenances, and demand forecastings [12], [13], [14]. With machine learning, a system can learn through data processing without previous explicit programming. Such techniques can be applied to one of the main applications for edge computing: video streaming applications [15], [16].

With the applications' requirements diversity, each with its specificities, deploying and managing services in the edge can become complex [17]. For example, if we consider a video streaming application in a MEC scenario, we have to deal with the deployment and management of distinct components. These components can run on virtual machines or containers. Thus, for each different application, we have to deal with the environment configuration to initiate, manage, and stop services. We can describe this general problem as follows:

- Business problem: ease the deployment of AI services in a MEC server considering requirements as low-latency and reduced computing costs (use of CPU/memory);
- Technical problems:
  - Investigate open source solutions that can help the deployment of services in a MEC server, and can integrate with IoT devices;
  - Design an architecture that integrate open source solutions and help the deployment of AI services in a MEC server considering requirements as low-latency and reduced computing costs;
  - Implement a platform that follows the designed architecture;
  - Test the implementation, measuring latency and CPU/memory consumption.

In this paper, we present the VEF Edge Framework, a platform to ease the deployment and management of AI services in MEC servers. The VEF architecture considers open-source platforms and technologies, such as Openstack,[1] Open Source MANO,[2] and EdgeX Foundry.[3] The VEF proposes a deployment flow in which the developers can upload the containerized AI services to run inside virtual machines. This way, a developer can deploy and manage the AI services through a dashboard. To evaluate the VEF platform's performance, we carried out experiments to measure the communication latency and CPU/memory consumption considering its main components. We implemented a machine learning-based object detection service as a use case for the experimentation.

The main contributions of this work are:

- An architecture with components to manage and deploy containerized AI services running inside virtual machines in MEC servers, and enhance the internal communication flow for streams processing;
- The VEF Edge Framework, a platform implemented according to the proposed architecture, which aims at easing the development and deployment of AI services in MEC scenarios;
- An implemented use case, which considers an object detection system as intelligent service to be deployed and managed by the platform;
- An experimental evaluation of the platform, considering performance measurements, such as communication latency and CPU/memory consumption.

The remainder of this article is organized as follows. Section II presents a brief description about the main platforms and technologies considered for the VEF architecture. Section III introduces the VEF architecture, explaining its main components. Section IV presents the experimental design we followed to evaluate the VEF's performance. Section V details and discusses the results achieved with the evaluation. Section VI describes related works which also consider the deployment of AI services in MEC scenarios. Finally, Section VII concludes this work, giving suggestions for future work.

II. BACKGROUND

A. EdgeX FOUNDRY

EdgeX is an open-source platform to facilitate the deployment of IoT applications through interoperability across devices and systems [18], [19]. All EdgeX components are provided as microservices running in Docker containers [20]. Each microservice communicates through a REST API. EdgeX is multi-protocol, allowing devices to communicate through different protocols, and has components for

[1]https://www.openstack.org/
[2]https://osm.etsi.org/
[3]https://www.edgexfoundry.org/

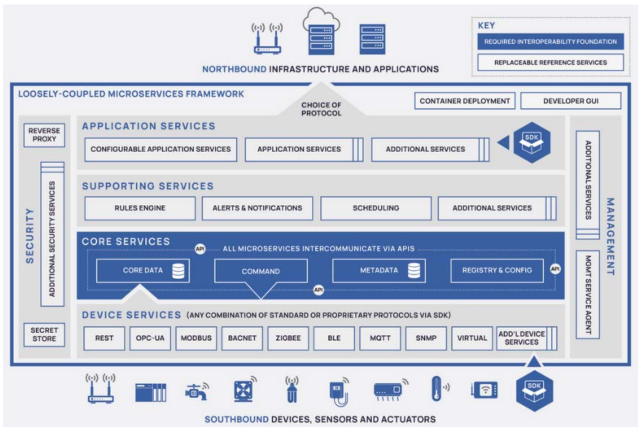


FIGURE 1. **EdgeX architecture[4].**

integration with cloud services. The EdgeX architecture can be seen in Figure 1. It is divided in six main layers (represented by the blocks in the figure): device services, core services, supporting services, application services, security, and management.

B. OPENSTACK

Openstack is an open-source platform, created by NASA and Rackspace Hosting in 2010, for deploying and managing cloud computing services. It is widely used with the infrastructure as a service (IaaS) paradigm to provide processing, network, and storage resources for various applications types.

C. OSM–OPEN-SOURCE MANO

Open Source MANO (Management and Orchestration) [21], [22] emerged in April 2016, after ETSI (European Telecommunications Standards Institute)[5] created the OpenSource Group (OSG) to develop open-source (free software) projects related to ETSI specifications. A MANO solution is responsible for orchestrating VNFs (Virtualized Network Functions) and managing hardware and software resources that support infrastructure virtualization. OSM brings together some existing open source projects: Telefonica's OpenMANO project, Riftware's Rift.io software, and Canonical's Juju charms.

D. EMQ X KUIPER

EMQ X Kuiper[6] is a lightweight, cross-platform, and highly extensible data analytics and streaming tool. Implemented in Golang, it can run in many resource-constrained edge devices. Among Kuiper's advantages, we can list: reduced system response latency, reduced network bandwidth and storage costs, and improved system security. Kuiper works as a rules engine, allowing users to perform fast data processing

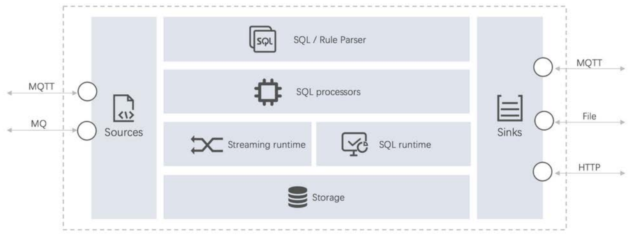


FIGURE 2. **Kuiper architecture[7].**

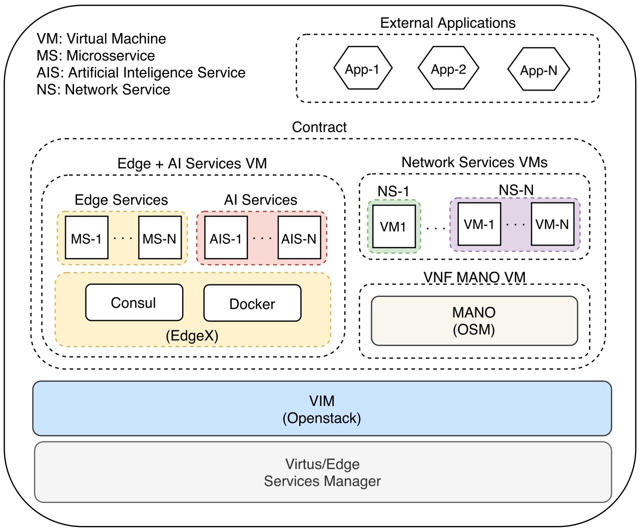


FIGURE 3. **Edge framework architecture.**

in the edge, using rules written in SQL. Kuiper is based on the following three components, as shown in Figure 2: data sources, SQL rule, and result sinks.

III. EDGE FRAMEWORK–ARCHITECTURE

The VEF Edge Framework aims to facilitate the development and deployment of intelligent edge computing applications. For this, we planned an architecture based on microservices to deal with the data that need to be collected, transmitted, processed, and stored. The VEF will act as an MLaaS (Machine Learning as a Service) edge platform, focusing on IoT applications based on fog or edge computing and using machine learning services. Figure 3 presents the defined VEF architecture. The architecture contains three main components: services manager, Virtualized Infrastructure Manager (VIM), and contracts. The external applications are general applications that consume results from intelligent services (e.g., inference or training services). A contract contains all the components needed to run an intelligent service. These components are deployed in two computing instances: one for the edge and intelligent services, and another for the network services. The VIM is in charge of deploying the virtual machine (VM) locally or remotely.

We describe the three main components below:

- Services Manager (Maestro) - The Services Manager, named Maestro, is the control panel (dashboard)

[4]https://www.edgexfoundry.org/why_edgex/why-edgex/
[5]https://www.etsi.org/
[6]https://docs.emqx.io/en/kuiper/latest/

responsible for managing and monitoring the components of a project/contract;
- VIM (Openstack) - A VIM is required to manage the provision of virtualized services. For the Edge Framework, we use Openstack as VIM;
- Contract - A project/contract involves the deployment and management of intelligent and edge services.

Maestro is also responsible for creating and managing users. It uses Openstack as a VIM to manage the virtual machines that run the edge, AI, and network services. Each contract initially has two virtual machines: one running EdgeX, for edge and AI services, and one running OSM, for VNFs (Virtualized Network Functions).

We use Openstack to manage the virtual machines, which run the essential services for the Edge Framework, and the virtualized network services to compose the communication infrastructure when needed. For this, we register, in the Openstack Glance (image manager), the operating system images used in the virtual machines. This way, whenever there is any update in the images used as a base, they must be updated in Openstack. In addition, images containing virtualized network services must also be registered and updated in Glance whenever necessary.

A. CONTRACT

As mentioned, each contract creates two virtual machines, which are described in this subsection.

1) EDGE + AI SERVICES VM

This VM runs the edge and artificial intelligence services. Both run on the EdgeX platform. The "Edge + AI Services" VM (virtual machine) runs services in Docker containers, which are managed through the Consul,[7] a service discovery system. We represented the edge services as MS (Microservice) and the AI services as AIS (Artificial Intelligence Service).

For each project, Maestro starts a VM with EdgeX running. The intelligent services provided by the Edge Framework run in the "Application Services" layer. The communication services for the applications' devices run in the EdgeX's "Device Services" layer. Thus, applications need to implement communication services with devices (communication with a camera, for example) and artificial intelligence services (model training or inference, for example).

Maestro communicates with the EdgeX Consul to verify the health status of Edge (MSs) and AI (AISs) Services. In this way, it is possible to query whether an inference or model training service is properly running.

Edge services are the services already offered by the EdgeX platform, represented in the architecture as MS (Fig. 3). They are responsible for collecting, storing, and distributing data on the framework. Through these services, it is possible to register devices and configure how communication occurs with them. Each device has a "profile" that presents information regarding which commands can interact with it. EdgeX's non-native edge services need to have their images available in a repository with Maestro. This way, when starting EdgeX, all necessary services run in containers.

AI services are integrated with EdgeX, represented in the architecture as AIS (Fig. 3). The Edge Framework already provides some services ready to be deployed and allows development teams to implement their training and optimization services of models/inference. All AI services need to be available in Maestro's image repository to be initialized in containers when contracts are created.

[7]https://www.consul.io/

2) VNF MANO VM

The VNF MANO VM runs OSM, which enables the creation and management of virtualized network services (NSs). OSM also uses Openstack as a VIM to manage the virtual machines (Network Services VMs) that run the virtualized NSs. Each virtualized NS can run on one or more virtual machines. The OSM will trigger the registered VIM (Openstack) to initialize the VMs with the settings properly informed (Network Service Descriptors and Virtualized Network Function Descriptors).

3) NETWORK SERVICES VMs

For the NSs managed by the VIM, we are considering that they can run in one or more VMs. Thus, the VIM can create and start one or more VMs depending on the NS architecture.

B. EXTERNAL APPLICATIONS

External applications can communicate with services available on the "Edge + AI Services" VM. Figure 4 shows how the communication between AI/ML services and external applications works.

External applications send a request to the inference process through the Handler. The Handler handles this request and prepares the specific AI/ML service to perform the processing. The Handler is also responsible for communication with the Broker to receive the results of the inference service and send them to external applications. Such applications can subscribe to topics to receive the results of the AI/ML service via notifications. Inference services send the results to the Broker through publications. Finally, whenever there is new information in a topic, indicating that new inference has been processed, the Broker sends notifications to external applications subscribed to this topic.

As Handler, we are using the EMQ X Kuiper. The AI services are available as plugins of the EMQ X Kuiper. Plugins can be loaded directly into EMQ X Kuiper or accessed externally via RPC or HTTP interfaces.

The Figure 4 southbound has two EdgeX service layers: Core Services and Devices Services. These components are responsible for communicating with the devices that generate the data used by the intelligent services (e.g., inference or training services). For instance, a video camera can send video streams to an appropriate component of the device

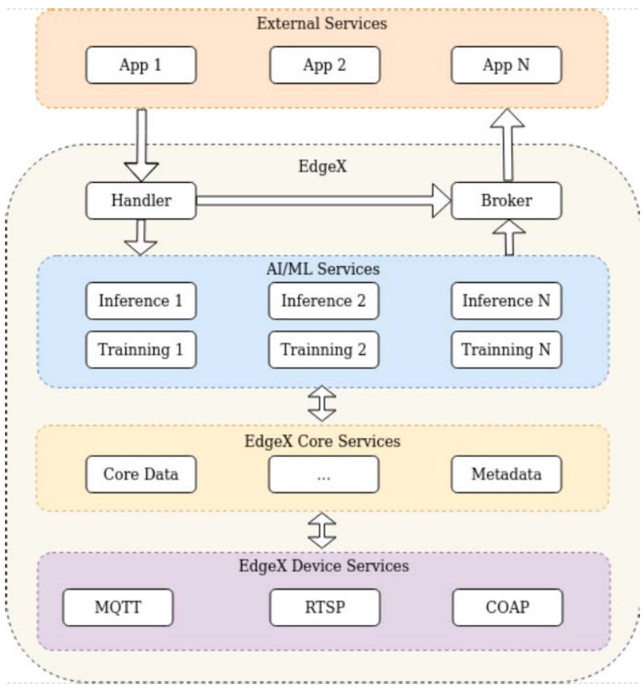


FIGURE 4. Communication flow with external services.

services layer, and it sends the received streams to the core services layer components.

IV. EXPERIMENTAL DESIGN

For testing purposes, we developed some use cases that perform inference based on machine learning models. Thus, we can use the VEF to deploy each inference service. To measure the VEF performance, we used an object detection inference application as the base case. This application performs the inference on video frames in real-time. This use case is similar to a scenario where video cameras can be used to detect fault objects at different places in a production line.

The experimental environment consists of three components: the video sources, the VEF, and the application clients. To deploy these components, we used three virtual machines (VMs), as seen in Figure 5. The communication flow considers:

1) the video streamings generated at the sources and sent to the VEF;
2) the inferences performed at the VEF, with the results sent to the clients;
3) the inference results received in the clients.

For the VEF VM, we changed the computational resources, considering the following flavors: small (1 vCPU, 2GB RAM), medium (2 vCPUs, 4GB RAM), and large (4 vCPUs, 8GB RAM). In each communication cycle, we measured the CPU and memory consumption for the following services in the VEF VM: EMQ X MQTT broker, Kuiper, machine learning inference model, and video streaming hub. We have also measured the frame processing time of video streaming hub and machine learning inference model.

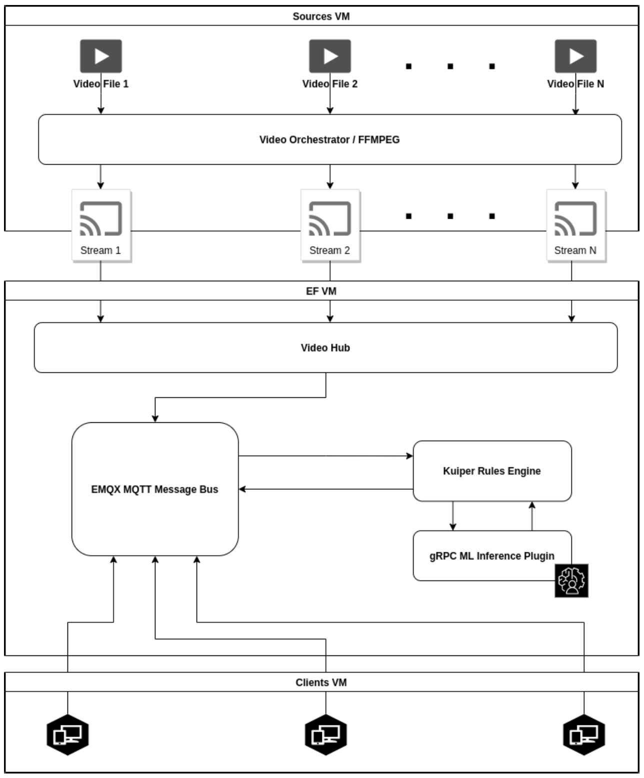


FIGURE 5. Experimental environment with communication flow.

In the Sources VM and Clients VM, we also varied the number of video sources and application clients, respectively. As a pre-experiment, we measured the VEF VM services' CPU and memory consumption in a scenario that kept only one source and varied the number of clients among one, ten, twenty, and forty. We then observed that the number of clients does not affect the result, i.e., regardless of how many clients must receive the inference results, the services' CPU and memory consumption do not change significantly. For this reason, we fixed the number of clients in 10, varying only the VEF VM flavor and the number of sources as listed below:

- Small (1 vCPU, 2GB RAM) - 1, 5, and 10 sources;
- Medium (2 vCPUs, 4GB RAM) - 1, 5, 10, 25, and 50 sources;
- Large (4 vCPUs, 8GB RAM) - 1, 5, 10, 25, 50, and 100 sources.

For each VEF VM flavor, we evaluated the following metrics:

- CPU Usage - how much CPU each of the four services used?
- Memory Usage - how much memory each of the four services used?
- Processing Time - how long did each step of the inference pipeline take within VEF VM?

For each experiment factors combination, we performed two executions to generate a confidence interval, bringing more robustness to the experiment results. The experiments considered the sources sending an MP4 Full HD video streams (approximately 9 min of video, at 30 fps) to the VEF VM and the clients receiving the inference results.

Our main experiment goal is to evaluate the impact of our architecture implementation in the video stream processing flow, considering that low latency is one of the main requirements for this type of application. Finally, we want to evaluate the best setup in terms of processing power and memory, as computing costs are an essential requirement when considering industrial scenarios.

V. RESULTS AND DISCUSSION

We conducted the experiments using VMs deployed on a MEC Server. We instrumented the code to collect and save metrics and used Jupyter Notebooks[8] to perform individual and comparative analyses of the results. Below, we describe the evaluation of each metric across the different experimented scenarios.

A. CPU EVALUATION

Figure 6 presents the CPU usage for each VEF component (e.g., Kuiper, MQTT Broker, Inference Model, and Video Stream service) in each VEF VM flavor (e.g., Small with 1, 5, and 10 sources, Medium with 1, 5, 10, 25, and 50 sources, and Large with 1, 5, 10, 25, 50, and 100 sources). Analyzing the Figure 6, we can observe the following:

- Kuiper and MQTT Broker services perform similarly, with low CPU usage overall, but with a slight increase with the number of sources followed by a decrease in the higher number of sources per VM size.
- Inference (ML-Inference-Vino) service is usually second in CPU usage, nonetheless decreasing its usage with the number of sources - probably due to the threshold used to avoid processing frames that are too old;
- Video Stream service usually dominates the CPU usage, rising with the number of sources;

Considering the platform streaming components (MQTT Broker and Kuiper), we see they present low CPU usage, utilizing less than 30% of one CPU core even in the cases with a high number of sources (50 and 100). It is important to note they process the data stream twice (when going from Video Stream to Inference and when going from Inference to the Client), thus demonstrating they present high efficiency.

When we analyze the Video Streaming specific components (Video Streaming and ML-Inference-Vino), we notice they dominate the CPU usage, with the Video Streaming service being responsible for the higher CPU usage (up to 3 cores in the 4-vCPU VM). This result indicates that Video Streaming is the bottleneck of this architecture, which leads us to employ efforts to make it more efficient, case necessary.

We used a Frame Age Threshold to specify the maximum tolerated "frame age" for a frame to be inferred upon. The frame age is defined as the difference between VEF frame arrival timestamp and Inference Service frame arrival timestamp. The idea is that frames that are too old should not be processed in order to prioritize more recent ones, aiding the live aspect of the application. If the frame age exceeds the threshold value (80ms in this setting), it is discarded by the Inference Service. In this experiment, as the number of sources grows, the Video Streaming processing time increases, causing inference to be performed to fewer frames due to the use of the aforementioned threshold and, as a result, making the Inference Service use fewer resources.

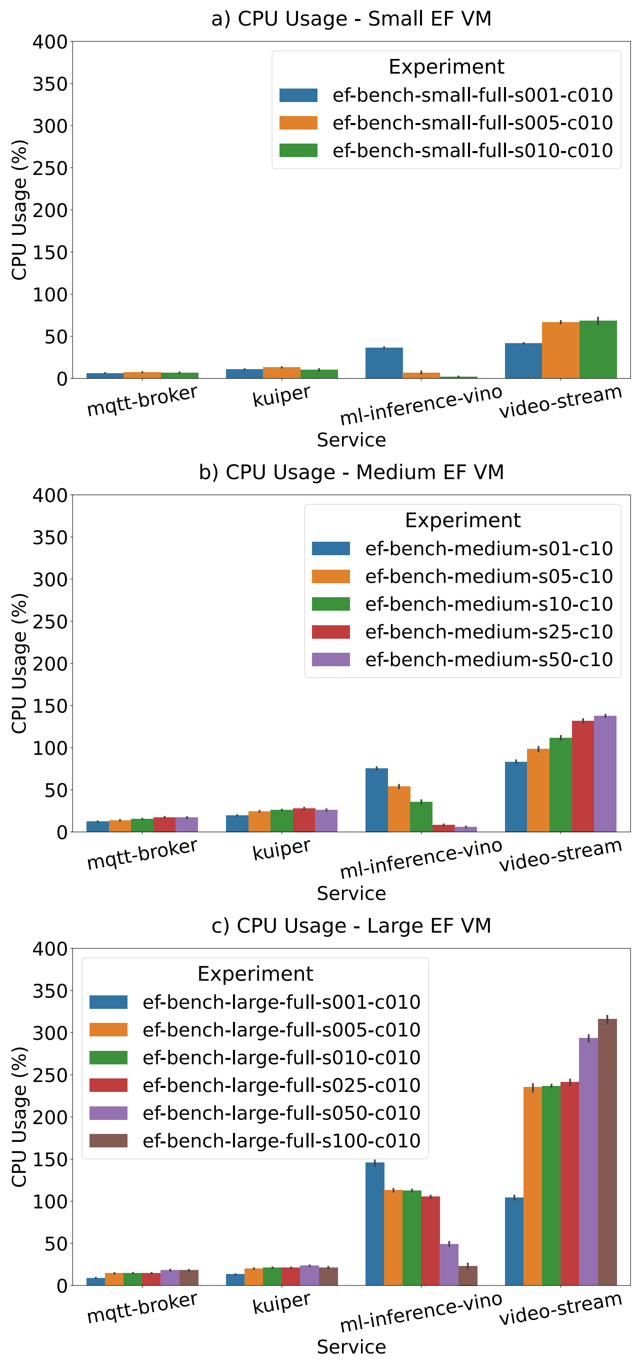


FIGURE 6. VEF service CPU usage per VM flavor and number of sources.

[8]https://jupyter.org/

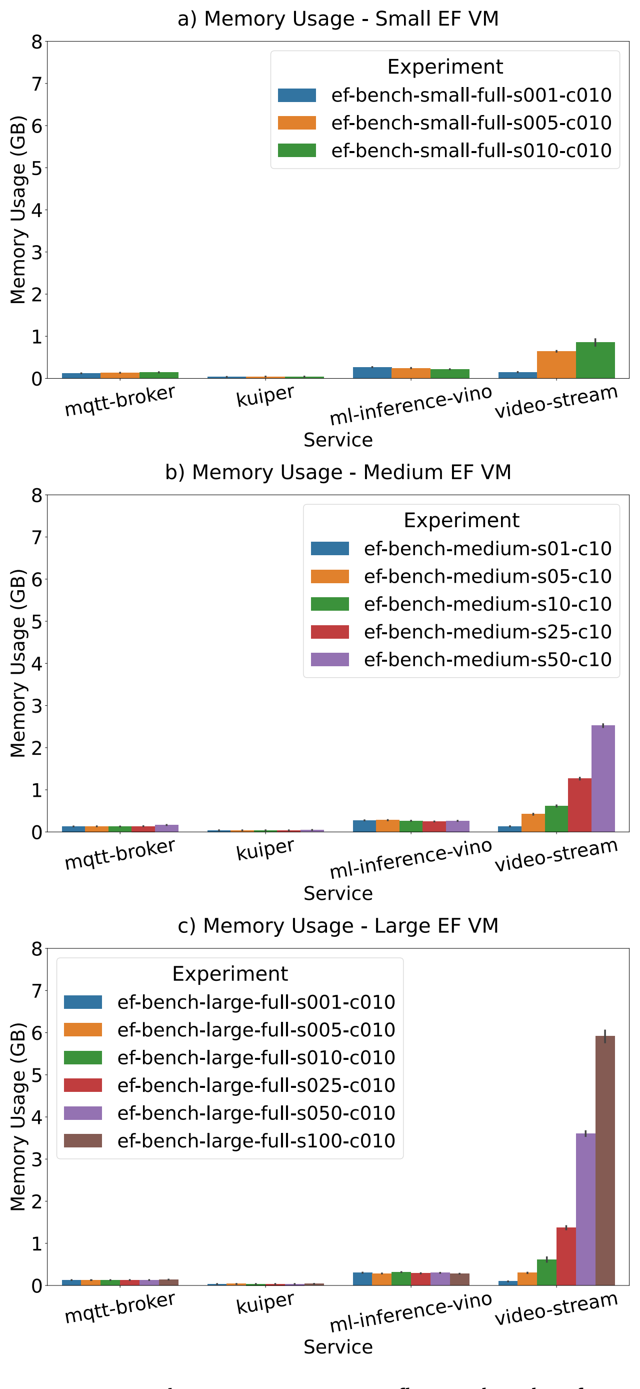


FIGURE 7. VEF service memory usage per VM flavor and number of sources.

B. MEMORY EVALUATION

Figure 7 presents the memory usage for each of the VEF components, also considering each of the VEF VM flavors. When analyzing the memory usage for each VEF component and VM flavor combination, we can see that:

- Kuiper and MQTT Broker services perform with low memory usage overall (less than 50MB for Kuiper and less than 200MB for MQTT broker, for all VM flavors), but with a slight increase with the number of sources.

TABLE 1. Processing time per VEF VM flavor and number of sources.

VM Flavor	# Sources	Video Stream (ms)	Inference (ms)
small	1	99.0	165.0
small	5	154.0	148.0
medium	1	62.0	103.0
medium	5	119.0	109.0
medium	10	122.0	112.0
medium	25	125.0	107.0
medium	50	120.5	106.0
large	1	19.0	54.0
large	5	40.0	65.0
large	10	55.0	64.0
large	25	100.0	65.0
large	50	134.0	60.0
large	100	132.0	63.0

- ML-Inference-Vino service is the second in memory usage, not varying significantly with the number of sources, but presenting a somewhat constant behavior - probably due to the use of the Frame Age Threshold, which prevents the processing of frames that are considered too old;
- Similarly to the CPU usage, Video Stream service dominates the memory usage, increasing its utilization with the number of sources;

In memory evaluation, analogously to the CPU evaluation, we observe the platform streaming components have a low resource usage. This result indicates they can process high loads of data without consuming much memory. This characteristic enables and empowers a variety of high-throughput-low-latency applications, bringing new possibilities to businesses and users.

Again, the Video Stream service presents high memory usage, indicating that we should optimize it if more efficient resource utilization is required.

C. PROCESSING TIME EVALUATION

When analyzing processing time, we measured the duration of 2 steps of the pipeline (in chronological order) within the core VM:

- Video Streaming Processing Time - time to process frame within the Video Stream service;
- Inference Processing Time - time to process frame within the Inference service;

Table 1 presents the latency (processing time) values for each combination with VM flavor and number of sources. Analyzing the table, we can notice that:

- Video Streaming Processing Time usually increases with the number of sources;
- Inference Processing Time usually increases when going from 1 to 5 sources but then roughly stabilizes and

then decreases for the highest number of sources of each scenario (this last behavior is probably due to the infrastructure resources overflow).

From the behavior observed in latency measurements for the Video Stream and Inference services, we can conclude that the Video Stream service benefits from a more robust infrastructure (it went from 100ms of processing time in the small-01-source setting to 19ms in the large-01-source setting). However, the Video Streaming Processing Time significantly increases as the number of sources grows. Hence, we must carefully define the infrastructure specifications for the VEF VM based on the final application requirements to deliver a smooth user experience.

The Inference Processing Time roughly stabilizes when adding more sources, given that there is processing power available to be used. Nonetheless, this is likely caused by the use of the Frame Age Threshold, which makes the Inference service skips more and more frames from inference as the number of sources rises, keeping processing time constant. Therefore, we cannot say the current implementation of the Inference service is efficient since it does not get to process an increasing amount of frames as the number of sources rises.

VI. RELATED WORK

This Section gathers works demonstrating the application of edge computing and artificial intelligence techniques. We can see different efforts regarding the challenges, opportunities, and focuses. However, all the works present benefits of the use of edge computing when compared to cloud computing.

Aytaç and Korçak [5] described some benefits of edge computing and proposed an edge-based IoT architecture for use in quick-service restaurants. The architecture comprises sensors, actuators, and external data sources, such as social network interactions, and proposes basic data protection mechanisms (nonce and digest) and machine learning techniques. With the generated data, the authors proposed to estimate the service level in the restaurants, to improve efficiency and decrease waste of resources. A proof of concept was implemented, considering a Raspberry Pi 3 with Windows IoT Core as the edge gateway and two machine learning techniques to predict the service level (K-means clustering and Naïve Bayes classifier). Although the authors mentioned they performed experiments, there are no results presented regarding these tests.

Cao *et al.* [23] explored the quality of service challenges when employing edge computing for cyber-physical systems (CPS) applications. The authors present a systematic classification, summarizing experiences from the surveyed works and suggesting directions for future research. Zou *et al.* [4] presented the main characteristics of edge and fog computing, summarizing the challenges regarding the enablement of AI for edge/fog-based IoT scenarios. One of the main challenges for edge devices running complex AI algorithms is decreasing resources consumption, improving energy efficiency. The authors presented a table containing AI processors solutions for specific algorithms with their respective energy efficiencies.

Sittón-Candanedo *et al.* [11] presented a review of edge computing reference architectures (RAs). They described the main characteristics of the following four edge computing RAs: FAR-EDGE RA, INTEL-SAP RA, Edge Computing RA 2.0, and Industrial Internet Consortium RA. Based on these RAs, the authors proposed the Global Edge Computing Architecture, which was evaluated through an agroindustry application. The authors carried out two tests during two months, considering a Raspberry Pi 3 as the edge node and the Google Cloud Platform as the cloud. The first test did not consider the edge pre-processing activities, gathering the data from the collecting nodes and sending them directly to the application in the cloud. The second test considered the edge node and its pre-processing, including data filtering and encryption. The results achieved a reduction of 38.84% in the data transferred to the cloud.

Sun *et al.* [12] proposed the deployment of AI techniques in edge and cloud servers, considering delay and service accuracy to deal with IIoT devices' requests. To reduce the processing complexity, the authors proposed the use of transfer learning, deploying pre-trained models in the edge servers. Then, each IIoT device's request is answered based on its required values for delay and service accuracy. The proposed framework identifies the edge server that will process the request matching the required delay and accuracy. Simulations were performed with an NVIDIA TITAN V GPU, considering ten edge servers and 100 IIoT devices, running an image recognition application. Results showed that the proposed solution achieves better average accuracy than an experiment scenario that does not consider the proposal (i.e., without considering the accuracy of the servers). For future work, the authors suggest the use of caching allocation in the edge servers.

Dimithe *et al.* [24] developed a machine learning environment based on a TX2 board acting as an edge server. The TX2 receives images from a drone and a robot and performs object detection and classification. The authors mentioned problems with reduced latency due to the high volume of transmitted data and also with the accuracy of the used models. In future work, they plan to improve the trained model and increase the processing speed.

Bellavista *et al.* [6] proposed a Machine Learning infrastructure based on edge and cloud computing. The proposal suggests running models locally at the edge. Several experiments were performed, considering a faces recognition application in a smart city scenario, an Android smartphone as the mobile node, a Raspberry Pi 3 as the edge node, and a virtual machine deployed at AWS as the cloud. The authors used OpenCV and Python for the ML procedures. Results demonstrated good operation performance, mainly due to the lower network communication latencies achieved at the edge compared to the values achieved for communications with the cloud. The result is interesting because the lower latency at the edge makes the total processing time better than in the cloud, even considering that the cloud server has more computational power than the edge node. As future work, the authors suggest deploying the proposal in an industrial environment and develop an optimizer module for the models.

Huang *et al.* [25] designed and implemented an edge computing platform to deploy machine learning applications. The platform consists of training models at the cloud and deploying predictive models at the edge server. The authors used Docker and Kubernetes to deploy the services in containers and a Raspberry Pi as the edge server. Simple experiments measured the CPU and memory overhead. As future work, they intend to optimize the processing rates.

We can see there is a trend to adopt edge computing scenarios for the deployment of artificial intelligence services, mainly when these services serve applications requiring lower communication latencies (e.g., an industrial application employing IIoT devices). From the literature, we notice that many works have been published regarding different applications, considering the benefits of edge computing. Nonetheless, we should consider any tool to help in the deployment and management of such applications, and this is why we proposed the VIRTUS Edge Framework.

VII. CONCLUSION

Edge computing empowers the Industrial IoT applications, offering more computational resources (compared with the IoT devices) and reduced communication time (compared with cloud servers) for data processing and storage. When edge computing meets the telecommunication infrastructures, Multi-access Edge Computing (MEC) arises. Given that many of the applications deployed in MEC scenarios apply Artificial Intelligence (AI) techniques, this work proposed the VEF Edge Framework, which aims at easing the deployment of AI applications, also considering the communication with IIoT devices.

To evaluate the VEF, we deployed the framework and an AI application, which worked as a base case. For this application, we considered a machine learning model to detect objects in video streamings. We varied the number of video sources and the sink clients. We performed experiments to measure communication latency and CPU/memory consumption, considering different scenarios, alternating the number of sources and sinks. Analyzing the results, we can see that the main VEF components do not present a high overhead during data processing and communication. This is due our integration of an IoT processing platform (EdgeX) with a data analysis streaming and communication tool (EMQ X Kuiper) for the processing of streamings and data in machine learning inference models.

As future work, we are going to use our testbench to run experiments considering other AI applications deployed through the VEF. We also want to advance the functionalities regarding the management and deployment of VNFs as support for AI services and applications.

REFERENCES

[1] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020.

[2] R. V. de Omena, D. Santos, and A. Perkusich, "An approach to reduce network effects in an industrial control and edge computing scenario," in *Proc. CLOSER*, 2021, pp. 296–303.

[3] Q. Qi and F. Tao, "A smart manufacturing service system based on edge computing, fog computing, and cloud computing," *IEEE Access*, vol. 7, pp. 86769–86777, 2019.

[4] Z. Zou, Y. Jin, P. Nevalainen, Y. Huan, J. Heikkonen, and T. Westerlund, "Edge and fog computing enabled AI for IoT–An overview," in *Proc. IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Mar. 2019, pp. 51–56.

[5] K. Aytac and O. Korcak, "IoT edge computing in quick service restaurants," in *Proc. 16th Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2018, pp. 1–6.

[6] P. Bellavista, P. Chatzimisios, L. Foschini, M. Paradisioti, and D. Scotece, "A support infrastructure for machine learning at the edge in smart city surveillance," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2019, pp. 1189–1194.

[7] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2017.

[8] T.-D. Nguyen, E.-N. Huh, and M. Jo, "Decentralized and revised content-centric networking-based service deployment and discovery platform in mobile edge computing for IoT devices," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4162–4175, Jun. 2019.

[9] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[10] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6722–6747, Aug. 2020.

[11] I. Sittón-Candanedo, R. S. Alonso, J. M. Corchado, S. Rodríguez-González, and R. Casado-Vara, "A review of edge computing reference architectures and a new global edge proposal," *Future Gener. Comput. Syst.*, vol. 99, pp. 278–294, Oct. 2019.

[12] W. Sun, J. Liu, and Y. Yue, "AI-enhanced offloading in edge computing: When machine learning meets industrial IoT," *IEEE Netw.*, vol. 33, no. 5, pp. 68–74, Sep. 2019.

[13] M. A. Rahman, M. S. Hossain, A. J. Showail, N. A. Alrajeh, and A. Ghoneim, "AI-enabled IIoT for live smart city event monitoring," *IEEE Internet Things J.*, early access, Sep. 1, 2021, doi: 10.1109/JIOT.2021.3109435.

[14] S. Zhu, K. Ota, and M. Dong, "Green AI for IIoT: Energy efficient intelligent edge computing for industrial Internet of Things," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 79–88, Mar. 2022.

[15] M. Fatih Tuysuz and M. Emin Aydin, "QoE-based mobility-aware collaborative video streaming on the edge of 5G," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7115–7125, Nov. 2020.

[16] W. Dou, X. Zhao, X. Yin, H. Wang, Y. Luo, and L. Qi, "Edge computing-enabled deep learning for real-time video optimization in IIoT," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2842–2851, Apr. 2021.

[17] J. Portilla, G. Mujica, J.-S. Lee, and T. Riesgo, "The extreme edge at the bottom of the Internet of Things: A review," *IEEE Sensors J.*, vol. 19, no. 9, pp. 3179–3190, May 2019.

[18] K. Han, Y. Duan, R. Jin, Z. Ma, H. Rong, and X. Cai, "Open framework of gateway monitoring system for Internet of Things in edge computing," in *Proc. IEEE 39th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Nov. 2020, pp. 1–5.

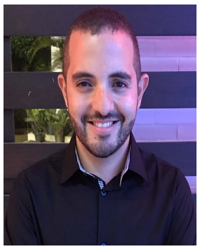
[19] R. Xu, W. Jin, and D. Kim, "Enhanced service framework based on microservice management and client support provider for efficient user experiment in edge computing environment," *IEEE Access*, vol. 9, pp. 110683–110694, 2021.

[20] J. Kim, C. Kim, B. Son, J. Ryu, and S. Kim, "A study on time-series DBMS application for EdgeX-based lightweight edge gateway," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2020, pp. 1795–1798.

[21] P. Trakadas, P. Karkazis, H. C. Leligou, T. Zahariadis, F. Vicens, A. Zurita, P. Alemany, T. Soenen, C. Parada, J. Bonnet, E. Fotopoulou, A. Zafeiropoulos, E. Kapassa, M. Touloupou, and D. Kyriazis, "Comparison of management and orchestration solutions for the 5G era," *J. Sens. Actuator Netw.*, vol. 9, no. 1, p. 4, Jan. 2020. [Online]. Available: https://www.mdpi.com/2224-2708/9/1/4

[22] G. M. Yilma, Z. F. Yousaf, V. Sciancalepore, and X. Costa-Perez, "Benchmarking open source NFV MANO systems: OSM and ONAP," *Comput. Commun.*, vol. 161, pp. 86–98, Sep. 2020.
[23] K. Cao, S. Hu, Y. Shi, A. Colombo, S. Karnouskos, and X. Li, "A survey on edge and edge-cloud computing assisted cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7806–7819, Nov. 2021.
[24] C. O. Bitye Dimithe, C. Reid, and B. Samata, "Offboard machine learning through edge computing for robotic applications," in *Proc. SoutheastCon*, Apr. 2018, pp. 1–7.
[25] Y. Huang, K. Cai, R. Zong, and Y. Mao, "Design and implementation of an edge computing platform architecture using Docker and kubernetes for machine learning," in *Proc. 3rd Int. Conf. High Perform. Compilation, Comput. Commun.*, New York, NY, USA, Mar. 2019, pp. 29–32.

DALTON CÉZANE GOMES VALADARES received the bachelor's, master's, and Doctor degrees in computer science, and the M.B.A. degree in project management, and a technical course in informatics. He is currently a Professor with the Federal Institute of Pernambuco (IFPE) and a Researcher with the Embedded Laboratory, Federal University of Campina Grande (UFCG). He has over 15 years of experience in IT, having worked on several Research and Development Projects assuming different roles: systems analyst, embedded systems/web developer, quality/testing analyst, and project manager. He currently develops research collaboration with FCS Group, Embedded Laboratory (UFCG), and GPRSC, UTFPR. His current research interests include the Internet of Things, software engineering, fog/edge computing, data security, and wireless networks.



TARCISO BRAZ DE OLIVEIRA FILHO received the bachelor's and master's degrees in computer science from UFCG. He works as a Data Scientist at VIRTUS Innovation Center, Federal University of Campina Grande (UFCG). He has over 12 years of experience in IT, having worked on several Research and Development Projects assuming different roles: distributed systems/full-stack developer, data scientist, and project manager. His research interests include data science, artificial intelligence, big data, and distributed systems.

THIAGO FONSECA MENESES received the master's degree in computer science from UFCG. He works as a Software Engineer at VIRTUS Innovation Center, Federal University of Campina Grande (UFCG). He has experience in computer science, with emphasis on information and communication technologies, working mainly on the following topics: 5G networks, IaaS, systems analysis, and software requirements specification.

DANILO F. S. SANTOS (Member, IEEE) received the Ph.D. degree in computer science from the Department of Electrical Engineering, Federal University of Campina Grande (UFCG), Brazil. He works as a Professor at the Department of Electrical Engineering, UFCG, where he coordinates Research, Development, and Innovation Projects in partnership with technology companies. He serves as an Innovation Coordinator at VIRTUS-UFCG Innovation Center. His research interests include intelligent software engineering, pervasive and edge computing, and wireless communication, with over 60 published articles.

ANGELO PERKUSICH (Member, IEEE) received the Ph.D. and master's degrees in electrical engineering from the Federal University of Paraiba, in 1987 and 1994, respectively. He was a Visiting Researcher at the Department of Computer Science, University of Pittsburgh, PA, USA, from 1992 to 1993. He is currently a Professor with the Electrical Engineering Department (DEE), Federal University of Campina Grande (UFCG), since 2002. He is the Founder and the Director of the VIRTUS Innovation Center and Embedded and the Pervasive Computing Laboratory. He has over 400 papers published and advised 85 master's thesis and 25 Ph.D. dissertations. His main research interests include embedded systems, software engineering, and cyber-physical systems.