Serverless Computing Approach for Deploying Machine Learning Applications in Edge Layer

Ta Phuong Bac
School of Electronic Engineering
Soongsil University, Seoul, Korea
bactp@dcn.ssu.ac.kr

Minh Ngoc Tran
School of Electronic Engineering
Soongsil University, Seoul, Korea
mipearlska1307@dcn.ssu.ac.kr

YoungHan Kim*
School of Electronic Engineering
Soongsil University, Seoul, Korea
younghak@ssu.ac.kr

2022 International Conference on Information Networking (ICOIN) | 978-1-6654-1332-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICOIN53446.2022.9687209

***Abstract*— Serverless computing- a stateless cloud computing model, is an emerging solution that has shown significant benefits to efficiency and cost for event-driven applications in the cloud environment, including artificial intelligence (AI), machine learning applications. With serverless computing, the machine learning system's complexity is minimized, flexible and straightforward in management. However, operating and managing serverless machine learning services on clouds faces many limitations such as latency and data privacy. Local distributed edge computing nodes which are closed to users can address these challenges of cloud-serverless AI applications. Based on this motivation, in this paper, we propose an architecture for deploying machine learning workload as serverless functions in the edge environment. We illustrate our proposed approach and evaluate its performance and effectiveness by exploiting a holistic end-to-end image classifier, a famous machine learning use case in the MNIST dataset. Our proof of concept provides comprehensive assessments that prove its effectiveness in latency reduction and distributed machine learning deployment.**

Keywords—Serverless computing, machine learning, Edge-cloud computing architecture.

I. INTRODUCTION

In recent years, the rapid development of machine learning applications, such as smart cities, self-driving cars, or augmented reality [1], has necessitated new ways to cope with the massive amounts of data created by IoT sensors. These applications are typically deployed on the cloud to analyze large amounts of data and provide valuable services to users.

At the same time, serverless computing [2][3](known as Function as a Services) has emerged as a novel paradigm for deploying applications and services in the cloud environment[4]. Serverless computing has shown particular promise for event-driven applications under the increase of containers and microservices. The cost-effectiveness of the serverless platform is a fundamental reason for its popularity; the price model is based on the execution time rather than the resource allocated. Thus, rather than having long-running servers on the virtual machine, consumers only pay for the time they utilize, resulting in idle time for those infrequent events and an undesired monetary cost.

Because of these benefits, applying serverless computing in deploying machine learning applications in the cloud has attracted lots of attention. Several useful integrations of serverless computing and AI has been demonstrated in [5][6][7]

However, deploying the whole AI workflow (training and serving) over serverless computing on the cloud suffers from several drawbacks. Firstly, because of the time required to communicate a significant quantity of data, shifting computationally heavy operations to a cloud center may cause a delay[8]. The latency affects the performance of machine learning applications, which requires fast data processing and low latency to ensure service level agreement and quality of experience[9]. Second, collecting raw data is difficult due to privacy concerns. For example, hospitals are unwilling to share their patients' personal data for healthcare applications.

To solve the mentioned issues above, moving serverless AI workflow to the edge is a promising candidate.

First, edge computing can significantly reduce latency, reduction in traffic and geographic distance. Deploying serverless on edge nodes instead of in a cloud data center would benefit both users and network operators, achieve fast response time. Especially, serverless packages machine learning applications as discrete functions, controlled and delivered on-demand using dynamically created containers. Container engineering [10] is also used to overcome the requirement use many libraries to support machine learning applications related to the constraints and limitations of the underlying platform, which includes the runtime environment (e.g., Python) and required machine learning libraries (e.g., Pandas, Tensorflow, and sklearn) to complete applications. With minimum configuration and setup, these containers may be halted, destroyed, rebuilt, and replaced. This event-approach service execution paradigm allows for on-demand access to applications, which has the potential to reduce costs as well as latency[11]. Therefore, moving the machine learning application to the edge layer can reduce the expenses and bottlenecks in performance, computing resources near the data source (e.g., users, sensors), minimizing the overall latency and faster response [1].

Second, the edge computing paradigm allows distributed machine learning model training between local edge data to secure data privacy and save the cloud resources, which is enabled by a recent advanced intelligence technology - Federated Learning[12]. In federated learning, the machine learning model is distributedly trained over multiple nodes. Each node train with their local data, then only the weights of the trained model is sent to the aggregation server in the cloud instead of all raw data. This mechanism allows raw data to be kept private at local nodes and saves network bandwidth and cloud storage for handling these vast amounts of data.

Motivated by this potential, we propose a general architecture for deploying machine learning applications as serverless functions in distributed edge environment. We also illustrate the operation of the proposed approach and evaluate its performance and effectiveness by exploiting a holistic

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center)

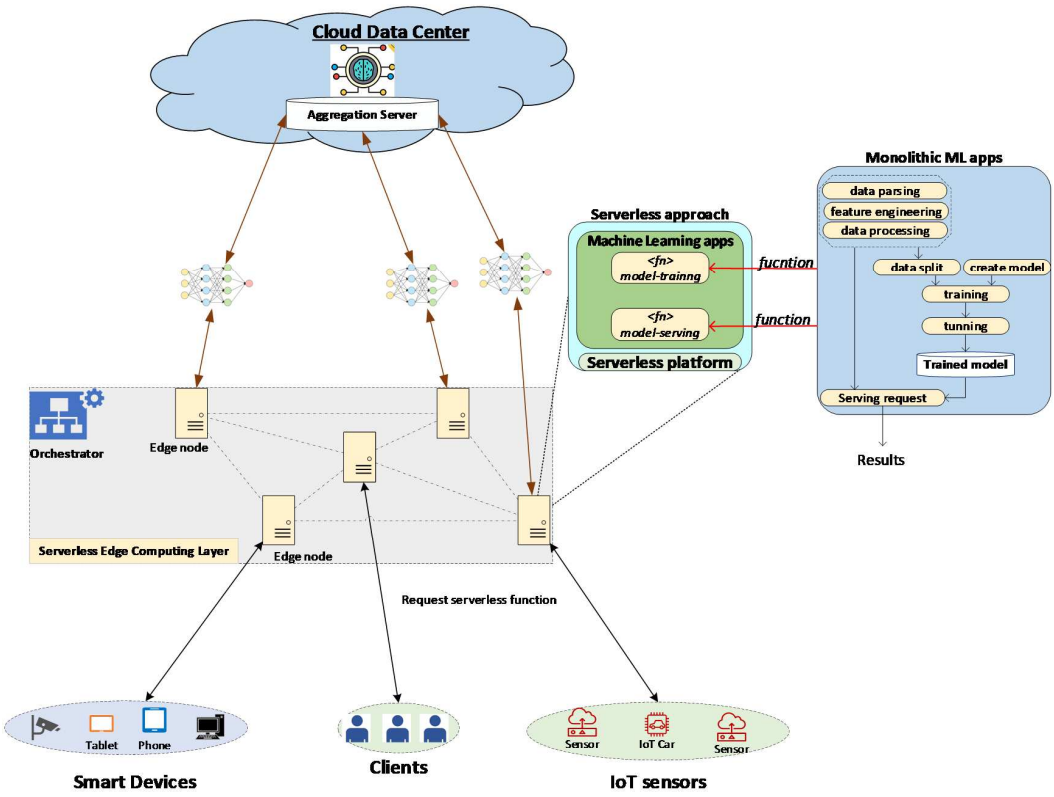

Fig. 1 Architecture for serverless edge computing approach

end-to-end image classifier, a famous machine learning use case in the MNIST dataset[13].

Our main contributions in this paper include:

- Proposing an approach leverages the serverless computing paradigm at the edge layer to reduce latency and fast response time.
- We provide the proof of concept that show our approach benefits to implementing machine learning workflow on serverless computing environment: (1) allowing distributed models training and deployment through federated learning, (2) improving the latency of serverless machine learning workflows compared with deploying them on the cloud

The rest of this paper is structured as follows. Section II shows the related work. The details of the serverless edge computing architecture are described in Section III. Section IV evaluates the performance of the proposed approach both in the training and serving process. Finally, Section V is for the conclusion and our future work.

II. Related work

Serverless computing has received much attention in the development of machine learning applications on commercial cloud computing platforms [5][6][14]. Cirrus is a serverless framework proposed by Carreira et al. [14]to assist end-to-end machine learning workflows. It only supports homogenous commercial cloud platforms like AWS Lambda, and the worker runtime does not support a popular machine learning framework, making it challenging to integrate and train various models in Cirrus. In the same direction, [5] investigates serverless cloud computing for double machine learning. The author proposes a prototype Python implementation for estimating double machine learning models with AWS Lambda's serverless computing architecture and shows its value with a case study examining estimated times and costs. However, studies towards machine learning on serverless have not considered a comprehensive evaluation in implementing a machine learning application. These works [5][6][7] have primarily been used for inference models.

In recent years open-source serverless frameworks have been introduced to avoid vendor lock-in, whose features and performance have been evaluated in [11][15][16][17]. In[17], the authors performed many tests on several cloud serverless platforms (AWS Lambda, ...) In [11] and [15], the authors assessed popular open-source serverless platforms (e.g., Knative, Kubeless, Nuclio) with a simple HTTP workload in cloud and edge environments. Junfeng Li et al. in [16]delves into the inner workings of various central open-source serverless platforms and identifies their distinguishing features. The evaluation results are performed on the cloud environment (CloudLab testbed). In [11], the author assessed a resource-constrained, edge computing environment regarding response time, throughput, and success rate. Nevertheless, these studies do not evaluate with a machine learning workload implementation, instead of a simple HTTP workload that sends requests and receives responses.

Thus, our work proposes an approach by using serverless computing for edge computing to implement machine applications. In addition, we show a proof of concept implementation and thoroughly evaluate with a machine learning case study in both the training and serving process.

III. Proposed Serverless Edge Computing Approach

This study proposes a serverless edge computing approach for implementing machine learning applications at the edge layer. We designed the machine learning model followed by the event-driven application, which decoupled into training events and serving events like in Fig. 1, and each event is a serverless function, respectively.

As illustrated in Fig. 1, the serverless edge computing approach is based on edge computing architecture with federated learning mechanisms to enable distributed machine learning, including the cloud data center, serverless edge layer, and data source layers.

- *Cloud data center*: with unlimited resources and high computing power, the cloud is better suited to resource-intensive, time-sensitive applications such as big data storage and data mining with a large amount of data. In addition, the cloud data center has hosted the artifacts (e.g., docker images) of serverless functions in a registry by using container engineering, which is easy to manage, flexible in updating new functions for the application. Serverless functions at the edge use artifacts that are available at any time in many places. A cloud data center is also the global processing unit in charge of federating the model information from different edge nodes, aggregating it, constructing a global federated model.
- *Serverless Edge layer:* an edge zone comprises an orchestrator and many distributed serverless edge computing nodes. Located orchestrator at the edge layer effectively manages complex, heterogeneous, and large-scale edge environments. Each edge computing node is deployed with a serverless platform to host the machine learning application in this layer. Applications are based on the function as a service programming paradigm and offer computing capabilities to execute serverless functions to satisfy client requests. The application will invoke the machine learning model that is closest to their current location. By eliminating the round trip to the cloud, latency is reduced.
- *Data source layer*: contains end devices that may use serverless applications such as smart gadgets, wearable devices, IoT sensors, direct users.

In our approach, the monolithic machine learning applications map to serverless functions with training and serving functions and deploy by serverless platform. The *model-training function* performs related tasks such as data processing, model creation, training, tunning, and exporting trained model weights to the database. The *model-serving function* will receive requests access to the application from the user to perform the prediction process right at the edge layer. The distributed training of edge nodes operates according to the federated learning mechanism. First, each edge node receives an initial model from the cloud aggregate server. Then, the serverless training function is invoked to perform the training process with the local data of the edge node. After, local training finish edge node sends the weights of the trained model to the cloud for aggregate and continues training until the global model is converged.

Using the federated learning method in our serverless edge computing approach saves bandwidth, ensures data privacy of edge nodes and resource management when each function is configured with an appropriate amount of resources for the workload it performs. Serverless has the agile ability to scale the number of functions to respond to the different situations to ensure application performance and service quality. Furthermore, edge layer deployment significantly reduces end-to-end latency compared to conventional cloud deployments. Ensure the best response speed for users. The following section will present an experiment on the machine learning case study and the serverless edge computing approach.

IV. Experiment Case Study

In this section, we set up a testbed to evaluate our proposed approach. The main case study is image classification with the MNIST dataset[13].

A. Workload and testbed setup

Testbed As illustrated in Fig. *2*, our testbed involves the cloud, serverless edge layer, and event sources.

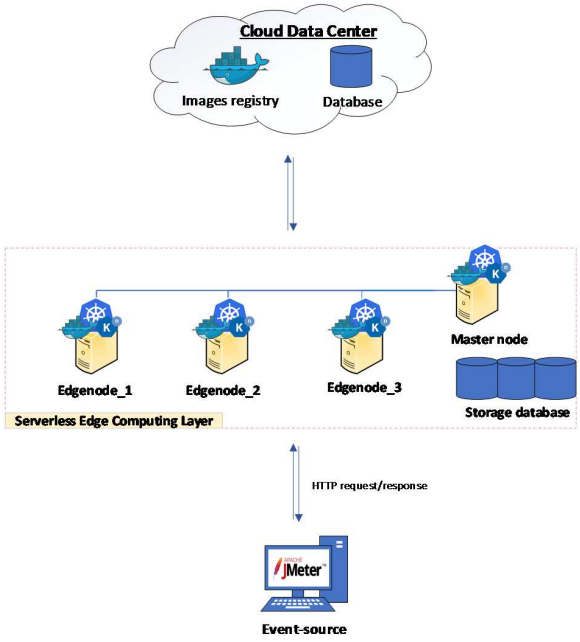

Fig. 2 Serverless edge computing testbed setup

We use the docker image registry [10] to store artifacts of functions in the cloud. With serverless edge layer, we deploy a serverless edge cluster over Kubernetes [18] includes; 3 physical worker nodes (Mini PC Intel® Core i5, 2.6Ghz, 4Gb memory, Ubuntu 18.04.5 LTS) denoted as edge nodes (clients) to enable distributed machine learning with federated learning and one master node (Intel® Xeon ® x86_64, 12 cores, 1.5Ghz, 47Gb memory).

We utilize Knative [19]- an open sources serverless platform to deploy machine learning applications as functions, due to many advantages over other serverless platforms [16]. The resources for serverless functions (200MB memory, 1CPU) and (100MB memory,0.4CPU) are allocated for the training and serving functions. This resource

management helps to efficiently utilize and respond to the limited resources of edge devices, optimizing for each function's workload. Besides, we deploy a database in the master node to store the weights of the trained model. Apache JMeter [20], a pure Java application, will be used to send HTTP POST requests that invoke serverless functions.

Workload In this testbed, we implement an image classifier and use the MNIST dataset to build this application because of the dataset's availability reproducibility. In this approach, we designed an image classifier followed by event-driven with training and serving events corresponding to model-training and model-serving serverless functions and hosted in all edge clients for federated learning. The operation of each function is shown in Fig. 3

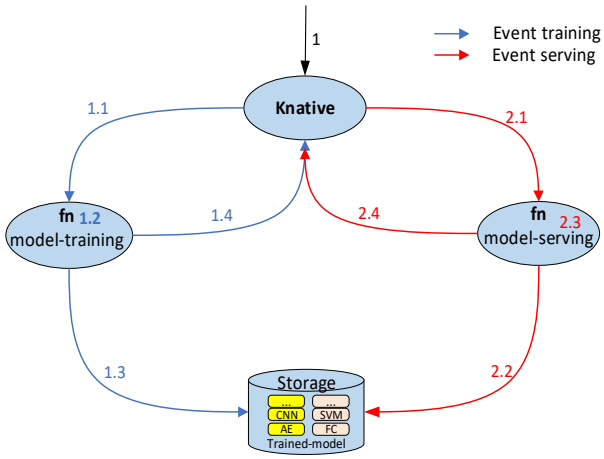

Fig. 3 The execution-graph of function workflow

To invoke a serverless function:
(1) Knative ingress gateway detects a function triggered by the HTTP request.
Training process: if the event of request is "training."
(1.1) the Knative framework triggers the execution of the training function "fn-model training."
(1.2) the training function loads and processes local data from storage. Afterward, the data is used to create the classifier.
(1.3) The weights matrix of the complete trained model is exported and used for aggregated processes in the cloud.
(1.4) As soon as the training function finished when the global converged

Serving process: if the event of the request is "serving."
(2.1) the Knative framework triggers the execution of the serving function "fn-model serving."
(2.2) the serving function call to the external storage to get a trained model.
(2.3) serving function performs input data processing, put through the trained model to perform predict.
(2.5) After the serving is finished, it notifies the Knative ingress gateway with the serving results with the corresponding request.

Our experimental evaluation is done with popular image classification models implemented with TensorFlow. The convolutional neural network (CNN) with the structure is shown in Table 1, with 317 066 parameters trainable.

Table 1 Structure of CNN model

Conv2D (32, (5x5), relu)
MaxPooling2D (2x2)
Dropout (0.25)
Conv2D (64, (5x5), relu)
MaxPooling2D (2x2)
Flatten ()
Dense (256, relu)
Dropout (0.25)
Dense (output, softmax)

When deploying an image classifier in a serverless edge environment, besides evaluating distributed deployment performance of federated learning, we focus on measurements in two specific situations of serverless, cold start and warm start. The *warm start* refers to the serverless function that can be rerun in a container that has previously completed its layers, runtime, and startup, ready to serve. If a new container is required to start and run a serverless function, this is known as a *cold start*. Based on that, in the evaluation section, we capture the following metrics:

- Model performance: the client and global accuracy when performing the training function.
- The response time: The total end-to-end time it takes for the function to return with results in a warm and cold start environment

B. Evaluation of serverless edge computing benefits on machine learning workflow deployment

1. Distributed machine learning applications capability

The federated learning training process of the image classifier model is executed through our proposed architecture. The training performance is shown in Fig. 4 and Fig. 5.

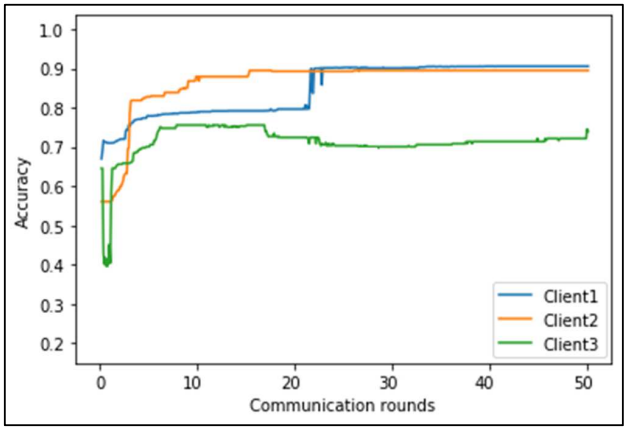

Fig. 4 The training accuracy in each edge node.

Fig. 4 shows the accuracy and the loss after 50 communication rounds of the test. The number of local epochs was set to 10 after empirical testing showed that this dramatically reduces the training time. As we can see, the accuracy stops increasing, and the loss of the global model stops decreasing much after about 22 communication rounds, so this can be a good cut-off point to finish the training process. The accuracy also reaches close to 90% for Client 1,2 and 70% for Client3. The accuracy of the global model reaches 82%, as shown in Fig. 5. The training accuracy of the

global model is good enough for the image classification task when implemented by the serverless function with a limited resource for each function. Thus, this result proves the capability of distributed machine learning deployment through our proposed approach.

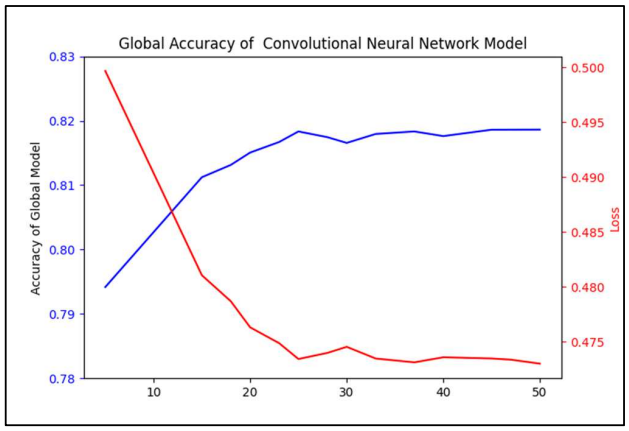

Fig. 5 The accuracy of global model in cloud

2. Response time reduction

We evaluate the effect between cold start and warm start situations on the function response time. As we can see in Fig. 6, the response time of the serverless training function is the same pattern as the training time when the number of epochs increases. The training time is a runtime of the training classifier process, and it is close to the response time in the warm start situation, a slight difference of about 0.05s.

On the other hand, the cold start scenario takes about a 5s delay compared to the actual time of the training process due to the time required to start the serverless training function for serving the request (scale from zero). Because the training process does not occur continuously over time, it only happens when there is a new update of the model's settings or a change in the system's training data. Therefore, the delay time in the cold start context is acceptable. The cold start training function helps reduce the problem of wasting resources when the system has to maintain a large number of resources to be ready for training, even when there is no requirement to perform model training.

In general, the serverless approach can ensure optimal request response time and execution time of the training function, minimizing the end-to-end delay of machine learning applications compared with deploying serverless functions in the cloud, which suffers from significant response time because of far data transmission distance between cloud and users.

Because the impact of cold start in serverless architecture on latency is well-documented, we evaluate the effect size of workloads on the response time of the serving function with various test samples in the test request (number of the images required to predict). The response time represents the end-to-end user latency when they request to image classifier to perform prediction.

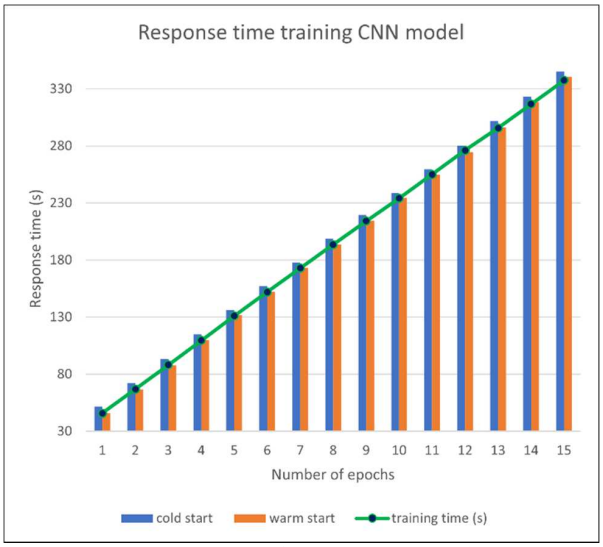

Fig. 6 Response time when trigger training function

As depicted in Fig. 7, the response time of requests with less than 500 samples is roughly the same, about 5s with cold start and 0.3s with warm star. On the contrary, there is a significant difference as the number of samples increases gradually from 500 to 10 000 samples; the response time value increases considerably as the number of samples increases. The experimental results show that besides the influence of the startup method, the size of the workloads is an essential factor affecting the response time of the serverless function during serving. So using the serverless approach can be flexible in optimizing the processing time for each invocation.

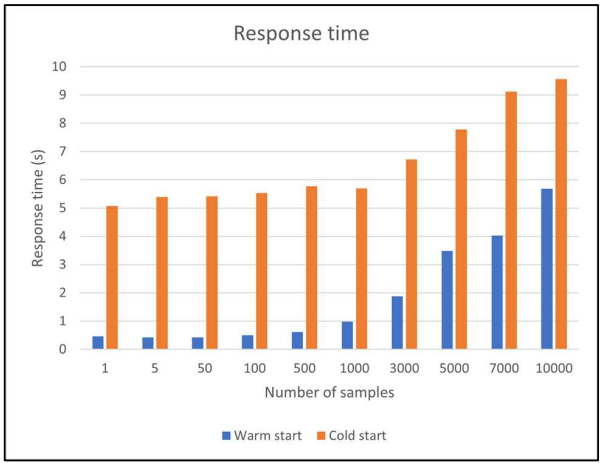

Fig. 7 Response time when trigger serving function

To minimize the effects of these factors, we should carefully choose the function startup method following the scenarios and user demands

V. Conclusion and Future Work

In this paper, we have proposed the serverless edge computing approach for implementing machine learning applications. The overall method has shown the suitability of serverless computing for machine learning workloads performed on distributed edge environments. We also comprehensively evaluate the performance with a famous case study image classification with the MNIST dataset in the

training and serving process. Our results indicate that the serverless approach can guarantee optimal response and running time, reduce the end-to-end delay of the machine learning application and show its capability to support distributed machine learning. We will further focus on collecting other metrics to evaluate the proposed approach's effectiveness in future works. Furthermore, we are investigating to reduce the impact of the startup method, besides optimization efficiency of deploying machine learning applications in serverless architecture by dividing machine learning applications into more serverless functions.

ACKNOWLEGMENT

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2021-2017-0-01633) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation)

REFERENCES

[1] Zhihan Lv, Dongliang Chen, Ranran Lou, Qingjun Wang, “Intelligent edge computing based on machine learning for smart city,” Future Generation Computer Systems, Volume 115, 2021, Pages 90-99, ISSN 0167-739X.

[2] Cloud Native Computing Foundation (CNCF), “Serverless Whitepaper v1.0,” Available on https://github.com/cncf/wgserverless/tree/master/whitepapers/serverless-overview. Last accessed: 10th Sep 2021

[3] Aslanpour, M., Toosi, A., Cicconetti, C., Javadi, B., Sbarski, P., Taibi, D., Assunção, M., Gill, S.S., Gaire, R., & Dustdar, S. (2021). “Serverless Edge Computing: Vision and Challenges.” 2021 Australasian Computer Science Week Multiconference

[4] Eismann, S., Scheuner, J., Eyk, E.V., Schwinger, M., Grohmann, J., Herbst, N., Abad, C.L., & Iosup, A. (2020). A Review of Serverless Use Cases and their Characteristics. ArXiv, abs/2008.11110.

[5] Malte S. Kurz. 2021. “Distributed Double Machine Learning with a Serverless Architecture.” In Companion of the ACM/SPEC International Conference on Performance Engineering (ICPE '21). Association for Computing Machinery, New York, NY, USA, 27–33. DOI:https://doi.org/10.1145/3447545.3451181

[6] D. Chahal, R. Ojha, M. Ramesh, and R. Singhal, "Migrating Large Deep Learning Models to Serverless Architecture," 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2020, pp. 111-116, DOI: 10.1109/ISSREW51248.2020.00047.

[7] Christidis, Angelos et al. “Serving Machine Learning Workloads in Resource Constrained Environments: a Serverless Deployment Example.” 2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA) (2019): 55-63.

[8] hang, X., Wang, Y., Lu, S., Liu, L., Xu, L., & Shi, W. (2019). “OpenEI: An Open Framework for Edge Intelligence.” 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 1840-1851.

[9] P. K. Gadepalli, G. Peach, L. Cherkasova, R. Aitken and G. Parmer, "Challenges and Opportunities for Efficient Serverless Computing at the Edge," 2019 38th Symposium on Reliable Distributed Systems (SRDS), 2019, pp. 261-2615, DOI: 10.1109/SRDS47363.2019.00036.

[10] Registry – Official Image, Docker Hub, Available on https://hub.docker.com/_/registry Last accessed: 10th Sep 2021

[11] A. Palade, A. Kazmi and S. Clarke, "An Evaluation of Open Source Serverless Computing Frameworks Support at the Edge," 2019 IEEE World Congress on Services (SERVICES), 2019, pp. 206-211, DOI: 10.1109/SERVICES.2019.00057.

[12] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi and M. Guizani, "A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond," in IEEE Internet of Things Journal, vol. 8, no. 7, pp. 5476-5497, 1st April 2021, DOI: 10.1109/JIOT.2020.3030072.

[13] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6), 141–142.

[14] Joao Carreira, Pedro Fonseca, Alexey Tumanov, Andrew Zhang, and Randy Katz. 2019. “Cirrus: a Serverless Framework for End-to-end ML Workflows.” In Proceedings of the ACM Symposium on Cloud Computing (SoCC '19). Association for Computing Machinery, New York, NY, USA, 13–24. DOI:https://doi.org/10.1145/3357223.3362711

[15] H. Lee, K. Satyam and G. Fox, "Evaluation of Production Serverless Computing Environments," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 442-450, doi: 10.1109/CLOUD.2018.00062.

[16] Li, Junfeng, et al. “Analyzing Open-Source Serverless Platforms: Characteristics and Performance.” ArXiv abs/2106.03601 (2021): n. pag.

[17] Wen, Jinfeng et al. “Understanding Characteristics of Commodity Serverless Computing Platforms.” ArXiv abs/2012.00992 (2020): n. pag.

[18] Kubernetes, Available on https://kubernetes.io/ Last accessed: 10th Sep 2021

[19] Knative, Available on https://knative.dev/docs/ Last accessed: 10th Sep 2021

[20] Apache JMeter Available on https://jmeter.apache.org/ Last accessed: 10th Sep 2021