Design and Development of Server-Client Cooperation Framework for Federated Learning

Jongbin Park and Seung Woo Kum
Information Media Research Center, Korea Electronics Technology Institute, Seoul, South Korea
Email: jpark@keti.re.kr

***Abstract*—Federated learning is a machine learning technique that enables distributed training without explicitly data sharing between multiple heterogeneous devices. In this paper, we propose and develop a practical federated learning framework to effectively support model deployment, aggregation, and client device monitoring. The proposed approach is designed as a micro-architecture service using container-related technologies such as Docker, Kubernetes, and Prometheus.**

***Index Terms*—Federated Learning Framework, Edge Computing, Micro Service Architecture**

I. Introduction

A. Introduction

Federated learning is a machine learning technique that enables distributed training without explicitly data sharing between multiple heterogeneous devices. This avoids the handling issue of sensitive information such as personal data, secure business data, and confidential information in machine learning. Its applications span multiple industries and services, including personal mobile devices, defense, surveillance, healthcare, and smart agriculture [1]–[4].

To build a federated learning platform, it is necessary to easily and effectively provide frequent communication traffic between the server and the device in the learning process [3]. In addition, proper load balancing between devices is required for cooperative training. To do this, it is necessary to monitor each device's resources and status [5]. In terms of improving learning performance, determining the reliability of device training results is also critical [6].

Therefore, we propose and develop a practical federated learning framework to effectively support model deployment, aggregation, and device monitoring. The proposed method utilizes container-related technologies such as Docker [7], Kubernetes [8], and Prometheus [9]. Through this, stable model distribution was supported and load balancing by device became possible. In addition, each device's resources and state can be monitored, enabling resource-aware federated learning to be supported. Training time is measured in terms of testing the developed platform. In addition, when federated learning on our platform, it was found that the overall learning speed was improved by reducing the data transfer overhead.

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2021-0-00907, Development of Adaptive and Lightweight Edge-Collaborative Analysis Technology for Enabling Proactively Immediate Response and Rapid Learning).

B. Related Work

1) Federated Learning: Fig 1 shows the concept of federated learning [1]–[4]. For federated learning, it is usually composed of a centralized server and several distributed devices. They do not directly share training samples from data sources, but they can transmit training results between the server and distributed devices. The centralized training server aggregates the distributed training results sent by multiple participating devices and updates the common machine learning model. The common updated model is redeployed to the devices, and new training is performed again in participating devices. By repeating this, the performance of the trained model gradually improves [1]–[4].

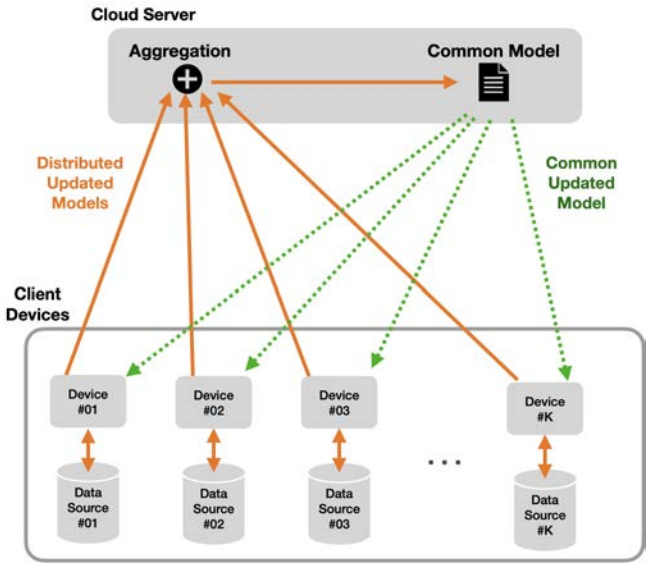


Fig. 1. Concept of federated learning [1]–[4].

2) Containerization: Containerization is operating system-level or application-level virtualization to deliver a software package called containers [10]–[12]. Multiple containers share the OS kernel, but they operate in their configuration files or dependency libraries in an isolated execution environment. However, they can also communicate with each other through application programming interfaces(APIs), etc.

Docker is a well-known and popular containerization technology [7], [10], [11]. These container technologies such as Docker [10] and container-d [13] can be effectively managed and controlled through a container orchestration system such as Docker Swarm [14] and Kubernetes [8], [10].

C. Proposed Framework and Implementations

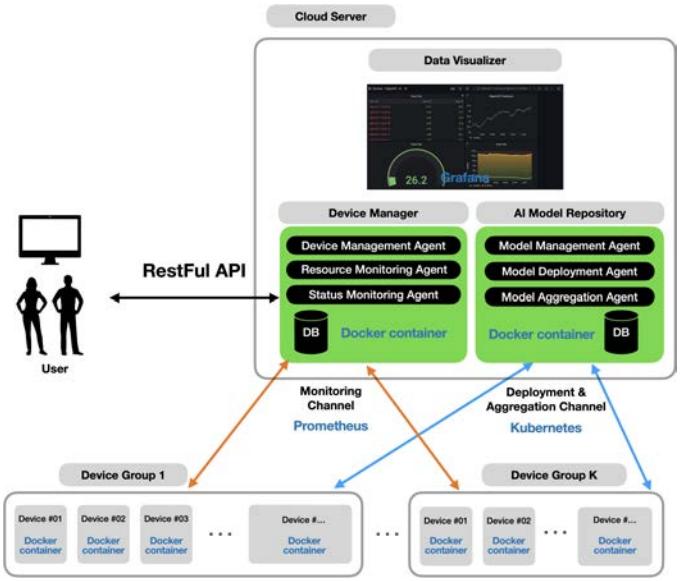


Fig. 2. Proposed framework for federated learning

1) Proposed Federated Learning Framework: Fig 2 shows the structure of the proposed framework for federated learning. The framework consists of a centralized server and distributed devices. The main modules of the cloud server are "Device Manager", "AI Model Repository", and "Data Visualizer". The "Device Manager" monitors the state and resources of the device. It also provides a RESTful(Representational state transfer) API [15] for registering devices. "AI Model Repository" serves to distribute and aggregate models of artificial neural networks to desired target devices.

The central server and distributed devices are grouped into a Kubernetes cluster [8]. Artificial neural network models are packaged based on Docker containers. In a cluster network, each Docker container can be deployed and controlled on a schedule or API command. Parallel processing and load balancing are also possible, and RESTful API [15] is provided for resource-aware federated learning [5], [6], [16]. The proposed framework periodically observes and records the operational status, computational capabilities, available memory, and network bandwidth.

2) Implementation: To monitor the device, for each device, Node Exporter [17] is installed and a container for network speed measurement is executed. Prometheus [9] then collects monitoring information provided by Node Exporter and the network bandwidth monitor process on each device. It periodically observes and records the operational status, computational capabilities, available memory, network bandwidth, etc. Grafana [18] is used for data visualization as shown in Fig 3. The user registers the desired AI model and is assigned device resources. This is done through the provided RESTful API [15]. The user then deploys the model to the desired devices and runs federated learning.

Device List (Cloud Server + Devices)

Device ID	Description	Group ID	Group Description	ID in Group
0	server	0	cloud group	0
1	Model management & Deployment	0	cloud group	1
2	Visualization	0	cloud group	2
3	Training and Inference (CPU)	1	network group 1	0
4	Training and Inference (GPU)	1	network group 1	1
5	Sensing	1	network group 1	2
6	Camera NVR	2	network group 2	0
7	IP Camera	2	network group 2	1
8	Training and Inference (GPU)	2	network group 2	2

(a) List of devices for federated learning

(b) Monitoring client devices

(c) AI model's statistics registered in repository

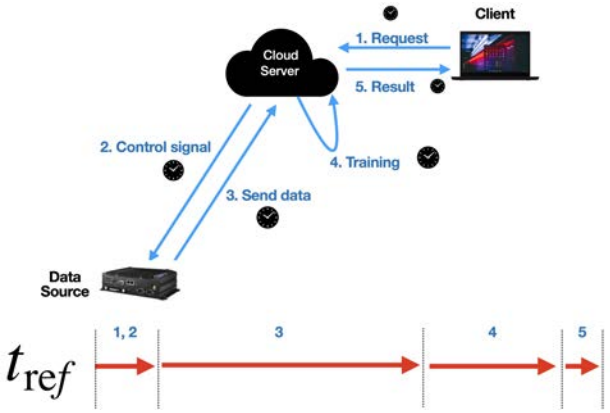
Fig. 3. Example of data visualization

3) Experimental Result: Compared to the cloud-based training method, when using federated learning, data transmission overhead can be reduced. Therefore, it has the advantage of reducing the total training time. Table I and Fig 4 show a brief configuration for training time measurement. We used the ImageNet dataset [19], [20] for the training. As a result, a speed improvement of about 11% was achieved as shown in Table II. Where the time-saving rate is expressed as (1). t_{ref} is a cloud server-based training time, and t_{new} is a client device-based training time.

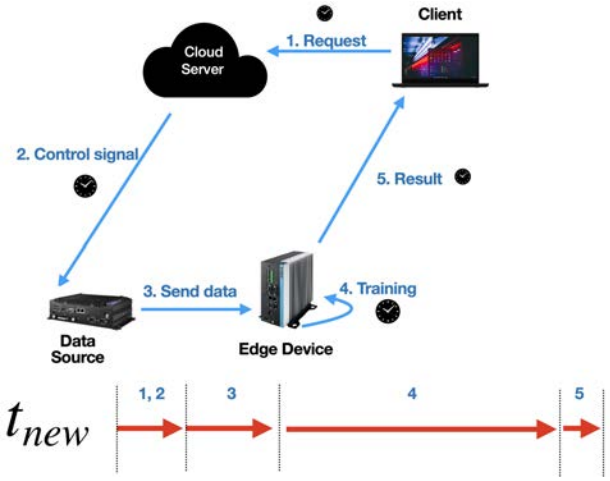
$$\Delta t = \frac{1}{n}\sum_{i=1}^{n}\frac{t_{ref} - t_{new}}{t_{ref}} \quad (1)$$

TABLE I
EXPERIMENTAL CONDITION FOR TRAINING TIME MEASUREMENT

	S/W	**H/W**
Server	Ubuntu 20.04 LTS, python 3.8.10, Model:YOLOv4 [21]	Xeon 56cores@2.2Ghz, 187 Gbytes RAM, Titan XP GPU x 8ea
Client	Ubuntu 20.04 LTS, python 3.8.10, Model:YOLOv4 [21]	i7 12 cores@3.30GHz, 15 Gbytes RAM, GTX1080 GPU x 1ea



(a) t_{ref} : Cloud server-based training time



(b) t_{new} : Client device-based training time

Fig. 4. Configuration for training time measurement

TABLE II
AVERAGE TRAINING LATENCY FOR 50 REPETITIONS

Cloud server-based training time: t_{ref}	**Client device-based training time:** t_{new}	**Time-saving rate:** Δt
6.02 [s]	6.72 [s]	$\sim$ -11.6 [%]

II. CONCLUSION

In this paper, we proposed and developed a practical federated learning framework to effectively support model deployment, aggregation, and device monitoring. The proposed method utilized container-related technologies such as Docker, Kubernetes, and Prometheus. Through this, stable model distribution was supported and load balancing by device became possible. In addition, each device's resources and state can be monitored, enabling resource-aware federated learning to be supported.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, 2020.

[3] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[4] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[5] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. Nguyen, and C. S. Hong, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 88–93, 2020.

[6] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.

[7] Docker, "[online] https://www.docker.com/."

[8] Kubernetes, "[online] https://kubernetes.io/."

[9] Prometheus, "[online] https://prometheus.io/."

[10] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.

[11] J. Fink, "Docker: a software as a service, operating system-level virtualization framework," *Code4Lib Journal*, no. 25, 2014.

[12] C. Pahl, "Containerization and the paas cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015.

[13] ContainerD, "[online] https://containerd.io/."

[14] F. Soppelsa and C. Kaewkasi, *Native docker clustering with swarm*. Packt Publishing Ltd, 2016.

[15] M. Masse, *REST API design rulebook: designing consistent RESTful web service interfaces*. " O'Reilly Media, Inc.", 2011.

[16] Z. Xu, Z. Yang, J. Xiong, J. Yang, and X. Chen, "Elfish: Resource-aware federated learning on heterogeneous edge devices," *Ratio*, vol. 2, no. r1, p. r2, 2019.

[17] NodeExporter, "[online] https://github.com/prometheus/node_exporter/."

[18] Grafana, "[online] https://grafana.com/."

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[21] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.