Rail Camera: An Edge Resource Management for Smart Farm Application

Seungwoo Kum
Information and Media Research Center
Korea Electronics Technology Institute
Seongnam, Republic of Korea
swkum@keti.re.kr

Youngkee Kim
Information and Media Research Center
Korea Electronics Technology Institute
Seongnam, Republic of Korea
youngkee.kim@keti.re.kr

Seungtaek Oh
Information and Media Research Center
Korea Electronics Technology Institute
Seongnam, Republic of Korea
stoh@keti.re.kr

Jaewon Moon
Information and Media Research Center
Korea Electronics Technology Institute
Seongnam, Republic of Korea
jwmoon@keti.re.kr

Alejandro Barrera Carvajal
CT Engineering Group
Madrid, Spain
alejandro.barrera@ctingenieros.es

Francisco Andres Perez
CT Engineering Group
Madrid, Spain
francisco.andres@ctingenieros.es

***Abstract*—Edge computing is one of the most important technology for the industry to realize analysis with deep learning technology. The edge computing architecture provides many benefits over cloud computing, since it reduces cloud communication requirements by processing data in itself. Among many industries for edge computing applications, smart agriculture is a promising one since it needs to handle various kinds of data such as ones from sensors and/or video streams. However, applying edge computing has been remaining a challenging task since it has different environment than other industries. It has less computational resources in the field and it should be able to deal with the target crop change. In this paper, an edge management platform with a rail camera for a smart farm is presented. First, A rail camera is implemented as an edge device which has accelerator resource where the deep learning application container can be deployed. The edge resource is federated with other cloud resources. And then, software for the edge management are presented - the Container Deployment Service and AI Model Repository, for container deployment and model deployment management, respectively.**

***Index Terms*—edge computing, deep learning inference, inference architecture, cloud to edge, smart farm**

I. Introduction

Edge computing is one of the most important technology for the industry to realize AI services, and also facing new challenges from applying it on the vertical industries. Edge has been extended its range in last decades from a standalone device such as IoT gateway to a component between edge-cloud continuum for service realisation. As a standalone device, it needs consideration on lightweight software such as operating systems or deep learning models, but as a component in edge-cloud continuum, it needs more consideration on networking and distribution. Many methods have been proposed to deal with this and actively employed on various use cases. For example, container technology is used for the packaging and delivery of a (micro)service, Kubernetes for the management of deployment of containerised applications, and Prometheus for the monitoring of resources. Those technologies are already widely used on the cloud computing, and extending its applications for the edge devices. However, it is also true that there are not many references for vertical industry applications that make use of the edge-cloud continuum. The authors presents a practical service for strawberry growth monitoring on a smart farm in this paper. More specifically, this paper will present the benefits of using Container Deployment Service (CDS) and AI Model Repository (AMR) for realization of vertical smart farm service.

This paper consists as follows. Section II presents description of the proposed architecture along with the two software components. Also the Rail Camera, which is a resource in a smart farm, is beneficial to the smart farm application. The detailed implementation on actual smart farm environment is presented in Section III, followed by the concluding section.

II. Smart Farm Edge Architecture

Applying deep learning inference to the agriculture is getting more focus these days. Zhou et al. [1] has proposed a method for plant phenotyping with deep learning and Bernotas et al. [2] has proposed a method for plant growth tracking. Parico and Ahamed [3] proposed more practical approach on detecting and counting crops with real-world images with Yolo v4. Zhang et al. [4] proposed for the analysis on the field using UAV such as drones. Although the results of those studies seems promising, it would not be easy to apply those methods since it requires kinds of specific hardware such as rig or drone, which is not suitable for the actual field, such as greenhouse. In addition, it also requires resources with high computational power. In this paper, an edge management tools are designed and implemented to provide more practical methods for providing deep learning-

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2021-0-01578, Smart Farm Platform for high-quality and traceable yield. A multi-purpose DDS based on proximal and remote sensing).

based service on agriculture. First, to provide a deep learning capable resource on the agricultural field, a Rail Camera is implemented. The Rail Camera is designed to have sufficient computational resources for data processing. And then, a set of software is implemented for the intuitive management of edge resources. Container Deployment Service (CDS) is a software that enables container deployment with a simple GUI, and AI Model Repository (AMR) is another software for the deep learning model management.

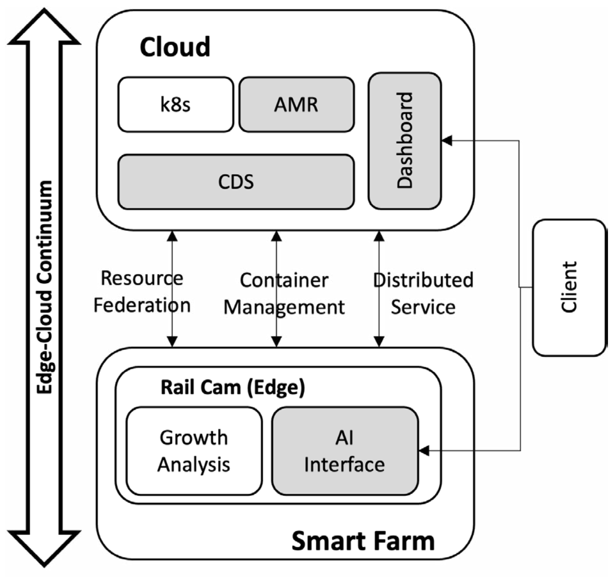


Fig. 1. Edge Management Architecture

Fig. 1 depicts the configuration of the proposed system. The gray boxes indicate the components that are implemented and presented in this paper. Those are the facilities to make it easy to implement distributed deep learning services on the edge-cloud continuum. In the design and implementation of the system, the authors have focused on two things. The first one is to make the rail camera a multi-purpose edge. The smart farm is an environment that is sensitive to the cost, both in installation and maintenance. Instead of having dedicated edge device for each use case, it would be effective if the intelligence (deep learning application) on the edge resource is replaceable to another one according to the use case. To this end, all the applications are implemented with AI-interface, so that they can rely on the same set of interfaces. Also the models for applications are managed in a common repository, the AI Model Repository. The next one is that the implemented service should be easy-to-use while reflecting the cloud-edge characteristics of distributed deep learning service. The target user of the system is not the cloud operator, but an a farmer who may not posses enough information on cloud technologies. Therefore, it is required to have an intuitive GUI interface. The Container Deployment Service (CDS) is designed to have a web browser-friendly interfaces to manage deployment of a container to a specific edge resource.

A. *Container Deployment Service (CDS)*

The Container Deployment Service (CDS) is a service to manage distribution of deep learning application on cloud and edge resources in an intuitive way. It has two window panes, the list of resources are displayed on the Node List Pane at the top, and the Container List Pane at bottom pane lists the containers that can be used on the cluster. The Node List Pane displays the list of nodes and status that are federated in a cluster. Each node displays the containers that are running on it. Also, it displays on where the resources are located, for example on the cloud or on the edge, and/or the specific location of the resources.

The CDS provides two links to manage deep learning services that are deployed on the cluster with CDS. If the deployed deep learning service is built with the AI Interface, that is explained in the next subsection, CDS will provide a GUI to connect to the service. And if the deployed deep learning service manages its model with AMR, that is presented in II-C, it will display a button to update or rollback the model being used on the deep learning service application. Those services are using intuitive interaction based on web GUI.

B. *AI Interfaces*

AI Interfaces is a software package to provide interfaces for a deep learning application. It differentiates itself from other model-serving platforms by including pre-processing and post-processing of a deep leaning inference process in it. It has internal REST interfaces to configure and control the whole inference pipeline of the application.

C. *AI Model Repository*

AI Model Repository is a repository of deep learning models with corresponding metadata. Users can register a trained model on the AMR with a set of metadata, from the basic information like name or version of a model, to the extended information such as the data set used for the training and intended use of it. It provides RESTful interfaces to access the metadata, model and retrieval via HTTP protocols.

III. Implementation and Workflow

This section presents the implementation of an agricultural service with the proposed architecture - strawberry crop detection with deep learning service on the edge resource. How the above mentioned facilities (CDS, AMR and AI Interface) are beneficial to the implementation is presented in this section.

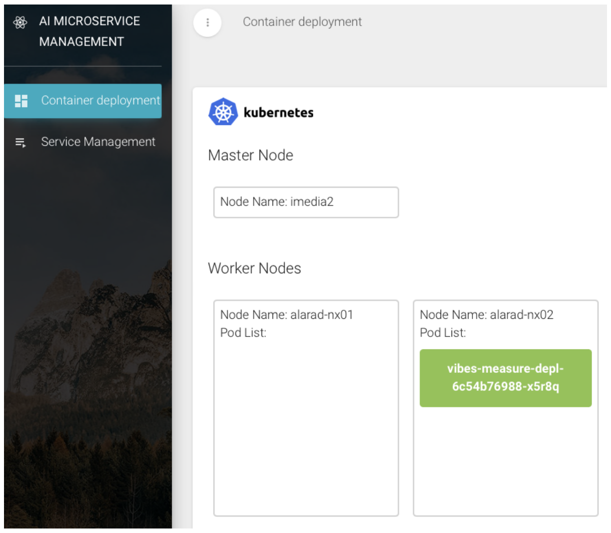
Rail Camera - an Edge resource A Rail Camera is a device which has a motor, a deep learning capable hardware and a camera for the detection of growth of a strawberry. For the acceleration hardware, Nvidia Xavier NX is used and federated with a cluster. Fig. 3 depicts the actual hardware implementation of the rail camera, that is installed on the actual strawberry farm.

Cloud resource Microsoft Azure cloud is used on the installation of CDS and AMR. Two virtual machines, each with 4 vCPUs, 16GB memory and 512GB storage are prepared and two software are installed with Python (version 3.8) environment. All the interfaces are implemented with Flask for RESTful APIs.

Kubernetes Cluster All the resources mentioned above are federeated in a cluster with Kuberenetes. All the resources are connnected via a virtual private network, and federated with Kuberentes version 1.22.4.

The implementation consists of two parts, the first part is to present use of CDS for the deployment of container, and the second part is to present AMR for the run-time-update of a model on a running container.

The CDS is implemented to show the characteristics of each resources. The characteristics of edge are displayed on the GUI of CDS, to make it able to choose proper resource for the distributed deep learning container. The GUI interface is depicted in Fig. 2. A simple drag-and-drop operation will deploy the container without needs of configuring YAML scripts or helm charts. Users can select the type of deployment between Pod and Deployment. And then, the interface to control the Rail Camera is presented. The Rail Camera container is implemented with the AI Interface, and the CDS can open the interface using Kubernetes NodePort resource. The interfaces are depicted in Fig. 4.



Container deployment

Master Node

Node Name: imedia2

Worker Nodes

Node Name: alarad-nx01
Pod List:

Node Name: alarad-nx02
Pod List:
vibes-measure-depl-6c54b76988-x5r8q

Fig. 2. GUI of CDS.

While CDS is focusing management of the containers, the AMR is designed for the management of the trained model to be used on the deployed containers. A metadata file (manifest.json) that describes the configuration of the model is used for the identification of a model with AMR. The model files can be registered to the AMR with the metadata file. Here in this implementation, a Yolo model file is used for the detection of strawberry from the images taken by the Rail Camera from the smart farm.

IV. Conclusion

The design and implementation of management of container deployment on edge-cloud cluster is presented in the paper.

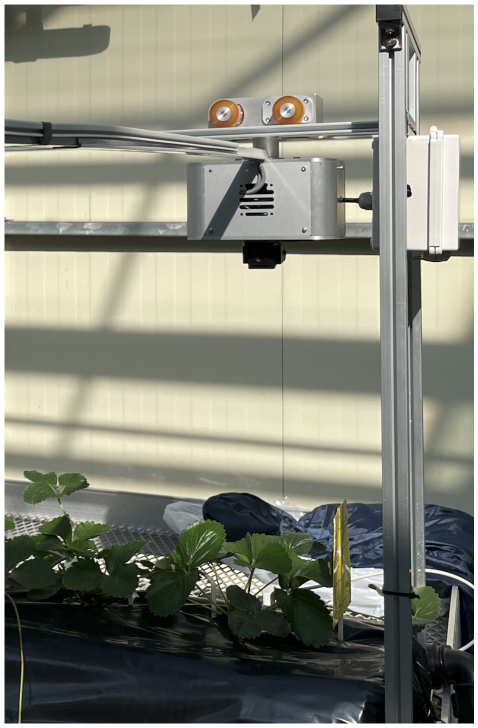
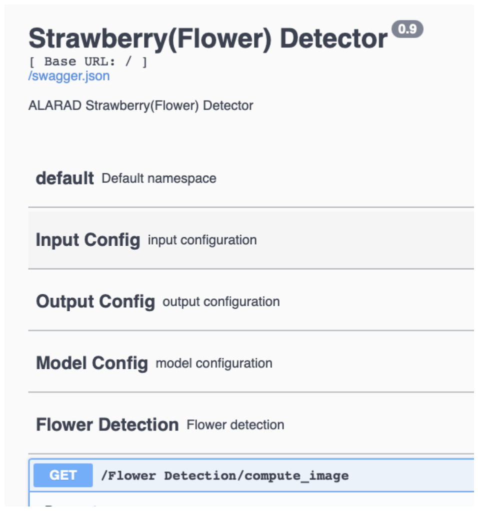


Fig. 3. Rail Camera in a Strawberry Farm.



Strawberry(Flower) Detector 0.9
[Base URL: /]
/swagger.json

ALARAD Strawberry(Flower) Detector

default Default namespace

Input Config input configuration

Output Config output configuration

Model Config model configuration

Flower Detection Flower detection

GET /Flower Detection/compute_image

Fig. 4. Deep learning container on a Rail Camera.

The set of tools - CDS, AMR and AI Interfaces - is presented for the management of deep learning-based service on the mixed edge-cloud resources. The characteristics of edge resources on edge-cloud continuum are easily identified with the help of CDS, and access of the deployed deep learning container provided with the intuitive GUI and AI interface module. Further, the AMR enables run-time update of a deep learning model which ensures better quality of service. In this paper, the use case of strawberry detection is presented, and it shows that Rail Camera can host many kinds of deep leaning applications with the help of CDS and AMR. The authors plan to continue developing applications with the Rail Camera for the yield prediction of the strawberry.

References

[1] J. Zhou, C. Applegate, A. D. Alonso, D. Reynolds, S. Orford, M. Mackiewicz, S. Griffiths, S. Penfield, and N. Pullen, “Leaf-gp: An open and automated software application for measuring growth phenotypes for arabidopsis and wheat,” *Plant Methods*, vol. 13, 12 2017.

[2] G. Bernotas, L. C. Scorza, M. F. Hansen, I. J. Hales, K. J. Halliday, L. N. Smith, M. L. Smith, and A. J. McCormick, “A photometric stereo-based 3d imaging system using computer vision and deep learning for tracking plant growth,” *GigaScience*, vol. 8, 5 2019.

[3] A. I. B. Parico and T. Ahamed, “Real time pear fruit detection and counting using yolov4 models and deep sort,” *Sensors*, vol. 21, 7 2021.

[4] Y. Zhang, J. Yu, Y. Chen, W. Yang, W. Zhang, and Y. He, “Real-time strawberry detection using deep neural networks on embedded system (rtsd-net): An edge ai application,” *Computers and Electronics in Agriculture*, vol. 192, 1 2022.