Deploy Anywhere: An End to End Kubernetes based Mobility Service Framework

Seiichi KOIZUMI Yong JUN KAI Aman GUPTA

We have been developing a mobility edge/cloud service framework called 'misaki', which integrates vehicle edge, network edge (e.g. MEC) and cloud seamlessly. It also deploys and operates mobility service modules on this heterogeneous environment. Service developers are able to develop their apps on cloud, test it on virtual vehicle environment and deploy it anywhere. It also removes the concern for differences of environment (e.g. vehicle edge/cloud locational differences and vehicle edge such as each type of vehicle has differences of computing resources)

Key words :
edge computing, connected vehicle, MaaS, IoT, Orchestration, Service Mesh

1. Introduction

Software service economy has been rapidly expanding and service providers are doing DevOps by leveraging public clouds and cloud native OSS. Thus, it is said that 'Kubernetes is becoming the Linux of the cloud' and Kubernetes related eco-system are becoming more mainstream[1]. In the automotive industry, vehicles are connected to the cloud and the demand for development of mobility services is growing rapidly. Responding appropriately to this demand is thus, a concern for all big mobility service providers . As mobility service users'demands are ever-changing, developers require a flexible and agile development environment, like the cloud native approach. However, there are some technical barriers tin introducing cloud native approach to vehicles; 1) Vehicle embedded software skill sets are different from those of cloud service software, 2) Limited vehicle edge computing resources and lack of scalability, 3) Vehicle constraints; Unstable LTE/3G connections, limited power supply when engines are switched off, CAN BUS data decoding/encoding and embedded software environment challenges.

To solve these technical issues, we developed 'misaki', end to end mobility service development, deployment and operation framework. It is a cloud native framework for vehicle/network edge and cloud environment. We defined each vehicle/network edge as a small cloud. By breaking down and deploying Kubernetes functions on the edge and cloud by resource availability, using functionalities of service-mesh and digital twin technology, it seamlessly integrates vehicle/network edge and cloud. Mobility service developers are enabled to develop, test and deploy their software modules/applications easily.

misaki has two technical features: End-to-End (E2E) orchestrator and Vehicle service mesh. The E2E orchestrator decouples from application runtime and infrastructure so that application developers are able to allocate the appropriate amount of resources and location for their product. Once they prepare their container images, our Kubernetes then deploys and manages those containers across edge and cloud according to user configuration and available resources.

A service-mesh, according to the early definition in [18], is a “dedicated infrastructure layer for handling service-to-service communication. It’s responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application”. A service-mesh takes care of service discovery, load balancing, error handling as well as monitoring and management functions. For an overview of cloud oriented solutions, we refer to [14].

For a vehicle service mesh, we have to consider the specific challenges of edge computing inside vehicles, regarding computing and networking resources. This means in particular the scalability of containers toward more limited computing platforms (also called vertical scalability. While there are promising works container technologies for various kinds of devices, e.g. [19][13], a main open challenge is the support for application mesh, e.g. discussed in [14]. Thus, this paper presents a first approach for combining container-based approaches with Kubernetes and service mesh technologies in the vehicle-edge context.

The remainder of this paper is organized as follows. Section 2 briefly surveys related works. Section 3 explains the approach in detail. Section 4 evaluates workloads on the edge environment. Section 5 discusses effectiveness of our approach.

2. Related Works

Service frameworks that use data and devices as leverage are defined in IEEE P2413[2], ITU-T Y.2060[3]. This framework accelerates service developments by smoothly collecting data from sensor devices, comprehending situations and orchestrating applications appropriately[4]. Public cloud’s IoT services (e.g. AWS IoT, Azure IoT) provide fundamental functions of this framework. Vehicles have vehicle-specific characteristics such as mobility, wireless, hetero-data, and battery characteristics[5]. To cover these characteristics, a mobility services framework[6] and vehicular cloud computing[7] has been proposed. However, these approaches are not able to create service ecosystem which are supported by developer communities and keep generating new services.

From the perspective of cloud service ecosystem, cloud native approach[8] is getting supported by innovative service providers and developer communities. Cloud-native is a software design, development and operation approach that takes advantage of the cloud to develop services. Containers, service meshes, micro-services, immutable infrastructure, and declarative APIs exemplify this approach. These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow developers to make high-impact changes frequently and predictably with minimal toil.

Current commercial cloud systems offer service mesh concepts, see e.g.[16][17] offer service mesh for cloud but not for edge components. For a survey, we refer to[14], which also introduces the idea of edge support for service mesh but does not offer any specific solution or discussion. For some challenges regarding service mesh in IoT situations, the problem of latencies is considered in[15]. Here, we focus on the flexible deployment in edge computing as well as the management and orchestration.

To accelerates mobility service development and operation, OEM service providers already used cloud native approach. BMW provides multiple services, DriveNow, ParkNow and ChargeNow and these services utilizes cloud native approach[9]. The Ford Motor Company also defined “ Cloud Native Reference Application” framework[10] to share best practices of mobility service.

3. E2E Deploy Anywhere Framework, misaki

3.1 Overview

Our misaki framework defines three layers through vehicle/network edge and cloud:

- ●Application processing layer: Developing applications as containers on the cloud and deploying it to vehicle edge without being affected by the vehicle’s resource constraints.
- ●Service mesh network layer: It connects or disconnects containers based on service mesh controller. It also enables vehicle-to-cloud container migration by control traffic and manage IP addresses.
- ●Distributed data layer (under development): containers is able to retrieve/register data from any distributed data cache on vehicle/network edge and cloud.

To cover various requirements of mobility software service, such as minimizing uploading cost, maximizing data processing throughput and real time processing, misaki orchestrates these three layer’s resources (computing resource, network resource and storage resource) allocation and deployment for three locations (vehicle edge, mobile edge and cloud).

3.2 Technical Features and Prototype Architecture

To follow the latest software development practices, the framework must allow the mobility service developer to have full ownership of their application. They can choose their language, libraries, and software environment to build the application and they can fully control the application lifecycle. This increases developer productivity and scalability. Developer productivity is one of the most important factors for a successful service development. Furthermore, scalability in software development enhances business scalability and agility. Therefore, misaki is carefully designed with openness in mind and it is based on robust and popular OSSs like Kubernetes and Envoy.

misaki prototype provides the following functionalities;

- ●End to end (E2E) orchestrator: It provides unified operation of design, development, deployment and updates of containers on vehicle/network edge and cloud environment.
- ●Vehicle service mesh: Split application logic layer and network layer and orchestrate the network traffic in operation phase between service components (containers).

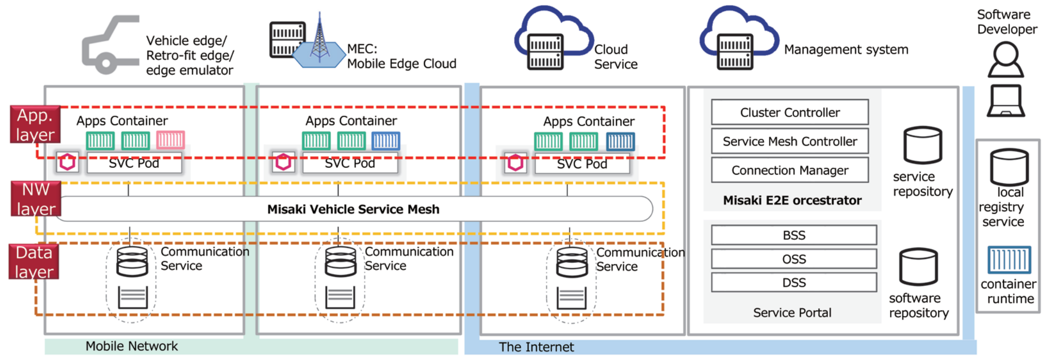

Fig. 1 End to end deploy anywhere framework

3.2.1 E2E Orchestrator

The E2E orchestrator decouples application runtime and infrastructure so that the service developers can choose the most suitable technology stack for their product. Once they prepare their container images, our Kubernetes then deploys and manages those containers on both edge and cloud according to user's configuration and available resources. This is particularly important if the services need API access or data sources from the vehicle or from the cloud. The platform can orchestrate the access to these via a service mesh. However, it remains to the developer to find a suitable placement of service components based on service needs, service context, and capacity of computing resources.

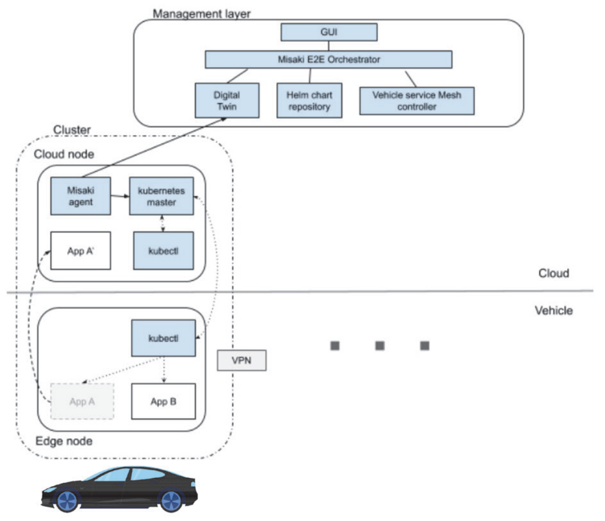

Fig. 2 E2E Orchestrator Architecture

E2E Orchestrator mainly consists of three components (Fig. 2): Kubernetes as edge and cloud container orchestrator, digital twin for edge and cloud configuration management, Helm chart repository for container package management and Vehicle service mesh controller for seamless communications between services. We first decompose Kubernetes functions into edge kubectl and the other functionalities and only the edge kubectl is allocated to vehicle/network edge to minimize workload footprints. This architecture gives positive impacts to developer efficiency: Service developers can easily create their application on the cloud, test it on the cloud and deploy their application to the edge. There are no special technology for edge application development with E2E Orchestrator. All of the concepts and technologies are widely known in service developers: Container and Kubernetes are de facto of in modern software development. Fig. 3 shows GUI console of E2E Orchestrator. Service developer is able to register their container applications in Helm chart format and deploy containers to edge and cloud.

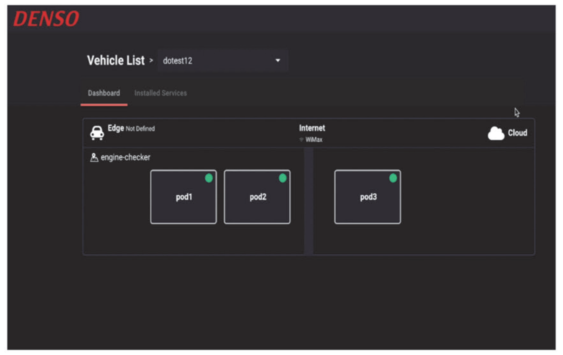

Fig. 3 E2E Orchestrator Console

3.2.2 Vehicle Service Mesh

Vehicle service mesh decouples vehicle's network concerns from applications and connects the service components. In service mesh[11], data plane proxies ("sidecars") run alongside of containers and they handle cloud network connection to other service components. The service mesh with the use of these proxies takes care of timeouts, retries, traffic volume controls, service discoveries, load balancing and more. However, vehicle edges have additional network concerns like no network connection occasionally, heterogeneous connection (4G, 3G, Wi-Fi, DSRC, LPWA) and traffic priorities. To address these concerns, we developed Vehicle service mesh as an enhancement of service mesh. As Vehicle service mesh interconnects vehicle/network edge and cloud seamlessly and addresses network concerns. Service developers are then able to focus on their business values/logic implementation. Thus, improving developer's productivity.

Fig. 4 illustrates Vehicle service mesh functionalities. We chose Envoy proxy, the most widely adapted software for service mesh proxies. We implement custom-made control plane and network functions (e.g. data queue, network connection selector, data compression and more).

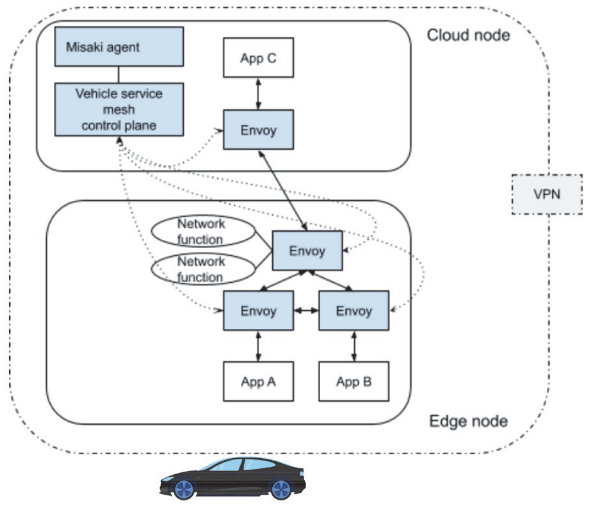

Fig. 4 Vehicle Service Mesh Architecture

4. Evaluation

As computing resources of vehicle edge are limited, workload footprints are an important factor. To compare workloads, we set up four type of edge container orchestrator: Kubernetes, k3s[12], Balena IoT[13] and misaki. Kubernetes is a full set of Kubernetes and it contains entire master node and worker node functions. k3s is a light weight Kubernetes. Balena IoT is an edge IoT solution and support container deployment. Its functions are similar to AWS IoT and Azure IoT but Balena IoT footprint is smaller than them. For the edge environment, we prepare 'Intel Core-i5 6500TE 2.3GHz, 8GB RAM computer' and five sensor data processing containers.

Fig. 5 shows time series data of each orchestrator's CPU usage. k3s is higher than Kubernetes, Balena IoT is lower than them and misaki shows minimum CPU usage. As misaki off loads master node and related functions to cloud, we expects total misaki workload is almost same as Kubernetes.

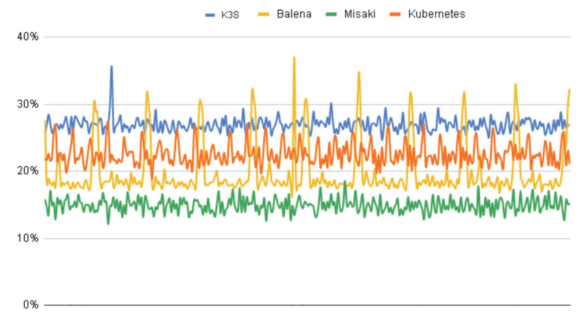

Fig. 5 CPU Workloads on Core-i5 6500TE 2.3GHz

5. Discussion

To launch and accelerate mobility service, developer's productivity is key factor. Existing mobility service frameworks requires to get use to their environment and it reduces productivity. Service developers are already fond of using cloud native approach and possess technical assets. To maximize their productivity, misaki provides the following features;

- Utilizing skillsets and reusing software development assets, such as CI/CD pipeline, install package template (like helm chart), test environment, sidecar containers, monitoring environment and reusing policy.
- Seamless edge/cloud integration: Developers is able to deploy their containers to vehicle/network edge. According to the workload or low battery issues in vehicle, containers can be migrated to the cloud.
- Software-defined networking (SDN): Each mobility service have different processing priorities, so we provides programmable network layer based on service mesh. Developer is able to allocate appropriate network connection like LTE/Wi-Fi

and insert network functions (e.g. IDS/IPS, FW, encryption, queue)

To evaluate productivity improvement, we will have user trials of mobility service development.

6. Conclusion

In this paper, we proposed a mobility edge/cloud service framework called ‘misaki’, which integrates vehicle edge, network edge and cloud seamlessly. It also deploys and operate mobility service modules on this heterogeneous environment. Service developers can develop their applications on cloud, test it on virtual vehicle environment and deploy it anywhere. Also there is no need to consider differences of environment.

Toward the realization of the misaki framework, further studies need to be conducted on an implementation study on distributed data layer and consideration on security.

References

1) CNCF Cloud Native Landscape, https://landscape.cncf.io/
2) IEEE P2413 “Standard for an Architectural Framework for the Internet of Thing”, https://standards.ieee.org/standard/2413-2019.html
3) ITU-T Y.2060 “Overview of the Internet of things”, 2012, https://www.itu.int/rec/T-REC-Y.2060-201206-I
4) A. Al-Fuqaha, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications”, IEEE Communications Surveys & Tutorials, Volume: 17, Issue: 4
5) N.Lu, et. al., “Connected Vehicles: Solutions and Challenges”, IEEE Internet of Things Journal, Volume: 1 , Issue: 4 , 2014
6) M. Whaiduzzaman, et. al., “A survey on vehicular cloud computing”, Journal of Network and Computer Applications, 2014
7) W. He, et. al., “Developing vehicular data cloud services in the IoT environment”, IEEE Transactions on Industrial Informatics, Volume10, Issue2, 2014
8) “CNCF Cloud Native Definition v1.0”, https://github.com/cncf/toc/blob/master/DEFINITION.md
9) BMW “Cloud Native@BMW Group, Technology for the agile transition”, RedHat Summit 2017
10) Ford Motor Company, “Ford Motor Company’s Cloud Native Reference Application”, SpringOne Platform 2017
11) W. Li, Y. Lemieux, et. al., “Service Mesh: Challenges, State of the Art, and Future Research Opportunities,” IEEE SOSE, 2019.
12) k3s: light weight kubernetes, https://github.com/rancher/k3s
13) Balena IoT, https://www.balena.io/
14) Li, Wubin, et al. “Service mesh: Challenges, state of the art, and future research opportunities.” 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE). IEEE, 2019.
15) X. He and F. Deng, “Research on Architecture of Internet of Things Platform Based on Service Mesh,” 2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Phuket, Thailand, 2020, pp. 755-759,
16) AWS App Mesh, https://aws.amazon.com/de/app-mesh/
17) Google Cloud, Traffic Director, https://cloud.google.com/traffic-director/docs/traffic-director-concepts
18) https://buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/
19) S. Liu and Y. Zu, “Design and Research of Edge Layer Service Platform Based on Flexible Service Architecture,” 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2019, pp. 555-560, doi: 10.1109/ICSESS47205.2019.9040827.

著者

小泉 清一
こいずみ せいいち
情報通信事業部 クラウドサービス開発部
車載エッジコンピュータ，コネクティッド基盤技術の開発に従事

Yong Jun Kai
ヨン ジュンカイ
情報通信事業部 クラウドサービス開発部
Edge computing and cloud software engineer

Aman Gupta
アマン グプタ
情報通信事業部 クラウドサービス開発部
Edge computing and cloud oftware engineer