# A Comparative Research on Open Source Edge Computing Systems

Jiayue Liang, Fang Liu(✉), Shen Li, and Zhenhua Cai

Sun Yat-sen University, Guangzhou 510006, China
{liangjy77,lish286,caizhh8}@mail2.sysu.edu.cn,
liufang25@mail.sysu.edu.cn

**Abstract.** With the development of edge computing, open source communities have put forward several edge computing systems. This paper discussed about edge computing and its current situation, then presented typical open source edge computing systems such as EdgeX Foundry, Azure IoT Edge, CORD, Apache Edgent and Akraino Edge Stack, and gave a comparison about them on their characteristics. A comparison study on their characteristics were given to help users to understand these open source edge computing systems and make choices.

**Keywords:** Edge computing systems · Open source

## 1 Introduction

Cloud computing is the dominated computing paradigm in the past decade, providing efficient, centralized computing platform for big data. Recently, with the development of Internet of Things (IoT) [1] and the popularization of 4G/5G, massive devices are connected into the network, and a huge amount of data are generated all the time, which challenges the linearly increasing capability of cloud computing. Meanwhile, many applications of IoT demand lower response time and improved security, and the cloud computing cannot meet the requirements. Under this background, edge computing [2] is put forward to solve these problems. Edge computing takes advantage of the compute resources of edge nodes to perform computation so that the data can be processed at the edge. Several software systems for edge computing have been designed to help the popularization of edge computing, such as cloudlet [3] and Firework [4]. Open source communities began to develop open source systems such as EdgeX Foundry [5] and Apache Edgent [6]. However, there is no public literature to help users to select and use these open source edge computing systems. In this paper, our research analyzes several open source systems for edge computing to provide an overview of these systems.

The remaining parts of this paper are organized as follows. In Sect. 2, we discuss the current situation of edge computing. In Sect. 3, we present five open source edge computing systems. In Sect. 4, we give a comparison about these open source systems. Finally, Sect. 5 summarizes our conclusions.

X. Sun et al. (Eds.): ICAIS 2019, LNCS 11633, pp. 157–170, 2019.
https://doi.org/10.1007/978-3-030-24265-7_14

## 2 Edge Computing

Edge computing is a new paradigm and it calls for technologies to enable computation to be performed at the edge of the network in close proximity to data sources. W. Shi et al. gave their definition on “edge” as “any computing and network resources along the path between data sources and cloud data centers” [2]. Edge computing is expected to solve problems that challenge cloud computing. One problem is the pressure of bandwidth. Not all the data from the IoT objects are necessary to be sent to the cloud for processing, and most of them can be processed at the edge. Edge computing performs computing tasks on edge devices to process part of the data so as to reduce the data volume sent to the cloud and the pressure of bandwidth. Another problem is the latency. Centralized processing in the cloud may not meet the need of real-time processing for IoT applications. Edge computing processes data at the edge so as to get a shorter response time. The third problem is security and privacy [7]. In many cases, the data are not expected to share due to privacy concern and a local solution is needed. Edge computing provides support to process data at the edge without sending them to the cloud. Edge computing has a wide application area such as smart city, smart home, intelligence manufacturing and lots of other use cases in IoT.

Edge computing has obtained a rapid development. In 2015, the Open Edge Computing initiative [8] was published by Vodafone, Intel, Huawei and Carnegie Mellon University. Then the Open Fog Consortium [9] was established by Cisco, Microsoft, Intel, Dell, ARM and Princeton University. In 2016, the Edge Computing Consortium (ECC) [10] was established in Beijing by Huawei, Chinese Academy of Sciences, Intel, ARM, etc. What’s more, software systems developed specifically for edge computing were launched gradually such as Cloudlet, ParaDrop [11] and Firework. A cloudlet is a small-scale cloud datacenter at the edge of the network, which provides compute resources to support running interactive mobile applications provided by mobile devices with low latency [3]. ParaDrop, as an edge computing framework, leverages the computing resource of wireless gateways or access points at the edge for developers to create and run services. Firework focus on the data sharing problem among the applications for the Internet of Everything (IOE), and put forward a computing framework to enable distributed data processing and sharing among multiple stakeholders such as edge nodes or the cloud in the hybrid cloud-edge environment [4]. As for commercial companies, especially cloud service providers, Amazon published AWS Greengrass [12], which can deploy applications to local devices for data processing with the help of the AWS Cloud capabilities such as deployment and management. Microsoft and Alibaba Cloud published solutions namely Azure IoT Edge [13] and Link IoT Edge [14] respectively, which are similar to AWS Greengrass with the concept “hybrid cloud-edge analytics”. Open source communities also launched several open source edge computing systems, and we present them in Sect. 3.

Open source edge computing systems, supported by multiple companies with latest technologies, accelerate the development of edge computing. It is meaningful to perform a study on these systems so as to help users to understand these systems. In this paper, we studied five of these systems in detail.

## 3 Open Source Edge Computing Systems

The Linux Foundation published two projects, EdgeX Foundry in 2017 and Akraino Edge Statck [15] in 2018. The Open Network Foundation (ONF) launched a project namely CORD (Central Office Re-architected as a Datacenter) [16]. The Apache Software Foundation published Apache Edgent. Microsoft published Azure IoT Edge in 2017 and announced it was available and open source in 2018.

Among these open source edge computing systems, EdgeX Foundry and Apache Edgent focus on IoT, they try to solve problems which bring difficulties to practical application of edge computing in IoT. Azure IoT Edge provides with hybrid cloud-edge analytics, which helps to migrate cloud solutions to IoT devices. CORD and Akraino Edge Stack focus on providing edge cloud services.

### 3.1 EdgeX Foundry

EdgeX Foundry is a standardized interoperability framework for IIoT edge computing, whose sweet spots are edge nodes such as gateways, hubs, routers [5]. It can connect with various sensors and devices via different protocols, manage them and collect data from them, and export the data to a local application at the edge or the cloud for further processing. IIoT, where a huge amount of data are produced all the time, has urgent need for combination with edge computing. However, the diversity of hardware, software and communication protocols among sensors and devices in IIoT brings a great difficulty on managing them and collecting data from them. EdgeX Foundry is such a framework to provide interoperability to solve this problem, aiming to simplify and standardize the foundation of computing architectures in the IIoT market, and create an ecosystem of interoperable components. EdgeX is designed to be agnostic to hardware, CPU, operating system, and application environment. It can run natively or run in docker containers.

Figure 1 [5] shows the architecture of EdgeX Foundry. “South side” at the bottom of the figure includes “all IoT objects, within the physical realm, and the edge of the network that communicates directly with those devices, sensors, actuators, and other IoT objects, and collects the data from them” [5]. Relatively, “north side” at the top of the figure includes “the Cloud (or Enterprise system) where data are collected, stored, aggregated, analyzed, and turned into information, and the part of the network that communicates with the Cloud” [5]. EdgeX Foundry connects these two sides regardless of the differences of hardware, software and network. EdgeX tries to unify the manipulation method of the IoT objects from South Side to a common API, so that those objects can be manipulated in the same way by the applications of North Side.

EdgeX uses a Device Profile to describe a south side object. A Device Profile defines the type of the object, the format of data that the object provides, the format of data to be stored in EdgeX and the commands used to manipulate this object. Each Device Profile involves with a Device Service, which is a service that converts the format of the data, and translates the commands into instructions that IoT objects know how to execute. EdgeX provides SDK for developers to create Device Services, so that it can support for any combination of device interfaces and protocols by programming.

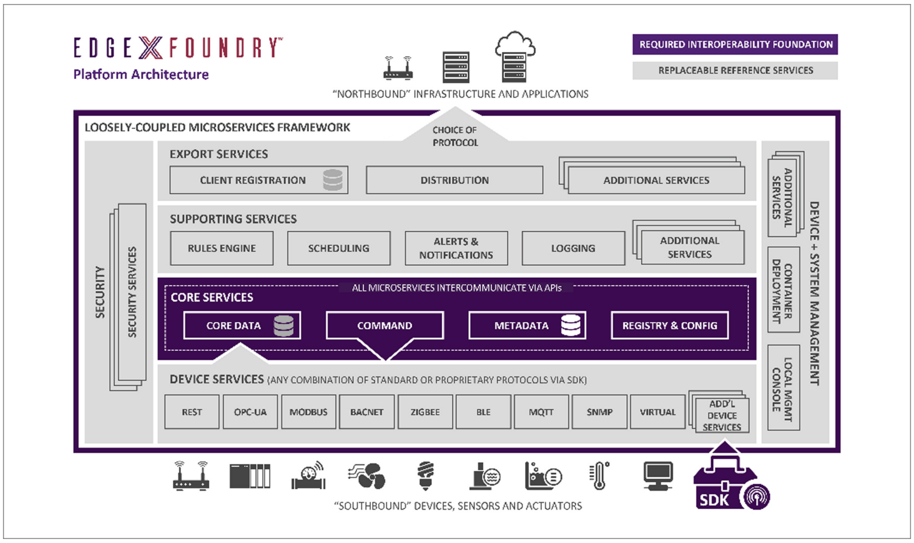

**Fig. 1.** Architecture of EdgeX Foundry

EdgeX consists of a collection of microservices, which allows services to scale up and down based on device capability. These microservices can be grouped into four service layers and two underlying augmenting system services as depicted in Fig. 1. The four layers are Device Services Layer, Core Services Layer, Supporting Services Layer and Export Services Layer. Two underlying augmenting system services are System Management and Security. Each layer consists of several components and all of these components use a common Restful API for configuration.

**Device Services Layer.** This layer consists of Device Services. According to the Device Profiles, Device Service Layer converts the format of the data, sends them to Core Services Layer, and translates the command requests from Core Services Layer.

**Core Services Layer.** This layer consists of four components Core Data, Command, Metadata, Registry and Configuration. Core Data is a persistence repository as well as a management service. It stores and manages the data collected from the south side objects. Command is a service to offer the API for command requests from north side to Device Services. Metadata is a repository and management service for metadata about IoT objects. For example, the Device Profiles are uploaded and stored in Metadata. Registry and Configuration provides centralized management of configuration and operating parameters for other microservices.

**Supporting Services Layer.** This layer is designed to provide edge analytics and intelligence [5]. Now the Rules Engine, Alerting and Notification, Scheduling and Logging microservices are implemented. We can set a target range of data to trigger a specific device actuation as a rule and Rules Engine helps to realize by monitoring the incoming data. Alerting and Notifications can send notifications or alerts to another system or person by email, REST callback or other methods when a urgent actuation or a service malfunction happens. Scheduling can set up a timer to regularly clean up the stale data. Logging is used to record the running information of EdgeX.

**Export Services Layer.** This layer connects EdgeX with North Side and consists of Client Registration and Export Distribution. Client Registration enables clients like a specific Cloud or a local application to register as recipients of data from Core Data. Export Distribution distributes the data to the Clients registered in Client Registration.

**System Management and Security.** System Management provides with management operations including installation, upgrade, start, stop and monitoring as EdgeX is scalable and can be deployed dynamically. Security is designed to protect the data and command of IoT objects connected with EdgeX Foundry.

EdgeX is designed for use cases which deal with multitudes of sensors or devices, such as automated factories, machinery systems and lots of other cases in IoT. Now EdgeX Foundry is in the rapid update phase, more features will be added in future release. An EdgeX UI is in development as a web-based interface to add and manage the device.

## 3.2 Azure IoT Edge

Azure IoT Edge, provided by Microsoft Azure as a cloud service provider, tries to move cloud analytics to edge devices. These edge devices can be routers, gateways or other devices which can provide compute resources. The programming model of Azure IoT Edge is the same as that of other Azure IoT services [17] in the cloud, which enables user to move their existing application from Azure to the edge devices for lower latency. The convenience simplifies the development of edge applications. In addition, Azure services like Azure Functions, Azure Machine Learning and Azure Stream Analytics can be used to deploy complex tasks on the edge devices such as machine learning, image recognition and other tasks about artificial intelligence.

Azure IoT Edge consists of three components: IoT Edge modules, IoT Edge runtime and a cloud-based interface as depicted in Fig. 2. The first two components run on edge devices, the last one is an interface in the cloud. IoT Edge modules are containerized instances running the customer code or Azure services. IoT Edge runtime manages these modules. The cloud-based interface is used to monitor and manage the former two components, in other words, monitor and manage the edge devices.

IoT Edge modules are the places that run specific applications as the units of execution. A module image is a docker image containing the user code. A module instance, as a docker container, is a unit of computation that running the module image. If the resources of edge devices supports, these modules can run the same Azure services or custom application as in the cloud because of the same programing model. In addition, these modules can be deployed dynamically as Azure IoT Edge is scalable.

IoT Edge runtime acts as a manager on the edge devices. It consists of two modules, IoT Edge hub and IoT Edge agent. IoT Edge hub acts as a local proxy for IoT Hub which is a managed service as a central message hub in the cloud, and is responsible for communication. As a message broker, IoT Edge hub helps modules communicate with each other, and transport data to IoT Hub. IoT Edge agent is used to deploy and monitor the IoT Edge modules. It receives the deployment information

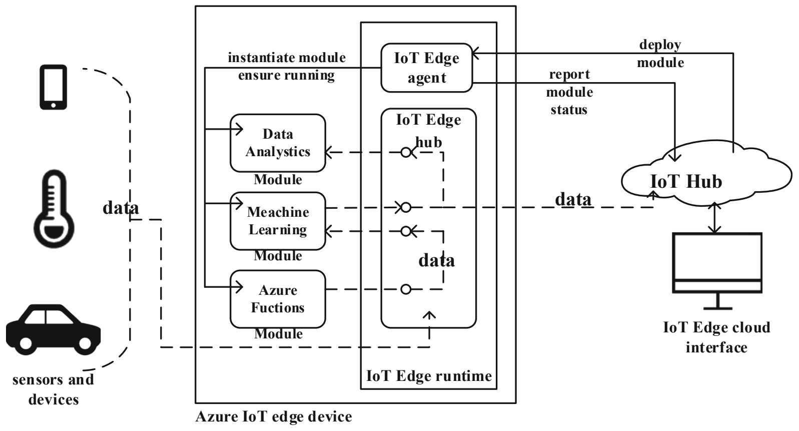

**Fig. 2.** Diagram of Azure IoT Edge

about modules from IoT Hub, instantiates these modules, and ensures they are running, for example, restarts the crashed modules. In addition, it reports the status of the modules to the IoT hub.

IoT Edge cloud interface is provided for device management. By this interface, users can create edge applications, then send these applications to the device and monitor the running status of the device. This monitoring function is useful for use cases with massive devices, users can deploy applications to devices on a large scale and monitor these devices.

A simple deployment procedure for applications is that, users choose a Azure service or write their own code as an application, then build it as an IoT Edge module image, and deploy this module image to the edge device with the help of the IoT Edge interface. Then the IoT Edge runtime receives the deployment information, pulls the module image and instantiates the module instance.

Azure IoT Edge has wide application area, now it has application cases on intelligent manufacturing, irrigation system, drone management system and so on. It's worth noting that Azure IoT Edge is open-source but the Azure services charge for fee.

## 3.3 CORD

CORD is an open source project of ONF initiated by AT&T and is designed for network operators. Current network infrastructure is built with closed proprietary integrated systems provided by network equipment providers. Due to the closed property, the network capability cannot scale up and down dynamically. And the lack of flexibility results in inefficient utilization of the compute and networking resources. CORD plans to reconstruct the edge network infrastructure to build datacenters with SDN [18], NFV [19] and Cloud technologies. It attempts to slice the compute, storage and network resources so that these datacenters can act as clouds at the edge, providing agile services for end-user customers.

CORD is an integrated system built from commodity hardware and open source software. Figure 3 [16] shows the hardware architecture of CORD. It uses commodity servers that are interconnected by a Fabric of White-box switches. These commodity servers provide with compute, storage resources and the fabric of switches is used to build the network. This switching fabric is organized to a Spine-Leaf topology rather than traditional three-tier network topology, because it can provide scalable throughput for greater East-to-West network traffic. In addition, specialized access hardware is required to connect subscribers. The subscribers can divided into three categories for different use cases, mobile subscribers, enterprise subscribers and residential subscribers. Each category demands different access hardware due to different access technology. In terms of software, Fig. 4 [16] shows the software architecture of CORD. Based on the servers and the fabric of switches, OpenStack provides with IaaS capability for CORD, it manages the compute, storage and networking resources as well as creating virtual machines and virtual networks. Docker is used to run services in containers for isolation. ONOS (Open Network Operating System) is a network operating system which is used to manage network components like the switching fabric and provide communication services to end-users. XOS provides a control plane to assemble and compose services. Other software projects provide with component capabilities, for example, vRouter (Virtual Router) provides with virtual routing functionality.

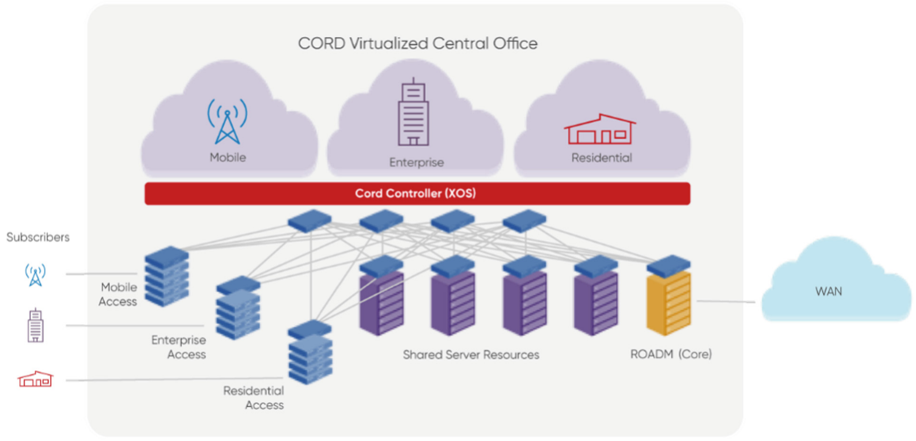

**Fig. 3.** Hardware architecture of CORD

The edge of the operator network is a sweet spot for edge computing because it connects customers with operators and is close to customers' applications as data sources. CORD takes edge computing into consideration and moves to support edge computing as a platform to provide edge cloud services from the 4.1 release version. CORD can be deployed into three solution, M-CORD (Mobile CORD), R-CORD (Residential CORD) and E-CORD (Enterprise CORD) for different use cases. M-CORD focus on mobile network, especially 5G network, it plans to disaggregate and virtualize cellular network functions to enable services be created and scaled dynamically.

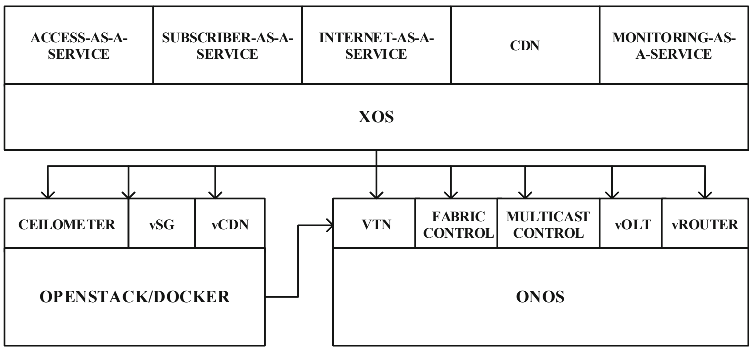

**Fig. 4.** Software architecture of CORD

This agility helps to provide multi-access edge services for mobile applications. For those use cases like driverless cars or drones, users can rent the edge service to run their edge applications. Similarly, R-CORD and E-CORD are designed to be agile service delivery platforms but for different users, residential and enterprise user relatively.

So far, deployment of CORD is still in test among network operators, and it needs more researches to combine CORD with edge applications.

## 3.4 Apache Edgent

Apache Edgent, which was known as Apache Quarks previously, is an Apache Incubator project at present. It is an open source programming model and lightweight runtime for data analytics, used in small devices such as routers and gateways at the edge. Apache Edgent focuses on data analytics at the edge, aiming to accelerate the development of data analysis.

As is a programming model, Edgent provides API to build edge applications. Figure 5 illustrates the model of the Edgent applications. Edgent uses a topology as a graph to represent the processing transformation of streams of data which are abstracted to a Tstream class. A connector is used to get streams of data from external entities such as sensors and devices in physical world, or to send streams of data to back-end systems like a cloud. The primary API of Edgent is responsible for data analysis. The streams of data can be filtered, split, transformed or processed by other operations in a topology. Edgent use a provider to act as a factory to create and execute topologies. To build an Edgent applications, user should firstly get a provider, then create a topology and add the processing flow to deal with the streams of data, and finally submit the topology. The deployment environments of Edgent are Java 8, Java 7 and Android.

Edgent provides API for sending data to back-end systems and now supports MQTT, IBM Watson IoT Platform, Apache Kafka and custom message hubs. Edgent applications analyze the data from sensors and devices, and send the essential data to the back-end system for further analysis. For IoT use cases, Edgent helps to reduce the cost of transmitting data and provide local feedback.

Edgent is suitable for use cases in IoT such as intelligent transportation, automated factories and so on. In addition, the data in Edgent applications are not limited to sensor

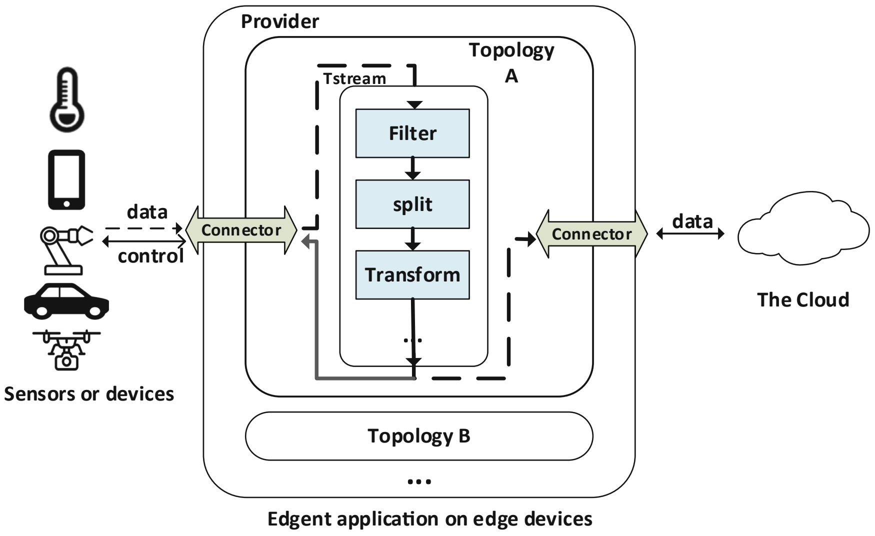

**Fig. 5.** Model of the Edgent applications

readings, they can also be files or logs. Therefore, Edgent can be applied to other use cases. For example, it can perform local data analysis when embedded in application servers, where it can analyze error logs without impacting network traffic [6].

## 3.5 Akraino Edge Stack

Akraino Edge Stack, initiated by AT&T and now hosted by Linux Foundation, is a project to develop a holistic solution for edge infrastructure so as to support high-availability edge cloud service [15]. An open source software stack, as the software part of this solution, is developed for carrier to facilitate optimal networking and workload orchestration for underlying infrastructure in order to meet the need of edge computing such as low latency, high performance, higher availability, scalability and so on.

To provide a holistic solution, Akraino Edge Stack has a wide scope from infrastructure layer to application layer. Figure 6 [15] shows the scope with three layers. In the application layer, Akraino Edge Stack wants to create an app/VNF ecosystem and calls for edge applications. The second layer consists of middleware which supports applications in the top layer. In this layer, Akraino Edge Stack plans to develop Edge API and framework for interoperability with 3rd party Edge projects such as EdgeX Foundry. At the bottom layer, Akraino Edge Stack intends to develop an open source software stack for the edge infrastructure in collaboration with upstream communities. It interfaces with and maximize the use of existing open source projects such as Kubernetes, OpenStack and so on. Akraino Edge Stack provides different edge use cases with blueprints, which are declarative configurations of entire stack including hardware, software, point of delivery, etc. [15] The application domains of these blueprints start from Telco industry, will expend to more domains like Enterprise and industrial IoT. Now Akraino Edge Stack has put forward several blueprints such as Micro-MEC and Edge Media

Processing. Micro-MEC intends to develop a new service infrastructure for smart cities, which enables developing services for smart city and has high data capacity for citizens. Edge Media Processing intends to develop a network cloud to enable real-time media processing and edge media AI analytics with low latency.

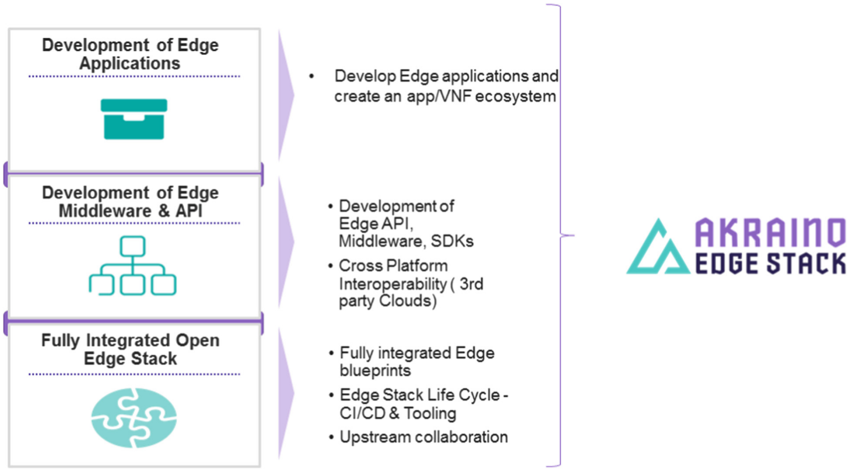
Development of Edge Applications
Development of Edge Middleware & API
Fully Integrated Open Edge Stack
• Develop Edge applications and create an app/VNF ecosystem
• Development of Edge API, Middleware, SDKs
• Cross Platform Interoperability ( 3rd party Clouds)
• Fully integrated Edge blueprints
• Edge Stack Life Cycle - CI/CD & Tooling
• Upstream collaboration
AKRAINO EDGE STACK

**Fig. 6.** Akraino Edge Stack's scope

As an emerging project, Akraino Edge Stack moved to execution since August 2018, thus more researches need to be done with the development of this project.

## 4 Comparative Study

In this section, we summarize the features of the systems discussed above in Table 1. Then compare these open source edge computing systems under different aspects as shown in Table 2. These aspects includes: the main purpose of the systems, the application area, target user, the virtualization technology, system characteristic and limitation. We believe they are of importance on giving a better understanding of these systems. Finally, we discuss some use scenarios to help to choose a suitable system for researchers.

### 4.1 Main Purpose

The main purpose shows the target problem that a system tries to fix. It is a key factor for us to choose a suitable system to run edge applications. Therefore, we choose main purpose as the first aspect we use to compare.

As an interoperability framework, EdgeX Foundry aims to communicate with any sensor or device in IoT. And this ability is necessary for edge applications with data from various sensors and devices. Azure IoT Edge offers an efficient solution to move

**Table 1.** Features of open edge systems

| Feature | EdgeX Foundry | Azure IoT Edge | Apache Edgent | CORD | Akraino Edge Stack |
|---|---|---|---|---|---|
| Scalability | Scalable | Scalable | Not scalable | Scalable | Scalable |
| User access interface | Restful API or EdgeX UI | Web service, Command-line | API | API or XOS-GUI | N/A |
| Deployment | Dynamic | Dynamic | Static | Dynamic | Dynamic |
| OS support | Various OS | Various OS | Various OS | Ubuntu | Linux |
| Programming Framework | Not provides | Java, .NET, C, Python, etc. | Java | Shell script, python | N/A |

**Table 2.** Comparison of open edge system characteristics

| Aspect | EdgeX Foundry | Azure IoT Edge | Apache Edgent | CORD | Akraino Edge Stack |
|---|---|---|---|---|---|
| Main purpose | Provide with Interoperability for IoT edge | Support hybrid cloud-edge analytics | Accelerate the development process of data analysis | Transform edge of the operator network into agile service delivery platforms | Support edge clouds with an open source software stack |
| Application area | IoT | Unrestricted | IoT | Unrestricted | Unrestricted |
| Target user | General users | General users | General users | Network operators | Network operators |
| Virtualization technology | Container | Container | JVM | Virtual Machine and Container | Virtual Machine and Container |
| System characteristic | A common API for device management | Powerful Azure services | APIs for data analytics | Widespread edge clouds | Widespread edge clouds |
| Limitation | Lack of programable interface | Azure Services is chargeable | Limited to data analytics | Unable to be offline | Unable to be offline |

the existing applications from cloud to edge, and to develop edge applications in the same way with the cloud applications. Apache Edgent helps to accelerate the development process of data analysis in IoT use cases. CORD aims to reconstruct current edge network infrastructure to build datacenters so as to provide agile network services for end-user customers. From the view of edge computing, CORD provides with multi-access edge services. Akraino Edge Stack provides an open source software stack to support high-availability edge clouds.

## 4.2 Application Area

This subsection discusses about the suitable application area of these systems, considering that some systems are created to meet the need of specific application area.

EdgeX Foundry and Apache Edgent both focus on IoT edge, and EdgeX Foundry is good at communication with various sensors and devices, while Edgent is good at data analysis. They are suitable for intelligent manufacturing, intelligent transportation and smart city where various sensors and devices generate data all the time. Azure IoT Edge can be thought as the expansion of Azure Cloud. It have an extensive application area but depends on the compute resources of edge devices. Besides, it is very convenient to deploy edge applications about artificial intelligence such as machine learning and image recognition to Azure IoT Edge with the help of Azure services. CORD and Akraino Edge Stack support edge cloud services, which has no restriction on application area. If the edge devices of users don't have sufficient computing capability, these two systems are suitable for users to run resource-intensive and interactive applications in connection with operator network.

## 4.3 Target User

Though these open source systems focus on edge computing, but their target user are not the same. EdgeX Foundry, Azure IoT Edge and Apache Edgent have no restriction on target users. Therefore, every developer can deploy them into local edge devices like gateways, routers and hubs. Differently, CORD and Akraino Edge Stack are created for network operators because they focus on edge infrastructure.

## 4.4 Virtualization Technology

Virtualization technologies are widely used nowadays [20]. Virtual machine technology can provide better management and higher utilization of resources, stability, scalability and other advantages. Container technology can provide services with isolation and agility but with negligible overhead, which can be used in edge devices [21]. Using OpenStack and Docker as software components, CORD and Akraino Edge Stack use both of these two technologies to support edge cloud.

Different edge devices may have different hardware and software environment. For those edge systems which are deployed on edge devices, container is a good technology for services to keep independence in different environment. Therefore, EdgeX Foundry and Azure IoT Edge choose to run as docker containers. As for Edgent, Edgent applications run on JVM.

## 4.5 System Characteristic

System characteristics show the unique features of the system, which may help user to develop, deploy or monitor their edge applications. It will save lots of workload and time if making good use of these characteristics. EdgeX Foundry provides with a common API to manage the devices, and this brings great convenience to deploying and monitoring edge applications in large scale. Azure IoT Edge provides powerful

Azure services to accelerate the development of edge applications. Apache Edgent provides a series of functional APIs for data analytics, which lowers the difficulty and reduces the time on developing edge analytic applications. CORD and Akraino Edge Stack provide with multi-access edge services on edge cloud, we only need to keep connection with operator network, then we can apply for these services without the need to deploy an edge computing system on edge devices by ourselves.

### 4.6 Limitation

This subsection discusses the limitation of the latest version of them to deploy edge applications. The latest version of EdgeX Foundry has not provided a programmable interface in its architecture for developers to write their own application. Although EdgeX allows us to add custom implementations, but it demands more workload and time. As for Azure IoT Edge, though it is open-source and free, but Azure services are chargeable as commercial software. For Apache Edgent, it is lightweight and it focuses on only data analytics. As for CORD and Akraino Edge Stack, these two systems demand stable network between data sources and the operators because the edge applications are running on the edge of operator network rather than local devices.

### 4.7 Scenarios

Based on the differences of aspects discussed above, now we can discuss where and when each system should be chosen from the view of edge application.

In the first scenario, suppose we want to run edge applications on local area network, and use local enterprise system as back-end system with no need for third-party clouds. In this case, we can choose EdgeX Foundry or Apache Edgent. Further, suppose we want a good device management ability with various and devices added in this system, we should choose EdgeX Foundry because it provides APIs to manage and control the devices. If we focus on data analysis, a best approach would be to use Apache Edgent because it helps to accelerate the development of edge analytic applications.

Then suppose we want to build edge applications about artificial intelligence. In this case, Azure IoT Edge can reduce the difficulty of development by providing powerful Azure services like Azure Machine Learning, as well as commercial support.

At last, suppose we want to run mobile edge applications on drones or autonomous cars. In this case, we should choose edge cloud services with wireless access, so CORD or Akraino Edge Stack are the best choice.

## 5 Conclusions

In this paper, we presented five open source edge computing systems, EdgeX Foundry, Azure IoT Edge, CORD, Apache Edgent and Akraino Edge Stack. We also listed features of them, gave a comparative study about them and described user scenarios. We hope this paper can give the readers a better understanding of their characteristic and help readers to choose an appropriate systems according to their requirement.

**Acknowledgments.** This work is supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000302 and the National Natural Science Foundation of China under Grant Nos. 61832020.

## References

1. Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
2. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
3. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2009)
4. Zhang, Q., Zhang, Q., Shi, W., Zhong, H.: Firework: data processing and sharing for hybrid cloud-edge analytics. IEEE Trans. Parallel Syst. **29**(9), 2004–2017 (2018)
5. EdgeX Foundry Homepage. https://www.edgexfoundry.org. Accessed 16 Nov 2018
6. Apache Edgent Homepage. http://edgent.apache.org. Accessed 16 Nov 2018
7. Cui, J., Zhang, Y., Cai, Z., Liu, A., Li, Y.: Securing display path for security-sensitive applications on mobile devices. Comput. Mater. Contin. **55**(1), 17 (2018)
8. Open Edge Computing initiative Homepage. http://openedgecomputing.org. Accessed 16 Nov 2018
9. The Open Fog Consortium. https://www.openfogconsortium.org. Accessed 16 Nov 2018
10. Edge Computing Consortium. http://www.ecconsortium.org. Accessed 16 Nov 2018
11. Willis, D., Dasgupta, A., Banerjee, S.: ParaDrop: a multi-tenant platform to dynamically install third party services on wireless gateways. In: ACM Workshop on Mobility in the Evolving Internet Architecture. ACM (2014)
12. AWS Greengrass. https://aws.amazon.com/greengrass. Accessed 16 Nov 2018
13. Azure IoT Edge. https://azure.microsoft.com/zh-cn/services/iot-edge. Accessed 16 Nov 2018
14. Link IoT Edge. https://www.aliyun.com/product/iotedge?spm=5176.224200.loT.7.34666e-d6dQ8fjt. Accessed 16 Nov 2018
15. Akraino Edge Statck Homepage. https://www.akraino.org. Accessed 16 Nov 2018
16. CORD Homepage. https://www.opennetworking.org/cord. Accessed 16 Nov 2018
17. Azure IoT Homepage. https://azure.microsoft.com/en-us/overview/iot/. Accessed 16 Nov 2018
18. Nunes, B.A.A., Mendonca, M., Nguyen, X.N., Obraczka, K., Turletti, T.: A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutor. **16**(3), 1617–1634 (2014)
19. Hawilo, H., Shami, A., Mirahmadi, M., Asal, R.: NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC). IEEE Netw. **28**(6), 18–26 (2014)
20. Xie, X., Yuan, T., Zhou, X., Cheng, X.: Research on trust model in container-based cloud service. Comput. Mater. Contin. **56**(2), 273–283 (2018)
21. Morabito, R.: Virtualization on Internet of Things edge devices with container technologies: a performance evaluation. IEEE Access **5**, 99 (2017)