

The Application of Kata Containers in Baidu AI Cloud

White Paper

Author Zhang Yu October 2019

Contents

1.0		Introduction	4
2.0		Background - Baidu AI Cloud and Container Services	5
	2.1	Baidu AI Cloud	5
	2.2	Baidu AI Cloud Container Services	5
		2.2.1 Cloud Container Engine (CCE)	6
		2.2.2 Cloud Function Computing (CFC)	
		2.2.3 Baidu Container Instance (BCI)	
		2.2.4 Baidu Edge Computing (BEC)	6
3.0		Container Service: Challenges and Solutions	8
	3.1	Kata Containers	9
	3.2	Function Computing	
	3.3	Cloud Container Instances	11
	3.4	Edge Computing	12
4.0		Applications of the Kata Containers Schema	14
	4.1	Practice of Function Computing Based on Kata Containers	14
		4.1.1 Function Performance at Cold Start and Warm Start	14
		4.1.2 Accelerating the Start-up at the Scheduling Layer	
		4.1.3 Support Dynamic Mounting of User Code	
	4.2	The Integration of Kata Containers and OpenStack Architectures	
		4.2.1 Computing - Overall Architecture Built on NOVA	
		4.2.2 Storage - Optimization and Acceleration	
	4.3	4.2.3 Network - Customized Plug-ins Support OVS	
	4.3	Practice of Kata Containers in Edge Computing 4.3.1 Edge Computing using BEC	
		4.5.1 Edge Computing using BEC	2 1
5.0		Achievements in Practice	
	5.1	Achievements after the Application of Kata in Function Computing	
		5.1.1 DuerOS Skill Development	
		5.1.2 Development of Smart Applications	
	БЭ	5.1.3 NPM Package Source Station Hosting	
	5.2	Achievements of BCI in Big Data Applications	
	5.3 5.4	Execution of Scenarios on BEC	
	5.4	DuEdge Network Computing Services	
6.0		Outlook	
	6.1	Ongoing Evolution of Lightweight Virtualization	
	6.2	AI Acceleration Hardware	
	6.3	Trustworthy Computing Environment	31
7.0		Acknowledgements	32

33

Figures

Figure 1.	Traditional Containers	8
Figure 2.	Kata Containers	9
Figure 3.	Implementation of CaaS based on Kata Containers	12
Figure 4.	Cold Start of Function	14
Figure 5.	Pre-warm Start for Kata	16
Figure 6.	Integration of Kata Containers and OpenStack Architectures	17
Figure 7.	EmptyDir Implementation from the Kata Community using virtio-9pfs	18
Figure 8.	EmptyDir Implementation Based on virtio-blk	19
Figure 9.	Docking of Kata Containers and Neutron Network	20
Figure 10.	BEC Dual Network Interface Card Architecture	
Figure 11.	Multus Multiple Network Interface Plug-in	
Figure 12.	DuerOS Development Skills	
Figure 13.	Smart Application Development	
Figure 14.	NPM Source Station Hosting	
Figure 15.	DuEdge Function Computing	
Figure 16.	Evolution of Virtualization Technologies	30
Figure 17.	Set of Virtualization Rust Crates for Easy VMM Implementation	

Tables

Table 1.	Comparison of Start-up Time of Containers in Different Modes	15
Table 2.	References	33

1.0 Introduction

This white paper describes Baidu's journey to offer **AI Cloud** and **Edge Computing** services at massive scale by taking advantage of innovative cloud technologies such as **Kubernetes, Kata Containers and OpenStack**. It begins with background on **Baidu AI Cloud and Container** offerings and how they take advantage of the greater isolation offered by Kata Containers. It then describes in detail how Baidu integrates Kata Containers within its customers' **OpenStack** infrastructures in a performant way. An **OpenStack Neutron** networking use case is also shared. Next, it explains how Baidu Edge Computing takes advantage of the **Multus** container network plugin to create multiple container network interfaces (CNI) for Kubernetes pods. Finally, it touches upon new **hypervisor innovations** as well as hardware acceleration for AI inference, before concluding with acknowledgements.

Background - Baidu AI Cloud and Container 2.0 **Services**

2.1 **Baidu AI Cloud**

Baidu AI Cloud [Ref. 1 in Table 2], officially launched in 2015, is Baidu's intelligent cloud-computing platform for enterprises and developers that undertakes the mission of manifesting Baidu's technological capabilities. Rooted in technology innovation, Baidu AI Cloud is dedicated to providing all-in-one ABC (Artificial Intelligence, Big Data, Cloud Computing) services across industries.

Baidu AI Cloud has thus far released more than 260 products and nearly 40 solutions, not only in the areas of computing, storage, network, database and security, but also comprehensive solutions for big data and artificial intelligence. Its cloud services consist of public, private and hybrid clouds.

In the field of artificial intelligence, Baidu AI Cloud inherits Baidu's leading AI technological advances that enable more than 210 capabilities including voice and image technologies, facial recognition, human body recognition, and natural language processing. These capabilities have already been successfully deployed in autonomous driving and smart home applications. In addition, Baidu has independently developed the Kunlun chip, China's first-ever cloud-to-edge AI chip; and Xiaodu, which is the first conversational AI operating system, already with an installed base of 100 million users in China. All these components of the Baidu product and service portfolio give Baidu AI Cloud a competitive edge to excel even further.

Equipped with the powerful resources of this portfolio, Baidu AI Cloud has enjoyed rapid development. Owing to its revenue growth and significant increase in market share, it successfully entered the "Strong Performers" quadrant in The Forrester Wave™: Full-Stack Public Cloud Development Platforms In China, Q3 2018 [Ref. 2 in Table 2], which is a report released by the renowned Forrester consulting firm. According to the Report on Asia-Pacific Public Cloud Market 2019 [Ref. 3 in Table 2] published by the influential research organization Synergy Research Group in May 2019, Baidu AI Cloud ranked among the top four and has firmly established itself as a leader in the Chinese market.

2.2 **Baidu AI Cloud Container Services**

Baidu is one of the pioneers in container technologies. Since the launch of Kubernetes v0.8, Baidu has already achieved production-level service and practice through secondary optimizations.

In June 2017, Baidu AI Cloud unveiled Cloud Container Engine (CCE), marking the beginning of the transformation of the company's rich experience in container technologies to commercial products and services success. In December of the same year, Baidu joined the Cloud Native Computing Foundation (CNCF) as a Gold Member. In March 2018, the CCE became one of the first offerings to be certified under the Certified Kubernetes Conformance Program. In April 2019, Baidu AI Cloud participated in the Kubernetes Certified Service Provider (KCSP) program.

Currently, Baidu AI Cloud provides the following Container services:

2.2.1 Cloud Container Engine (CCE)

CCE [Ref. 4 in <u>Table 2</u>] provides Docker container lifecycle management, operational management for large-scale container clusters, and one-click publish and run for applications. It provides highly flexible, highly available, highly efficient and convenient platform services through seamless connections with other products of Baidu AI Cloud. It is suitable for scenarios such as microservices architecture, efficient DevOps operation and management, and AI applications for container-based deep learning.

CCE features flexible container cluster management, integrating capability for underlying network/storage/secure infrastructure and high-performance elastic scaling.

2.2.2 Cloud Function Computing (CFC)

Cloud Function Computing (CFC) [Ref. 5 in <u>Table 2</u>], one of the first CCE-based scenarios, provides event-driven, highly flexible, available, scalable and responsive serverless cloud computing. It enables users to solely focus on writing code without the need to manage servers. In addition, it supports multiple function triggers. The CFC is applicable to diverse event-triggering scenarios.

CFC features a cost effective, quick-to-start, "zero" operations management solution with automated scaling.

2.2.3 Baidu Container Instance (BCI)

Baidu Container Instance (BCI) [Ref. 6 in <u>Table 2</u>] provides the Serverless container service. To create and run containers, users only need to provide container images and basic parameters required for container operation, without the need to manage the servers and clusters. As an added benefit, users only pay for resources actually consumed by the containers.

2.2.4 Baidu Edge Computing (BEC)

Based on a vast network of CDN nodes, Baidu Edge Computing (BEC) provides one-stop elastic computing resources in close proximity to end users. It greatly reduces response latency and lowers central bandwidth costs, thanks to computing and processing on

the edge. BEC container services include BCI and DuEdge [Ref. 7 in <u>Table 2</u>]. DuEdge, which is Baidu's Network Edge Computing solution, provides the following four core features: customizable programming, content distribution, security defense, and artificial intelligence. With the help of edge network computing, BEC aims to solve data transmission and network traffic problems between the cloud and end users with the goal of improving business flexibility and operational efficiency.

3.0 Container Service: Challenges and Solutions

Baidu AI Cloud has developed powerful container technologies centered on Kubernetes after overcoming and learning from technological challenges in its own complex networks. These challenges involved huge amounts of traffic and complicated deployment scenarios exemplified by peak traffic of a single cluster of 1 billion+ page views (PVs) per day, and 50,000+ containers for a single tenant.

While resource sharing brings flexibility in business and higher efficiency in resource utilization, container services also face higher security risks. These risks stem from the fact that containers on a shared host share the host kernel, as shown in Figure 1. Since different tenants can run their container workloads on the same shared host, it is difficult to prevent malicious code in one container workload from attacking a neighboring container workload, thus posing security threats to the entire cloud infrastructure and to the data and business of cloud customers. Therefore, it was imperative for Baidu to figure out how to improve container isolation to protect customer workloads and data while leveraging the lightweight nature and agility of containers.

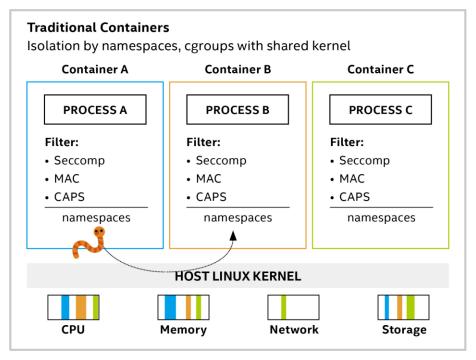


Figure 1. Traditional Containers

3.1 Kata Containers

Kata Containers [Ref. 8 in <u>Table 2</u>] is an open source community project working to build a secure container runtime that employs fast, lightweight virtual machines to increase container isolation. Kata Containers act and perform like classic containers but provide stronger workload isolation using hardware virtualization technology as a second layer of defense.

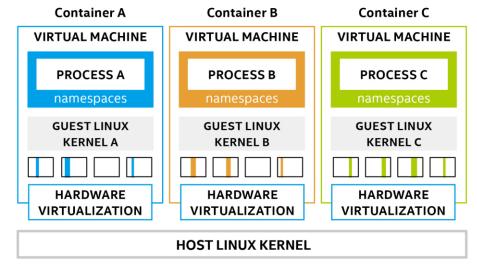
Back in 2015, engineers from the Intel Open Source Technology Center began looking to enhance security isolation in the container ecosystem using Intel® Virtualization Technology (Intel® VT), work that resulted in the launch of the Intel® Clear Containers project [Ref. 9 in <u>Table 2</u>]. At the same time, engineers at Hyper.sh, a Chinese high tech startup, launched the open source project runV [Ref. 10 in <u>Table 2</u>] with the same goal as Clear Containers and using a similar strategy of placing containers in a secure "sandbox." In late 2017, in order to accelerate adoption of and industry participation in this secure container technology, the Intel and Hyper.sh teams decided to merge the two projects into a single, new project called Kata Containers.

As shown in Figure 2, Kata Containers bridges the gap between the hardware isolation of traditional virtual machines (VMs) and the speed and relatively smaller footprint of containers. For Kata Containers, each container or container pod is launched into a lightweight VM with its own unique kernel instance. Since each container/pod is now running in its own VM, malicious code can no longer exploit the shared kernel to access neighboring containers. Kata Containers also makes it possible for container-as-a-service (CaaS) providers to more securely offer containers running on bare metal since each container/pod is isolated by a lightweight VM. Kata Containers allows mutually untrusting tenants – or even production and pre-production (unproven) apps – to safely run in the same cluster, thanks to this hardware isolation.

Figure 2. Kata Containers

Kata Containers

Each container or pod is more isolated in its own lightweight VM



Baidu has closely followed Clear Containers -- now Kata Containers-- since its launch and had many exchanges with the community to study and test the technology. Based on extensive research on secure container technologies, Baidu believes that Kata Containers represents a highly secure and practical container technology for the following reasons:

- Kata Containers is a highly secure container technology that uses lightweight virtual machines to isolate containers. Hardware virtualization technology and the isolation it provides is a well-proven methodology to share resources in a secure manner and has been in use for decades.
- Kata Containers has high performance and reliability, along with additional security, as compared to traditional containers. Intel® Virtualization Technology (Intel® VT) [Ref. 11 in <u>Table 2</u>] and Intel® Virtualization Technology for Directed I/O (Intel® VT-d) provide hardware support and acceleration for virtualization, and ensure continuous performance improvement to support advanced cloud computing scenarios.
- Because Kata Containers is an **Open Container Initiative (OCI) compatible runtime**, it works seamlessly with container orchestration solutions such as Kubernetes. The Open Container Initiative is a Linux Foundation project to design open standards for operating-system-level virtualization, most importantly Linux containers.
- Some business applications require specific kernel versions, different from that of the host OS. In this scenario, the **customizable kernel feature** of Kata Containers solves this issue.
- Kata Containers is highly mature. As the merger of two established secure container projects Intel[®] Clear Containers and Hyper runV both of which started in 2015, the functionality and development infrastructure is proven. Thanks to contributions from ARM, IBM, and NVIDIA, Kata Containers is also **multi-architecture**. There are frequent releases of new features and bug fixes and the code has performed well in Baidu internal testing.
- Kata Containers is widely supported in the industry and has a well-developed and active open source community. Ant Financial (an Alibaba affiliate), ARM, Huawei and Intel sit on the Kata Architecture Committee which guides the technical direction of the project. Many other influential companies participate and/or contribute to the community as well including Alibaba, AMD, Baidu, Canonical, Google IBM, NVIDIA, Red Hat, SUSE, Tencent, Vexxhost and ZTE. The project is hosted by the OpenStack Foundation and has transparent community governance and well maintained CI/CD infrastructure, as well as comprehensive project documentation.

In summary, the features and capabilities of Kata Containers support the design requirements of Baidu Cloud container products, which include Function Computing, Cloud Container Instances, and Edge Computing, all of which are discussed in the following sections.

3.2 Function Computing

Function computing with Serverless features provides developers with great convenience, flexibility, and cost performance while leaving all the operational work and security assurance to service providers. Container technology is an infrastructure that is quite suitable for function computing. Invariably, the containers are lightweight, quick to start and easy to be orchestrated, which makes it possible for the CFC (Cloud Function Computing) of Baidu AI Cloud to grant numerous developers access to shared computing resources via an elastic resource pool—when the developer function is triggered to start, the container will be started to carry out the function operation; when the function operation is completed, resources are released to meet the computing needs of other developers. This model not only responds quickly to the requests of end users, but also reduces the use cost through resource sharing.

The primary requirement to implement function computing is that the underlying infrastructure is invisible to the applications and decoupled from it, so the isolation of resources should also be invisible to the applications and the users. The isolation mode among virtual machines adopted by Kata Containers not only ensures a safe isolation of the container in a multi-tenant environment, but also helps to make the isolation of virtual machines invisible to applications and users.

3.3 Cloud Container Instances

Function computing has been widely used and experimented in many lightweight application scenarios, such as mini programs, intelligent devices, automatic data processing, etc. However, function computing is not a panacea to enterprises, because it demands that the user's workloads are split on the function level of granularity, and the communication between functions is done entirely through the interfaces, while at the same time, it requires a high level of statelessness of the businesses. For complex business systems, it is not easy to meet these requirements. Even if the user's business structure can truly go through functional transformation, the technology for large-scale function collaboration, orchestration and management is not yet mature, preventing it from being put into full production.

When faced with a containerized and Serverless situation, users typically need to split their businesses and choose a more appropriate architecture according to the business attributes, which sometimes leads to technical dilemmas. For that purpose, Baidu Al Cloud has been exploring a product that fully takes advantage of the orchestration power of containers, while also allowing users to enjoy the low cost and high elasticity provided by the Serverless solution. BCI, the Baidu Container Instance service, comes to the foreground as the most appropriate product. BCI provides users with containerized resources that can be started directly, without the need to purchase any server or cluster. Users can start one or more containers at any time when required by their business, which can be released promptly when the task is completed. Compared with virtual machines, BCI gives full play to the advantages of containerization. By using standard Docker images, it allows for start and resource allocation on a fine level of granularity, start or stop in just a few seconds, and multi-copy replication at any time.

For the developers of small applications, BCI offers a Serverless feature comparable to function computing—the application can be packaged into a Docker image to start or stop the container. Developers are charged only for the resources actually consumed, without any need to take into consideration the management and scheduling of the underlying resources.

Because Kata Containers uses a lightweight virtual machine as part of its container implementation, the same Kubernetes (K8S) cluster can be shared by multiple tenants, as shown in Figure 3. Traditional cloud container services require that users apply for a virtual machine cluster as an isolated tenant before they launch K8S clusters. In contrast, Kata Containers, due to its lightweight virtual machine isolation, enables cloud providers to provide users with a Serverless container cluster—a true CaaS—in which users merely need to focus on their own applications.

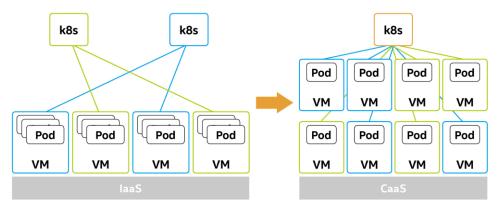


Figure 3. Implementation of CaaS based on Kata Containers

3.4 Edge Computing

Baidu edge computing products employ an edge node to deploy services for the cluster, and nodes are distributed in different geographical locations. Allowing tenants that do not trust each other to use resources within the same node requires high security and multi-tenancy isolation. Traditional container architectures involve shared kernels between host operating systems and guest containers. If one container fails, the workloads of the other containers in the cluster can be exposed to attacks. In addition, because the edge area lacks the complete network virtualization and security services that the central cloud enjoys, and the edge is directly exposed to the external network environment, it becomes a huge challenge for the products in terms of network security.

BEC users consuming edge computing resources prioritize performance over other demands. Because physical resources may be limited at the edge, it's critical to maximize product performance so as to avoid excessive costs or waste of resources. Because Kata Containers are within lightweight virtual machines, they're able to provide high performance while ensuring security without the need for compromise.

4.0 Applications of the Kata Containers Schema

Although Kata Containers is fully developed with great performance, there are still some key technical points that need to be resolved in the implementation of specific product forms and actual production deployment.

In this section, we mainly focus on the application practice of Baidu AI Cloud based on Kata Containers from the aspects of function computing, combined with OpenStack and edge computing.

4.1 Practice of Function Computing Based on Kata Containers

Because of the extra isolation provided by the Kata Containers virtual machine, the container launch time for the function computing scenario was too long. Baidu has developed clever solutions to improve the launch time, while still maintaining the extra isolation provided by Kata Containers. In order to solve this problem, Baidu was unrelenting in its development efforts, which resulted in significant achievements.

4.1.1 Function Performance at Cold Start and Warm Start

A cold start of a function, as shown in <u>Figure 4</u>, means that when the function is invoked for the first time, the container must be prepared, and the user function must be deployed before the language runtime is started.

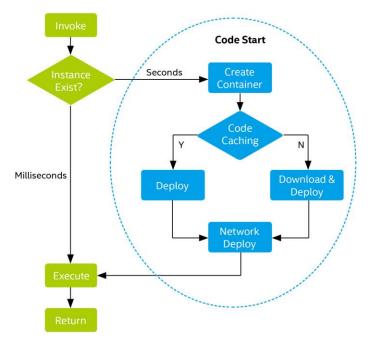


Figure 4. Cold Start of Function

Cold start performance can vary depending on the size of the user's code. The start-up time should be on the order of hundreds of milliseconds, which does not include the time for the Virtual Private Cloud (VPC) switch. Why is cold start performance so important? The reason lies in the fact that the access timeout of the upper layer of a business' function is set to a standard value and cannot be changed dynamically, whether the CFC function is triggered by a cold or warm start. If the timeout value for the function is very small, the longer time required by a cold start can easily lead to the function not being completed. Test results show that it took around 1 second to start Kata Containers, making it impossible to meet the requirements of function computing.

To solve this problem, Baidu starts containers in advance to guarantee a quick response. To further reduce start time, certain containers are turned into pre-warm containers, which can then become computing containers with only minor initialization operations required when the user function needs to be run. The following table compares the start-up times of containers in different modes. Note that lower values are better.

Comparison of Start-up Time of Containers in Different Modes Table 1.

Start Time of Containers (ms)	RunC	Kata
Runtime for starting the containers	526ms	1501ms
Runtime for starting the containers in advance + docker cp	441ms	463ms
Runtime for starting the containers in advance + dynamic mounting	155ms	98ms

4.1.2 Accelerating the Start-up at the Scheduling Layer

The scheduling layer of CFC must be integrated with Kubernetes. In addition, a scheme featuring perfect compatibility is also required so that the upper interface can be integrated with Kubernetes CRI.

Kata-runtime boots one lightweight Virtual Machine (VM) per pod for Kubernetes and multiple containers can run in a pod. To boost the start-up speed of the function containers, the Baidu team improved the pod creation process by starting a pod first and then inserting the containers. In this solution, a pod for Kata is deployed at runtime through Kubernetes before an additional business container is inserted in the pod through the CRI interface.

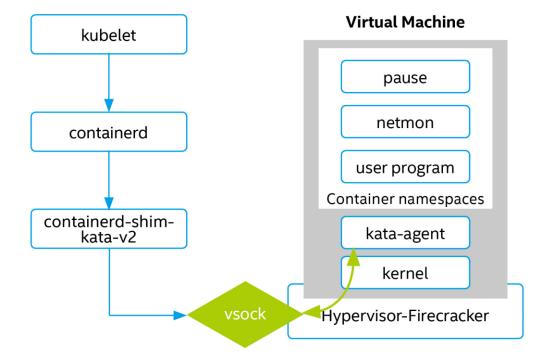


Figure 5. Pre-warm Start for Kata

<u>Figure 5</u> shows that at initialization, only the pause container and the netmon container are started inside the virtual machine. When user traffic appears, then the code directory is prepared and the container is created in which the user program resides.

4.1.3 Support Dynamic Mounting of User Code

A basic requirement of function computing is to mount user code.

A disadvantage to using a pre-warm container is that it may not contain any user code, instead the user code is loaded in a computing container, which must be mounted into the container when it is running. In the docker runc context, the host machine and the container share the same kernel and can be easily mapped using the -v parameter. Alternatively, the user code can be mounted across the namespace after the container has been started, by means such as nsenter. However, in the Kata Containers scenario where the host and guest machines do not share the same kernel, it becomes a huge challenge to mount the user code dynamically into the Kata Containers.

The team resorted to a method called Mount Bind to mount the user code dynamically into the Kata Containers. Please refer to the discussions on GitHub for details [Ref. 12 in Table 2].

4.2 The Integration of Kata Containers and OpenStack Architectures

The majority of Baidu's clients have already established their infrastructures, most of which are based on OpenStack [Ref. 13 in <u>Table 2</u>]. In order to align the BCI with such existing facilities, a set of customized new systems that can integrate seamlessly with existing client systems have been launched by BCI, as shown in <u>Figure 6</u>. In this way, the team made the best use of such existing systems and at the same time, improved it by deploying lightweight virtualization systems, resulting in a mix of both lightweight and heavyweight virtual machines. The solution not only saves resources, but also reduces development and maintenance costs.

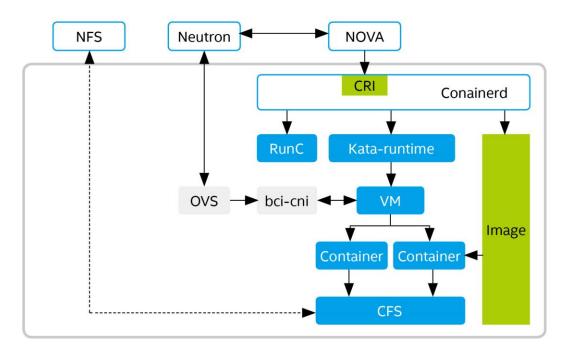


Figure 6. Integration of Kata Containers and OpenStack Architectures

The performance of the integrated technologies of BCI and OpenStack architectures is extraordinary in terms of computing, storage, and network.

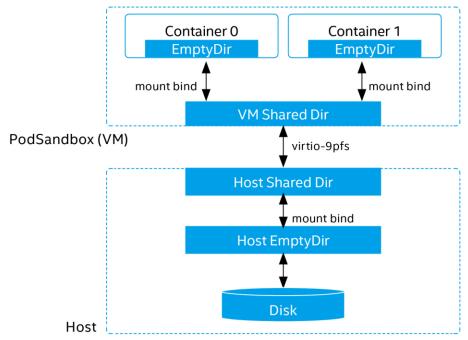
4.2.1 Computing - Overall Architecture Built on NOVA

For the original OpenStack virtual machines, a lightweight virtual machine (Safe Container) suitable for NOVA terminals was developed, which can be mixed with the existing virtual machines. The NOVA terminal unifies scheduling resources, by supporting containerd scheduling in the lower layer and Container Runtime Interface (CRI) in the upper layer to facilitate user scheduling through the customized Kubernetes nodes.

4.2.2 Storage - Optimization and Acceleration

EmptyDir is the non-persistent storage of BCI products. It provides the container group with an empty directory that is shared by all containers in the container group at the same time. The key to the EmptyDir design is how to implement this empty directory in Kata Containers. The community provides an implementation method that uses virtio-9pfs, a file system virtualization technology, and shares the empty directory on the host machine to the container group as the carrier of EmptyDir, as shown in Figure 7.

Figure 7. EmptyDir Implementation from the Kata Community using virtio-9pfs



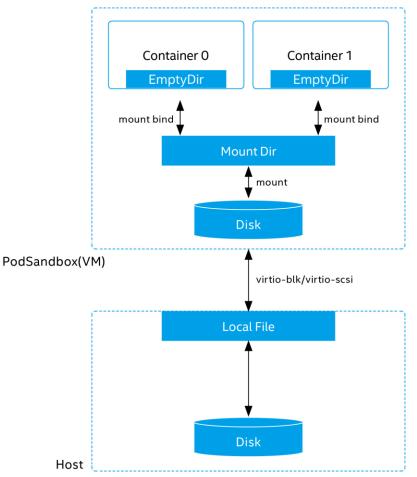
Nevertheless, this scheme has two obvious disadvantages:

- 1. For multi-tenancy scenarios such as the public cloud, this scheme requires the file system to provide quota administration. Without such administration, the local disk of the host machine could be easily filled up by any user's container group and prevent other users' container groups from working properly. However, the quota administration of the existing file system is not flexible enough to support features such as expansion, multi-user management, and others.
- 2. 9pfs is not a file system protocol designed specifically for virtualization, so the performance will be significantly reduced (by about 70%) compared to the host machine.
- **Note:** An alternative and more optimal shared file system protocol, virtio-fs, is in development by Red Hat and others which allows VMs to access a directory tree on the host. It is more performant and secure than 9pfs, however, at the time of writing this document, virtio-fs is still in development. An early version of virtio-fs is planned for an upcoming release of Kata Containers so users can start experimenting with it.

Due to these disadvantages, the team chose to use the virtualization technology for block devices, combined with the local storage approach of Baidu virtual machine products, to virtualize a file on the host machine into a block device of the virtual machine. The file size on the host is controllable and so is the quota. In terms of performance, because virtio-blk/virtio-scsi has been proven to be a mature virtualization solution, the virtual machine does not suffer much loss in performance, compared with the host machine. Based on these two advantages, this scheme can be applied to the emptydir implementation of BCI.

The emptydir implementation of BCI is shown in <u>Figure 8</u>. Test results show that the performance of this scheme is close to 80% of the host machine (single queue, single thread).

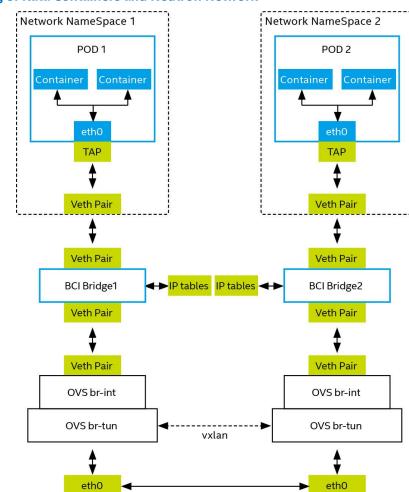
Figure 8. EmptyDir Implementation Based on virtio-blk



4.2.3 Network - Customized Plug-ins Support OVS

The network is a rather complicated part of lightweight virtualization, which requires not only high performance, but also a high-level of maintainability. Baidu has

developed a solution which reuses the OpenStack Neutron components and a network scheme incorporating Neutron with Open vSwitch, making the isolation and speed limit on the network possible. Due to the powerful functionality and performance of Neutron + Open vSwitch and by reusing the former network architecture, the network requirements can be met at a lower cost. The overall architecture is shown in Figure 9.





Its main features are as follows:

- Utilizing the original ovs + iptables rules, the isolation between the virtual machine and the physical machine is realized.
- The traffic flowing through OVS is basically the same as the original virtual machine. There is no need to change the topology structure because multi-host network communication can be realized with the overlay network provided by OVS.
- The IP address of the container is injected by Neutron, and through the interaction between containerd and the bci-cni plug-ins of the Baidu scheme, it becomes

possible to specify IP addresses. The eip module is supported by the original Neutron.

A new bci-cni plug-in was developed for the scheme, which can be adapted to the CNI part of the containerd, and can create the specified network architecture before OVS is accessed. To facilitate communication between neutron and containerd, when a pod is created, a specific annotation is passed in through nova to carry relevant information such as the IP address of the bci-cni plug-in.

(Additional details about networking and Kata Containers may be found at: <u>https://github.com/kata-</u> <u>containers/documentation/blob/master/design/architecture.md#networking</u>.)

4.3 Practice of Kata Containers in Edge Computing

4.3.1 Edge Computing using BEC

Unlike the central cloud, BEC is deployed and run on edge nodes, such as on CDN nodes. Network security can be a huge challenge, because the edge lacks the complete network virtualization and security services that the central cloud enjoys, and the edge service is directly exposed to the external network environment. However, network security is the most important part of BEC, because it relates to the security of the computing region and the public network and it is closely related to multi-tenancy isolation in the computing region. In addition, certain clients may need to create a dual-network device in the container to control the traffic in a self-dependent and more refined manner, which makes the network architecture more complicated.

BEC also echoes some of the performance requirements of the BCI. For example, the optimization of the storage performance and the possibility of mounting GPUs have been similarly mentioned in the aforesaid BCI practice, while the network security will be the focal point in the practice session of the BEC technology.

In the Kata scenario, there is another virtualization layer in the container's Network Namespace, which requires the use of a Linux tap device because a veth Network card cannot be directly used. The Edge Computing of Baidu Al Cloud, however, adopts the Macvtap mode: it can create a parent device as the Macvtap of veth for VM to use.

1. When it comes to supporting dual network cards, BEC has launched a customized CNI implementation scheme that makes it possible to manage both internal and external network cards through two types of plug-ins. In the internal network scheme, the Calico tool is adopted to implement a two-layer network through BGP protocol, while macvlan is adopted in the external network scheme to minimize losses caused by excessive network devices, thus making the network more efficient. The overall architecture is shown in Figure 10.

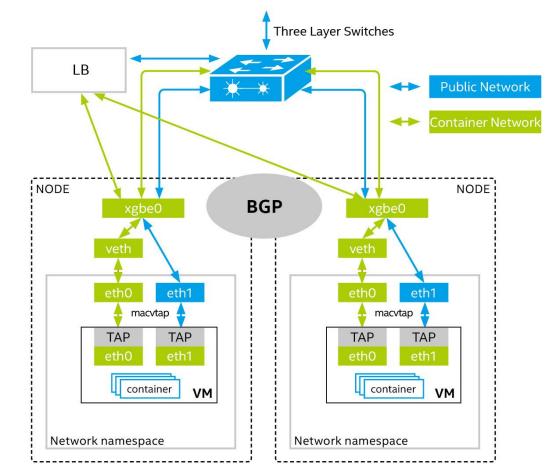


Figure 10. BEC Dual Network Interface Card Architecture

The pod of the native Kubernetes only supports one network interface. Multus [Ref. 14 in <u>Table 2</u>] is a container network plug-in developed by Intel, which can be used to create multiple network interfaces for the pods in Kubernetes as shown in <u>Figure 11</u>. During its development, the Multus project was assigned to the Kubernetes network working group [Ref. 15 in <u>Table 2</u>] and laid the foundation for the establishment of the group. It also became the open-source version of the standard multi-network interfaces specification.

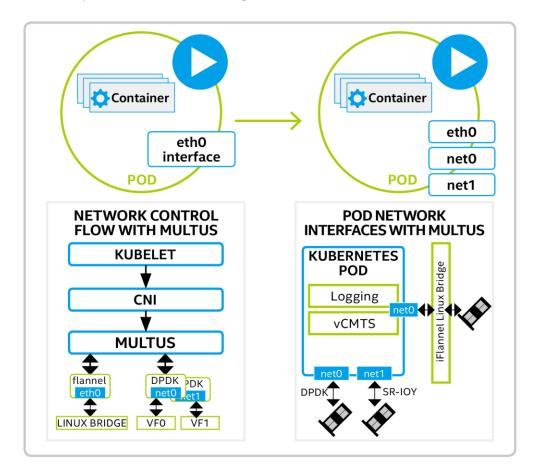


Figure 11. Multus Multiple Network Interface Plug-in

As the agent of other CNI plug-ins, Multus deploys additional plug-ins to create the CNI network interface. To configure Multus, another plug-in needs to be assigned as the main one to configure and manage the main network interface.

The bottom right corner in Figure 11 demonstrates a container virtual firewall with the logging function deployed in a pod with triple network interfaces, among which eth0 enables the said pod to communicate with the other pods in the K8S cluster as the management interface. On top of that, it also serves as the main network interface in K8S.

Multus can use the interfaces and software stack such as SR-IOV, DPDK and the like as well. In Figure 11, the pod is also equipped with two SR-IOV VF interfaces, which are shown as net0 and net1. Those interfaces are created to expedite the network data plane. For instance, when a virtual firewall requires dual networks, that enables the firewall rules to be isolated from each other.

A dual network interface card not only presents challenges to the network architecture, but it also greatly complicates the control plane. The BEC of the control plane is made possible via a customized CNI, which supports Calico and macvlan. In addition, the routing table rules are automatically injected into the Kata Containers to ensure the separation of the intranet and outer net traffic as well as the accuracy of the return to the source station.

2. In terms of network isolation, adopting the complete VPC virtualization plan on the edge is apparently not viable. Therefore, Baidu used the concept of native cloud and implemented the network policy mechanism in Kubernetes. By mapping out the tailored isolation strategy, the tenants are isolated. By default, the isolation is effective inside the container intranet and not so in the outer net. Besides, the container of the intranet utilizes Calico, and the native version supports the network policy.

3. Although the container outer net does not necessitate isolation, it does require speed limit. BEC brought in a speed-limit module in CNI, which was realized through the TC tool. During the process, in order to address conflict between TC and Kata's default tcFilter network model when creating PodSandbox, BEC altered the Kata network model as macvtap, hence ensuring the execution of functions.

5.0 Achievements in Practice

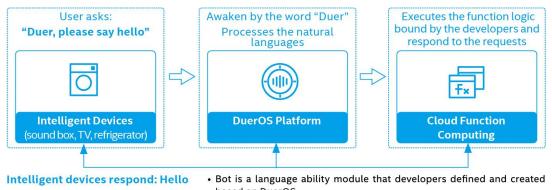
5.1 Achievements after the Application of Kata in Function Computing

Baidu Cloud Function Computing (CFC) has been widely applied in a multitude of areas. For example, based on function computing, Xiaodu provides a platform (DuerOS) for the skill developers of the Xiaodu intelligent hardware; the mobile Baidu app provides a one-stop cloud-development platform for smart application developers; various online content media outlets carry out the pre-screening and categorization of mass streaming data; and game publishers conduct the automated packaging and distribution of the game setup programs.

5.1.1 DuerOS Skill Development

In the DuerOS skill development scenarios, function computing was provided for more than **3,000** developers as well as nearly **20,000** skills, and responded to more than a million user requests from the intelligent terminals every day. Thanks to reduced costs made possible by resource sharing, skill developers accessed **a million function calls** and **400,000 GB second function** operations per month, substantially lowering the threshold for use and enabling developers to focus on the development of business logic.

Figure 12. DuerOS Development Skills



based on DuerOSFunction is an event and response mechanism defined and created by

 Function is an event and response mechanism defined and created by developers based on CFC

5.1.2 Development of Smart Applications

CFC also backs the cloud development services of Baidu AI Cloud and supports users in connecting the development side and the Baidu AI Cloud resource side with userdefined functions. It thus allows them to access the computing, storage and network resources of CFC more flexibly, elevating development efficiency and **lowering costs** for more than 50% of the smart application developers.

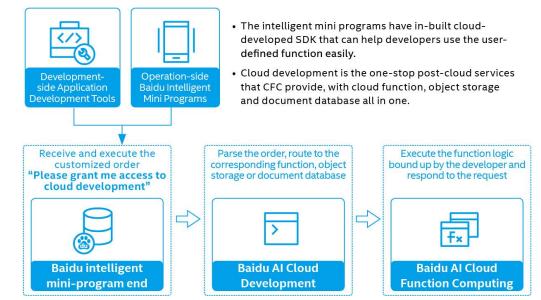


Figure 13. Smart Application Development

5.1.3 NPM Package Source Station Hosting

CFC is responsible for the Node package manager (NPM) package source station hosting of ShineScrum, meaning it automatically checks the version changes of the NPM package through the function activated regularly. When there is any upgrade available, the downstream function is automatically triggered, allowing for batched updates and downloading of the NPM packages. Currently, **CFC has already supported the check on updates for more than 800,000 packages** under various categories inside and outside Baidu, and has been reliably serving the company's DevOps production line automation integration process.

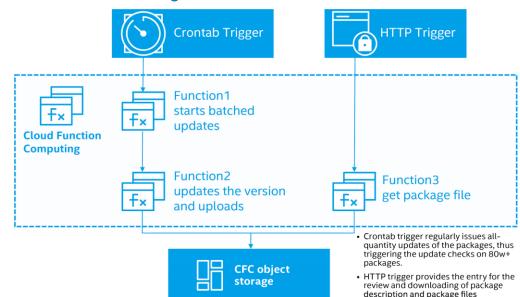


Figure 14. NPM Source Station Hosting

5.2 Achievements of BCI in Big Data Applications

The CFC container instance BCI provides powerful basic architecture for the big data business inside Baidu and creates a Serverless data processing platform accessible to multiple tenants for the big data department. For the R&D personnel in a big data group, BCI fulfills the needs of the developers by using mature Spark community technologies via its integration with Kubernetes, without additional cost or attention to the management as well as operation and maintenance of resources. For developers, the big data processing platform based on BCI provides a more convenient model for the use of Spark. The developers only need to package the data, handle the tasks and submit them to the platform before they wait for the tasks to automatically run and generate results. They do not need to monitor scheduling of resources nor maintain any basic architecture of Spark.

In such a multi-tenant big data platform scenario, the provider of Spark services wishes to maximize the use of resources and provide sufficient capabilities for mass computing. However, the traditional Kubernetes cluster model does not have the flexibility to satisfy the need. Therefore, a container instance that requires no servers and is able to start within seconds becomes the optimum option. By deploying BCI, the task arrangement services can be carried out in the Kubernetes cluster of the platform, while the tasks submitted by the users leverage the BCI resources in the accounts for operation through Virtual-Kubelet. On the one hand, the way the platform uses it does not differ from that of traditional Kubernetes; on the other, the platform is not required to prepare resources for the user tasks beforehand, which improves resource utilization. In contrast, Kata Containers, based on underlying technologies, ensures that the user tasks run in the isolated system kernel and cyberspace, therefore ensuring higher security for task execution and data.

At present, the Baidu stream-oriented computation service BSC based on Spark has integrated the Serverless container capability of BCI. It can reduce the cost for forging the Kubernetes cluster by 40%~60%, and bring in a great number of valuable platform creation practices. In addition, the Baidu big data team has been working in tandem with its BCI counterparts to promote the Serverless trend of more big data technology platforms (such as MapReduce, machine learning, etc.) and to provide additional comprehensive and convenient big data services for developers.

5.3 Execution of Scenarios on BEC

As a standardized cloud product accessible to the public, Baidu Edge Computing (BEC) provides open services to all clients, isolates their operations, and frees them from mutual interference, utilizing the features of Kata Containers. Thus far, BEC has taken on a multitude of businesses that have demanding requirements for the specifications and performance of resources, with the specific application scenarios as follows:

• Interactive Livestreaming / VOD

It provides capabilities for stream reception, stream convergence, transcoding and bullet screen distribution nearby. While significantly reducing network latency and optimizing user experience, it eases the pressure on the bandwidth of the IT room at the client media center (with the bandwidth cost down by approximately 80%). It is also able to provide guaranteed service levels for the link network during localized livestreaming.

• Bullet Screen Distribution

A *bullet screen* shows user comments in real-time that are displayed moving across a video. As a type of user-defined data, bullet screen does not support CDN distribution. The common practice is that the bullet screen is processed and distributed via the IDC data center. In the scenario of bullet screen distribution, edge computing nodes take care of processing and distribution, which drastically increases the success ratio of processing, sending and reception, and saves midbandwidth cost.

• Smart Security / Smart Cities

Edge computing nodes can provide the functions of video gathering and processing nearby, and carry out the convergence, storage and analysis of video streams as well as photos within the same area, thus enabling low-cost and high-quality smart security surveillance capabilities. In this scenario, the requirements for the construction of smart security and smart cities can be met, and the expandability and plasticity of BEC solutions allows for unified management at the edge.

• Creating Brand New CDN Architecture

This new architecture delivers CDN services generated by the edge computing nodes, provides flexible asset-light hash rate resources throughout the region, and helps users quickly access CDN services.

• Cloud Games, AR/VR

Cloud games, AR, VR and the like are extremely sensitive to latency. Edge computing nodes can provide GPU computing resources with high real-time performance, and effectively cope with the requirements for scene rendering in games.

5.4 DuEdge Network Computing Services

Scenarios where Kata can be securely applied to DuEdge include:

- User development and test code computing containers: usually there are some bugs in the process of test code development such as infinite loop, memory leaks, incorrect network calls, etc. Without the isolation provided by Kata, trusted container workloads wouldn't want to run on the same host as unproven / untrusted container images.
- 2. Computing containers with high risk codes: some codes fail to pass the automatic check or malicious codes are spotted during the auto check process.
- 3. Computing containers without secure sandbox language: for some programming languages, for example, Python and Go, it is difficult to create a secure sandbox. To overcome this issue, the code is run in Kata Containers.
- 4. Computing containers with isolated businesses: some businesses wish to have more independent and secure computing container execution.
- 5. Computing containers with special requirements for the kernel: some business applications require specific kernel versions, different from that of the host OS. In this scenario, the customizable kernel feature of Kata Containers solves this issue.

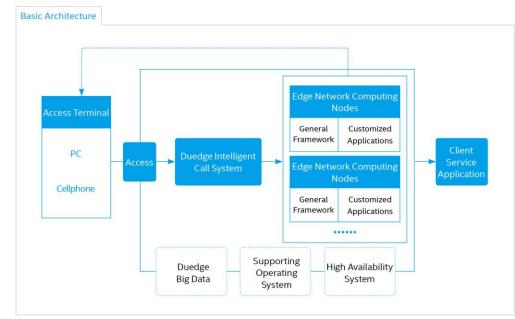


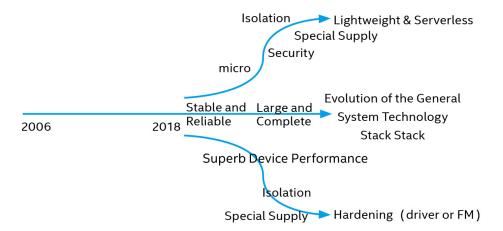
Figure 15. DuEdge Function Computing

6.0 Outlook

6.1 Ongoing Evolution of Lightweight Virtualization

As virtualization technologies evolve from general virtualization to the different paths shown in Figure 16, the development of lightweight virtualization is meant to satisfy more container requirements. The differentiating factors for the scenarios include isolation, performance and special requirements, however, safe isolation is of paramount importance. Throughout the evolution of CFC, a number of lightweight virtualization products appeared that were applicable to various scenarios. Per ascending order of the lightweight levels, function computing, Baidu container instances and edge computing products have been created.

Figure 16. Evolution of Virtualization Technologies



As a secure container solution, Kata Containers plays a vital role in the container services provided by Baidu by meeting diverse customer use cases through the support of multiple KVM-based Virtual Machine Managers (VMMs). This dovetails nicely with the **Rust-VMM** community project [Ref. 16 in <u>Table 2</u>]. Rust-VMM is a virtualized component library that enables users to build their own custom hypervisor. The **Intel**[®] **Cloud Hypervisor project** [Ref. 17 in <u>Table 2</u>] was launched on the basis of Rust-VMM. By taking advantage of the safer Rust language and only including modern hypervisor features, users can reduce both their attack surface and their memory footprint.

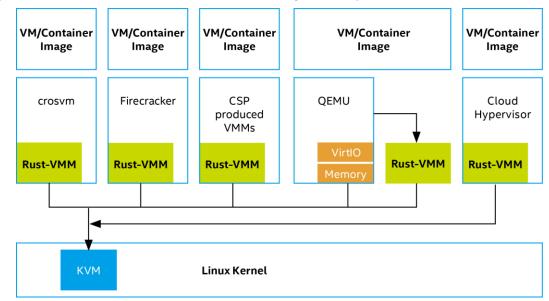


Figure 17. Set of Virtualization Rust Crates for Easy VMM Implementation

6.2 Al Acceleration Hardware

The edge-oriented trend of AI inference technologies boasts a promising future and the AI acceleration hardware is also driving the market forward. The virtualization of AI hardware and its support for intra-container access are also popular for the existing research on container technologies. It is the users' common hope that GPU can be shared in the multi-tenant environment so that those resources can fully be used. Moreover, AI acceleration has its dedicated FPGA and a plug-in AI acceleration chip that targets embedded devices. Preferably, it should also be compatible with the Baidu AI computing platform such as PaddlePaddle [Ref. 18 in Table 2].

6.3 Trustworthy Computing Environment

In the high security level edge computing scenarios, users do not want to hand their code or encrypted keys to the cloud computing provider, which calls for reliable computing containers by the latter. While the Intel® Software Guard Extensions [Ref. 19 in <u>Table 2</u>] technology can mitigate this issue, how to apply it to the edge computing scenarios is something worthy of further research.

7.0 Acknowledgements

In the writing of this white paper, we are so honored to have the support of our colleagues from all groups within Baidu Smart Cloud. We are particularly appreciative of the ever-lasting attention given to us by Xie Guangjun, the deputy general manager of Baidu Intelligent Cloud in the implementation of new technologies, especially in promoting the application of Kata technology within Baidu. We must also extend our sincere thanks to our product team members including Sun Li, Zhou Yueqian and Wu Qiucai for compiling and editing the requirement part of this white paper, to Song Fei, the head of the product team, for proofreading the contents, and to other colleagues including He Fangshi, Ni Xun, Shen Jiale, Han Ding, Zhou Ti, Bai Yu, Wang Hui, Xie Yongji and Li Yuxuan for writing the materials relating to research and development as well as their support for the entire project. We shouldn't, of course, forget the much-needed support given by Chen Shousong from the Strategic Planning & Management Department of Baidu Smart Cloud and Tan Zhongyi from the Open Source Technology Committee of Baidu.

We acknowledge the great support given by Intel team members including Eric Ernst, Yang Jiguo, Hu Zhiming, Shen Xiaochen, Jia Pei and Lyu Huijing who actively communicate with Baidu in technical details of Kata Containers and contribute in its optimization for customized scenarios of Baidu. We also thank Intel team members including Xu Libing, Liang Bing, Jin Tongtong, Chen Ke, Margaret LaBrecque, Eric Adams, Samuel Ortiz, Mary Camp and Sebastien Boeuf for their assistance with proofreading the white paper.

Finally, we are very indebted to the Kata Community. No words can fully express our appreciation. Without Kata, there would never be such great tools with which to solve these complex problems and this white paper would never come into being.

However, we understand this is only the beginning of our great journey. We will join hands and stride forward together!

8.0 References

Table 2. References

Number	Reference URL
1.	https://cloud.baidu.com/
2.	https://www.forrester.com/report/The+Forrester+Wave+FullStack+Public+Cloud+Developm ent+Platforms+In+China+Q3+2018/-/E-RES142373#figure3
3.	https://www.srgresearch.com/articles/chinese-companies-now-account-40-apac-public- cloud-market
4.	https://cloud.baidu.com/product/cce.html
5.	https://cloud.baidu.com/product/cfc.html
6.	https://cloud.baidu.com/product/bci.html
7.	https://duedge.baidu.com
8.	https://katacontainers.io
9.	https://github.com/clearcontainers/runtime
10.	https://github.com/hyperhq/runv
11.	https://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel- virtualization-technology.html
12.	https://github.com/kata-containers/runtime/issues/808
13.	https://www.openstack.org
14.	https://github.com/intel/multus-cni
15.	https://docs.google.com/document/d/1Ny03h6IDVy_e_vmElOqR7UdTPAG_RNydhVE1Kx54 kFQ
16.	https://github.com/rust-vmm
17.	https://github.com/intel/cloud-hypervisor
18.	https://github.com/PaddlePaddle
19.	https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard- extensions.html