Kunpeng BoostKit for Virtualization

Technical White Paper

Issue 23

Date 2025-12-30

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Overview

1.1 Introduction

Nowadays, digital transformation is a challenge for enterprises in all industries around the world. Application modernization is the core of digital transformation. It helps enterprises attract customers, train employees, optimize operations, and improve products. As the IT infrastructure of digital transformation, cloud computing technologies have developed rapidly in recent years. Enterprises benefit greatly from the development of cloud computing technologies such as virtualization, cloud services, and containerization in their digital transformation. The continuous innovation of cloud computing relies largely on the rapid development of open-source technologies and ecosystems. Open-source cloud computing technologies such as QEMU-KVM, OpenStack, Docker, and Kubernetes have broken the siloed computing architecture that was once closed and inefficient, continuously enriching IT infrastructure, making user applications evolve towards being more agile and efficient, and accelerating digital transformation.

Kunpeng BoostKit for Virtualization provides the following features to accelerate the implementation of cloud computing:

- Eliminates dependency on x86 servers to provide more computing platform options and reduce service continuity risks.
- Multi-core processor architecture that features higher density, lower power consumption, higher cloud infrastructure computing capability, and lower total cost of ownership (TCO).
- Replaces the cloud computing infrastructure platform without affecting user experience.
- Hybrid deployment of x86 and Kunpeng for higher flexibility and scalability.

This document describes the architecture, application scenarios, hardware, software, and typical configurations of Kunpeng BoostKit for Virtualization.

1.2 Overall Architecture

The overall architecture of Kunpeng BoostKit for Virtualization consists of the hardware infrastructure, OS, cloud platform, and cloud cluster management platform, as shown in **Figure 1-1**. For details about the components, see **Table 1-1**.

Figure 1-1 Overall architecture of Kunpeng BoostKit for Virtualization

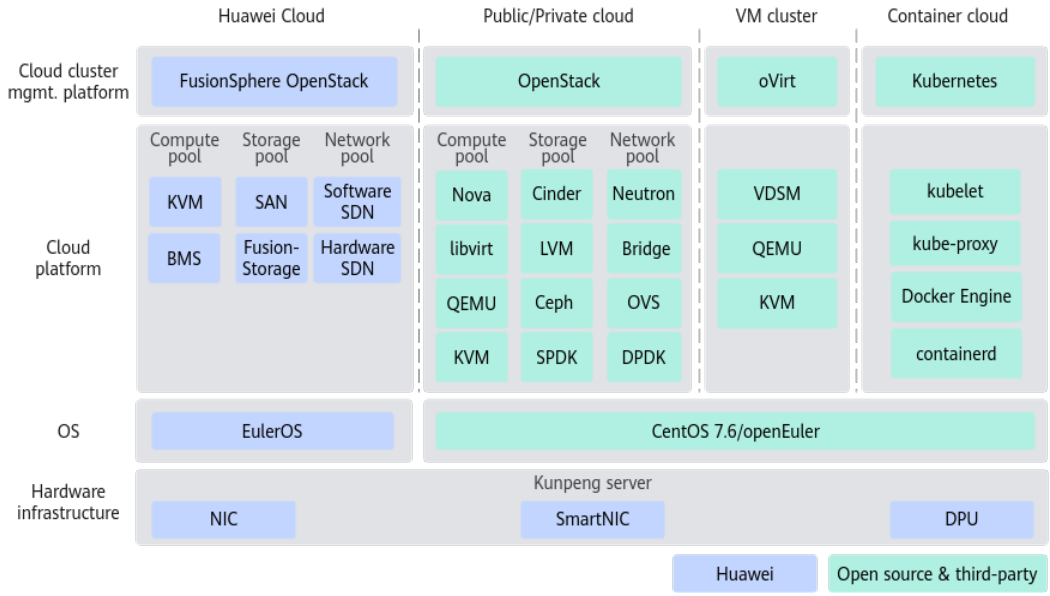


Table 1-1 Kunpeng BoostKit for Virtualization components

Name	Description
Infrastructure	Kunpeng servers powered by Kunpeng processors
OS	• Open-source CentOS 7.6 • openEuler 20.03 or later • EulerOS 2.8 for HCS commercial use
Cloud platform	HCS private cloud platform and open-source QEMU-KVM and Docker container platforms
GuestOS	CentOS 7.6, Ubuntu 16.04, SUSE 15.1, and Kylin V7.6 on VMs
Cluster management platform	Open-source OpenStack, Kubernetes, and oVirt management platforms

1.3 Advantages

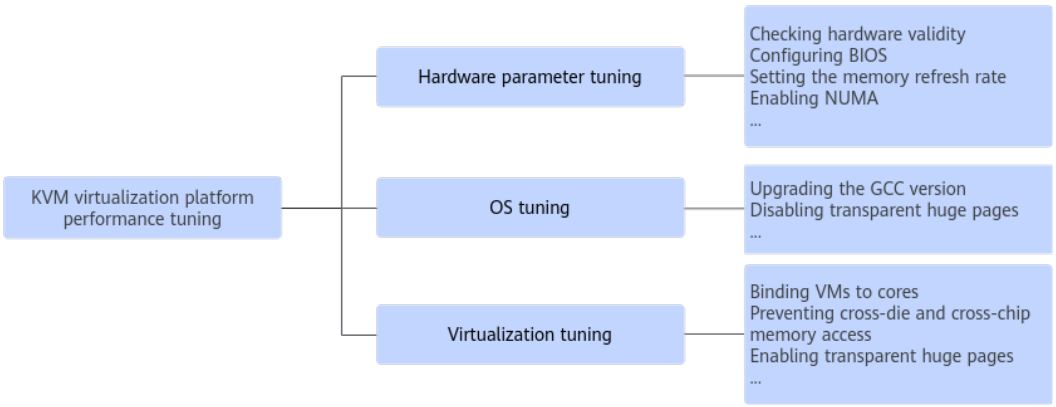
Based on Kunpeng servers, Kunpeng BoostKit for Virtualization provides a full stack of hardware, OSs, virtualization software, and OpenStack cloud

management software. It enables in-depth performance tuning for the Kunpeng architecture covering hardware parameters, OS kernels, and virtualization platforms. The solution features high performance and an open ecosystem.

High Performance

Kunpeng BoostKit for Virtualization enables in-depth performance tuning for hardware parameters, OS kernels, and virtualization platforms, making the most of Kunpeng multi-core processors.

Figure 1-2 Virtualization performance tuning items



Open Ecosystem

Kunpeng BoostKit for Virtualization has an open software ecosystem.

- It supports open-source components such as QEMU-KVM and Docker, and open-source cloud cluster management platforms such as OpenStack, Kubernetes, and oVirt.
- It supports HCS.

2 Typical Solutions

2.1 Virtualization Solution (Based on Open-Source KVM)

2.1.1 Architecture

The open-source KVM virtualization solution is designed for offline virtualization scenarios, including single-node, two-node high availability (HA), and multi-node cluster scenarios. It leverages VM migration and HA to ensure service reliability. Typical applications include databases, web servers, and cache servers.

- Single-Node Scenario Analysis

  In a single-node system, the QEMU-KVM open-source software is deployed on a single server. The Virt-Manager management software and virsh commands are used for out-of-band VM management. Both of them invoke libvirt APIs. The VNC software is used for in-band guest OS management. The local LVM virtual storage pool is used for VM storage. The VM network uses bridges (bridge mode) or physical NICs (host-only mode).

- Two-Node and Cluster Scenario Analysis

  The configurations of the two-node and cluster scenarios are based on that of the single-node system scenario. The computing virtualization, storage, and network configurations are the same as those of the single-node scenario. The difference is that the HA or live migration technology can be used to ensure cluster robustness in the two-node cluster and cluster scenarios. The following uses the Keepalived+LVS+MySQL two-node primary-secondary architecture as an example. Keepalived provides a floating IP address and

periodically checks the health status of servers in the cluster. When a faulty node is detected, a switchover is triggered.

The architecture of the open-source KVM virtualization scenario is divided into three layers. The underlying layer is the Kunpeng server hardware, and the middle layer is the host Linux kernel and the KVM virtualization software. The top layer is the QEMU, which virtualizes I/O devices. **Figure 2-1** shows the detailed system architecture and **Table 2-1** describes the components.

Figure 2-1 Open-source KVM virtualization architecture

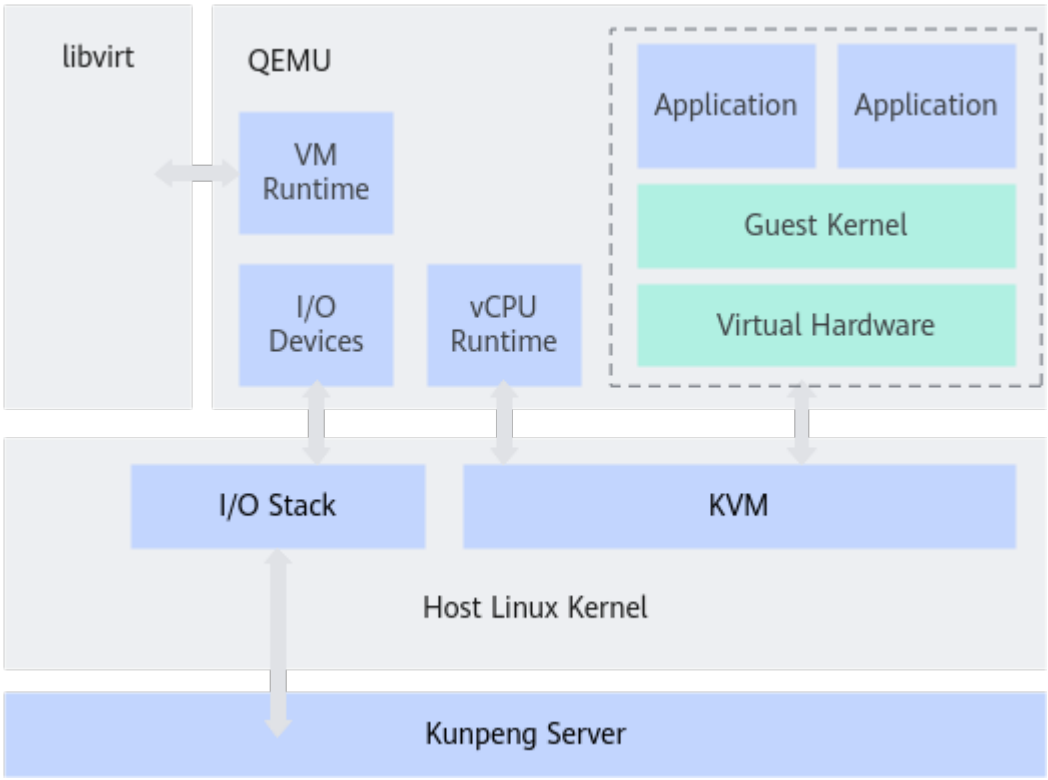


Table 2-1 Components in the open-source KVM virtualization scenario

Name	Description
KVM	The KVM is a kernel feature of the host Linux OS. It supports simulation of the CPU, memory, and I/O. The KVM is used as the hypervisor and works with QEMU to virtualize KVM VMs.
QEMU	QEMU runs in the user mode on the host as a process. Based on the KVM and kernel features, QEMU simulates hardware such as the CPU, memory, and I/O to support running of the guest OS in a process.
libvirt	The libvirt library provides APIs for Linux virtualization functions. Virtualization management services, such as virt-manager, manage and monitor VMs by using libvirt.

Name	Description
Virtual Machine	A virtual machine (VM) is a server resource that allows users to install Guest OSs including CentOS 7.6, SUSE 15.1, Ubuntu 16.04, and Kylin 7.6. You can run your own applications on the Guest OSs.

2.1.2 Typical Configuration

Table 2-2 lists the typical compute node configuration in the open-source KVM virtualization scenario.

Table 2-2 Typical compute node configuration for KVM VMs

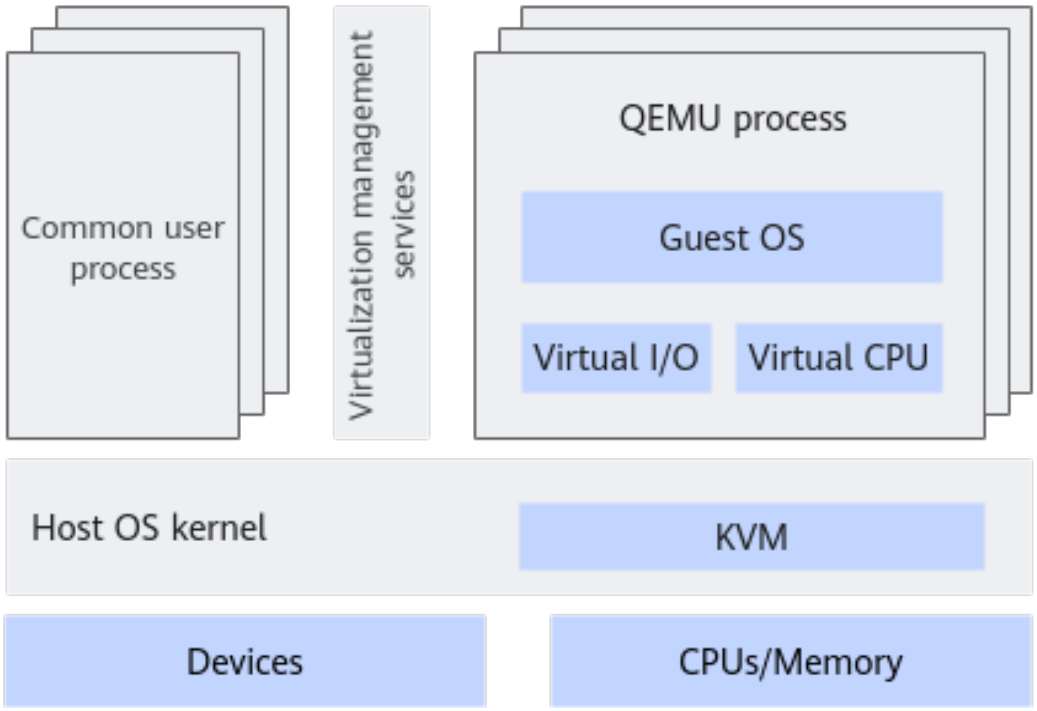
Node Type	Typical Configuration	Remarks
Compute node	Dual-socket rack servers • 2 x Huawei Kunpeng 920 processor (Kunpeng 920 7260 is recommended) • 256 GB or larger memory • 2 x 480 GB SSD • 6 x 600 GB SAS HDD (six or more) • Avago 3508 RAID controller cards • 1 x SP680 SmartNIC • An independent power supply	The number of compute nodes is calculated based on the number of VMs.

2.1.3 Component Principles

KVM

KVM uses features of the Linux kernel to implement functions such as CPU virtualization, memory virtualization, peripheral virtualization, and VM management. It is a bridge for software to simulate hardware and mode conversion. Figure 2-2 shows the KVM architecture.

Figure 2-2 KVM architecture



KVM is a hypervisor that runs in the host OS kernel. It simulates the CPU, memory, and I/O, monitors VMs, and provides entity support for the QEMU.

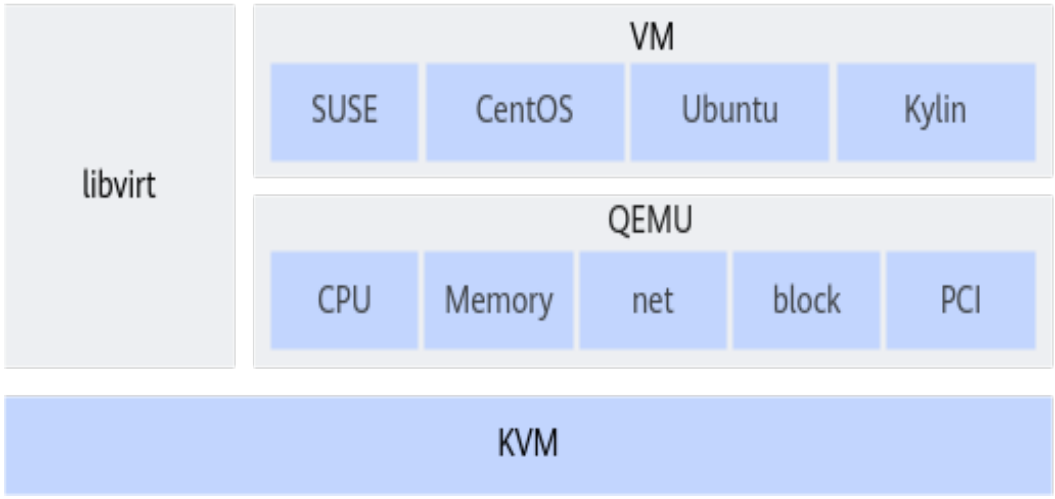
KVM has the following advantages:

- High compatibility
- Easy memory management. A VM is a process.
- High performance. The code resources of the KVM and OS kernel can be directly invoked.
- Focus on virtualization. Hardware is fully utilized to support virtualization.
- Good scalability. Memory management and multi-processor functions provided by the Linux kernel can be directly used.
- Simplicity. Currently, KVM I/O virtualization is implemented by using QEMU, which significantly reduces the implementation workload.

QEMU

QEMU is an emulator that provides a hardware environment for running VMs. It runs in the user mode on the host as a process. Based on the KVM and kernel features, QEMU simulates hardware such as the CPUs, memory, and I/O devices to support running of the guest OS in a process. QEMU supports full system emulation, full virtualization, and paravirtualization. Full system simulation and full virtualization are pure software simulation. A process constructed by a CPU can run on another CPU. The difference is that full system simulation allows the entire system to be simulated, including CPUs and peripheral devices, whereas in paravirtualization mode, QEMU and KVM are used together to simulate the CPU in KVM hardware-assisted virtualization mode. The paravirtualization mode is efficient and is commonly used. **Figure 2-3** shows the QEMU architecture.

Figure 2-3 QEMU architecture



QEMU works with KVM. KVM runs in kernel mode to simulate CPUs and memory. QEMU runs in user mode to simulate I/O devices and present virtual devices to external systems. QEMU provides the following functions for KVM:

- QMP interfaces for interaction with the upper-layer libvirt
- Virtual devices simulation
- ioctl interface for interaction with the lower-layer KVM

2.2 VM Cluster Solution (Based on Open-Source oVirt and KVM)

2.2.1 Architecture

oVirt applies to offline virtual cluster scenarios.

oVirt is an open-source virtualization management platform. It provides an easy-to-use WebUI, through which you can centrally manage VM, computing, storage, and network resources. The main components of oVirt include the oVirt engine, Virtual Desktop Server Manager (VDSM), KVM, storage, and databases. **Figure 2-4** shows the oVirt and KVM deployment architecture. For details about the components, see **Table 2-3**.

Figure 2-4 oVirt and KVM deployment architecture

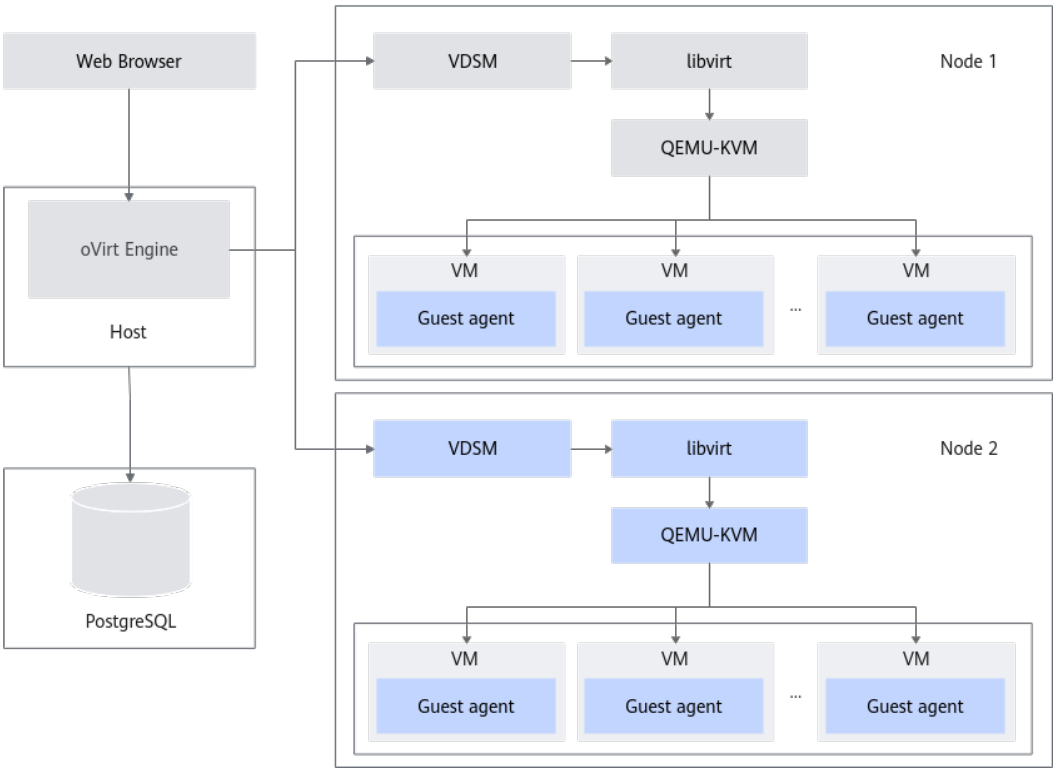


Table 2-3 Component description

Component	Description
Web Browser	A browser can be used to log in to the administrator portal or user portal. • The administrator portal is a website based on UI application programming on the engine and is used by the system administrator to perform advanced operations. • The user portal is a website based on UI application programming and is used to manage simple scenarios.
oVirt Engine	• Support for multiple virtual data centers and multi-cluster management • Support for different storage architectures (FC-SAN, IP-SAN, local storage, and NFS) • Hyper-converged deployment architecture • Unified management of the virtual computing, storage, and VM networks • Live VM and storage migration • High availability upon physical host breakdown • Load balancing resource scheduling policy

Component	Description
VDSM	Vdsm performs operations related to oVirt Engine requests. Its functions are as follows: • Starting and registering nodes • Managing the VM operations and life cycles • Managing the networks • Managing the storage • Monitoring and reporting the status of hosts and VMs • Externally interfering with VMs • Combining memory and storage and enabling memory and storage overcommitment The Vdsm system is designed based on the principles of high availability, high scalability, cluster security, backup and restoration, and performance optimization.
Guest agent	A guest agent runs on a VM, provides the oVirt engine with the VM resource usage information, and communicates with the oVirt engine through the virtual serial connection.
PostgreSQL	The oVirt engine uses PostgreSQL persistence for data storage.

2.2.2 Typical Configuration

Table 2-4 lists the typical configuration of the open-source oVirt and KVM solution.

Table 2-4 Typical configuration of open source oVirt and KVM

Node Type	Typical Configuration	Quantity	Remarks
Management node	Kunpeng server • Huawei Kunpeng 920 7260 processors • 256 GB or larger memory • 2 x 480 GB SATA SSD • Avago 3508/3516 RAID controller cards • 1 x 10GE SP680 SmartNIC (There are four network ports, and every two of them are bonded together to form a logical port.) • Independent power supply	Two or more	A management node can be self-hosted on a compute node.

Node Type	Typical Configuration	Quantity	Remarks
Compute node	Kunpeng server • Huawei Kunpeng 920 7260 processors • 256 GB or larger memory • 2 x 480 GB SATA SSD • Avago 3508/3516 RAID controller cards • 1 x 10GE SP680 SmartNIC (There are four network ports, and every two of them are bonded together to form a logical port.) • Independent power supply	Two or more	• The cluster scale is calculated based on the VM scale. • 2 x 10GE: network service + management • 2 x 10GE: network storage
Storage node (NFS or local storage)	Hot data: Kunpeng server • Huawei Kunpeng 920 5220 processors • 128 GB or larger memory • 2 x 480 GB SATA SSD • 12 x 3.2 TB NVMe SSD • 2 x 25GE NIC (LOM) (Every two network ports are bonded together to form a logical port and two logical ports are obtained. The two logical ports are connected to the public network and cluster network respectively.) • Independent power supply	Calculated based on the data volume	The calculation formula is as follows in terms of the data volume: Number of nodes = Planned data volume x 1.5 (Data expansion rate) x 1 (Data compression rate) x 3 (three copies)/0.8 (Drive utilization)/0.9 (Drive number system conversion)/(12 (Number of drives) x 3.2 TB (Drive capacity))

Node Type	Typical Configuration	Quantity	Remarks
	Warm data: Kunpeng server • Huawei Kunpeng 920 5220 processors • 192 GB or larger memory • 2 x 480 GB SATA SSD • 2 x 3.2 TB NVMe SSD for acceleration • 12 x 8 TB HDD • 2 x 25GE/10GE NIC (LOM) (Every two network ports are bonded together to form a logical port and two logical ports are obtained. The two logical ports are connected to the public network and cluster network respectively.) • Independent power supply	Calculated based on the data volume	The calculation formula is as follows in terms of the data volume: Number of nodes = Planned data volume x 1.5 (Data expansion rate) x 1 (Data compression rate) x 3 (three copies)/0.8 (Drive utilization)/0.9 (Drive number system conversion)/(12 (Number of drives) x 8 TB (Drive capacity))
	Cold data: Kunpeng server • Huawei Kunpeng 920 5220 processors • 128 GB or larger memory • 2 x 480 GB SATA SSD • 36 x 8 TB HDD • 2 x 25GE/10GE NIC (LOM) (Every two network ports are bonded together to form a logical port and two logical ports are obtained. The two logical ports are connected to the public network and cluster network respectively.) • Independent power supply	Calculated based on the data volume	The calculation formula is as follows in terms of the data volume: Number of nodes = Planned data volume x 1.5 (Data expansion rate) x 1 (Data compression rate) x 3 (three copies)/0.8 (Drive utilization)/0.9 (Drive number system conversion)/(36 (Number of drives) x 8 TB (Drive capacity))

2.2.3 Component Principles

oVirt Engine

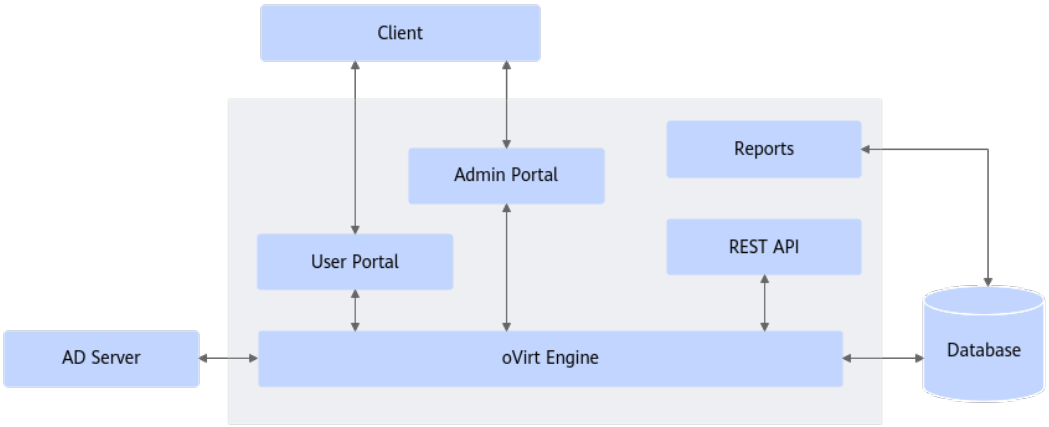
The oVirt engine runs on JBoss-based Java applications. The service communicates with Vdsm on the host to deploy, start, stop, migrate, and monitor VMs. In addition, the service can create new images on the storage by using templates. The oVirt engine uses a scalable, secure, and high-performance architecture to provide centralized management for large-scale servers and desktop virtualization.

oVirt provides the following functions:

- VM life cycle management.
- Network management: adding logical networks to the host.
- Storage management: managing storage domains (NFS, iSCSI and local storage) and VM disks.
- High availability: automatically restarting VMs of a faulty node on another node.
- Live migration: migrating running VMs between nodes without interrupting services.
- System scheduling: performing load balancing for VMs based on resource usage or policies.
- Energy saving: deploying VMs on a few servers during off-peak hours.
- Maintenance manager: ensuring that VMs are not stopped during the planned maintenance period.
- Image management: providing template-based configuration, thin provisioning, and snapshot.
- Monitoring: monitoring all objects in the system, such as VMs, nodes, networks, and storage.
- Importing and exporting: importing and exporting a VM or template by using an OVF file.

Figure 2-5 shows the oVirt engine component architecture.

Figure 2-5 oVirt engine component architecture



oVirt Node

An oVirt node is a compute node where VMs run. For details about how to install a common Kunpeng server as a compute node in the oVirt virtual environment, see **Kunpeng oVirt Lightweight Virtualization Management Platform Deployment Guide**.

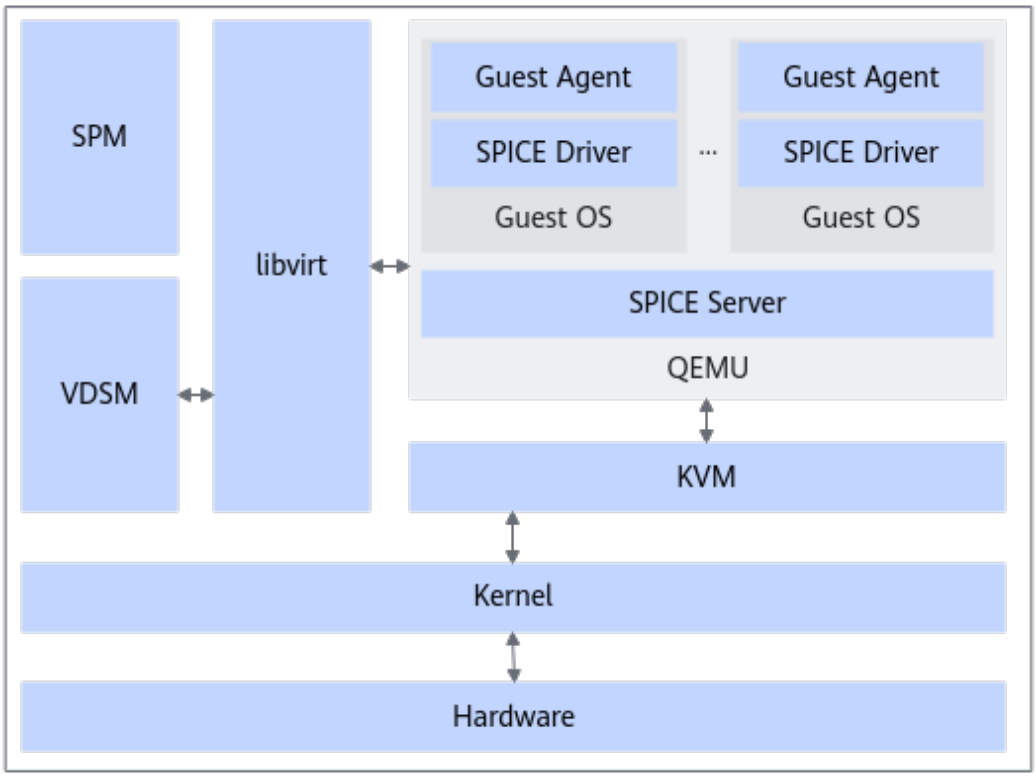
Vdsm on an oVirt node functions as the agent of the oVirt engine. It manages all resources in the oVirt virtual environment and performs client operations. A Vdsm command is run on each compute node. After receiving the instruction from the client, Vdsm invokes the libvirt underlying tool library to manage VMs and

hardware devices. QEMU supports the display of SPICE drivers. Therefore, clients can use SPICE client software to access VMs in graphical mode.

The Storage Pool Manager (SPM) is a role given to one of the hosts in the data center enabling it to manage the storage domains of the data center. Any host in the data center can function as an SPM. The system grants the role to one of the hosts in the data center. The SPM does not affect the normal functions of the host. The host running as the SPM can still provide virtual resources for running VMs.

Figure 2-6 shows the oVirt node architecture.

Figure 2-6 oVirt node architecture



2.3 Public Cloud and Private Cloud Solutions (Based on Open-Source OpenStack and KVM)

2.3.1 Architecture

The OpenStack+KVM solution applies to public cloud and private cloud scenarios.

OpenStack is an open-source project based on a community. It provides an operation platform and tool set for cloud deployment. It aims to help organizations run clouds that provide virtual computing or storage services and provide scalable and flexible cloud computing for public and private clouds.

OpenStack includes the following open-source components: Nova, Cinder, Neutron, Glance, Swift, Placement, Keystone, Horizon, Heat, and Ceilometer.

Figure 2-7 shows the OpenStack system architecture.

Figure 2-7 OpenStack component architecture

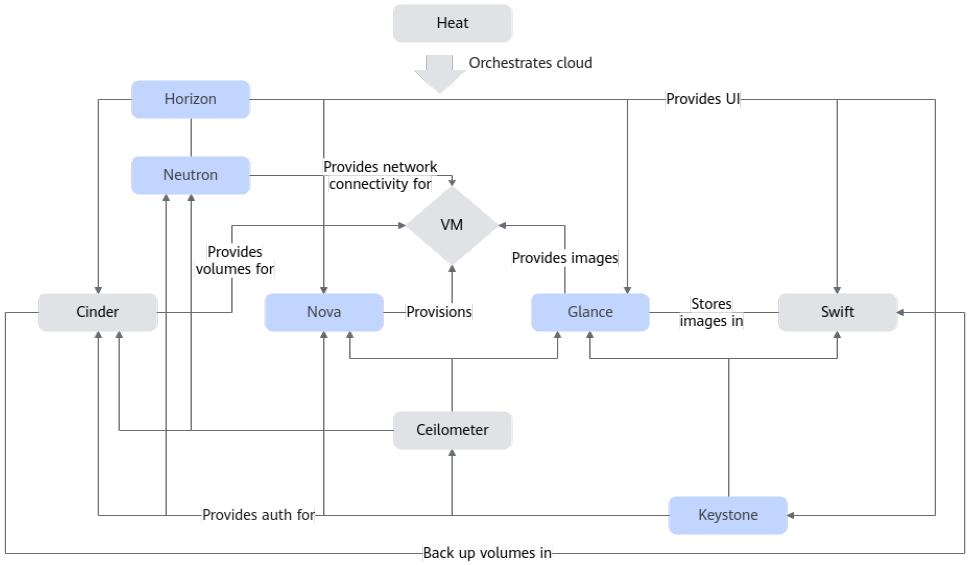


Table 2-5 Nodes in the OpenStack cloud scenario

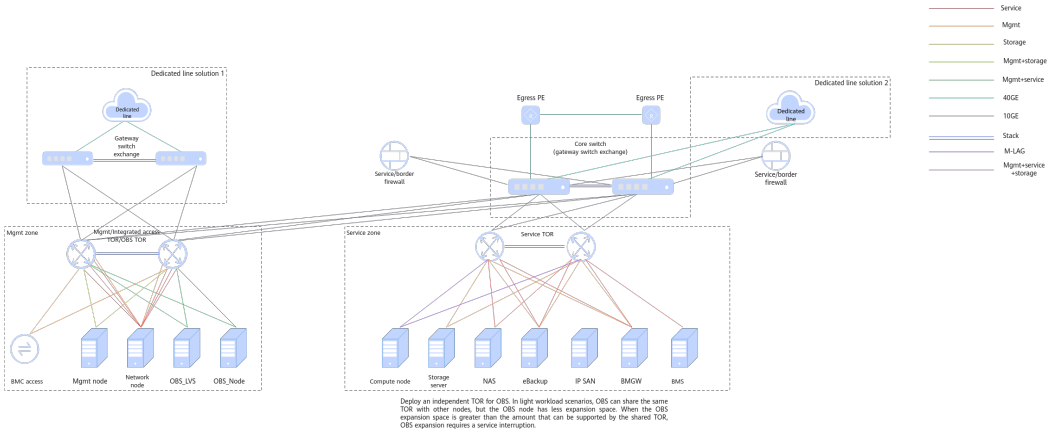
Name	Description
Nova	Nova is the core component of OpenStack and manages VM computing resources including the CPU and memory resources. When a VM creation request is received, Nova filters computing resources based on the computing resource requirements in the request, selects the resources that can be used to create VMs, sorts the resources based on a certain policy, and selects certain computing resources for VM creation.
Cinder	Cinder provides block storage resources for VMs. The OS operates drives by block. The OS divides a drive into blocks (clusters) to read and write the drive. Cinder uses drivers to manage, read, and write the physical storage media, and provides unified iSCSI storage (unified drive volumes) for VMs. VMs mount the unified drive volumes provided by Cinder.
Neutron	Neutron provides network resources for VMs. To implement communication and isolation between VMs and the external network, information such as IP addresses, routes, and VLANs need to be configured so that a grouped packet data forwarding channel that supports isolation, switching, and routing is set up on the network. Neutron configures interfaces, VLANs, and routes on hosts, switches, and routers to support VM data forwarding according to VM requirements.
Glance	Glance manages VM images and allows users to query, register, upload, obtain, and delete VMs.

Name	Description
Swift	Swift is an object storage component of OpenStack. It can interconnect with Ceph to provide object storage for OpenStack.
Horizon	Horizon provides a WebUI for almost all components. OpenStack components support only CLI commands. Horizon provides the graphical encapsulation function.
Keystone	Keystone provides the authentication function. In OpenStack, Keystone authenticates all operations of components that require authentication.
Ceilometer	Ceilometer provides resource metering and monitoring services.
Heat	Heat provides the service orchestration function.

2.3.2 Networking

The system networking of Kunpeng BoostKit for Virtualization consists of the management and service planes. For details, see **Figure 2-8**.

Figure 2-8 System networking of Kunpeng BoostKit for Virtualization



The network of Kunpeng BoostKit for Virtualization consists of the management and service planes. If big data services are deployed on the cloud, the big data service cluster is generally deployed independently. OpenStack components and BMC out-of-band management are deployed in the management zone. Compute nodes, storage nodes, bare metal servers (BMSs), and physical gateways are deployed in the service zone.

Table 2-6 BMC access network configuration

Port Type	Access Network	Description
GE electrical port	Management network	Connected to the management top-of-rack (TOR) switch.

Table 2-7 Management node network configuration

Port Type	Access Network	Description
2 x 10GE optical port	Management network	Connected to the management TOR switch. The two ports form a logical bond port in active/standby mode.
2 x 10GE optical port	Storage network	Connected to the storage TOR switch. The two ports form a logical bond port in active/standby mode.

The following table shows an alternative management node network configuration.

Port Type	Access Network	Description
2 x 10GE optical port	Management +storage network	Connected to the management TOR switch. The two ports form a logical bond port in active/standby mode.

Table 2-8 Network node network configuration

Port Type	Access Network	Description
2 x GE optical port	Management network	Connected to the management TOR switch. The two ports form a logical bond port in active/standby mode.
2 x 10GE optical port	Service network	Connected to the integrated access TOR switch to transmit vRouter traffic. The two ports form a logical bond port in Link Aggregation Control Protocol (LACP) mode.
2 x 10GE optical port	Service network	Connected to the integrated access TOR switch to transmit Load Balancer as a Service (LBaaS) traffic. The two ports form a logical bond port in LACP mode.
2 x 10GE optical port	Service network	Connected to the integrated access TOR switch to transmit Linux Virtual Server (LVS) traffic. The two ports form a logical bond port in active/standby mode.

Port Type	Access Network	Description
2 x 10GE optical port	Service network	Connected to the integrated access TOR switch to transmit Nginx traffic. The two ports form a logical bond port in active/standby mode.

Table 2-9 Compute node network configuration

Port Type	Access Network	Description
2 x 10GE optical port	Management +service network	Connected to the compute TOR switch. The two ports form a logical bond port in active/standby mode.
2 x 10GE optical port	Storage network	Connected to the compute TOR switch. The two ports form a logical bond port in active/standby mode.

Table 2-10 Storage node network configuration

Port Type	Access Network	Description
2 x 10GE optical port	Management +service network	Connected to the compute TOR switch. The two ports form a logical bond port in active/standby mode.

The following table shows an alternative storage node network configuration.

Port Type	Access Network	Description
2 x 25GE optical port	Management +service network	Connected to the compute TOR switch. The two ports form a logical bond port in active/standby mode.

Table 2-11 BMS node network configuration

Port Type	Access Network	Description
2 x 10GE optical port	Management +service network	Connected to the compute TOR switch. The two ports form a logical bond port in active/standby mode.

Port Type	Access Network	Description
2 x 10GE optical port	Storage network	Connected to the compute TOR switch. The two ports form a logical bond port in active/standby mode.

2.3.3 Typical Configuration

Table 2-12 lists the configuration of each component in the OpenStack scenario.

Table 2-12 Typical configuration in the OpenStack scenario

Node Type	Typical Configuration	Quantity	Remarks
Management node	Kunpeng server • Huawei Kunpeng 920 5220 processors • 256 GB or larger memory • 2 x 480 GB SATA SSD • Avago 3508 RAID controller cards • 1 x 10GE SP680 SmartNIC (four network ports) • Independent power supply	Three or more	PM: physical machines VM: virtual machines PM ≤ 50 and VM ≤ 500: 3 servers PM ≤ 100 and VM ≤ 1000: 4 servers PM ≤ 200 and VM ≤ 2000: 7 servers PM ≤ 500 and VM ≤ 5000: 10 servers PM ≤ 1000 and VM ≤ 10,000: 12 servers
Network node	Kunpeng server • Huawei Kunpeng 920 5220 processors • 256 GB or larger memory • 2 x 1.2 TB SATA SSD • Avago 3508 RAID controller cards • 2 x 10GE SP680 SmartNIC (four network ports) • 2 x GE LOM • Independent power supply	Two or more	Used only in the software SDN scenario. PM ≥ 200: four or more (recommended) 2 x 10GE: network service 2 x GE: network management

Node Type	Typical Configuration	Quantity	Remarks
Compute node	Kunpeng server • Huawei Kunpeng 920 7260 processors • 256 GB or larger memory • 2 x 480 GB SATA SSD • Avago 3508 RAID controller cards • 1 x 10GE SP680 SmartNIC (There are four network ports, and every two of them are bonded together to form a logical port.) • Independent power supply	Two or more	• The cluster scale is calculated based on the VM scale. • 2 x 10GE: network service + management • 2 x 10GE: network storage
Distributed storage node (Ceph)	Hot data: Kunpeng server • Huawei Kunpeng 920 5220 processors • 128 GB or larger memory • 2 x 480 GB SATA SSD • 12 x 3.2 TB NVMe SSD • 2 x 25GE NIC (LOM) (Every two network ports are bonded together to form a logical port and two logical ports are obtained. The two logical ports are connected to the public network and cluster network respectively.) • Independent power supply	Calculated based on the data volume	The calculation formula is as follows: Number of nodes = Planned data volume x 1.5 (Data expansion rate) x 1 (Data compression rate) x 3 (Three copies)/0.8 (Drive utilization)/0.9 (Drive number system conversion)/(12 (Number of drives) x 3.2 TB (Drive capacity))

Node Type	Typical Configuration	Quantity	Remarks
	Warm data: Kunpeng server • Huawei Kunpeng 920 5220 processors • 192 GB or larger memory • 2 x 480 GB SATA SSD • 2 x 3.2 TB NVMe SSD for acceleration • 12 x 8 TB HDD • 2 x 25GE/10GE NIC (LOM) (Every two network ports are bonded together to form a logical port and two logical ports are obtained. The two logical ports are connected to the public network and cluster network respectively.) • Independent power supply	Calculated based on the data volume	The calculation formula is as follows: Number of nodes = Planned data volume x 1.5 (Data expansion rate) x 1 (Data compression rate) x 3 (Three copies)/0.8 (Drive utilization)/0.9 (Drive number system conversion)/(12 (Number of drives) x 8 TB (Drive capacity))
	Cold data: Kunpeng server • Huawei Kunpeng 920 5220 processors • 128 GB or larger memory • 2 x 480 GB SATA SSD • 36 x 8 TB HDD • 2 x 25GE/10GE NIC (LOM) (Every two network ports are bonded together to form a logical port and two logical ports are obtained. The two logical ports are connected to the public network and cluster network respectively.) • Independent power supply	Calculated based on the data volume	The calculation formula is as follows: Number of nodes = Planned data volume x 1.5 (Data expansion rate) x 1 (Data compression rate) x 3 (Three copies)/0.8 (Drive utilization)/0.9 (Drive number system conversion)/(36 (Number of drives) x 8 TB (Drive capacity))

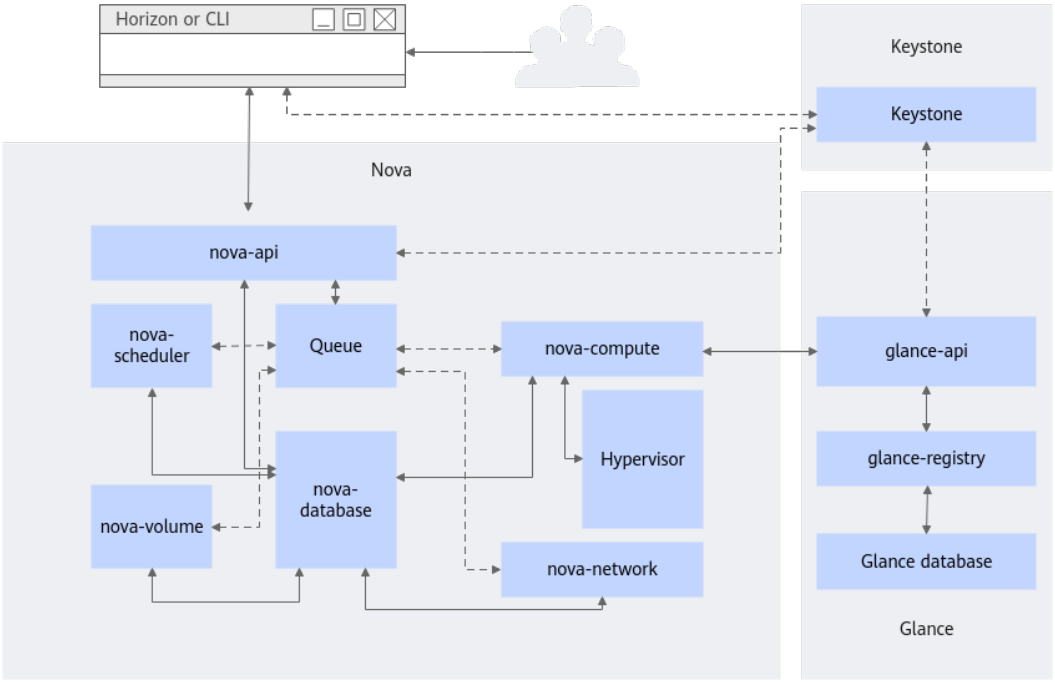
2.3.4 Component Principles

Nova

Nova is a tool for deploying the cloud and provides functions including running instances, managing the network, controlling users' and other projects' access to the cloud. Nova defines drivers that interact with underlying virtualization mechanisms that run on your host operating system, and exposes functionality over a web-based API. Nova provides API services for external systems, provides an endpoint for all API queries, initializes most deployment activities, and implements

certain policies. Nova supports single-node, two-node, and multi-node deployment. Multi-node deployment is recommended. Nova is a message-based architecture without sharing. Therefore, each nova-service can be installed on an independent server, but the dashboard must be installed on the nova-api server. **Figure 2-9** shows the position of Nova in the OpenStack framework.

Figure 2-9 Nova component architecture



The Nova services can be deployed on controller nodes and compute nodes. The nova-api provides web REST API services. The nova-scheduler selects the nodes to run VMs based on the CPU and memory weights. The nova-database is the database operation agent. The nova-compute manages the VM life cycles.

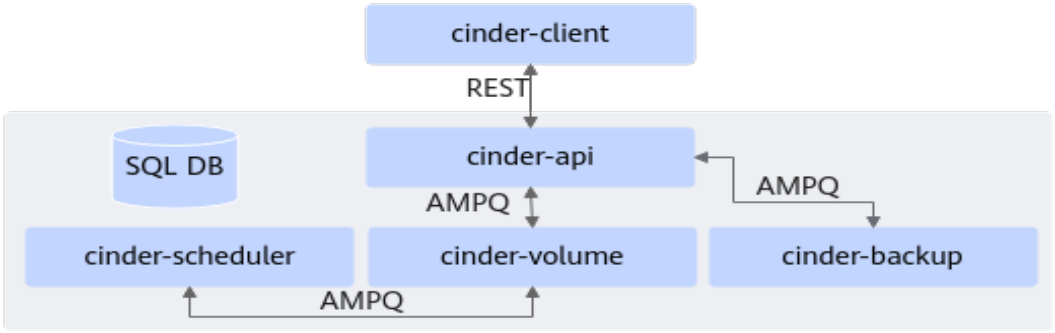
Cinder

Cinder is a block storage service that provides persistent block storage for VMs. **Figure 2-10** shows the architecture of Cinder.

- The cinder-client encapsulates REST APIs provided by Cinder, so that users can call these APIs in CLI mode.
- The cinder-api exposes REST APIs, parses operation requests, and routes APIs to handle the requests. It provides functions such as adding, deleting, modifying, and querying volumes (creating volumes from existing volumes, images, or snapshots); adding, deleting, modifying, querying, and backing up snapshots; managing volume types; attaching and detaching volumes.
- The cinder-scheduler collects capacity and capability information reported by the backend, and completes scheduling from volumes to specified cinder-volumes based on preset algorithms.
- The cinder-volume is deployed on multiple nodes. Different configuration files are used to enable connection to different backend devices. Storage vendors insert driver codes to enable interaction with storage devices to collect capacity and capability information and perform volume operations.

- The cinder-backup backs up volume data to other storage media.
- SQL DB provides data such as storage volumes, snapshots, backups, and services and supports SQL databases including MySQL and PostgreSQL.

Figure 2-10 Cinder component architecture



Neutron

Neutron is a virtual network management system that provides REST APIs, DHCP, and L2 network services. It uses a plugin architecture to support network devices and network technologies from various vendors. **Figure 2-11** shows the Neutron architecture.

- neutron-server

  Provides the REST API service and uses the relational database at the backend.
- Message Queue

  neutron-server uses message queues to exchange messages with other Neutron agents, but does not use message queues to exchange messages with other OpenStack components such as Nova.
- L2 Agent

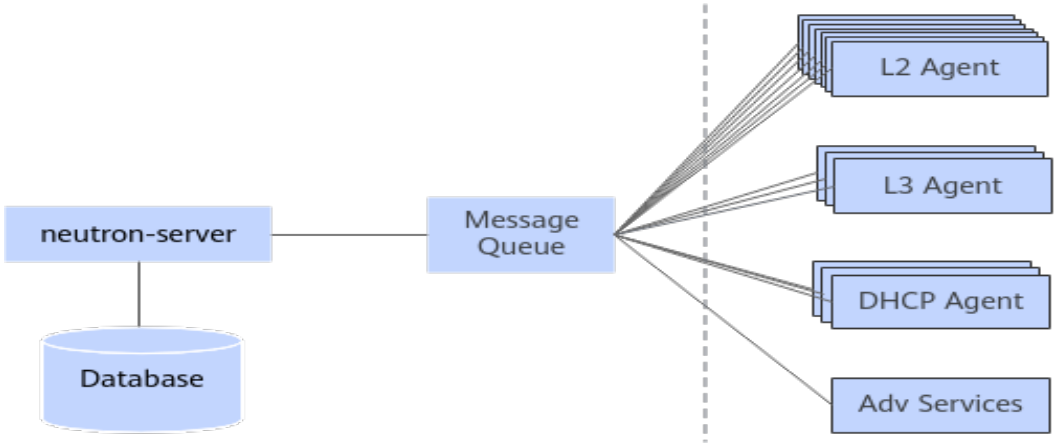
  Used to connect ports and devices so that they are in a shared broadcast domain. L2 agents usually run on the hypervisor.
- DHCP agent

  Used to configure the network of the host of VMs. In some cases (for example, the **config drive** command is used to configure the VM host), the DHCP agent may not be required.
- L3 Agent

  Used to connect the tenant network to the data center or Internet. In real deployment environments, multiple L3 agents run at the same time.
- Adv Services

  Provides high-level network services, such as load balancing, VPN, and firewall.

Figure 2-11 Neutron component architecture



2.4 Container Cloud Solution (Based on Open-Source Kubernetes and Docker)

2.4.1 Architecture

The Kubernetes+Docker solution applies to container cloud scenarios.

Linux Containers (LXC) is an OS-level virtualization method for running multiple isolated containers on a control host using a single Linux kernel. The cgroups functionality and namespace isolation functionality are used to achieve low host resource usage and high startup speed. Docker is a Linux container engine technology that enables application packaging and quick deployment. Docker uses Linux Containers to turn applications into standardized, portable, and self-managed components, enabling the "Build Once, Run Everywhere" features of applications. Features of Docker technology include: quick application release, easy application deployment and capacity expansion, high application density, and easy application management.

Kubernetes groups Docker container host machines into a cluster for unified resource scheduling, automatic container life cycle management, and cross-node service discovery and load balancing. It provides better support for the microservice concept and division of the boundaries between services, such as the introduction of concepts of label and pod.

A Kubernetes cluster consists of at least one cluster master and multiple nodes. It features lightweight plugin-based architecture, easy migration, quick deployment, and scalability. **Figure 2-12** shows the architecture.

Figure 2-12 Kubernetes and Docker container architecture

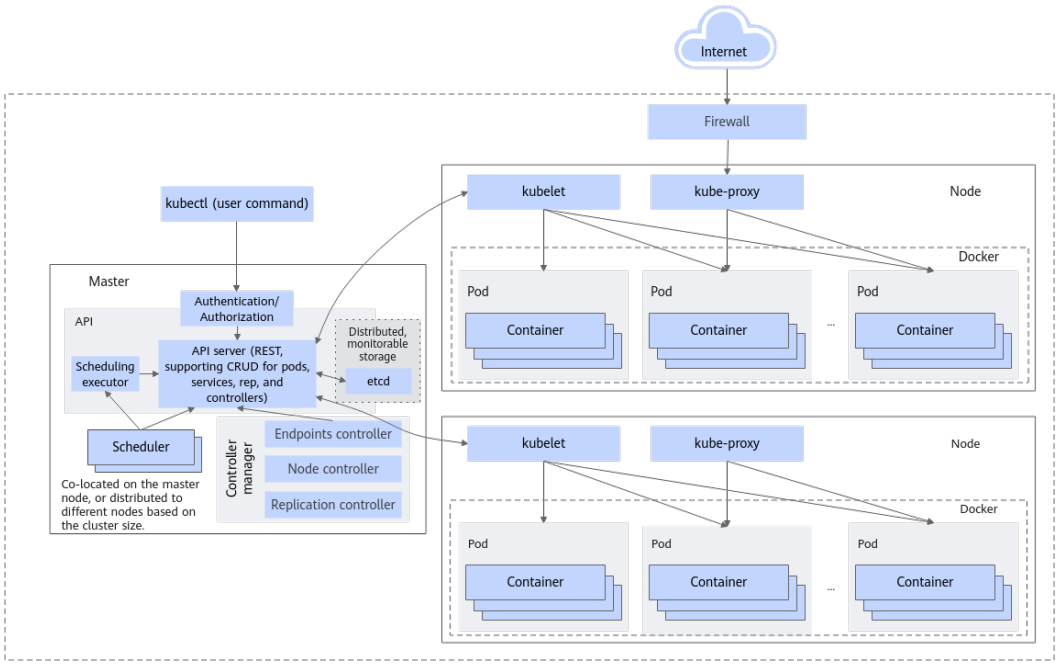


Table 2-13 Nodes in the Kubernetes and Docker container scenario

Name	Description
Pod	Pods are the smallest deployable units of containers that can be created, scheduled, and managed in Kubernetes. A pod is a collection of containers rather than a single application container. A pod is a group of one or more containers. Pods are always co-located and co-scheduled, and run in a shared context. Containers within the same pod share the same network namespace, IP address, port space, and volume. Pods are short-lived applications. Pods remain on the nodes where they are scheduled until being deleted.
API server	Exposes Kubernetes APIs. No matter whether the kubectl or API is invoked to operate various resources of the Kubernetes cluster, the operations are performed through the interfaces provided by the kube-apiserver.
kube-controller-manager	Manages the entire Kubernetes and ensures that all resources in the cluster are in the expected state. When the status of a resource in the cluster is abnormal, the controller manager triggers the corresponding scheduling operation. The controller manager consists of the following parts: • Node controller • Replication controller • Endpoints controller • Namespace controller • Service accounts controller

Name	Description
Scheduler	Schedules the Kubernetes cluster, receives scheduling operation requests triggered by the kube-controller-manager, performs scheduling calculation based on the request specifications, scheduling constraints, and overall resource status, and sends scheduling tasks to the kubelet component of the target node.
etcd	etcd is an efficient KV storage system used to share configurations and discover services. It features distribution and strong consistency. It is used for storing all data that needs to be persisted in Kubernetes.
kubelet	kubelet is the most important core component on a node. It is responsible for computing tasks of the Kubernetes cluster and performs the following functions: • Monitors task assignment of kube-scheduler. • Mounts volumes for pod containers. • Downloads secrets for pod containers. • Runs the Docker container by interacting with the Docker daemon. • Periodically performs container health check. • Monitors and reports pod status to kube-controller-manager. • Monitors and reports node status to kube-controller-manager.
kube-proxy	Forwards service requests from services to pod instances and manages load balancing rules.

2.4.2 Typical Configuration

Table 2-14 lists the configuration of each component in the Kubernetes and Docker container scenario.

Table 2-14 Typical configuration in the Kubernetes and Docker container scenario

Node Type	Typical Configuration	Quantity	Remarks
Management node	Kunpeng server • Huawei Kunpeng 920 5220 processors • 256 GB or larger memory • 2 x 480 GB SATA SSD • Avago 3508 RAID controller cards • 1 x 10GE SP680 SmartNIC • Independent power supply	Three or more	• Used to deploy core Kubernetes components, including kube-apiserver and kube-controller-manager. • Used to deploy HA and load balancing components such as Keepalived, HAProxy, and Nginx. • Supports deployment of etcd and SWR. • Supports deployment of Elasticsearch and monitoring, log, and alarm components.
Container node	Kunpeng server • Huawei Kunpeng 920 7260 processors • 256 GB or larger memory • 2 x 480 GB SATA SSD • Avago 3508 RAID controller cards • 2 x 2 x 10GE NIC • Independent power supply	Two or more	• The cluster scale is calculated based on the container scale. • 2 x 10GE: storage • 2 x 10GE: management + service
Distributed storage node (Ceph)	Hot data: Kunpeng server • Huawei Kunpeng 920 5220 processors • 128 GB or larger memory • 2 x 480 GB SATA SSD • 12 x 3.2 TB NVMe SSD • 2 x 25GE NIC (LOM) (Every two network ports are bonded together to form a logical port and two logical ports are obtained. The two logical ports are connected to the public network and cluster network respectively.) • Independent power supply	Calculated based on the data volume	The calculation formula is as follows: Number of nodes = Planned data volume x 1.5 (Data expansion rate) x 1 (Data compression rate) x 3 (Three copies)/0.8 (Drive utilization)/0.9 (Drive number system conversion)/(12 (Number of drives) x 3.2 TB (Drive capacity))

Node Type	Typical Configuration	Quantity	Remarks
	Warm data: Kunpeng server • Huawei Kunpeng 920 5220 processors • 192 GB or larger memory • 2 x 480 GB SATA SSD • 2 x 3.2 TB NVMe SSD for acceleration • 12 x 8 TB HDD • 2 x 25GE/10GE NIC (LOM) (Every two network ports are bonded together to form a logical port and two logical ports are obtained. The two logical ports are connected to the public network and cluster network respectively.) • Independent power supply	Calculated based on the data volume	The calculation formula is as follows: Number of nodes = Planned data volume x 1.5 (Data expansion rate) x 1 (Data compression rate) x 3 (Three copies)/0.8 (Drive utilization)/0.9 (Drive number system conversion)/(12 (Number of drives) x 8 TB (Drive capacity))
	Cold data: Kunpeng server • Huawei Kunpeng 920 5220 processors • 128 GB or larger memory • 2 x 480 GB SATA SSD • 36 x 8 TB HDD • 2 x 25GE/10GE NIC (LOM) (Every two network ports are bonded together to form a logical port and two logical ports are obtained. The two logical ports are connected to the public network and cluster network respectively.) • Independent power supply	Calculated based on the data volume	The calculation formula is as follows: Number of nodes = Planned data volume x 1.5 (Data expansion rate) x 1 (Data compression rate) x 3 (Three copies)/0.8 (Drive utilization)/0.9 (Drive number system conversion)/(36 (Number of drives) x 8 TB (Drive capacity))

2.4.3 Component Principles

Kubernetes

Kubernetes is an open-source container cluster deployment and management system, and has become a de facto standard in the PaaS field. Kubernetes is developed based on container technology and provides functions like resource scheduling, deployment, execution, service discovery, and capacity expansion and reduction for containerized applications. Kubernetes is the scheduling platform

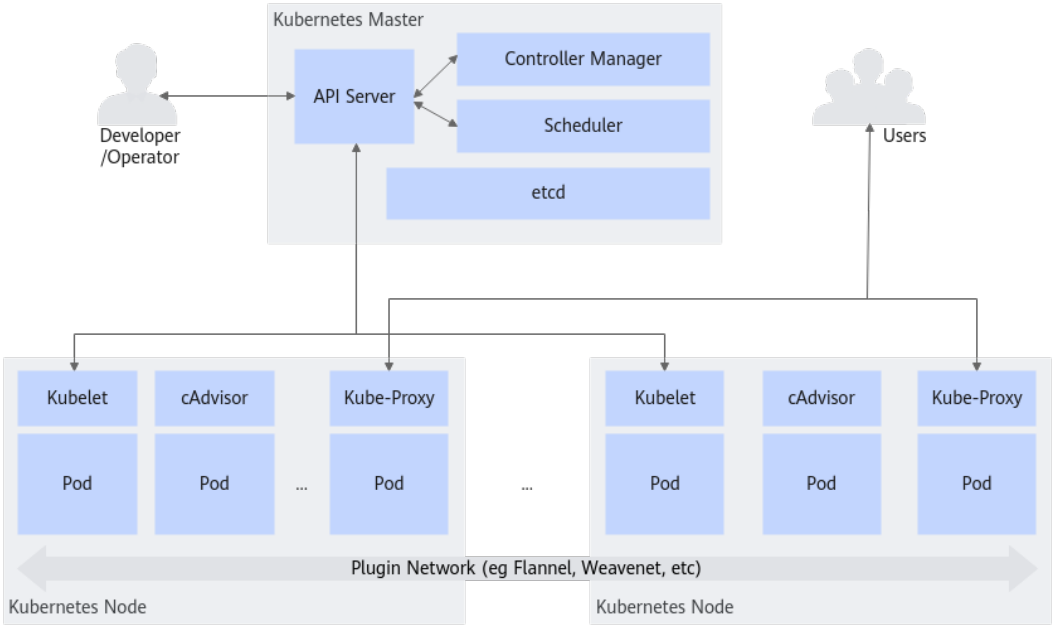
indeed, which is based on container technologies. **Figure 2-13** shows the architecture.

Kubernetes provides the following functions:

- Supports container-based application deployment, maintenance, and rolling upgrade.
- Supports load balancing and service discovery.
- Supports cluster scheduling cross different machines and areas.
- Supports auto scaling.
- Supports stateless and stateful services.
- Supports a wide range of volumes.
- Ensures plug-in mechanism scalability.

Kubernetes develops rapidly and has become a leader in the field of container orchestration.

Figure 2-13 Kubernetes component architecture



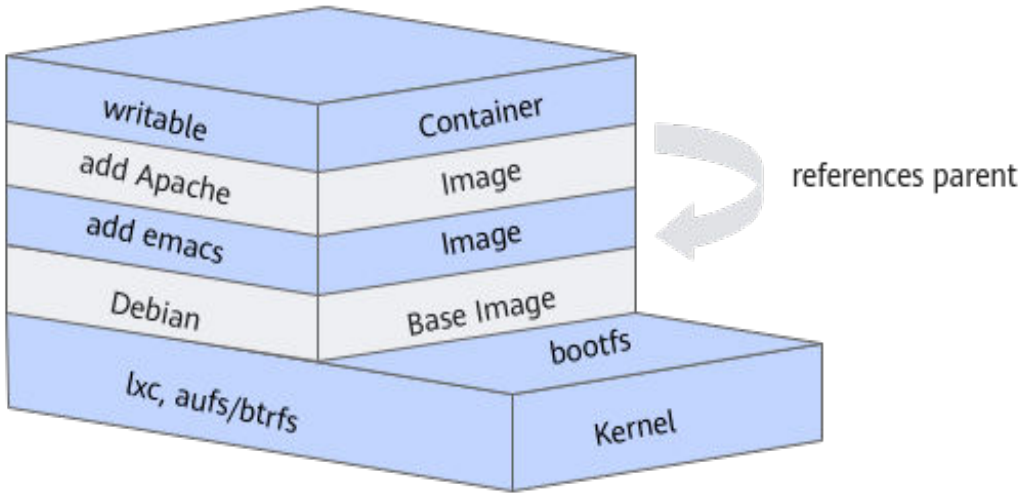
Docker

A Docker container is a process on the host OS. Docker uses namespaces and Cgroups to isolate and restrict resources. Namespaces in the Linux kernel are used to implement lightweight virtualization services. Processes in the same namespace can sense the changes of each other but are unaware of external processes. Users, hostnames, domain names, semaphores, networks, file systems, and processes in the same namespaces can be isolated. Cgroups can limit and record physical resources (including CPU, memory, and I/O resources) used by task groups to provide a basis for container virtualization. Cgroups provides the following functions:

- Resource limit: Cgroups can limit the total amount of resources used by tasks. For example, the upper limit of the memory used during running can be set.

- Priority allocation: controls the allocation of resource priorities based on the allocated CPU time slice and I/O bandwidth.
- Resource statistics: Cgroups can collect statistics on system resource usage such as CPU usage duration and memory usage.
- Task control: Cgroups can suspend and resume tasks.

Figure 2-14 Docker container architecture



Docker uses the typical C-S architecture. A user uses a Docker client to communicate with a Docker daemon and sends a request to the Docker daemon. The Docker client is used to send a request to the Docker daemon and perform container management operations. It can be a Docker command line tool or a Docker API client. The Docker daemon is the core Docker background process. It responds to requests from the Docker client and translates these requests into system calls to complete container management operations. The Docker image management module is Image Management. It downloads images from the Docker Registry and stores the images in the file system. The Docker network can be a host physical network, virtual bridge network, or overlay network.

2.4.4 Compatibility Between Kubernetes and Docker

Kubernetes v1.20 and later do not support the Docker container engine. However, according to the compatibility issue clarification on the official Kubernetes website:

1. Versions earlier than Kubernetes v1.20 are fully compatible with Docker.
2. Kubernetes v1.20 to v1.22 still support Docker, but alarms are generated when kubelet is being started.
3. Kubernetes v1.23 and later do not support the Docker engine but are still compatible with the original Docker images. In addition, the images generated by running the **docker build** command can be used properly in Kubernetes.
4. Kubernetes v1.23 and later can use the containerd or CRI-O container runtime.

After Kubernetes is incompatible with Docker, the possible countermeasures are as follows:

1. Use the containerd or CRI-O container runtime.
2. Use the Kunpeng iSula container engine. For details, see **iSula Container Engine User Guide**.
3. Use Swarm or MESOS to manage Docker.

3 Key Virtualization Features

3.1 Virtualized CPU Acceleration

3.1.1 Virtualized Topology-Aware Scheduling

Kunpeng BoostKit for Virtualization accelerates CPU scheduling for applications on VMs based on software-hardware collaboration.

- The CPU topology structure is directly passed to the VM through the NUMA awareness and cluster awareness features. The VM OS kernel utilizes the cluster task scheduling optimization to accelerate multi-thread/process calls.
- The lock mechanism during preemption is optimized to improve VM performance in overcommitment scenarios.
- The hardware deadlock mechanism is introduced to prevent VM suspensions and recovery failures caused by hardware deadlocks.

Advantages and Benefits

- The NUMA awareness feature improves the VM memory access performance by 5% to 15%.
- The cluster awareness feature improves the performance of multi-thread applications (such as big data applications) on VMs by 2% to 20%.
- The VM lock optimization feature improves Kunpeng CPU performance in overcommitment scenarios. When an 8-core VM has an overcommitment ratio of 1:2, the UnixBench score can be increased by about 40%.
- The hardware deadlock mechanism prevents an Arm VM from recovery failures after hardware deadlocks.

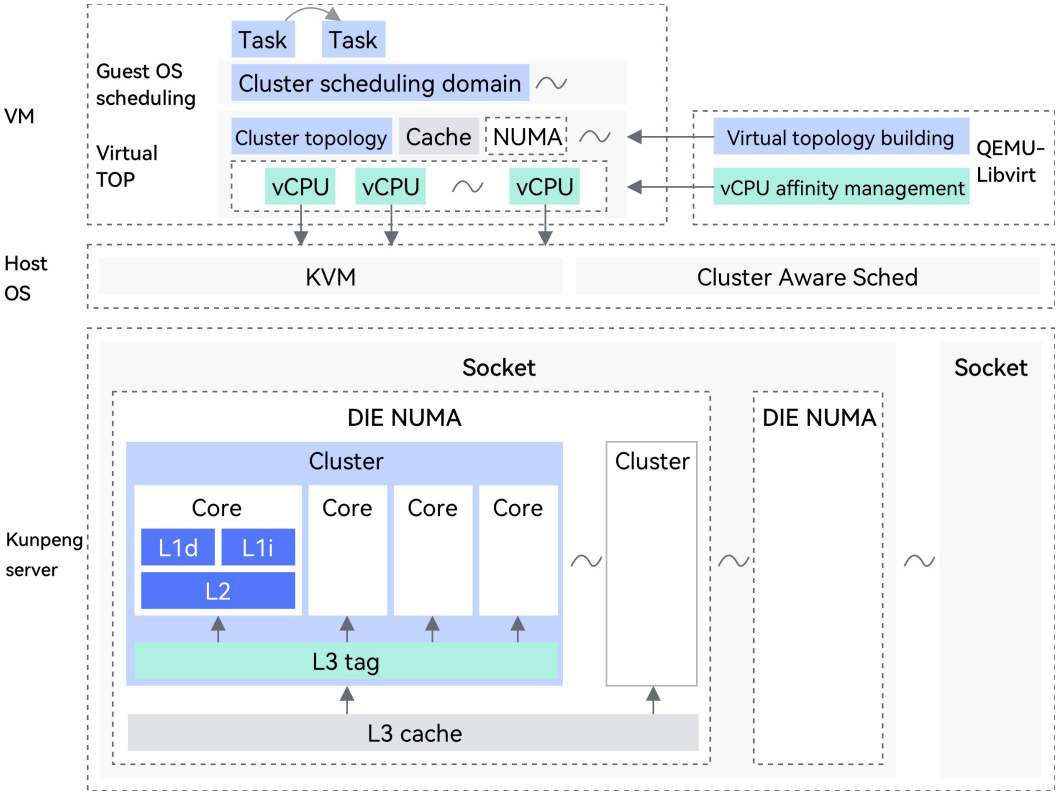
Key Technologies

- NUMA awareness: The NUMA topology is displayed on the VM to optimize the VM's memory access efficiency.
- Cluster awareness: The support for cluster awareness scheduling is added to the OS scheduler. The OS introduces the software-core synergy to improve process scheduling performance. The cluster topology is displayed on the VM to support cluster awareness scheduling.
- Lock optimization: When the VM OS is applying for a lock, the OS uses the shared memory to check whether the vCPU has been preempted by other VMs. If the vCPU has been preempted, the system exits lock wait. If the vCPU has not been preempted, the system enters lock wait.
- Deadlock detection: Performance monitoring interrupt (PMI) is configured to non-maskable interrupt (NMI), and the SDEI watchdog is disabled to trigger a high-priority NMI in the VM. When hard lockups occur on a VM, the exception can be recorded and the VM can be reset.

Application Scope

It is applicable in general Kunpeng virtualization scenarios and CPU overcommitment scenarios.

Figure 3-1 Virtualization scheduling optimization architecture



3.1.1.1 Virtual Cluster Topology Awareness Scheduling

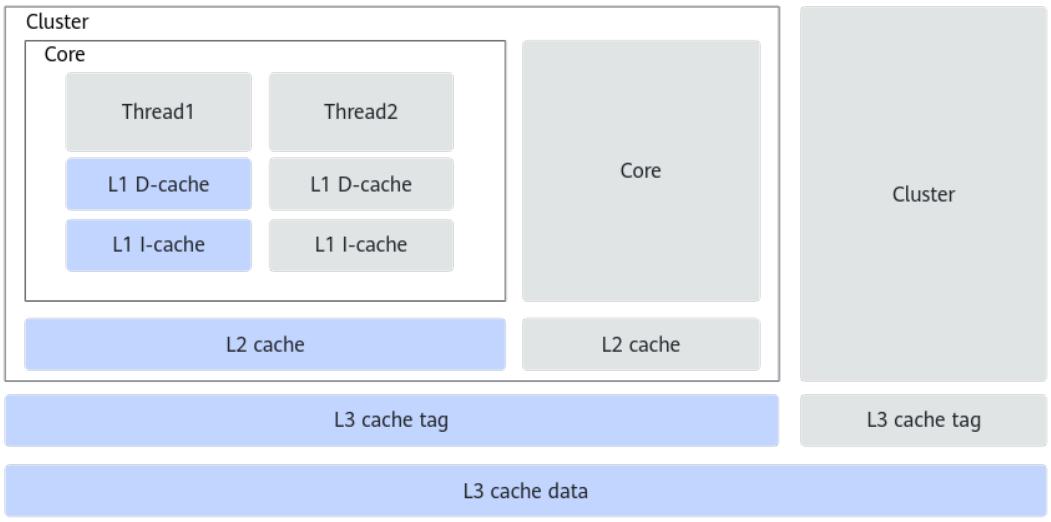
A cluster is a hardware unit of a CPU. Each cluster contains several cores. Cores in a cluster share the same L3 cache tag. By adding the option for cluster task scheduling tuning of the OS kernel, cross-cluster thread scheduling can be

prevented and L3 cache tag resources can be reused, improving the CPU scheduling efficiency and memory bandwidth utilization of multi-thread applications.

It also leverages hardware resources more efficiently, increases the system throughput, and speeds up responses to requests. In various application scenarios or test scenarios, especially in multi-core and multi-threaded application scenarios, enabling cluster scheduling tuning improves the performance by 2% to 20%.

By mapping the topology of a physical CPU cluster to a VM, the tuning effect of the VM system can be as good as that of the physical CPU.

Figure 3-2 Cluster topology awareness scheduling



Constraints

- openEuler 22.03 LTS SP2 or later is supported.
- The topology information of the physical CPU cluster must be correctly mapped to the VM.

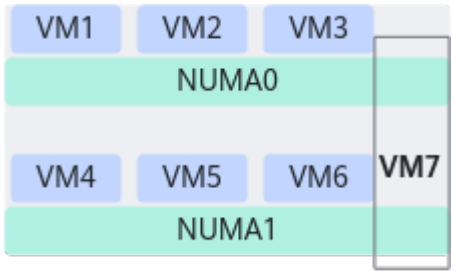
Application Scenarios

Apply to the 1:1 core binding scenario. The optimal cluster topology is displayed based on the topology of vCPUs bound to the physical CPUs.

3.1.1.2 VM NUMA Awareness

Guest NUMA can be configured for VMs. The vCPU NUMA status can be identified in VMs. Service software optimizes memory resource usage based on the guest NUMA topology.

Figure 3-3 NUMA awareness



Constraints

If the service software cannot identify or optimize NUMA, cross-NUMA memory access occurs and the performance deteriorates.

3.1.1.3 VM Lock Virtual-Real Synergy Optimization

Pain Points

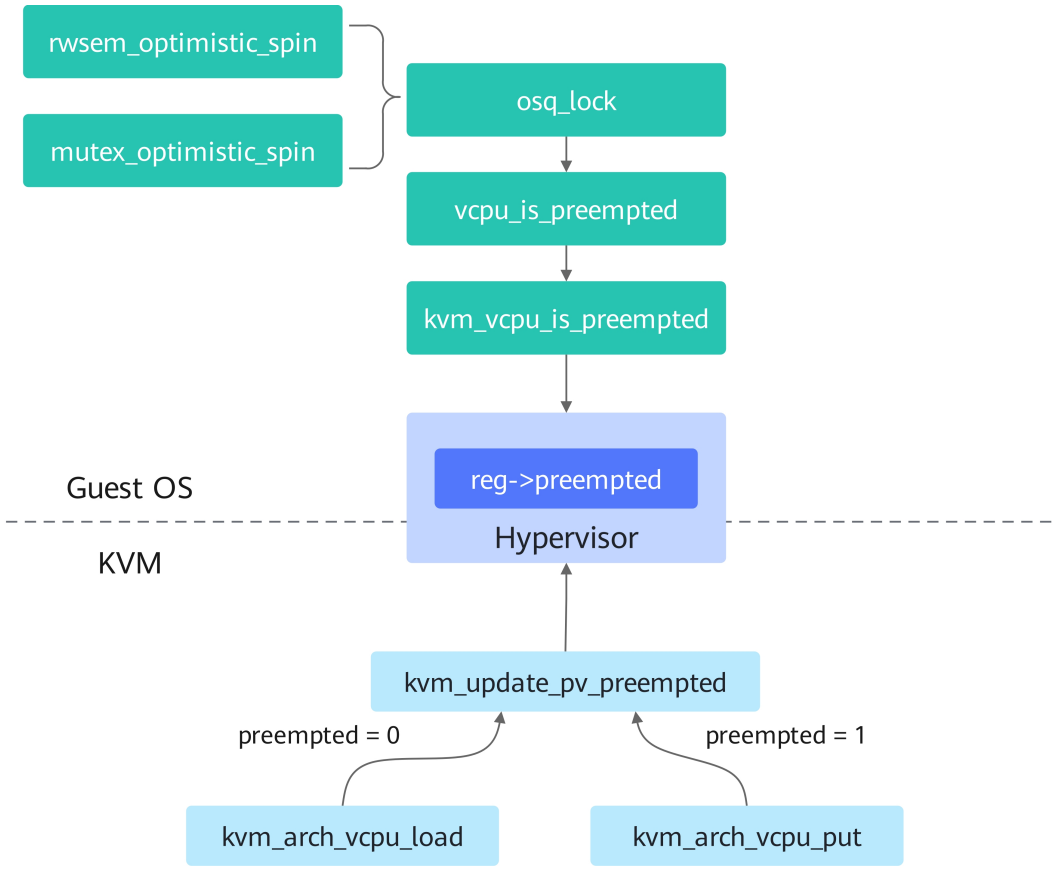
Some locks in the kernel enter the spin wait state during lock preemption. If the vCPU that holds the lock is scheduled by the hypervisor to another task, the vCPUs in the waiting state may waste a lot of time due to spin wait. In the VM overcommitment scenario, this problem significantly affects the overall performance of the VM.

Solution

To resolve this problem, this feature uses shared memory to relay vCPU preemption status from the hypervisor to VMs. In this way, a spinning vCPU aborts its wait state upon detecting that the lock holder has been preempted, avoiding unnecessary wait.

The **preempted** state value tracks whether the vCPU has been preempted on the host. Specifically, a value of **0** indicates that the hypervisor is actively scheduling the vCPU, while **1** signifies the vCPU has been stopped by the hypervisor.

Figure 3-4 Principles of VM lock optimization



Application Scenarios

1. The VM lock optimization feature targets vCPU overcommitment scenarios with core-bound vCPUs, where multiple vCPU threads contend for resources on shared physical cores.
2. Using a coordinated frontend-backend approach, the feature tracks vCPU thread preemption to refine scheduling and lock efficiency. It improves performance in CPU scheduling operations such as **mutex_spin_on_owner**, **mutex_can_spin_on_owner**, **rtmutex_spin_on_owner**, **osq_lock**, and **available_idle_cpu**.

3.1.1.4 VM Deadlock Detection

The NMI Watchdog is a mechanism dedicated to detect hard lockups in Linux. The NMI Watchdog generates non-maskable interrupts (NMIs) to interrupt code execution even in the code segments where the interrupts are disabled in the Linux kernel, ensuring that the system can detect hard lockups. The Arm hardware does not support NMIs. Therefore, the pseudo-NMI technology based on interrupt priority is used. The performance monitoring interrupts (PMIs) are configured as pseudo-NMIs to implement the NMI watchdog function, which is also called PMU Watchdog.

The implementation of NMI monitoring in the VM environment prevents the VM from failing to exit after entering an infinite loop. Otherwise, the management cannot be restored if this situation happens.

The feature is validated on VMs hosted on Kunpeng 920-based servers, running openEuler 22.03 LTS SP2 or later with libvirt 6.2.0 or later and QEMU 6.2.0 or later.

Constraints

- The internal OS of the VM needs to support pseudo-NMI and kernel parameters need to be configured correctly.
- The method of configuring the PMU Watchdog on the VM is the same as that on the host. You do not need to configure the XML file.
- The procedure for configuring the PMU Watchdog on the VM is the same as that on the host. You do not need to modify the VM configuration file (for example, the XML file).
- If the SDEI Watchdog is available, it has a higher priority than the PMU Watchdog. Therefore, you need to disable the SDEI Watchdog before enabling the PMU Watchdog. The VM environment does not support the SDEI Watchdog. Therefore, when configuring kernel parameters, ensure that the SDEI Watchdog has been disabled by configuring kernel parameters.

3.1.1.5 Virtualization Scenario Cache Topology Awareness

In an Arm-based virtual machine (VM) environment, a group of preset values are displayed by default when you view the cache size. These default values cannot accurately reflect the actual cache size used by the VM. In this case, the optimization of applications and OSs in the VM may be affected. To solve this problem, this feature provides a method of specifying the cache size in the VM XML configuration of libvirt or the QEMU command for starting VMs. By using this feature, the size of the cache used by a VM can be accurately reflected, and more accurate cache structure information is available for further optimizing the application performance in the VM.

Constraints

None

Application Scenarios

This feature enhances the accuracy of the cache structure information in VMs and improves the performance of a single service for gaming.

3.1.2 Interrupt Passthrough

This feature eliminates VM exit and entry overhead during interrupt processing based on the Kunpeng hardware virtualization features and the technology of directly injecting software and hardware interrupts into VMs. As the first in the industry to support the GICv4.1 driver, it optimizes the redistributor and distributor virtualization processing mechanism, and supports direct SGI passthrough.

Application scenario: Minimizes interrupt latency and maximizes the throughput of network- or I/O-heavy services.

Table 3-1 lists the interrupt passthrough types that the new 920 processor model supports.

Table 3-1 Interrupt passthrough types

Function	Principle & Effect	Application Scenarios/ Best Practices
GICv4.1 vLPI passthrough	Injects Virtual Function I/O (VFIO) device interrupts into VMs via GICv4.0/4.1, bypassing VM exit/entry.	Reduces interrupt latency and improves throughput for network- or I/O-intensive workloads.
GICv4.1 virtual device interrupt passthrough	Injects virtio device interrupts into VMs via Kunpeng-optimized GICv4.1, bypassing VM exit/entry.	Reduces interrupt latency and improves throughput for network- or I/O-intensive workloads.
GICv4.1 vSGI passthrough	Injects vCPU inter-processor interrupts (IPIs) into VMs via GICv4.1, bypassing VM exit/ entry.	Lowers IPI latency and boosts performance in multi-core IPI-heavy scenarios.

3.1.3 GICv4.1 Overcommitment Optimization

Generic interrupt controller v4.1 (GICv4.1) is an advanced interrupt controller architecture for Arm servers. It supports virtual locality-specific peripheral interrupts (vLPIs), hardware-accelerated virtual machine (VM) interrupt injection, and interrupt priority management, significantly improving interrupt processing efficiency and scalability in virtualization scenarios. When physical machine (PM) resources are limited and VM overcommitment is used, the GIC needs to use the VMOVP instructions to maintain the mapping between vCPUs and physical CPUs. The VMOVP instructions are executed on the GIC hardware in serial mode, which deteriorates the VM service performance. The GICv4.1 overcommitment optimization feature allows the GIC to skip VMOVP instructions when vCPUs are migrated between CPUs that share the same virtual processing element (vPE) table, thus improving VM service performance in overcommitment scenarios.

Constraints

- OS

  openEuler 24.03 LTS SP3 is supported.
- Hardware

  Only the new Kunpeng 920 processor model is supported.

Application Scenarios

In VM overcommitment scenarios where GICv4.1 is enabled, the vCPU ranges of multiple VMs are mapped to the same physical CPU range, improving VM service performance.

Benefits

In VM overcommitment scenarios, the service performance is improved by 10% compared to GICv3 overcommitment.

3.2 Virtualized I/O Acceleration

3.2.1 Hardware Accelerator

3.2.1.1 Virtualized KAE

Kunpeng Accelerator Engine (KAE) is a hardware acceleration solution based on the Kunpeng processor. It includes KAE encryption and decryption as well as KAEzip. Virtualized KAE (vKAE) devices can also enable KAE capabilities in VMs or containers.

The KAE encryption and decryption module uses KAE to implement the RSA, SM3, SM4, DH, MD5, and AES algorithms. It provides high-performance symmetric and asymmetric encryption/decryption algorithms based on the lossless user-mode driver framework. Compatible with OpenSSL 1.1.1a and later versions, it supports the synchronous and asynchronous mechanisms.

For details, see **Developer Guide (KAE Encryption & Decryption)**.

Advantages and Benefits

KAE encryption and decryption and KAEzip are used to accelerate SSL/TLS applications and data compression, respectively. They can significantly reduce processor consumption and improve processor efficiency. In addition, KAE shields the internal implementation details from the application layer. You can quickly migrate services by using the standard OpenSSL and zlib interfaces.

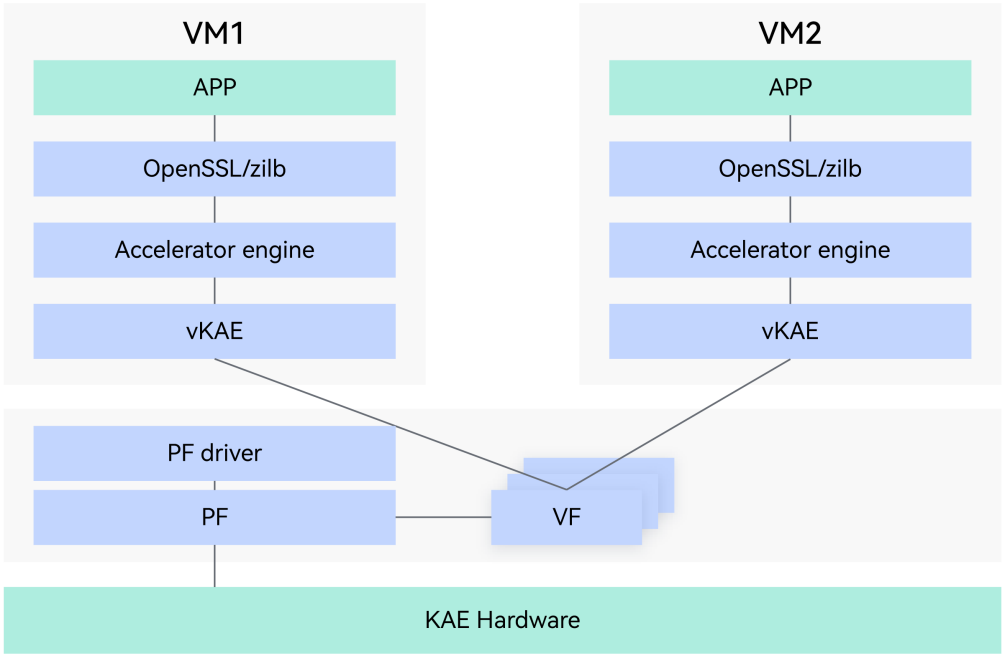
Key Technologies

- The KAE device on the host supports the SR-IOV technology, which enables the host KAE device to be passed through to the VM based on the virtual function (VF).
- The accelerator engine is installed on the VM and is associated with an open ecosystem library to implement acceleration that is transparent to applications.

Application Scope

It can be deployed on Kunpeng VMs and applied to services in which encryption, decryption, and decompression are frequently involved.

Figure 3-5 vKAE architecture



3.2.1.2 KAE Passthrough Live Migration

VM live migration is a critical operational capability that enables seamless transfer of running VMs between physical hosts without service interruption. KAE passthrough live migration specifically addresses the scenario where VMs are configured with KAE passthrough devices, offering enhanced operational flexibility and continuous service availability. The current live migration with passthrough devices relies on device-specific DMA dirty page marking, which many passthrough devices lack, limiting migration capabilities. The SMMU of the new Kunpeng 920 processor model introduces dirty page marking based on Hardware Translation Table Update (HTTU). By combining this hardware feature with an optimized software framework, robust passthrough device migration support is implemented, significantly strengthening competitiveness of Kunpeng virtualization.

Before configuring this feature, learn its specifications, supported version, license requirement, constraints, and application scenarios. For details, see **Virtualization Scenario vKAE Passthrough Live Migration Feature Guide**.

Version Requirements

- Version: For Arm-based KVM and QEMU virtualization platforms, only libvirt 6.2.0/9.10.0 and QEMU 6.2.0/8.2.0 are supported.
- License: KAE license.
- The application environment must meet the software and hardware environment requirements supported by KAE.

Application Scenarios

KAE passthrough live migration is primarily used in scenarios where VMs leverage KAE passthrough devices for compression, encryption/decryption, and other operations while requiring uninterrupted migration capabilities. This ensures seamless load balancing, hardware maintenance, disaster recovery, and high availability.

3.2.2 Virtualization DPU Offload

Virtualization DPU offload is to offload the acceleration software, such as OVS-DPDK and SPDK, to data processing unit (DPU) cards for running.

Advantages and Benefits

Virtualization DPU offload enables software running on the physical machine to be offloaded to DPU cards, reducing the physical machine's CPU load and improving VM density. DPU cards support protocols such as VirtIO-Net and VirtIO-blk. As the back end of the VirtIO device, DPU cards improve the virtualization network and storage performance.

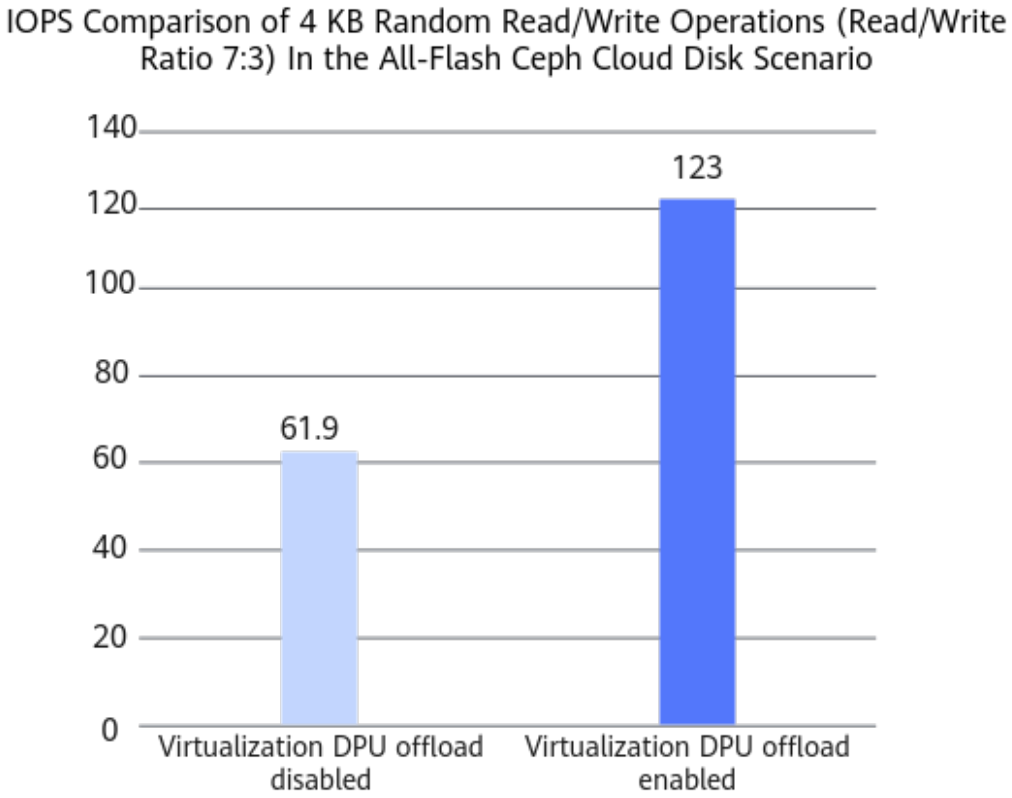
Key Technologies

DPUs can offload OVS-DPDK and SPDK software to the Arm cores. The DPU iNIC functions as the backend of VirtIO-Net and VirtIO-blk. VMs can directly communicate with DPU cards through the user-mode driver, bypassing the kernel-mode protocol stack and improving network and storage data processing capabilities. DPUs can offload management software, such as OpenStack and libvirt, to the Arm cores, facilitating VM management on the host.

Application Scope

It is applicable in supporting network and storage acceleration in virtualization scenarios.

Figure 3-6 Performance comparison before and after using virtualization DPU offload



3.3 Virtualization Management Optimization

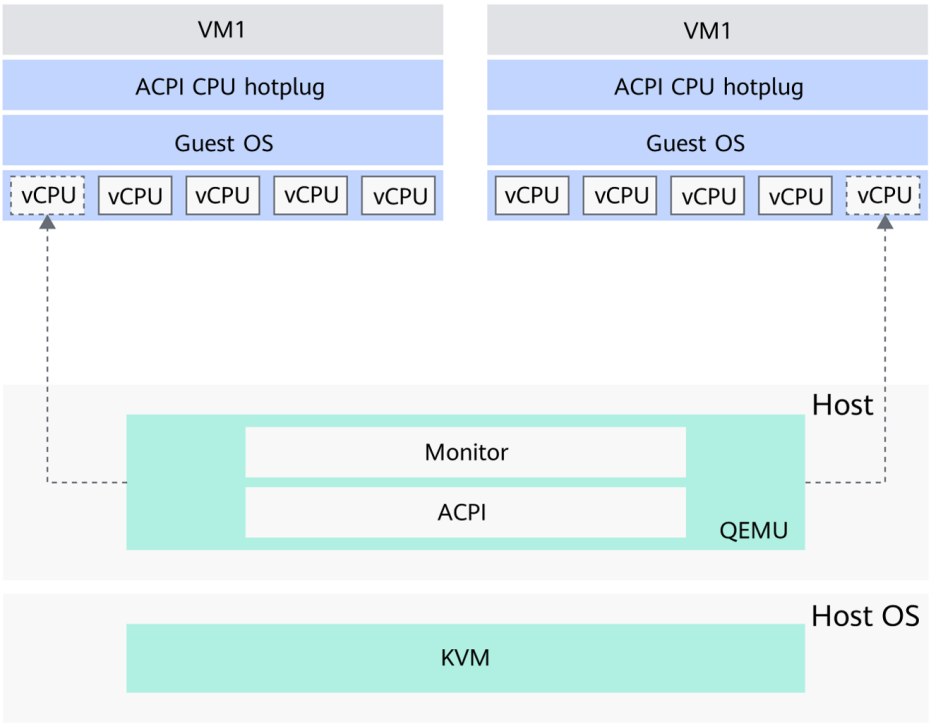
3.3.1 Hotplug

3.3.1.1 VM vCPU Hotplug

In the VM environment, vCPUs comply with the Advanced Configuration and Power Interface (ACPI) standard and implement functions by simulating ACPI GED. When a vCPU hotplug request is initiated in libvirt, QEMU operates the simulated ACPI GED device and sends interrupt signals to the VM. After receiving the interrupt, the ACPI firmware in the VM responds and invokes the corresponding CPU hotplug event handler to implement vCPU hotplug.

This mechanism not only supports online increase of vCPU threads, but also supports online decrease of vCPU threads.

Figure 3-7 VM vCPU hotplug principle



Application Scenarios

This feature applies to scenarios where the service load fluctuates dynamically and the CPU needs to dynamically adjust resources to reduce costs.

Constraints

NOTICE

In openEuler 24.03 and later versions, the CPU hot removal function is added to the AArch64 architecture. However, the new community mainline solution is actually used, which is incompatible with the CPU hotplug protocol of earlier versions. The guest version must match the host version. That is, if the guest version is openEuler 24.03 or later, the host version must be openEuler 24.03 or later as well. If the guest version is earlier than openEuler 24.03, the host version must be earlier than openEuler 24.03. Otherwise, the CPU hotplug function cannot be used.

- For processors using the AArch64 architecture, the specified VM chipset type (machine) needs to be virt-4.2 or a later version when a VM is created.
- For an AArch64 VM, CPUs that exist during the initial startup do not support hotplug.
- When configuring Guest NUMA, you need to configure the vCPUs that belong to the same socket in the same vNode. Otherwise, the VM may be soft locked up after CPU hotplug, which may result in VM panic.

- VMs do not support CPU hotplug during migration, hibernation, wakeup, or snapshot.
- Whether the hot added CPU can automatically go online depends on the VM OS logic rather than the virtualization layer.
- CPU hotplug is restricted by the maximum number of CPUs supported by the hypervisor and guest OS.
- When a VM is being started, stopped, or restarted, the hot added CPU may become invalid. However, the hot added CPU takes effect after the VM is restarted.
- When a VM is being started, stopped, or restarted, the CPU hot removal may time out and fail. Wait until the VM is running properly and try again.
- During vCPU hotplug, if the number of added CPUs is not an integer multiple of the number of cores in the VM CPU topology configuration item, the CPU topology displayed in the VM may be disordered. You are advised to add or remove CPUs whose number is an integer multiple of the number of cores each time.
- If the CPU that is hot added or removed needs to take effect online and is still valid after the VM is restarted, the **--config** and **--live** options need to be transferred to the **virsh setvcpus** API to persist the CPU hotplug operation.

3.3.1.2 QEMU VM Memory Hot Add

QEMU is a cross-platform computing emulator that is quick and open source. It can emulate a variety of hardware architectures and usually works together with libvirt to provide user VMs with emulated hardware operating environment, offering easy management and scheduling of large-scale VMs.

VM memory hotplug is a virtualization technology that dynamically expands the memory capacity of a running VM based on QEMU. However, on the current AArch64 architecture platform, the native QEMU 6.2.0 faces a technical limitation: It cannot start VMs whose initial memory in the NUMA configuration is configured **0**, and cannot provide the memory hotplug function to such VMs. This limitation narrows the QEMU application scenarios.

The Kunpeng BoostKit QEMU VM memory hotplug feature is based on QEMU's existing memory hotplug logic and extends functions through patches. The feature is open-sourced and released on Gitee. This feature makes the VM's XML configuration file contain a NUMA node with 0 initial memory and dynamically adds memory to the NUMA node using memory hotplug commands. This improvement greatly widens the QEMU application scenarios.

Application Scenarios

In large-scale VMs and centralized management scenarios, especially on cloud computing platforms, NUMA nodes that are temporarily allocated with 0 memory are often reserved in VMs in advance, aiming to efficiently manage large-scale VMs. This method ensures that memory resources can be quickly increased and allocated when more memory is required, facilitating dynamic memory expansion.

Constraints

- OS

openEuler 22.03 LTS SP4 is supported.

- VM specification

  Among the NUMA nodes configured for a VM, a maximum of one NUMA node whose initial memory is 0 is supported.

- Hotplug specification

  The VM XML configuration file must contain the **maxMemory** node as required by QEMU. This node specifies the number of memory slots (setting the slots attributes) dedicated for VM's memory hotplug and the maximum total memory that can be achieved through hotplug (setting the value of **maxMemory**).

NOTICE

In addition to the constraints in the VM XML configuration file, some configuration parameters in physical machines also put constraints on the maximum number of hotplug operations on the corresponding VMs and the maximum memory capacity provided by memory hotplug. The specific constraints depend on the OS type.

3.3.2 Virtualization Scenario KAE-Accelerated Live Migration

VM live migration allows a VM to be migrated from one physical host to another without interrupting the VM running. To reduce the data volume transmitted during migration, compression technologies (for example, zlib library) are usually used on the source physical host to compress memory pages before transmission, and then the memory pages are decompressed on the target physical host, thereby speeding up the VM live migration.

The Kunpeng Accelerator Engine (KAE) is a hardware acceleration solution based on the new Kunpeng 920 processor model. It includes the compression module KAEzip, which can significantly reduce processor consumption and improve processor efficiency. KAEzlib is a standard zlib interface provided by the KAE compression module. It uses the Kunpeng hardware acceleration module to implement the Deflate algorithm and works with the lossless user-mode driver framework. Therefore, KAE can replace the open-source compression library zlib to accelerate VM live migration.

Before configuring the KAE-accelerated VM live migration feature, learn the specifications, supported version, license requirement, constraints, and application scenarios of this feature. For details, see **Virtualization Scenario KAE Accelerated Live Migration Feature Guide**.

Specifications

Supported VM specifications include but are not limited to 2 vCPUs with 8 GB memory, 4 vCPUs with 8 GB memory, 4 vCPUs with 16 GB memory, 8 vCPUs with 16 GB memory, 16 vCPUs with 32 GB memory, and 32 vCPUs with 64 GB memory.

NOTICE

When the open-source compression library zlib is enabled during the Redis pressure test, live migration is restricted in scenarios with the following VM specifications and number of live migration threads:

- VMs of 2 vCPUs and 8 GB, and live migration threads fewer than 32.
- VMs of 4 vCPUs and 8 GB, and live migration threads fewer than or equal to 4.
- VMs of 4 vCPUs and 16 GB, and live migration threads fewer than or equal to 4.
- VMs of 8 vCPUs and 16 GB, 16 vCPUs and 32 GB, and 32 vCPUs and 64 GB, and live migration threads fewer than or equal to 3.

Hardware Constraints

Only the new Kunpeng 920 processor model is supported.

Version Requirements

- Version: For Arm-based KVM and QEMU virtualization platforms, only libvirt 10.0.0 and later versions, and QEMU 6.2.0 are supported.
- License: KAE license.
- The application environment must meet the software and hardware environment requirements supported by KAE.

Application Scenarios

VM live migration applies to load balancing, hardware maintenance, and disaster recovery (DR) high availability (HA) scenarios. It dynamically adjusts VM distribution to prevent a single physical host from being overloaded and improve resource utilization. VMs can be migrated without interrupting services to maintain or upgrade the source host. In addition, when a host is faulty or its performance deteriorates, VMs can be quickly migrated to ensure service continuity.

3.3.3 PMU Virtualization

The Performance Monitoring Unit (PMU) is a hardware performance monitoring system on Arm platforms, designed to track and quantify various microarchitecture events. With PMU virtualization enabled, users can leverage the perf tool to monitor PMU events within VMs, facilitating comprehensive performance analysis and optimization.

Specifications

Supported VM specifications include but are not limited to 2 vCPUs with 8 GB memory, 4 vCPUs with 8 GB memory, 4 vCPUs with 16 GB memory, 8 vCPUs with 16 GB memory, 16 vCPUs with 32 GB memory, and 32 vCPUs with 64 GB memory.

Version Requirements

- Physical machine: openEuler 22.03 LTS SP4 or openEuler 24.03 LTS SP1.
- VM: openEuler 22.03 LTS SP4 or openEuler 24.03 LTS SP1.
- License: none.

Application Scenarios

This feature enables PMU events to be collected on a VM, which helps analyze and optimize the performance of the VM OS and service software.

3.3.4 MPAM-enabled libvirt

VMs running on the same NUMA node share public resources, such as memory bandwidth and cache lines. When the shared resources are insufficient, resource contention may occur among VMs, causing performance deterioration. Customers want to limit the VM resource usage by configuring the bandwidth and cache upper limits. This can prevent interference between multiple tenants and improve the availability of cloud hosts.

Specifications

Supported VM specifications include but are not limited to 2 vCPUs with 8 GB memory, 4 vCPUs with 8 GB memory, 4 vCPUs with 16 GB memory, 8 vCPUs with 16 GB memory, 16 vCPUs with 32 GB memory, and 32 vCPUs with 64 GB memory.

Version Requirements

- Version: Only openEuler 24.03 LTS SP2 and later, libvirt 9.10.0, and QEMU 8.2.0 are supported.
- License: none.

Application Scenarios

This feature allows you to configure the XML file to limit the memory bandwidth and cache lines used by a VM. MPAM is used to allocate VMs to different bandwidth/cache partitions, implementing rate limiting or priority control. This ensures stable performance (such as real-time services) of high-priority VMs and prevents low-priority VMs from affecting key loads.

3.3.5 Cross-Generation VM Live Migration

VM live migration allows a VM to be migrated from one physical host to another without interrupting the VM running. The key benefit of cross-generation migration lies in its ability to maintain VM operation and service continuity during hardware upgrades by supporting migration between different hardware generations. Successful implementation typically requires sufficient compatibility between source and target hardware platforms, and hardware discrepancy tolerance at the VM OS and application layers. While technically challenging, this capability significantly enhances operational flexibility for data centers and cloud service providers by ensuring continuous service availability.

Specifications

Supported VM specifications include but are not limited to 2 vCPUs with 8 GB memory, 4 vCPUs with 8 GB memory, 4 vCPUs with 16 GB memory, 8 vCPUs with 16 GB memory, 16 vCPUs with 32 GB memory, and 32 vCPUs with 64 GB memory.

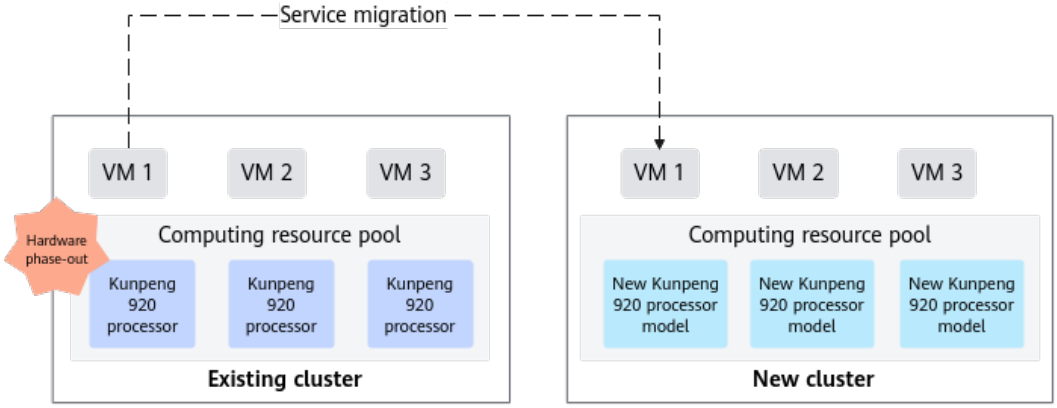
Version Requirements

- Version: Only kernel 6.6 and QEMU 8.2.0 are supported.
- License: none.

Application Scenarios

Cross-generation VM live migration applies to online service migration in hardware maintenance, disaster recovery (DR) for high availability (HA), and hardware upgrade. It performs memory dirty page replication and device status transfer on running VMs so that the VMs can be migrated from older-generation servers to next-generation servers without stopping the VMs. Enterprises can migrate services on old physical machines in batches during off-peak hours at night, achieving seamless hardware phase-out, rack replacement, and computing power upgrade in a short cutover window.

Figure 3-8 Cross-generation VM live migration



General Constraints

- VMs running on the Kunpeng 920 processor can only be unidirectionally migrated to the new Kunpeng 920 processor model.
- Cross-generation live migration requires disabling features that are not included in the minimum feature set, as they may cause performance deterioration. The minimum feature set includes features supported by both generations of processors.
- Cross-generation live migration applies only to VMs configured with features in the minimum feature set. As such, running VMs (usually configured with complete chip features) need to be restarted with features only in the minimum feature set enabled, while unstarted VMs can be directly started with features only in the minimum feature set enabled.
- All functions of cross-generation live migration are developed based on kernel 6.6 and QEMU 8.2.0. You need to upgrade the host kernel and QEMU to the

corresponding versions and apply the cross-generation live migration function patch.

- Customers need to upgrade the guest kernel to include the Errata live migration framework.
- The Syscount frequency must be consistent between processors. The value for the Kunpeng 920 processor is 100 MHz, while that for the new Kunpeng 920 processor model is 100 MHz or 1 GHz. Therefore, you need to set the frequency to 100 MHz in the BIOS menu.
- Only GICv3 and GICv4.1 support live migration, and live migration can be performed only when the same GIC version is used. When GICv3 is used, the BIOS GIC version is not restricted but the source and destination must use the same version, and GICv4.0-related content must be disabled in the OS boot option. When GICv4.1 is used, GICv4.1 must be enabled in the BIOS and the OS boot option.
- Whether VF passthrough devices support cross-generation live migration depends on the devices themselves.
- The functional compatibility of features in the minimum feature set between different generations depends on the actual verification.
- PMU migration is not supported.
- Do not directly migrate the programs compiled on a later processor model to earlier ones.

3.3.6 VM Single-Core and Single-Page Exception Handling

To ensure long-term stable running of Internet systems, Reliability, Availability, and Serviceability (RAS) capabilities are required to obtain the faulty hardware of the current physical machine and process the corresponding hardware information, minimizing the impact scope. After the single-core and single-page exception handling feature is enabled, single-core corrected errors (CEs) can be isolated online on Kunpeng servers without affecting service running; uncorrected errors (UEs) in a single page of memory affect only one process in a VM, preventing the VM from going offline.

Specifications

Supported VM specifications include but are not limited to 2 vCPUs with 8 GB memory, 4 vCPUs with 8 GB memory, 4 vCPUs with 16 GB memory, 8 vCPUs with 16 GB memory, 16 vCPUs with 32 GB memory, and 32 vCPUs with 64 GB memory.

Version Requirements

- Versions: openEuler 24.03 LTS SP3, QEMU 8.2.0, and libvirt 9.10.0-26.oe2403sp3 or later
- License: none.

Constraints

- The application environment must meet the hardware and software requirements.
- The single-page memory error handling is supported only when physical machines use 4 KB pages.

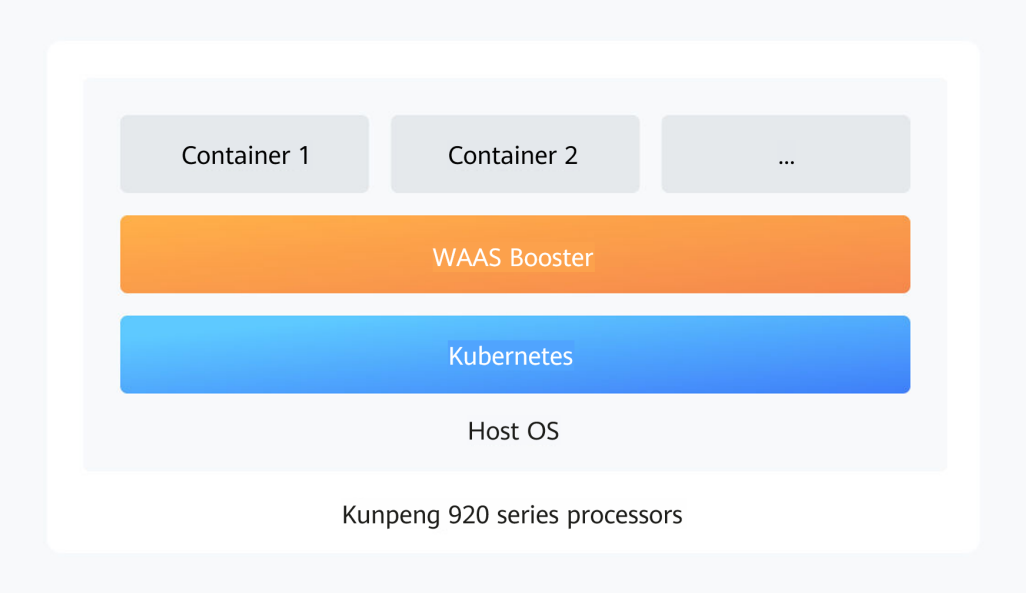
Application Scenarios

In common cloud computing scenarios, VMs can be recovered without interrupting other services when memory hardware errors occur.

3.4 General Optimization

3.4.1 Workload Aware Acceleration System Booster (WAAS Booster)

Cloud vendors want to use container overcommitment to improve the efficiency of container systems and maximize the resource utilization of server nodes without affecting the performance of services running in containers. However, the resource usage of some services is unstable and unpredictable, and the preset maximum resource usage cannot meet the actual service requirements. As a result, automatic tuning is expected for such container services so that idle resources can be allocated to containers with resource needs. This allows containers to temporarily obtain resources beyond the preset maximum value, improving service performance and resource utilization efficiency.



Benefits

In container overcommitment scenarios, the CPU requirements of some containers may exceed the preset maximum value. WAAS Booster detects and analyzes container loads in real time, and responds to CPU resource insufficiency caused by burst loads in a timely manner. In addition, it optimizes resource scheduling through load awareness and prediction.

Key Technologies

WAAS Booster has a built-in real-time load awareness module and long- and short-period load prediction module. It dynamically detects and learns container

service loads and adjusts container resources in real time. Based on the global resource management module, it ensures proper cluster resource allocation and improves the overall container service performance in overcommitment scenarios.

Application Scope

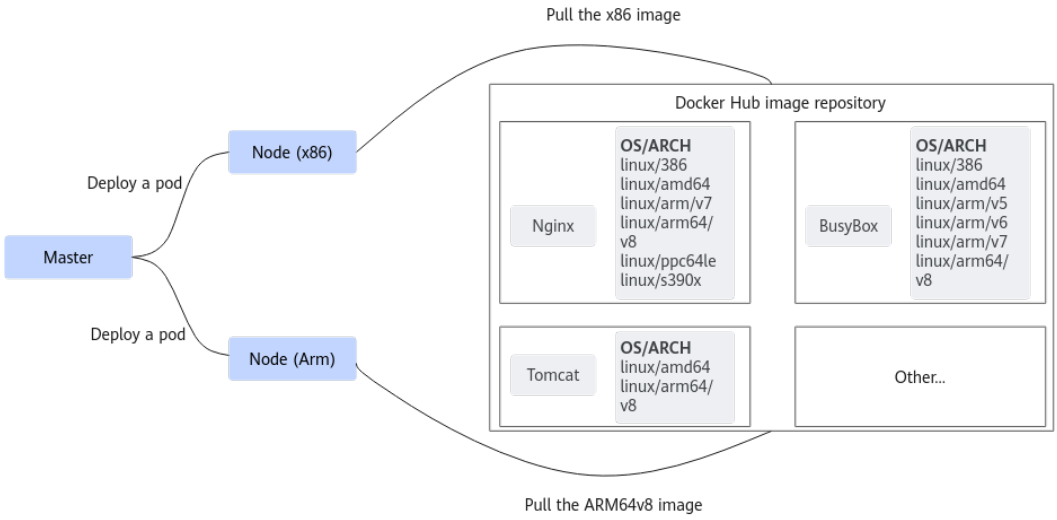
WAAS Booster is only available for tuning containers on servers equipped with Kunpeng 920 series processors. The system parameter configuration applies to only online application services deployed in containers. WAAS Booster is not responsible for any impact on other applications. After the tuning task of WAAS Booster is stopped, the container parameters are restored to the status before tuning.

3.5 Open Source Enablement

3.5.1 Hybrid Deployment on Kubernetes

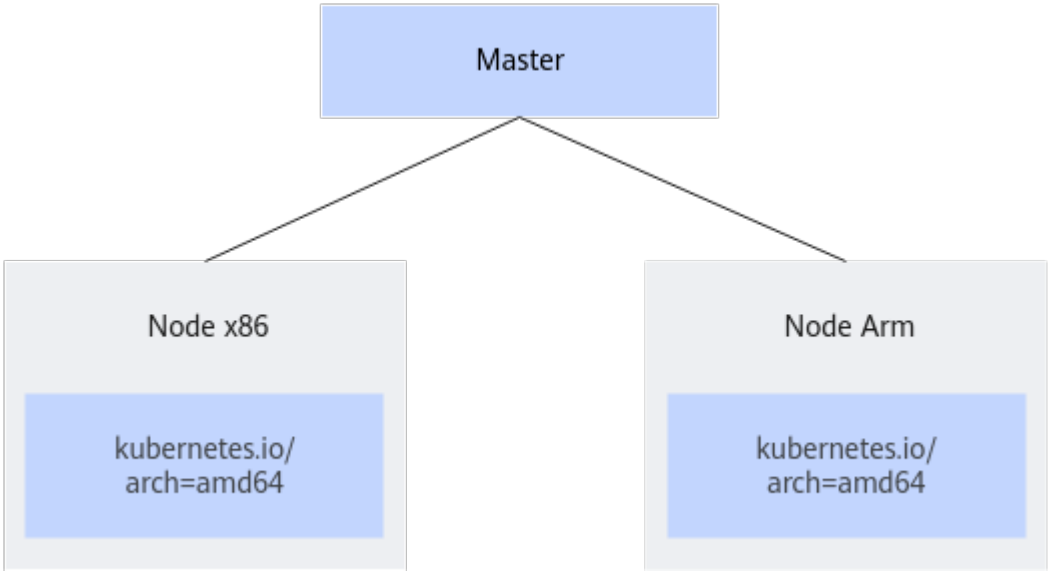
Kubernetes supports hybrid deployment of multiple architectures without modifying code. In addition, the Docker registry supports multi-architecture image pulling. You can pull required service images, such as Nginx, Tomcat, and BusyBox, from Docker Hub based on your architecture. For details about the Kubernetes architecture and components, see **2.4.1 Architecture**.

Figure 3-9 Hybrid deployment 1



Kubernetes provides the flexible Label+nodeSelector mechanism to support directional scheduling of pods based on specific labels. You can use the architecture-aware labels provided by Kubernetes and the **nodeSelector** field to implement directional scheduling of an architecture.

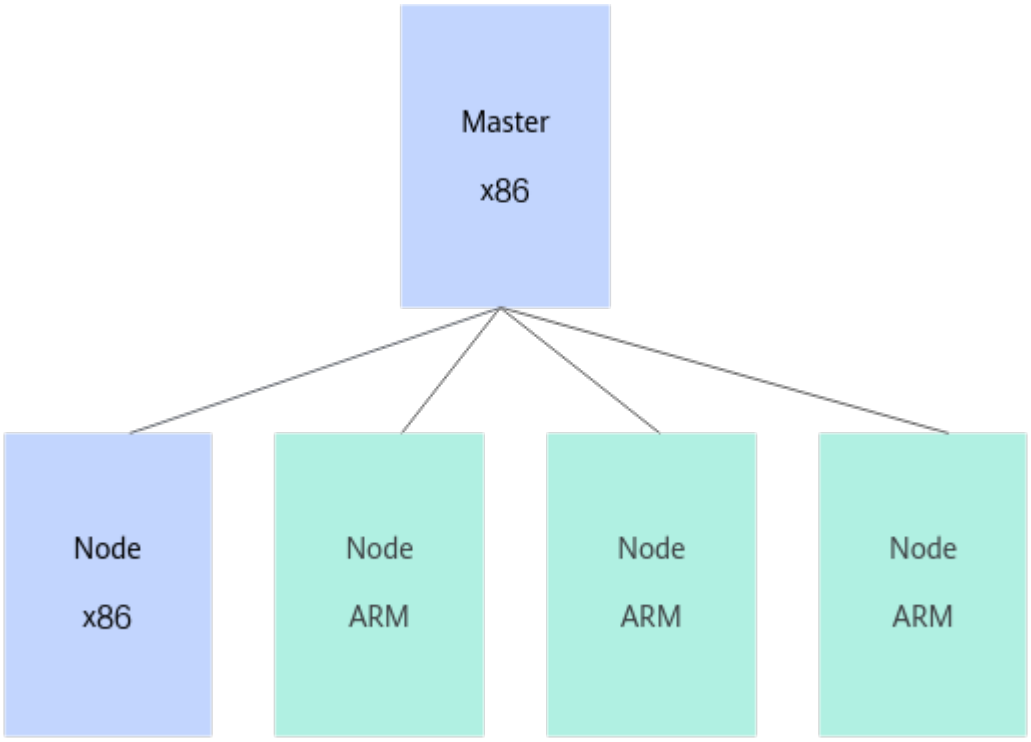
Figure 3-10 Hybrid deployment 2



Scale-out Scenario

In this scenario, Arm servers are added to the current x86 cluster to expand the cluster capacity.

Figure 3-11 Scale-out scenario



Main features in scale-out scenarios:

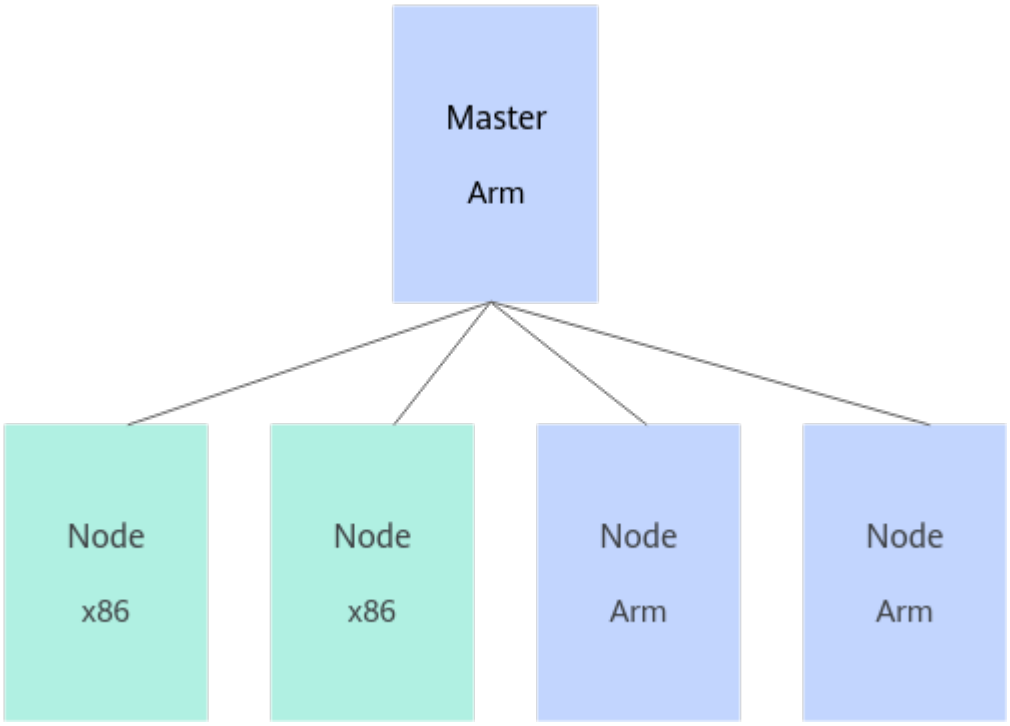
- Arm nodes can be added to a cluster.

- Arm images can be provisioned to Arm nodes.
- On an Arm node, containers in the same pod can communicate with each other.
- Containers in different pods on the same Arm node can communicate with each other.
- Containers in different pods on different Arm nodes can communicate with each other.
- On an Arm node, the communication between pods and services is normal.
- On an Arm node, services can be accessed from outside the cluster.
- Containers on Arm nodes can read and write volumes of the HostPath type.
- Containers on Arm nodes can read and write persistent volumes of the Ceph type.

New Deployment Scenario

Create a cluster and select an Arm server as the master node.

Figure 3-12 New deployment scenario



Main features in new deployment scenarios:

- Arm and x86 nodes can be added to a cluster.
- x86 images can be provisioned to x86 nodes.
- Arm images can be provisioned to Arm nodes.
- On an Arm node, containers in the same pod can communicate with each other.
- Containers in different pods on the same Arm node can communicate with each other.

- Containers in different pods on different Arm nodes can communicate with each other.
- On an Arm node, the communication between pods and services is normal.
- On an Arm node, services can be accessed from outside the cluster.
- Containers on Arm nodes can read and write volumes of the HostPath type.
- Containers on Arm nodes can read and write persistent volumes of the Ceph type.

3.5.2 Hybrid Deployment on OpenStack

- Hybrid deployment of VMs

  The OpenStack availability zones (AZs) are used to divide resources in the computing, network, and storage zones to manage VMs by AZ. Specifically, the image management service distinguishes x86 and Arm images based on their architecture attributes by using Glance. The computing service creates different compute resource pools through the compute AZ. The storage service connects to different storage AZs through different pools in the same Ceph cluster or connects to different storage AZs through different Ceph clusters, implementing management of independent storage services. The network service is divided into AZs through Neutron nodes. Each AZ provides network services independently.
- Hybrid deployment of BMSs

  A series of BMSs are used to manage and configure the x86 and Arm physical machines.

Hybrid Deployment Example

- Hybrid deployment of VMs
  - The controller node manages the entire OpenStack cluster and is the entry for all management operations.
  - The x86-compute node functions as both the x86 AZ network node and x86 compute node in the hybrid deployment scenario. This node provides network functions for all x86 compute nodes in the x86 AZ.
  - The arm-compute node functions as both the Arm AZ network node and Arm compute node in the hybrid deployment scenario. This node provides network functions for all Arm compute nodes in the Arm AZ.
  - Three Ceph nodes (ceph1, ceph2, and ceph3) provide backend block storage for the OpenStack cluster deployed in hybrid mode. Storage pools are created to provide storage services for different AZs.
- Hybrid deployment of BMSs
  - The controller node manages the entire OpenStack cluster and is the entry for all OpenStack management operations.
  - The baremetal node is the entry for all BMS management operations. BMSs reuse the network service of hybrid VM deployment to install and deploy BMS instances.

Figure 3-13 Cluster networking

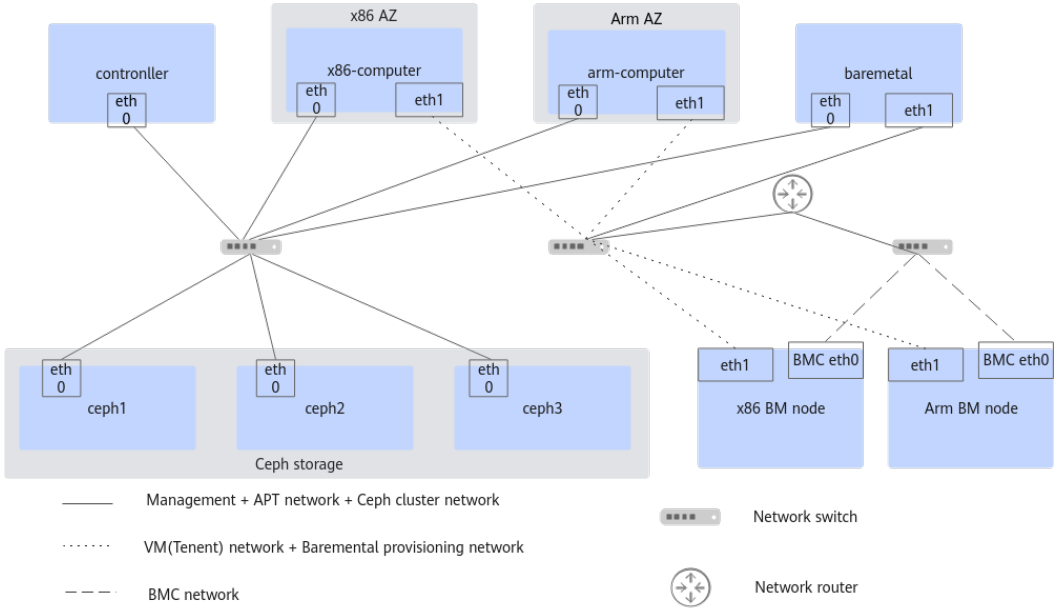


Table 3-2 Cluster nodes

Node Name	Description
controller	Used to deploy the controller node and Ceph client node together.
x86-compute	Used to deploy the x86 AZ network node, compute node, and Ceph client node together.
arm-compute	Used to deploy the Arm AZ network node, compute node, and Ceph client node together.
baremetal	BMS management node, which is an OpenStack compute node. • The eth0 network port is used for communication between OpenStack and other management services such as Keystone and MySQL. • The eth1 network port is used for the ironic-api and ironic-conductor services.
ceph1	Ceph storage node 1.
ceph2	Ceph storage node 2.
ceph3	Ceph storage node 3.

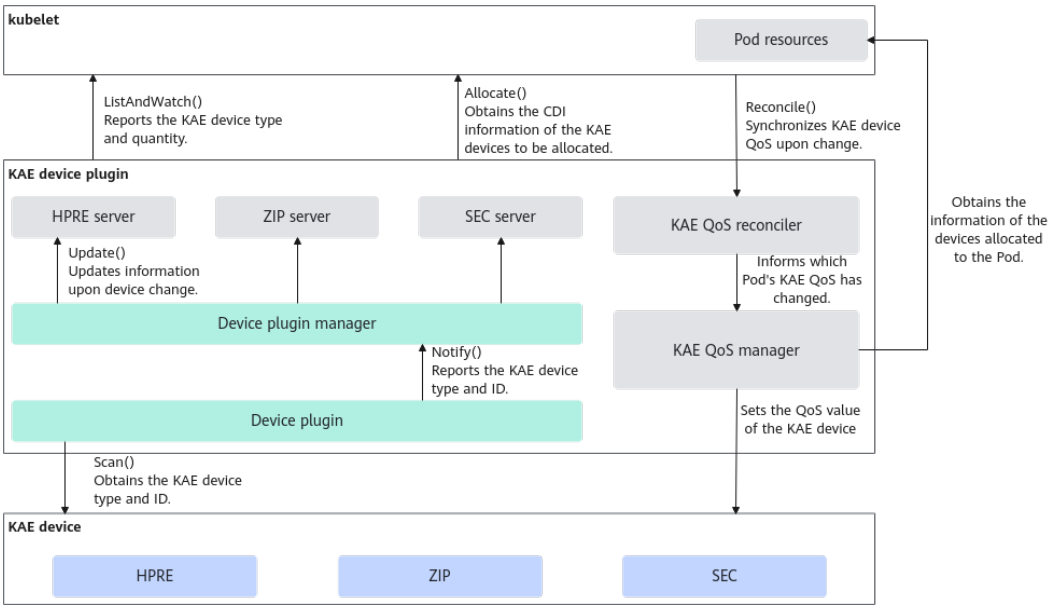
4 Key Container Features

4.1 Cloud Native Infrastructure

4.1.1 KAE Device Plugin

The Kunpeng Accelerator Engine (KAE) is a hardware-based acceleration solution built on Kunpeng 920 series processors. It supports encryption, decryption, and decompression. The KAE encryption and decryption module accelerates Secure Sockets Layer (SSL) and Transport Layer Security (TLS) applications. The KAE decompression modules accelerate data compression and decompression, greatly reducing processor consumption and improving efficiency. In addition, KAE abstracts the internal processing details from the application layer. You can quickly migrate services by using the standard OpenSSL, Tongsuo, BoringSSL, zlib, zstd, and LZ4 interfaces. This plugin automatically manages all KAE devices on the server and streamlines the KAE device passthrough operations. Using this plugin, you can pass through KAE devices to containers through simple statements to accelerate encryption, decryption, and data compression, and set QoS for KAE devices.

Figure 4-1 KAE device plugin architecture



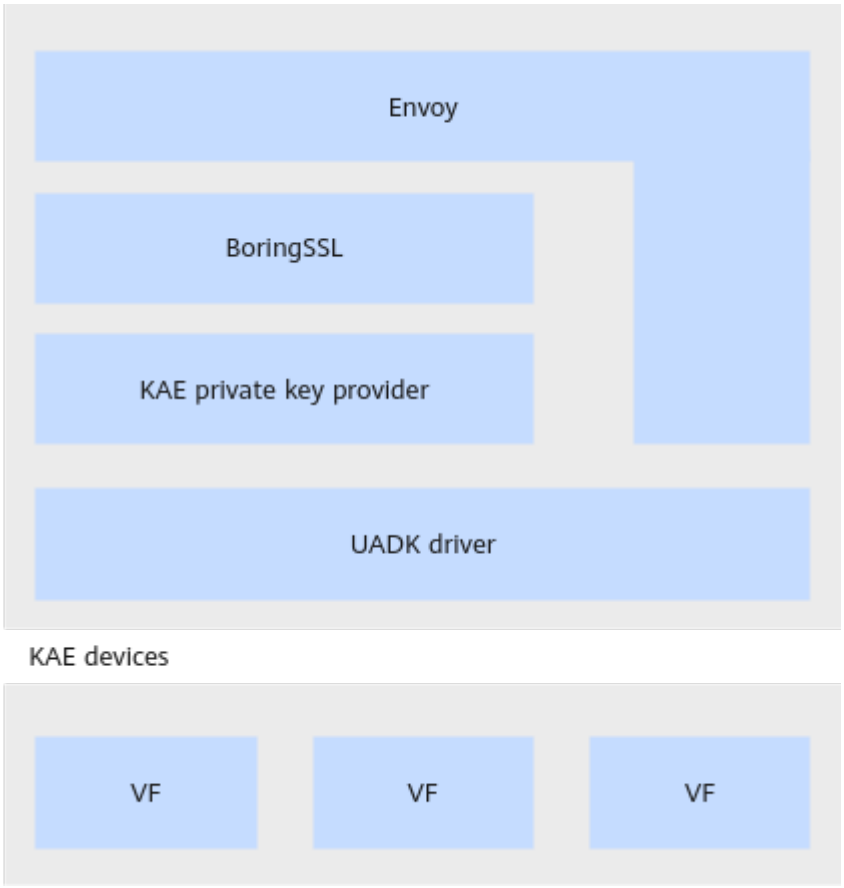
Constraints

Before using the KAE device plugin, ensure that the VF of the KAE device has been created.

4.1.2 KAE-enabled Envoy Acceleration

Envoy is a high-performance L7 (application layer) proxy and communication bus designed for modern cloud native architectures. It is widely used as a data-plane component in service meshes, typically operating in sidecar mode alongside each service. Envoy transparently handles inbound and outbound traffic between services, enabling core capabilities such as traffic management, security, and observability. With support for dynamic configuration, it serves as key infrastructure for service mesh technologies like Istio. In microservice scenarios, Envoy needs to process a large number of TLS requests, regardless of whether it functions as an ingress gateway or a microservice proxy. Especially in the handshake phase, asymmetric encryption and decryption consume a large number of CPU resources. If microservices are deployed on a large scale, this overhead may become a system performance bottleneck. To address this, the KAE private key provider of Envoy offloads time-consuming encryption and decryption operations from the CPU to KAE. This accelerates encryption and decryption while releasing CPU computing power for other service workloads.

Figure 4-2 KAE-enabled Envoy acceleration architecture

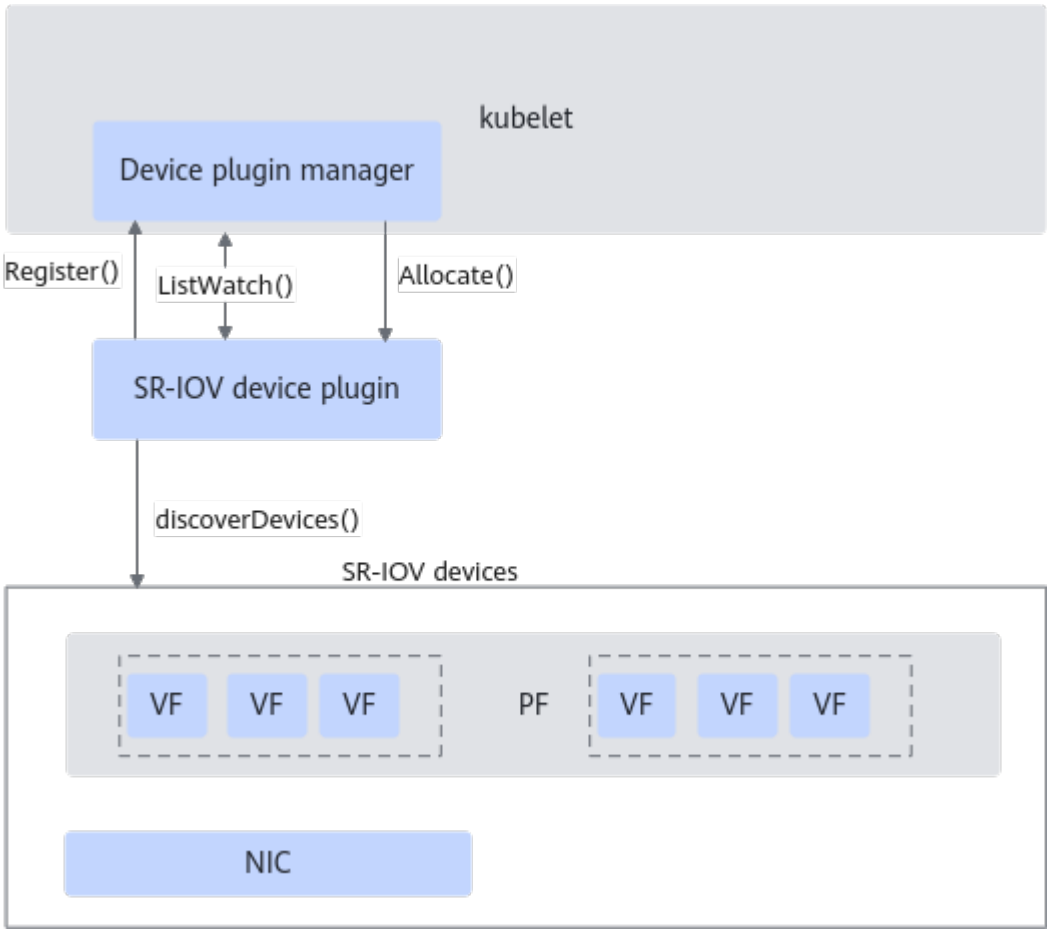


4.2 Cloud Native High-Performance Networking

4.2.1 Kubernetes SR-IOV Device Plugin

The SR-IOV technology virtualizes a single physical PCIe device into multiple virtual PCIe devices, enabling one physical device to support multiple VMs and containers. In Kubernetes environments, direct SR-IOV device usage traditionally requires manual container mounting, which is a cumbersome process for cloud providers managing numerous server containers, as each container needs individual mounting configurations. To address this challenge, the Kubernetes SR-IOV device plugin is introduced. This solution automatically detects and manages node SR-IOV devices and handles the mounting process according to device types specified in user configurations, thereby significantly simplifying operations. The plugin currently supports NICs for container passthrough, while maintaining compatibility with both SR-IOV CNI and Bond-CNI networks.

Figure 4-3 SR-IOV device plugin architecture



Specifications

The SR-IOV device plugin supports servers with Kunpeng 920 series processors, enabling automatic detection and management of SR-IOV devices in Kubernetes clusters.

4.3 Cloud Native Resource Affinity and Isolation

4.3.1 Kunpeng Topology Affinity Plugin

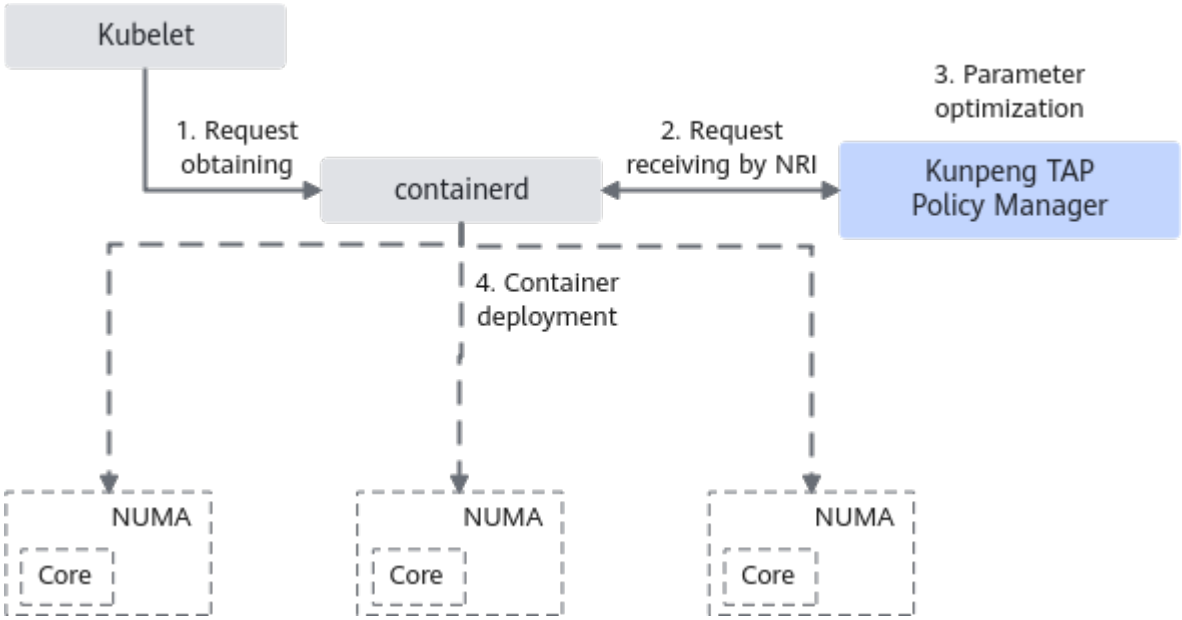
Currently, most cloud service providers need to deploy a large number of containers on a server, and the number of deployed containers even exceeds the number of physical cores of the server. In this scenario, the Kunpeng Topology Affinity Plugin (Kunpeng TAP) can automatically adjust the pod deployment range based on the node resource usage to ensure NUMA affinity for container resources like CPUs and GPUs and prevent cross-NUMA memory access of service containers.

The Kunpeng TAP runs at the node layer of the Kubernetes cluster. It is deployed through pods that use DaemonSet (Daemon pods), which ensures a plugin instance on each node. This plugin interconnects with the NRI of containerd to

dynamically adjust the CPU scheduling range of a pod, securing effective NUMA affinity.

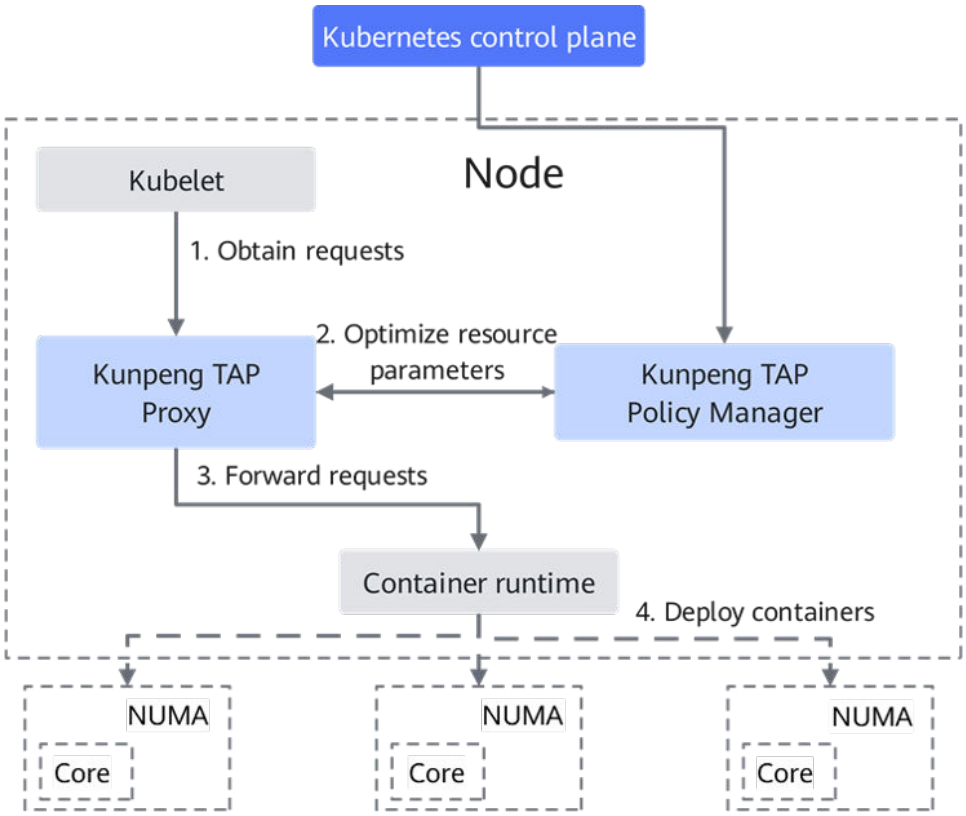
The following figure shows how the NUMA affinity scheduling plugin works based on the NRI function of the containerd runtime.

Figure 4-4 Working principle of the Kunpeng TAP in NRI mode



In addition, to be compatible with Docker and containerd in earlier versions of Kubernetes clusters, the Kunpeng TAP can obtain container requests and return responses as a proxy, and optimize and adjust resource parameters during request delivery to implement NUMA affinity. The following figure shows how the Kunpeng TAP works.

Figure 4-5 Working principle of the Kunpeng TAP



Constraints

The Kunpeng TAP has different usage restrictions and version requirements for the containerd and Docker scenarios. For details, see **Table 4-1**.

Table 4-1 Constraints

Plugin Type	Version Requirement			
	Kubernetes	containerd	Docker	Remarks
Kunpeng TAP (containerd, with NRI enabled)	1.28.4	1.7.0 or later	-	In the containerd scenario, use containerd as the container runtime.
Kunpeng TAP (Docker/containerd)	1.23.6	1.6.8+	20.10.14	In the Docker scenario, use Dockershim as the runtime communication component. containerd is also supported.

4.3.2 Kubernetes MPAM Access and Memory Isolation and Control Plugin

Memory System Resource Partitioning and Monitoring (MPAM) is an Arm64-based technology of isolating and monitoring access and memory resources. It solves system-wide or application-specific performance deterioration due to contention for shared resources in a server system that runs diverse types of services concurrently.

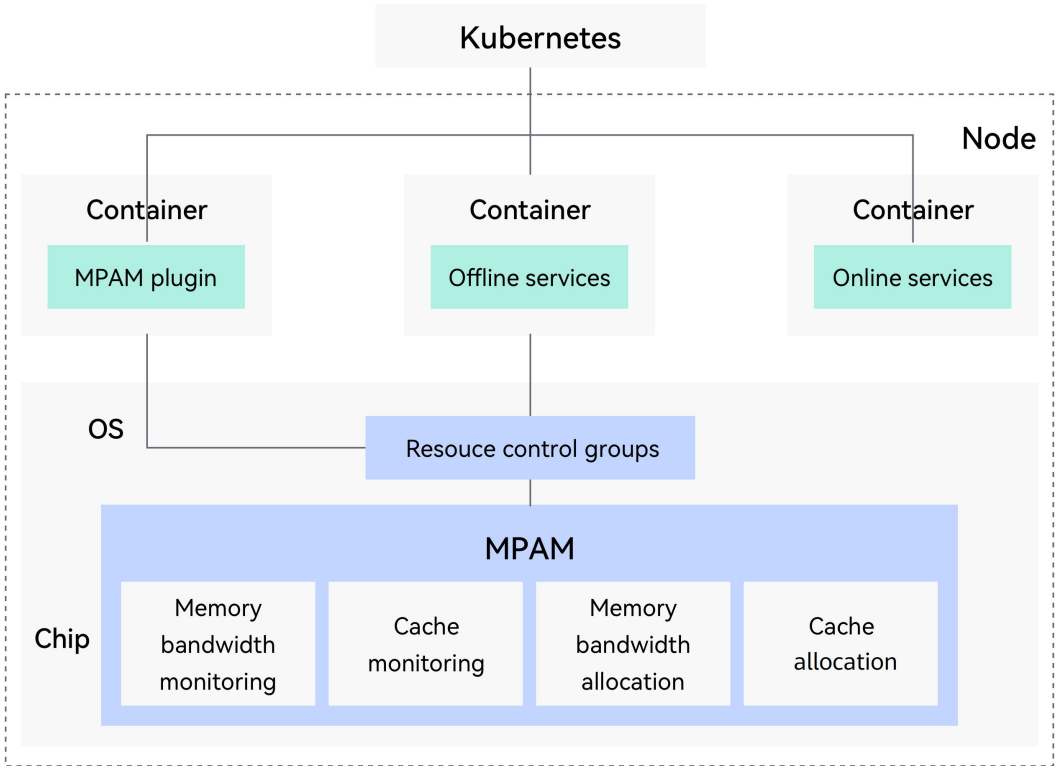
MPAM helps to restrict the memory bandwidth and L3 cache capacity occupied by offline services to prevent offline services from affecting the performance of real-time services.

- The MPAM plugin can be deployed on each compute node to configure resource groups in the YAML file. The memory bandwidth and L3 cache capacity are specified for each resource group.
- When an offline service is deployed, the resource group to which the service belongs can be specified in the YAML file.
- When offline services are deployed, adding YAML annotations places them in a dynamic control group that automatically adjusts their resource allocation based on online service performance.
- After the MPAM plugin detects a deployment task, the plugin allocates the process ID of the container service to the corresponding resource group. (The restriction information is configured on the hardware chip through the OS.)

  NOTE

  MPAM manages these shared resources: L2 cache, L3 cache, and DMC bandwidth.

Figure 4-6 MPAM plugin principle



5 Feature List

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualized CPU acceleration	Interrupt passthrough	When hardware-assisted virtualization acceleration is enabled on new Kunpeng 920 processor models, the direct interrupt injection of GICv4.1 (including vSGI passthrough) cuts interrupt response times and enhances throughput for demanding network and I/O workloads.	Supported OSs: openEuler 22.03 LTS SP3 and later Constraints: Only new Kunpeng 920 processor models are supported.	No	The virtualization software cannot be used on VMs.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualized CPU acceleration	Cluster awareness	A cluster is a hardware unit of a CPU. Each cluster contains several cores. Cores in a cluster share the same L3 cache tag. By adding the option for cluster task scheduling tuning of the OS kernel, cross-cluster thread scheduling can be prevented and L3 cache tag resources can be reused, improving the CPU scheduling efficiency and memory bandwidth utilization of multi-thread applications. Cluster awareness displays the cluster topology to the guest OS so that VM users can better use the cluster to optimize services. It also leverages hardware resources more efficiently, increases the system throughput, and speeds up responses to requests. For details, see **Cluster Tuning Guide**.	Supported OSs: openEuler 22.03 LTS SP2 and later Constraint: The topology information of the physical CPU cluster must be correctly mapped to the VM.	Yes	The enabling mode on a VM is the same as that on a physical machine. The OS must be openEuler 22.03 LTS SP2 or later.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualized CPU acceleration	NUMA awareness	NUMA awareness is an optimization technology for the non-uniform memory access (NUMA) architecture. Accessing the memory of other CPUs in the NUMA architecture causes access delay and performance deterioration. NUMA awareness displays the NUMA topology to the guest OS so that VM users can better use NUMA to optimize services. For details, see **Configuring Guest NUMA**.	Supported OSs: CentOS 7.6 and openEuler 20.03 LTS Constraints: After guest NUMA is configured, the service software may fail to identify NUMA. If the service software is not optimized for identifying NUMA, cross-NUMA memory access occurs, causing performance deterioration.	No	The virtualization software cannot be used on VMs.
Virtualized CPU acceleration	VM lock virtual-real synergy optimization	This feature uses shared memory to relay vCPU preemption status from the hypervisor to VMs, minimizing conflict-induced errors and improving system stability and reliability.	Supported OSs: openEuler 20.03 LTS SP1 or later libvirt 6.2.0 or later QEMU 6.2.0 or later	Yes	The enabling mode on a VM is the same as that on a physical machine. The OS must be openEuler 20.03 LTS SP1 or later.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualized CPU acceleration	VM deadlock detection	By leveraging non-maskable interrupts (NMIs), this feature monitors interrupt responses in real time to detect deadlocks within VMs, ensuring recovery from unresponsive states caused by deadlocks.	Supported OS: openEuler 22.03 LTS SP2 libvirt 6.2.0 or later QEMU 6.2.0 or later	Yes	The enabling mode on a VM is the same as that on a physical machine. The OS must be openEuler 22.03 LTS SP2 or later.
Virtualized CPU acceleration	Cache topology awareness	This feature provides a method of specifying the cache size in the VM XML configuration of libvirt or the QEMU command for starting VMs. By using this feature, the size of the cache used by a VM can be accurately reflected, and more accurate cache structure information is available for further optimizing the application performance in the VM.	Supported OS: openEuler 22.03 LTS SP4 libvirt 6.2.0 or later QEMU 6.2.0 or later	No	Physical machines transparently transmit topology to VMs.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualized CPU acceleration	Interrupt passthrough	When hardware-assisted virtualization acceleration is enabled on new Kunpeng 920 processor models, the direct interrupt injection of GICv4.1 (including vSGI passthrough) cuts interrupt response times and enhances throughput for demanding network and I/O workloads.	Supported OSs: openEuler 22.03 LTS SP3 and later Constraints: Only new Kunpeng 920 processor models are supported.	No	The virtualization software cannot be used on VMs.
Virtualized CPU acceleration	GICv4.1 overcommitment optimization	This feature allows the GIC to skip VMOVP instructions when vCPUs are migrated between CPUs that share the same vPE table, thus improving VM service performance in overcommitment scenarios.	Supported OS: openEuler 24.03 LTS SP3 Hardware constraints: Only the new Kunpeng 920 processor model is supported.	No	The virtualization software cannot be used on VMs.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualized I/O acceleration	vKAE	The Kunpeng Accelerator Engine (KAE) encryption and decryption module uses the KAE to implement the RSA, SM3, SM4, DH, MD5, and AES algorithms. It provides high-performance symmetric and asymmetric encryption/decryption algorithms based on the lossless user-mode driver framework. Compatible with OpenSSL 1.1.1a and later versions, it supports the synchronous and asynchronous mechanisms. This module can accelerate applications in VMs and containers. For details about how to use KAE in VMs, see **Using KAE on a KVM**. For details about how to use KAE in containers, see **Using KAE on Docker**. For details about how to use KAE-accelerated live migration in VMs, see **Virtualization Scenario KAE Accelerated Live**	Supported OSs: openEuler 20.03 LTS SP1 and later, and openEuler 22.03 LTS SP1 and later Constraint: You need to apply for a license on a physical machine. For details, see **Obtaining a License**.	Yes	The VM OS must be openEuler 20.03 LTS SP1 (or later) or openEuler 22.03 LTS SP1 (or later).

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
		Migration Feature Guide.			
Virtualized I/O acceleration	Virtualization scenario KAE passthrough live migration	VM live migration is a critical operational capability that enables seamless transfer of running VMs between physical hosts without service interruption. KAE passthrough live migration specifically addresses the scenario where VMs are configured with KAE passthrough devices, offering enhanced operational flexibility and continuous service availability.	Supported OSs: openEuler 22.03 LTS SP4 and openEuler 24.03 LTS SP2 and later Constraints: • Version: Arm-based KVM and QEMU virtualization platforms are required. For openEuler 22.03 LTS SP4, only libvirt 6.2.0 and QEMU 6.2.0 are supported. For openEuler 24.03 LTS SP2 and later, only libvirt 9.10.0 and QEMU 8.2.0 are supported. • License: KAE license. • The application environment must meet the software and hardware environment requirements supported by KAE.	No	-

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualized I/O acceleration	OVS flow table NIC acceleration (based on Mellanox)	Supports OVS flow table NIC acceleration (based on the Mellanox ConnectX-5 NIC) on the Kunpeng platform. For details, see **OVS Flow Table NIC Acceleration Feature Guide**.	Supported OSs: CentOS 7.6 and openEuler 20.03 LTS Constraints: • VM kernel-mode OVS scenario. • SR-IOV passthrough mode. • Live VM migration is not supported.	No	The virtualization software cannot be used on VMs.
Virtualization management optimization	vCPU hotplug	This feature increases or decreases the number of vCPUs of a running VM without interrupting services.	Supported OS: openEuler 24.03 LTS Constraints: The CPU specifications of the VM applying the hotplug feature cannot exceed the maximum CPU specifications supported by the hypervisor and the maximum CPU specifications supported by the guest OS.	No	The virtualization software cannot be used on VMs.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualization management optimization	QEMU VM memory hot add	This feature allows for starting a VM of a NUMA node whose initial memory configuration is 0 in the XML configuration file and dynamically adding memory to the NUMA node.	Supported OS: openEuler 22.03 LTS SP4 Constraints: • Among the NUMA nodes configured for a VM, a maximum of one NUMA node whose initial memory is 0 is supported. • The maximum memory that can be added for a VM through memory hot add is limited by the host system configuration and VM XML configuration. The host system configuration constraints depend on the OS type.	Yes	The enabling mode on a VM is the same as that on a physical machine. The OS must be openEuler 22.03 LTS SP4 or later.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualization management optimization	Virtualization scenario KAE-accelerated live migration	During the VM live migration, compression technologies (for example, zlib library) are usually used on the source physical host to compress memory pages before data transmission, and then the memory pages are decompressed on the target physical host, thereby speeding up the VM live migration. KAEzlib is a standard zlib interface provided by the KAE compression module. It uses the Kunpeng hardware acceleration module to implement the Deflate algorithm and works with the lossless user-mode driver framework. Therefore, KAE can replace the open-source compression library zlib to accelerate VM live migration.	Supported OSs: openEuler 22.03 LTS SP1/SP2/SP3/SP4 Constraints: ● Version: For Arm-based KVM and QEMU virtualization platforms, only libvirt 10.0.0 and later versions, and QEMU 6.2.0 are supported. ● License: KAE license. ● The application environment must meet the software and hardware environment requirements supported by KAE.	No	-

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualization management optimization	PMU virtualization	PMU virtualization enables PMU events to be collected on a VM, which helps analyze and optimize the performance of the VM OS and service software.	Supported OSs: openEuler 22.03 LTS SP4 and later	Yes	The enabling method on the VM is the same as that on the physical machine.
Virtualization management optimization	MPAM-enabled libvirt	MPAM is used to allocate VMs to different bandwidth/cache partitions, implementing rate limiting or priority control.	Supported OS: openEuler 24.03 LTS SP2 Constraints: Arm-based KVM and QEMU virtualization platforms are required. Only libvirt 9.10.0 and QEMU 8.2.0 are supported.	No	The virtualization software cannot be used on VMs.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualization management optimization	Cross-generation VM live migration	VM live migration allows a VM to be migrated from one physical host to another without interrupting the VM running. The key benefit of cross-generation migration lies in its ability to maintain VM operation and service continuity during hardware upgrades by supporting migration between different hardware generations. Successful implementation typically requires sufficient compatibility between source and target hardware platforms, and hardware discrepancy tolerance at the VM OS and application layers. While technically challenging, this capability significantly enhances operational flexibility for data centers and cloud service providers by ensuring continuous service availability.	Supported OSs: openEuler 24.03 LTS SP1 and later For details about constraints, see **General Constraints**.	No	The virtualization software cannot be used on VMs.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Virtualization management optimization	VM single-core and single-page exception handling	To ensure long-term stable running of Internet systems, Reliability, Availability, and Serviceability (RAS) capabilities are required to obtain the faulty hardware of the current physical machine and process the corresponding hardware information, minimizing the impact scope. After the single-core and single-page exception handling feature is enabled, single-core corrected errors (CEs) can be isolated online on Kunpeng servers without affecting service running; uncorrected errors (UEs) in a single page of memory affect only one process in a VM, preventing the VM from going offline.	Supported OS: openEuler 24.03 LTS SP3 libvirt 9.10.0-26.oe2403sp3 or later QEMU 8.2.0 or later	No	The virtualization software cannot be used on VMs.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Feature check tool	Hardware virtualization feature check tool	The hardware virtualization feature check tool (VirtCheck) is designed for new Kunpeng 920 processor models. It automatically scans and analyzes customer environments and scenarios to determine whether the virtualization features provided by Kunpeng BoostKit are supported or enabled.	Supported OS kernel versions: • The Linux kernel version of openEuler is 5.10 (for example, openEuler 22.03 LTS SP3) or 6.6 (openEuler 24.03 LTS).	Yes	Install the RPM package of the check tool on the VM and run the check command **virtcheck --guest**.
Open-source enablement	Hybrid deployment on OpenStack	Supports hybrid deployment for BMSs and KVMs in availability zones (AZs). Each AZ must contain only x86 or Kunpeng servers.	Supported OSs: CentOS 7.6 and openEuler 20.03 LTS Constraints: • The management and controller nodes do not support hybrid deployment. • Kunpeng and x86 servers are deployed together in AZs.	No	The virtualization software cannot be used on VMs.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Open-source enablement	Hybrid deployment on Kubernetes	Supports hybrid deployment of Kunpeng and x86 servers for Docker containers. For details, see **Hybrid Container Deployment Guide**.	Supported OSs: CentOS 7.6 and openEuler 20.03 LTS Constraints: Kubernetes management nodes do not support hybrid deployment.	No	The virtualization software cannot be used on VMs.
Open-source enablement	oVirt for VM cluster management	Supports oVirt use on the Kunpeng platform. For details, see **Kunpeng oVirt Lightweight Virtualization Management Platform Deployment Guide**.	Supported OSs: openEuler 20.03 LTS SP1 and openEuler 22.03 LTS SP2	No	The virtualization software cannot be used on VMs.
Open-source enablement	Large-scale Docker container networking	For details about the large-scale Docker container networking scenario, see **Large-Scale Docker Container Networking Feature Guide**.	Supported OSs: CentOS 7.6 and openEuler 20.03 LTS	Yes	The Docker container technology can be used on VMs.
Open-source enablement	Fixed IP address of the Docker container even after the porting	For details, see **Kube-OVN User Guide**.			

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Cloud native infrastructure	KAE device plugin	The Kunpeng Accelerator Engine (KAE) is a hardware-based acceleration solution built on Kunpeng 920 series processors. It supports encryption, decryption, and decompression. The KAE device plugin automatically manages all KAE devices on the server and streamlines the KAE device passthrough operations. Using this plugin, you can pass through KAE devices to containers through simple statements to accelerate encryption, decryption, and data compression, and set QoS for KAE devices.	Supported OSs: openEuler 22.03 LTS SP4 and later Constraints: Before using the KAE device plugin, ensure that the VF of the KAE device has been created.	No	This feature can be used only in container scenarios.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Cloud native infrastructure	KAE-enabled Envoy acceleration	In microservice scenarios, Envoy needs to process a large number of TLS requests, regardless of whether it functions as an ingress gateway or a microservice proxy. Especially in the handshake phase, asymmetric encryption and decryption consume a large number of CPU resources. If microservices are deployed on a large scale, this overhead may become a system performance bottleneck. To address this, the KAE private key provider of Envoy offloads time-consuming encryption and decryption operations from the CPU to KAE. This accelerates encryption and decryption while releasing CPU computing power for other service workloads.	Supported OSs: openEuler 22.03 LTS SP4 and later Constraints: Before using the KAE device plugin, ensure that the VF of the KAE device has been created.	No	This feature can be used in container and physical machine scenarios.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Cloud native high-performance networking	Kubernetes SR-IOV device plugin	The SR-IOV technology can virtualize a single physical PCIe device into multiple virtual PCIe devices, and then pass through these virtual devices to each VM, so as to be applied in scenarios where a single physical PCIe device supports running of multiple VMs. The Kubernetes SR-IOV device plugin automatically identifies and manages SR-IOV devices on nodes, and automatically completes the mounting process based on the SR-IOV device type specified in the configuration file, streamlining the usage process. For details, see **Cloud Native Scenario Kubernetes SR-IOV Device Plugin User Guide**.	Supported OSs: openEuler 22.03 LTS SP4 and later Constraints: Before using the SR-IOV device plugin, ensure that the VF of the SR-IOV device has been created.	No	This feature can be used only in container scenarios.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Cloud native resource affinity and isolation	Kunpeng Topology Affinity Plugin	In the Kubernetes container deployment scenario, the CPU scheduling range of containers can be automatically adjusted based on the CPU load of compute nodes to ensure NUMA affinity.	Supported OSs: openEuler 20.03 LTS SP3 and openEuler 22.03 LTS SP4 Kubernetes versions: 1.23.*x* and 1.28.4 containerd version: 1.7.14 Constraint: Use containerd as the container runtime. Docker version: 20.10.14 Constraint: Use Dockershim as the runtime communication component.	No	This feature can be used only in container scenarios.

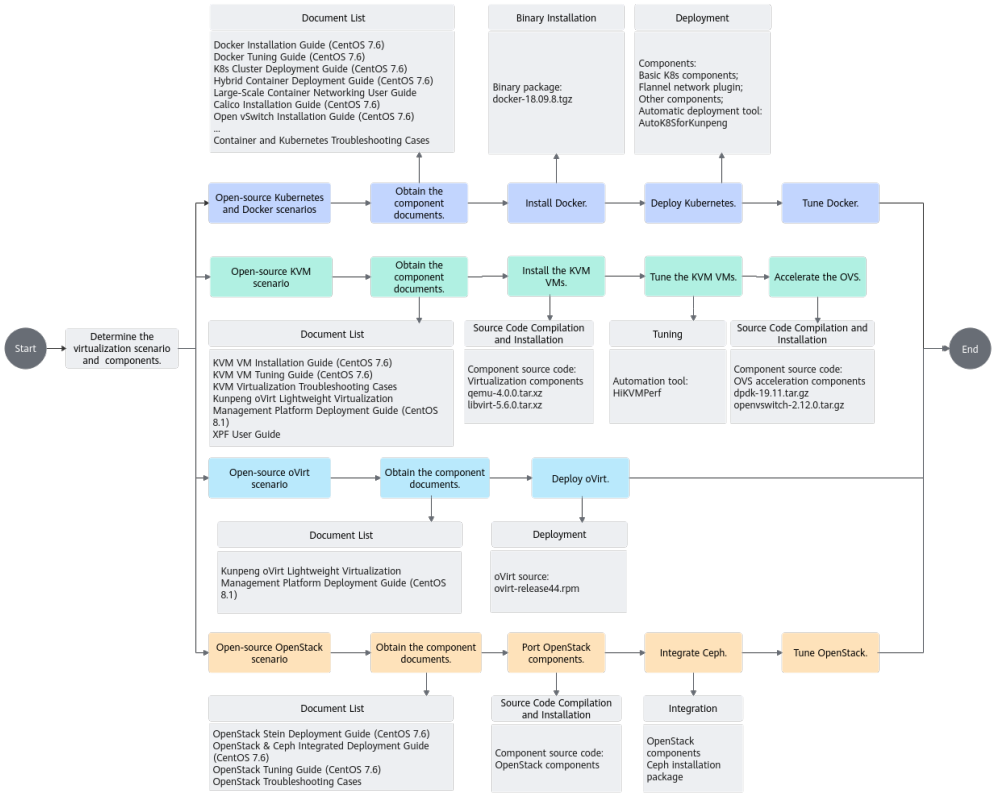
Feature	Sub-feature	Description	Constraints	VM Support	Remarks
Cloud native resource affinity and isolation	MPAM plugin	MPAM is a new feature of Armv8.4-A. The MPAM mechanism adds different labels (PARTID/PMG) to all requests of different service flows at the CPU/IO source so that the hardware can sense the service flows. Based on the information, resources, such as the cache capacity and DMC bandwidth, can be dynamically allocated to each component (such as the cache and DDR) of system resources to isolate different service flows and reduce interference. The Kubernetes MPAM plugin can dynamically isolate the resource usage of offline services in hybrid deployment scenarios. When resources are idle, offline services can use more resources. When resources are insufficient, the resource usage of offline services is limited to ensure the performance of online services and	Supported OSs: openEuler 22.03 LTS SP4 and later Constraints: Before using the MPAM plugin, ensure that MPAM has been enabled.	No	This feature can be used only in container scenarios.

Feature	Sub-feature	Description	Constraints	VM Support	Remarks
		improve the overall resource utilization of the system.			

6 Usage Process

Figure 6-1 shows the end-to-end process of porting, deploying, and tuning the Kunpeng BoostKit for Virtualization.

Figure 6-1 Kunpeng BoostKit for Virtualization end-to-end porting, deploying, and tuning process



7 Reference

- Information and figures on the official **QEMU**, **libvirt**, **OpenStack**, and **Ceph** websites are referenced in **2.1.1 Architecture** and **2.1.2 Typical Configuration**.
- Information and figures on the official **Kubernetes** and **Docker** websites are referenced in **2.1.3 Component Principles**.

A Change History

Date	Description
2026-03-30	This issue is the twenty-fourth official release. Reconstructed the entire technical white paper.
2025-12-30	This issue is the twenty-third official release. • Added **4.1.1 KAE Device Plugin** and **3.3.5 Cross-Generation VM Live Migration**. • Updated **5 Feature List**.
2025-09-30	This issue is the twenty-second official release. • Added **3.3.4 MPAM-enabled libvirt**. • Updated **5 Feature List**.
2025-06-30	This issue is the twenty-first official release. • Added **3.2.1.2 KAE Passthrough Live Migration** and **3.3.3 PMU Virtualization**. • Updated **4.2.1 Kubernetes SR-IOV Device Plugin**, **4.3.1 Kunpeng Topology Affinity Plugin**, **4.3.2 Kubernetes MPAM Access and Memory Isolation and Control Plugin**, and **5 Feature List**.
2025-03-30	This issue is the twentieth official release. • Added **3.1.1.5 Virtualization Scenario Cache Topology Awareness**, **4.2.1 Kubernetes SR-IOV Device Plugin**, and **3.3.2 Virtualization Scenario KAE-Accelerated Live Migration**. • Updated **4.3.1 Kunpeng Topology Affinity Plugin** and added the Topology Affinity Plugin (TAP) for Docker.

Date	Description
2024-12-30	This issue is the nineteenth official release. • Added **3.2.2 Virtualization DPU Offload**, **4.3.1 Kunpeng Topology Affinity Plugin**, **3.3.1.1 VM vCPU Hotplug**, **3.3.1.2 QEMU VM Memory Hot Add**, and **4.3.2 Kubernetes MPAM Access and Memory Isolation and Control Plugin**. • Updated Workload Aware Acceleration System.
2024-06-30	This issue is the eighteenth official release. • Added **3.4 General Optimization** and Workload Aware Acceleration System. • Updated **5 Feature List** and added descriptions about the Workload Aware Acceleration System.
2023-11-21	This issue is the seventeenth official release. Reconstructed the entire technical white paper.
2023-10-27	This issue is the sixteenth official release. Added the feature of cluster scheduling tuning to **5 Feature List**.
2022-12-30	This issue is the fifteenth official release. Added High-Performance Cloud Disk Optimization.
2022-10-19	This issue is the fourteenth official release. Added the section of "Key Features."
2021-11-10	This issue is the thirteenth official release. Added constraints on key features and description about whether these features can be used on VMs. For details, see **5 Feature List**.
2021-08-11	This issue is the twelfth official release. Modified the recommended Kunpeng processor model in the typical configuration of each solution.
2021-06-25	This issue is the eleventh official release. Changed the processor model from "Kunpeng 920 5230" to "Kunpeng 920 5220."
2021-06-21	This issue is the tenth official release. Added the paths for obtaining some installation packages in **5 Feature List**.
2021-06-09	This issue is the ninth official release. • Added **2.4.4 Compatibility Between Kubernetes and Docker**. • Added the feature of OVS flow table NIC acceleration to **5 Feature List**.

Date	Description
2021-05-27	This issue is the eighth official release. Changed the feature name from "OVS software offloading" to "OVS flow table normalization" in **5 Feature List**.
2021-03-23	This issue is the seventh official release. Changed the solution name from "Kunpeng virtualization solution" to "Kunpeng BoostKit for Virtualization."
2021-03-10	This issue is the sixth official release. Added the large-scale Docker container networking feature **5 Feature List**.
2021-01-07	This issue is the fifth official release. • Added open-source OVS flow table hardware acceleration. • Added the open-source oVirt platform to the cloud cluster management platforms in the solution overview.
2020-10-24	This issue is the fourth official release. • Added **2.2 VM Cluster Solution (Based on Open-Source oVirt and KVM)** and OVS Flow Table Normalization. • Added the following features in **5 Feature List**: VM lightweight management software, virtualized network acceleration, improved light-loaded VM performance, and reduced virtualization resource fragments.
2020-09-21	This issue is the third official release. Changed the solution name from "Kunpeng cloud platform solution" to "Kunpeng virtualization solution."
2020-07-03	This issue is the second official release. Modified the hybrid deployment features in **5 Feature List**.
2020-06-10	This issue is the first official release.

