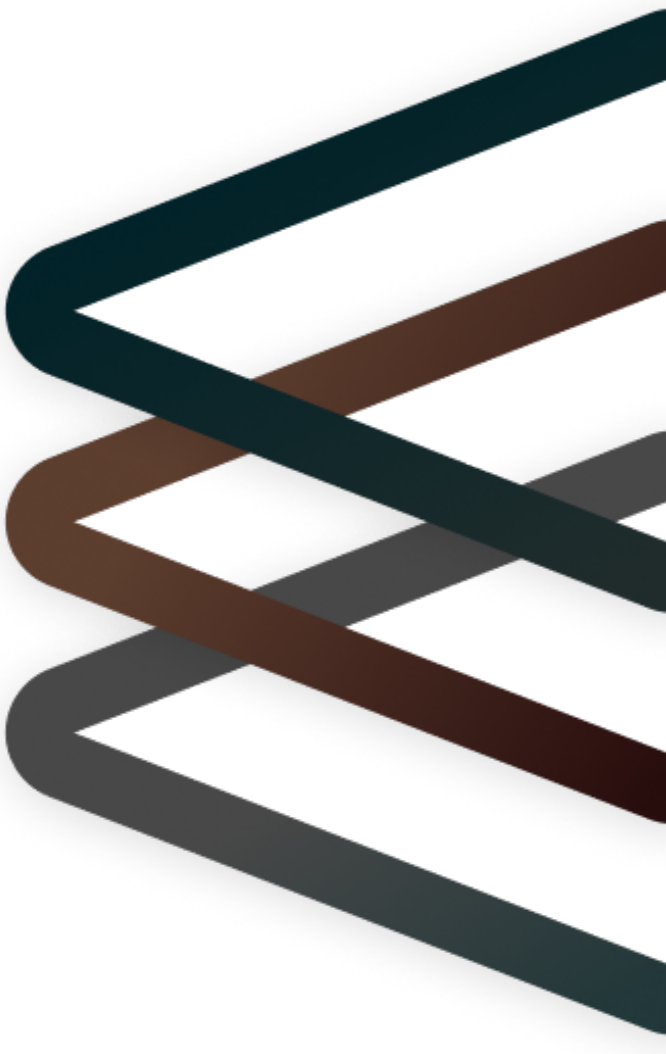
nSSV

Technical Whitepaper



Contents

1 Overview

Industry Background

Virtualization in IT data centers has become an industry trend. As IT infrastructure scales, its management complexity also increases. New-generation data centers use virtualization technologies to dynamically schedule and allocate IT infrastructure resources, effectively addressing many issues faced by traditional IT architectures, such as low resource utilization, high data security risks, complex system maintenance, and long business launch cycles. However, in the face of an increasing number of high-performance businesses, how to better mitigate challenges from performance, security, stability, and compatibility has become a key area of development for virtualization technologies and a fundamental requirement for platform selection.

Product Description

nSSV is a high-performance, high-security, high-stability, and high-autonomous virtualization software that features a or nCSSV engine and 4S characteristics. It uses advanced technologies such as server virtualization, network virtualization, and storage virtualization and has intelligent advanced maintenance capabilities. With nSSV, you can quickly build virtualized data centers and also build integrated solutions from IaaS to PaaS with Nexavm Technologies AG products.

Scenarios

Virtualization Transformation for High-Performance Business/Database：

Database is the core of modern applications, responsible for storing, managing, and processing large amounts of data. The performance and reliability of databases are crucial for the normal operation of applications.nSSV virtualization technology provides resource isolation and scalability, allowing for better management of the hardware resources of database servers. Through virtualization, you can dynamically allocate and adjust server memory, CPU, and storage resources as needed. In addition, nSSV virtualization also enables flexible backup and recovery, providing high availability and fault tolerance.

Smart Production Optimization：

Responding to the government's call to enhance the core competitiveness of manufacturing, more and more manufacturing enterprises are shifting their focus from processing-intensive operations to digital transformation through innovation applications and intelligent manufacturing to achieve high-quality growth.nSSV provides a powerful digital foundation, supporting the deployment of DMP/ERP/OA and other business systems and leveraging advanced AI technologies based on

the virtualization platform to achieve predictive, informationized, and dataized management, such as:

- Production testing scenarios, where you can use nSSV for product testing and simulation, simulating real production environments and processes to predict product performance, optimize production processes, and reduce actual testing and trial-and-error costs.
- Supply chain management scenarios, where you can use nSSV to achieve information sharing, real-time demand forecasting, and resource scheduling, improving supply chain responsiveness.
- Equipment maintenance scenarios, where you can use nSSV virtualization technology to convert physical devices into virtual instances, achieving remote monitoring, fault diagnosis, and remote maintenance, improving equipment availability, and reducing maintenance costs.
- Training and education scenarios, where you can use nSSV to create virtualized training environments, simulating real production scenarios and operations to help employees familiarize themselves with and master production skills, improving production efficiency and quality.

Resource Bearing for Production Environments：

In traditional production environments, you may face complex configuration and deployment, environment conflicts, physical resource limitations, data security issues, and difficulties in troubleshooting failures. With nSSV, you can easily address all these issues:

- **Fast Deployment and Environment Isolation**: nSSV supports fast deployment of virtual machine instances, simplifying environment setup and configuration processes. You can create multiple independent virtual environments as needed for different development and testing projects, ensuring that each project does not interfere with each other, thereby improving R&D efficiency.
- **Efficient Use of Physical Resources**: nSSV virtualization technology pools and shares physical server resources, enabling multiple virtual machine instances to run on the same physical server, thus improving resource utilization and reducing hardware investment costs. In addition, it makes it easier to dynamically adjust resource allocation to meet the changing needs of business projects.
- **Data Security Protection**: nSSV allows you to create snapshots for virtual machines as needed, enabling rollback when needed to ensure data security and business stability and reliability.

- **Flexible Development Environments**: nSSV virtualization technology provides flexible development environments for business teams. Each developer can develop and test in their own virtual machine, independently managing and configuring their development environments, avoiding conflicts and interference among development environments, thus improving developers' work efficiency and development quality.
- **Failure Isolation and Tolerance**: nSSV supports failure isolation and tolerance. If a virtual machine fails, it will not affect the normal running of other virtual machines, thus reducing downtime caused by software and hardware failures and improving business continuity and stability.

2 Technical Advantages

High Performance

High-performance businesses serve as a key lever for cost reduction and efficiency improvement, placing higher performance requirements on the underlying virtualized platform. To meet the growing performance demands, the virtualized platform has the following advantages:

- Strong high-concurrency capabilities that efficiently boost business deployment efficiency
- Maximum reduction of compute resource performance loss
- Efficient utilization of various network resources
- Flexible access to high-performance storage

High Security

As network infrastructure continues to improve and more data goes online, ensuring business security and stable operation has become a core requirement. The virtualized platform boasts all-around and multi-dimensional security capabilities:

- Network traffic control: Distributed firewalls and security group filtering control north-south and east-west network traffic, preventing virus attacks
- Data: Commercial cryptography is supported for application in various industries and fields, building a cryptographic barrier for network security
- Business: A kernel-level agentless virtualization security protection engine can promptly identify and clear security threats to protect business data

High Reliability

As a core software system in IT infrastructure, a virtualized platform is a strong support for rich business capabilities and the last line of defense for data center smooth running, serving as a basic prerequisite for cost reduction and efficiency improvement. As a key object for maintenance personnel to ensure stability, a highly reliable virtualized platform has the following advantages:

- Power failure auto-recovery
- No data loss
- Daily no crashes

High Self-Research

In recent years, the replacement of homegrown products has become a major trend in the domestic IT industry. Developing a homegrown and controllable product is an inevitable part

of this trend. Homegrown products are engines for achieving supply chain security. As a must-have in information system architecture, virtualization products occupy an important position in the replacement of homegrown products. As a core bottom platform, a highly self-researched virtualized platform has the following advantages:

- Support for establishing a domestic ecosystem
- Support for external API calls
- Support for self-downloading and trying out

3 Product Architecture

3.1 Software Architecture

The following lists the key features of the nSSV softwarearchitecture:

- Full Asynchronous Architecture: Asynchronous messages, asynchronousmethods, and asynchronous HTTP calls.
  - Asynchronous messages: Services communicate with each other via amessage bus. When a calling service calls a target service, the calling service sends amessage to the target service and registers a callback function. Then, the calling serviceimmediately returns. Once the target service completes the task, the callback function istriggered to reply the task result.Asynchronous messages support parallel processing.
  - Asynchronous methods: Services communicate with each other viaasynchronous messages . A series of related components or plugins within a service also calleach other via asynchronous methods, with the same calling method as that of asynchronousmessages.
  - Asynchronous HTTP calls: The plugin mechanism is adopted. A proxyprogram is set for each plugin, and a callback URL is set for each request in the HTTPheader. After the task is completed, the proxy program sends a response to the callingURL.
  - Based on these three ways of asynchronous messages, asynchronousmethods, and asynchronous HTTP calls, the virtualization platform constructs a layeredarchitecture, ensuring that all components can achieve asynchronous operations.
  - Thanks to the full asynchronous architecture mechanism, a singlen Management Node can concurrently process tens of thousands of APIrequests per second and can manage tens of thousands of hosts andhundreds of thousands of VM instances.
- Stateless Service: A single request does not depend on otherrequests.
  - Compute node agent, storage agent, network service, console agent,configuration service , and other services are not dependent on eachother. A single request can contain all information, and relevantnodes do not need to maintain any information.
  - Consistent hashing is used to authenticate the hash ring ofmanagement nodes, compute nodes, and other resources by using theUUID as the unique ID. The message sender does not need to know theservice instance that needs to process the message, and the servicealso does not need to maintain and exchange related resources. The service only needs toprocess messages.

- Information shared among management nodes is very little, and twomanagement nodes are enough to meet the high availability and scalabilityrequirements.
- The stateless service mechanism makes the system more robust. Rebooting a server will not lose any state information, and the scalability maintenance of the data center is simpler.

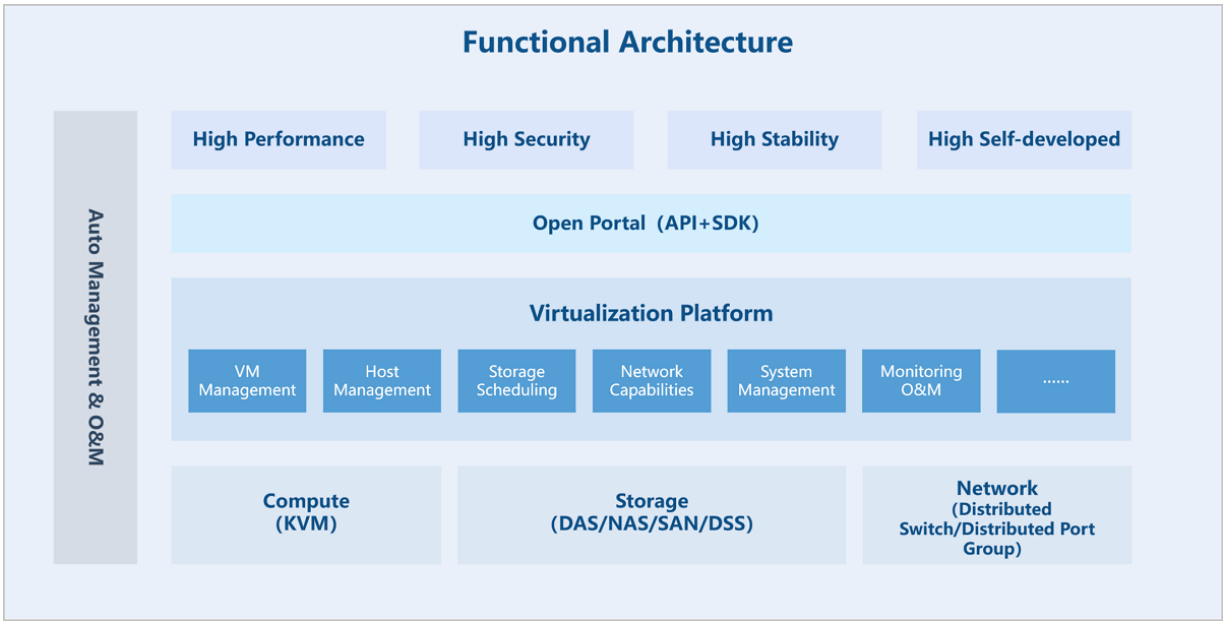
- Lock-free Architecture: Consistent hashing.
  - Consistent hashing ensures that all messages of the same resourceare processed by the same service instance. This method of aggregatingmessages to specific nodes reduces the complexity of synchronizationand parallelism.
  - Work queues are used to avoid contention for locks. Serial tasksare saved in memory in the form of work queues, and work queues canparallelize any operation on any resource to improve systemparallelism.
  - Based on queue-based lock-free architecture, tasks can simplycontrol parallelism, thereby improving system performance.
- Microservices in-process: Decoupled microservices.
  - Message buses are used to isolate and control each service, suchas VM service, identity authentication service, snapshot service,disk service, network service, and storage service. Allmicroservices are collected in the management node process, and eachservice interacts with each other through message buses. After allmessages are sent to the message bus , the consistent hashing ring isused to select the destination service for forwarding and processing.
  - The microservices in-process is implemented through a stararchitecture to achieve independent running of each service, decouplinghighly concentrated control businesses, achieving high autonomyand isolation for the system, and ensuring high reliability andstabili ty even if any service fails.
- Plug-in Structure: Plug-ins support horizontal expansion.
  - Newly added plug-ins do not affect each other and provide servicesindependently.
  - Strategy pattern and observer pattern are used to design plug-ins.Strategy plug-ins inherit from the parent class interface andimplement specific logic; observer plug-ins register Listener tomonitor internal business logic events. When an event occurs inthe application, the observer plug-in responds to the eventautomatically in its own code and executes the corresponding businessflow.
  - The plug-in horizontal expansion mechanism makes the virtualizationplatform able to quickly iterate while the overall systemarchitecture remains robust.

- Workflow Engine: Sequential management, rollback on errors.
  - Each workflow is clearly defined through XML. If an error occurs atany step, the workflow can be rolled back along the originalexecution path to clean up the garbage resources generated duringthe execution.
  - Each workflow can also contain sub-workflows to extend the businesslogic.
- Tag System: Support for business logic changes and addition ofresource attributes.
  - System tags and plug-in mechanisms are used to extend and changethe original business logic.
  - The tag mechanism is used to group and classify resources, and youcan use specified tags to search for resources.
- Waterfall Architecture: Support for cascading operations ofresources.
  - Resource management is implemented through a waterfall mechanism,which cascades operations on resources. For example, when youdetach or delete a resource, related resources will also becascade-deleted.
  - Resources can be added to or removed from the waterfallmechanism in plug-in form, which does not affect otherresources.
  - The cascading mechanism makes resource configurations flexibleand lightweight, and can quickly meet the configuration changerequirements of customers.
- Full Auto-Deployment: Ansible agentless auto-deployment.
  - Ansible is used for agentless full auto-deployment. Physicalresources are configured and agent programs are deployedautomatically. The entire process is transparent to users, and noadditional intervention is required. Reconnect the agent programto upgrade it.
- Full API Query: Any attribute of any resource can be queried.
  - Millions of conditions are supported for resource queries, and fullAPI queries and arbitrary combinations are supported.

3.2 Architecture

nSSV architecture is shown in

Figure 3-1: Architecture



.

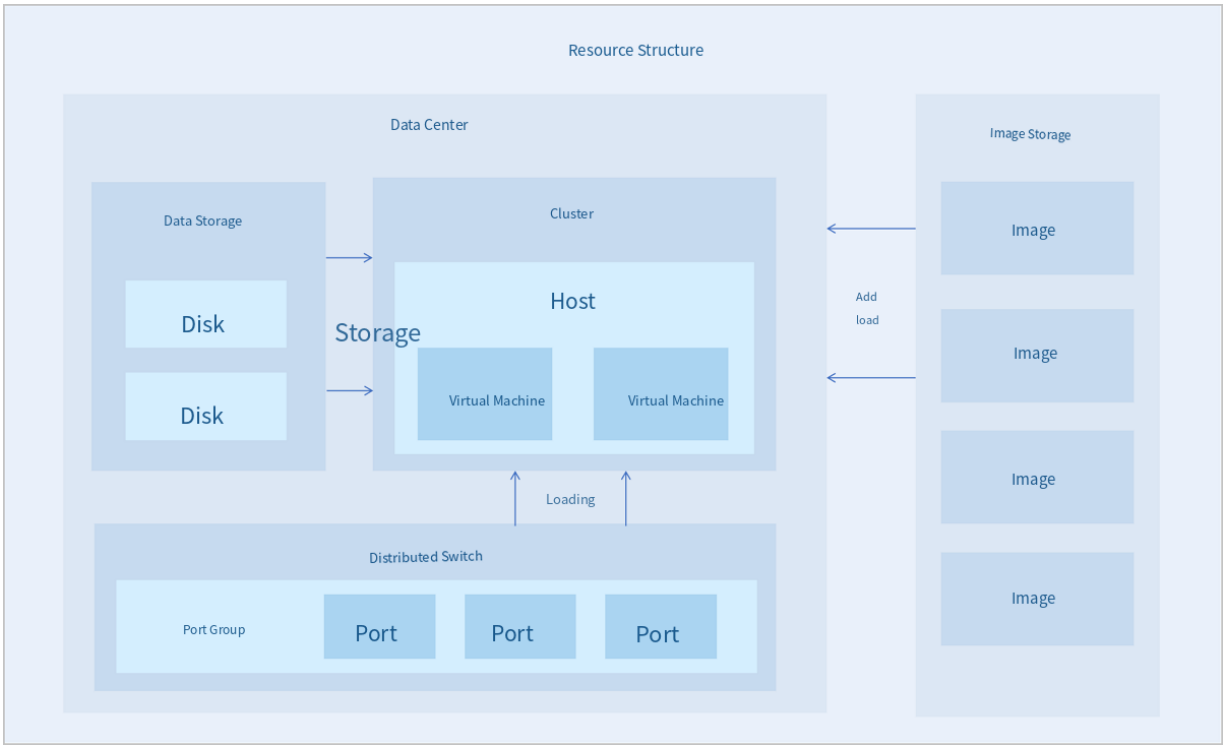
The key features of nSSV architecture include:

- Provides computing, storage, and network resource management services for enterprise -leveldata center infrastructures. The underlying supports KVM virtualization technology andmultiple storage types, including local storage, NFS storage, SAN storage, distribute dstorage, and DAS/NAS/SAN/DSS storage. In addition, it supports distributed switch anddistrib uted port group networks.
- Uses nSSV as its core engine. The engine communicates with thedatabase MariaDB and various service modules through a message bus to provide virtualmachine management, host management, storage scheduling, network functionality, systemmanagement, monitoring and maintenance, and other features.
- Provides Java and Python SDKs and supports RESTful APIs for resource scheduling andmanagement.

3.3 Resource Structure

nSSV resource structure is shown in

Figure 3-2: Resource Structure



.

The following resources mainly make up a nSSV:

- Data Center (DC): A data center is the largest resource namespace within a virtualization platform, including resources such as clusters, hosts, data storage, distributed switches, and distributed port groups.
- Cluster: A logical collection of a group of hosts (compute nodes).
- Host: A host is an x86 or ARM physical server running a KVM virtualization hypervisor, providing resources such as computing, networking, and storage to virtual machines.
- VM Instance: A virtual machine is a virtualized host running on a physical host, capable of running an operation system and applications just like a physical host.
- Data Storage (DS): A data storage is a virtualized resource that provides storage space for virtual machines and their application data. A data storage can be categorized into local storage and network shared storage.
- Distributed Switch (DSW): A virtual switching device that provides unified virtual network management and monitoring for virtual machines within a cluster.
- Distributed Port Group (DPG): A logical grouping of ports on a distributed switch, used for port configuration.

- Image Storage (IS): An image storage is a virtualized resource that provides storage space for image template files used by virtual machines or disks. An image storage can be categorized into standalone image storage and distributed image storage.
- Image: An image is a template file used by virtual machines or disks. Images are categorized into system images and disk images.

nSSV resources are related to each other in two ways:

- Parental relationship: similar to interpersonal relationships in the human society, including father-child relationship, brother-brother relationship, grandparent-grandchild relationship, and friend relationship.

  Specifically, a resource is defined as:

  - Father-child relationship: Resource A is the father or child of Resource B. For example, Cluster and Host, Host and VM Instance. Both Host and VM Instance run on Cluster.
  - Brother-brother relationship: Resource A and B have the same father and are brother to each other. For example, Cluster and DSW, Cluster and DS. Both Cluster and DS have Data Center as their father.
  - Grandparent-grandchild relationship: Resource A is the direct grandparent or grandchild of Resource B. For example, Data Center is the grandparent of Cluster, Cluster is the grandparent of Host, and Host is the grandparent of VM Instance.
  - Friend relationship: Resource A and B are not in the above three relationships but need to cooperate with each other in some scenarios, such as Data Storage and Image Storage. They need to work together to provide services for Cluster.
- Quantity relationship: similar to quantity restrictions in the human society, including 1:n, n:1, and n:n.

  Specifically, a resource is defined as:

  - **1:n**: indicates that Resource A can create/add/load multiple Resource B, for example, multiple hosts can be added to a cluster, and multiple clusters can be loaded by a DSW.
  - **n:1**: indicates that multiple Resource A can create/add/load one Resource B, for example, multiple hosts can be added to a cluster, and multiple clusters can be loaded by a DSW.
  - **n:n**: indicates that Resource A can create/add/load multiple Resource B, and multiple Resource A can create/add/load one Resource B, for example, an image storage can be loaded by multiple data centers, and a data center can also be loaded by multiple image storages.

4 Core Design

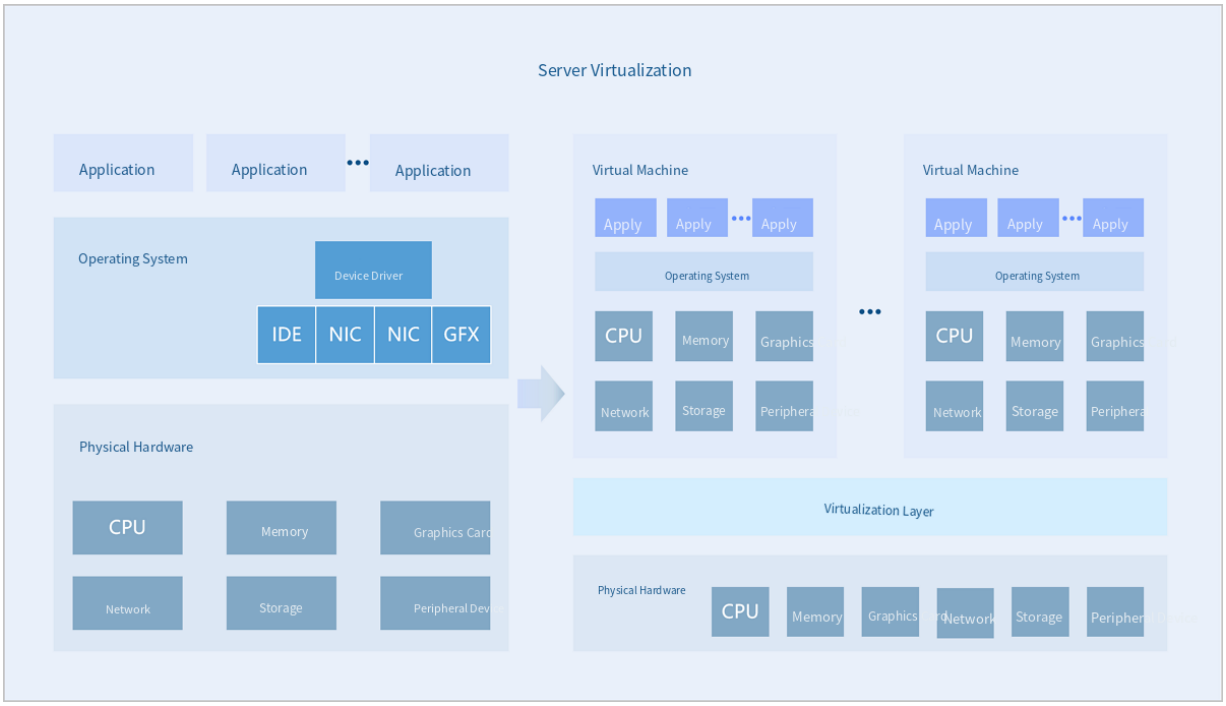
4.1 Resource Virtualization

4.1.1 Compute Virtualization

4.1.1.1 Overview

Server virtualization is a technology that abstracts physical server resources into logical resources through virtualization, making one physical server function as multiple logically isolated virtual servers. This way, various hardware resources such as CPU, memory, disk, and I/O devices are transformed into a pool of virtual resources for unified and dynamic management. This improves resource utilization and lowers system management costs, making IT more adaptable to business changes.

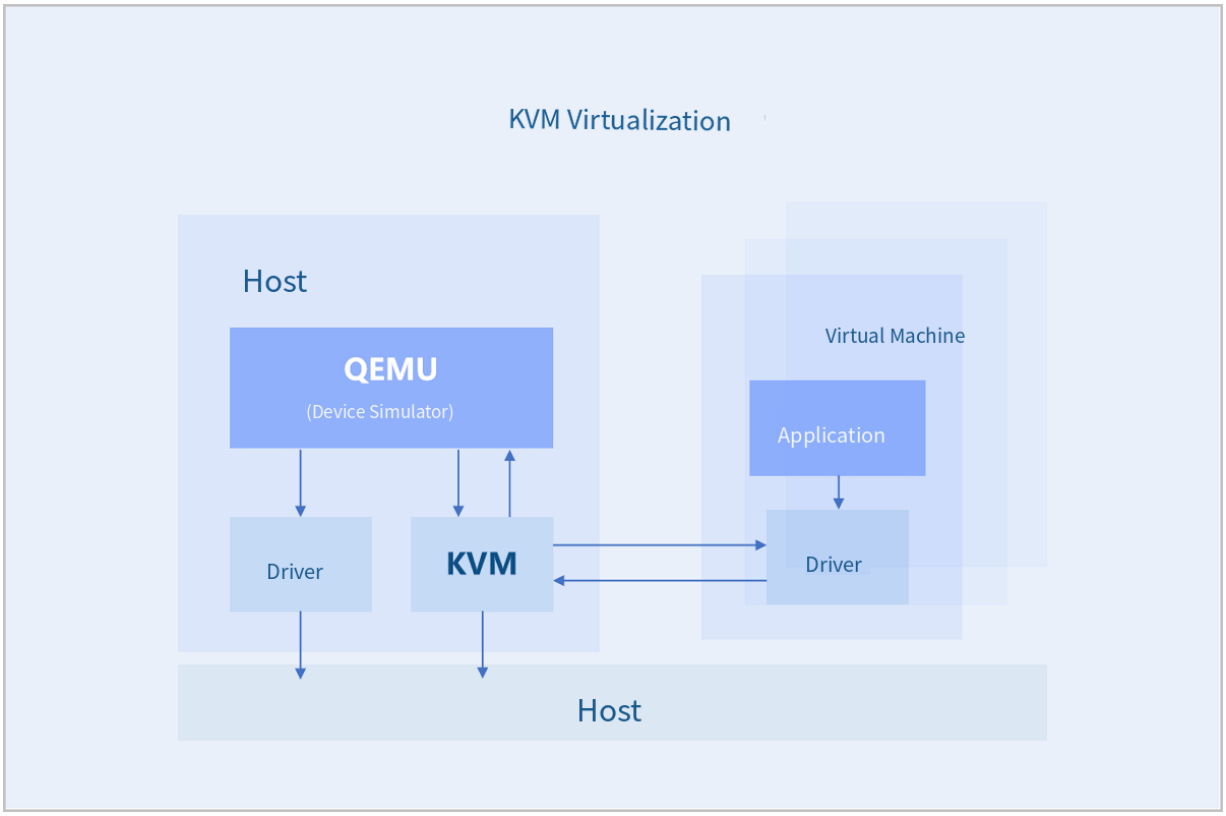
Figure 4-1: Server Virtualization



The virtualization platform supports KVM-based hardware virtualization. KVM is a Linux kernel module that turns the Linux kernel into a Hypervisor. KVM manifests as a process within the Linux system, scheduled by the standard Linux scheduler, which allows KVM to leverage existing Linux kernel features such as memory management and CPU scheduling. However, KVM only provides CPU and memory virtualization. To achieve I/O device virtualization, QEMU is used in conjunctio

n with KVM. QEMU is a user-space device emulator that provides virtual device models for VM instances, responsible for creating, calling, and managing various virtual devices.

Figure 4-2: KVM Virtualization



4.1.1.2 Technical Features

4.1.1.2.1 CPU Virtualization

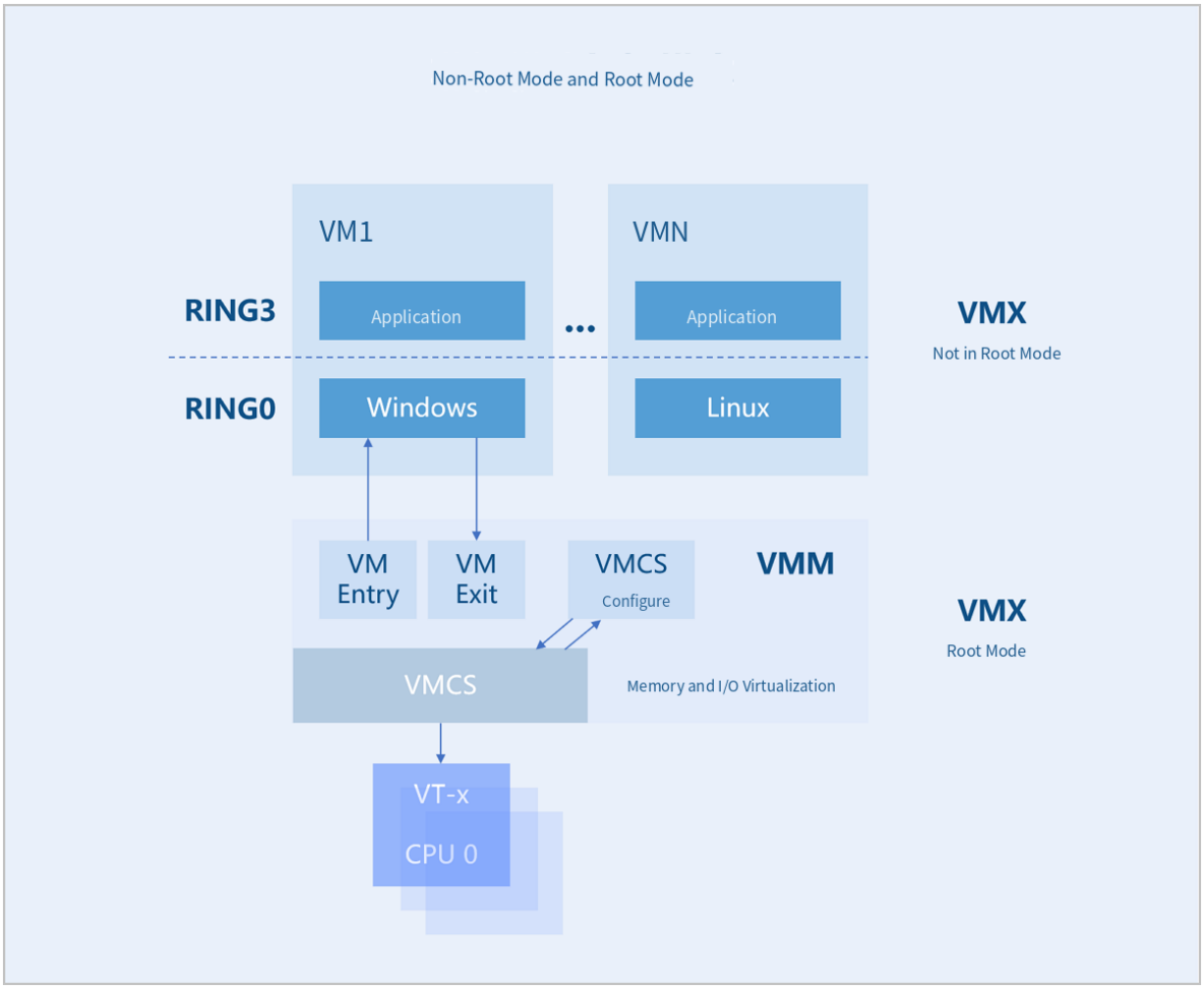
On the x86 architecture, CPUs generally have 4 privilege levels, namely, RING0 to RING3, used for allowing operating systems and applications to access hardware. In Linux, only 2 privilege levels are used, namely, RING0 (kernel mode) and RING3 (user mode).

About VMX root mode and non-root mode. For hardware-assisted virtualization, to allow virtual machines to be used without modifying operating systems, CPUs introduce 2 runtime modes, namely, VMX root mode and VMX non-root mode. Hosts run in root mode, with their kernels in RING0 and user-space programs in RING3. Virtual machines run in non-root mode, with their kernels in RING0 and user-space programs in RING3.

About VM exit and VM entry. When a virtual machine running in non-root mode is subject to an external interrupt, a page fault, or a VMCALL instruction call, the CPU switches from non-root mode to root mode, which is known as VM exit. In contrast, when a VMM switches a virtual

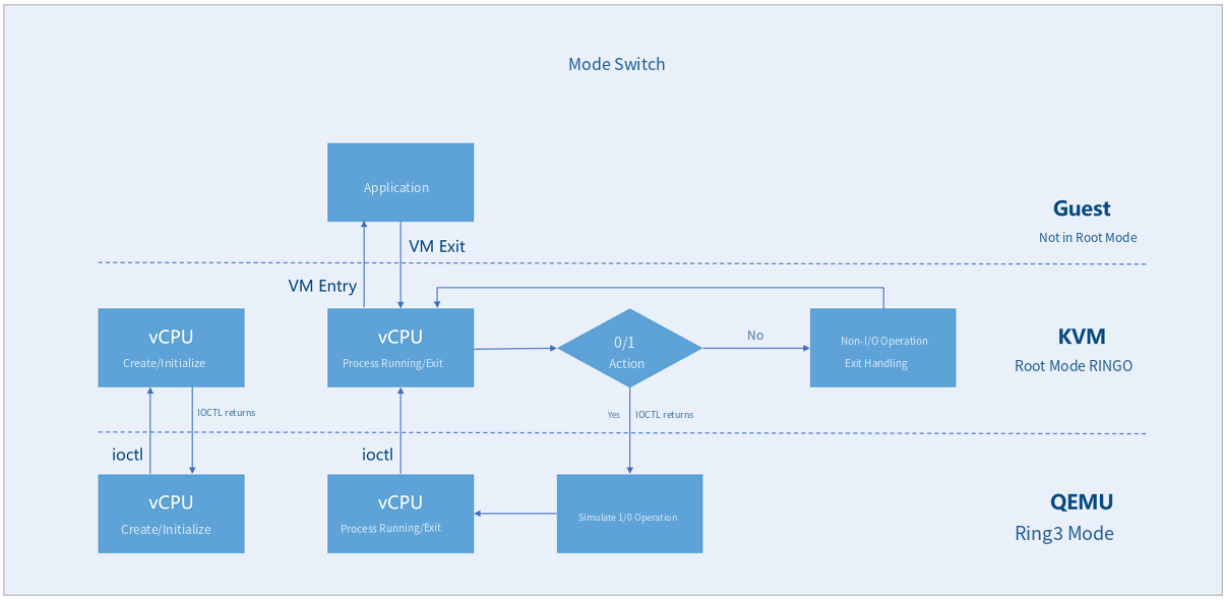
machine to non-root mode by explicitly calling the VMLAUNCH or VMRESUME instruction, the hardware automatically loads the virtual machine context and runs its instructions, which is known as VM entry.

Figure 4-3: Non-Root Mode and Root Mode



After a VM exits non-root mode and is subject to a VM exit, KVM will perform further actions based on the exit reason. If the exit is due to an I/O operation, QEMU will handle it. If the exit is due to a non-I/O operation, KVM will handle it on its own. After these actions are completed, KVM will re-enter non-root mode via VM entry.

Figure 4-4: Mode Switch



4.1.1.2.2 Memory Virtualization

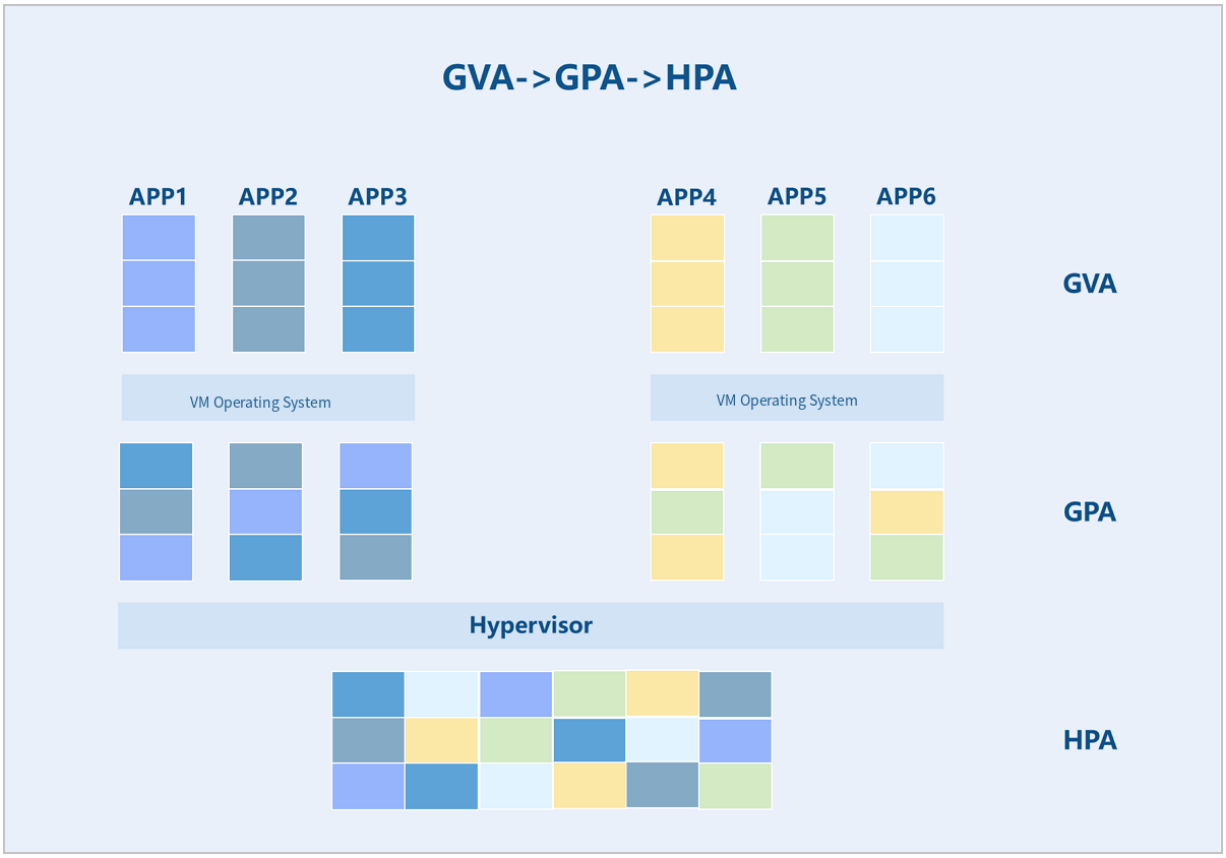
A virtual machine monitor (VMM) manages and allocates physical memory for eachvirtual machine . The operating system of a virtual machine sees a virtualphysical memory space. The memory management module of the operating systemmaps the virtual machine guest virtual address (GVA) to the guest physicaladdress (GPA). The instruction target address is also a virtual machinephy sical address. This address is actually a physical address when novirtualization is introduced.

However, when virtualization is introduced, this address cannot be useddirectly. The VMM converts the virtual machine physical address to a hostphysical address (HPA) first and then passes the HPA to the physicalprocessor for execution.

With the virtual machine physical address space introduced, memoryvirtualization mainly addresses the following two issues:

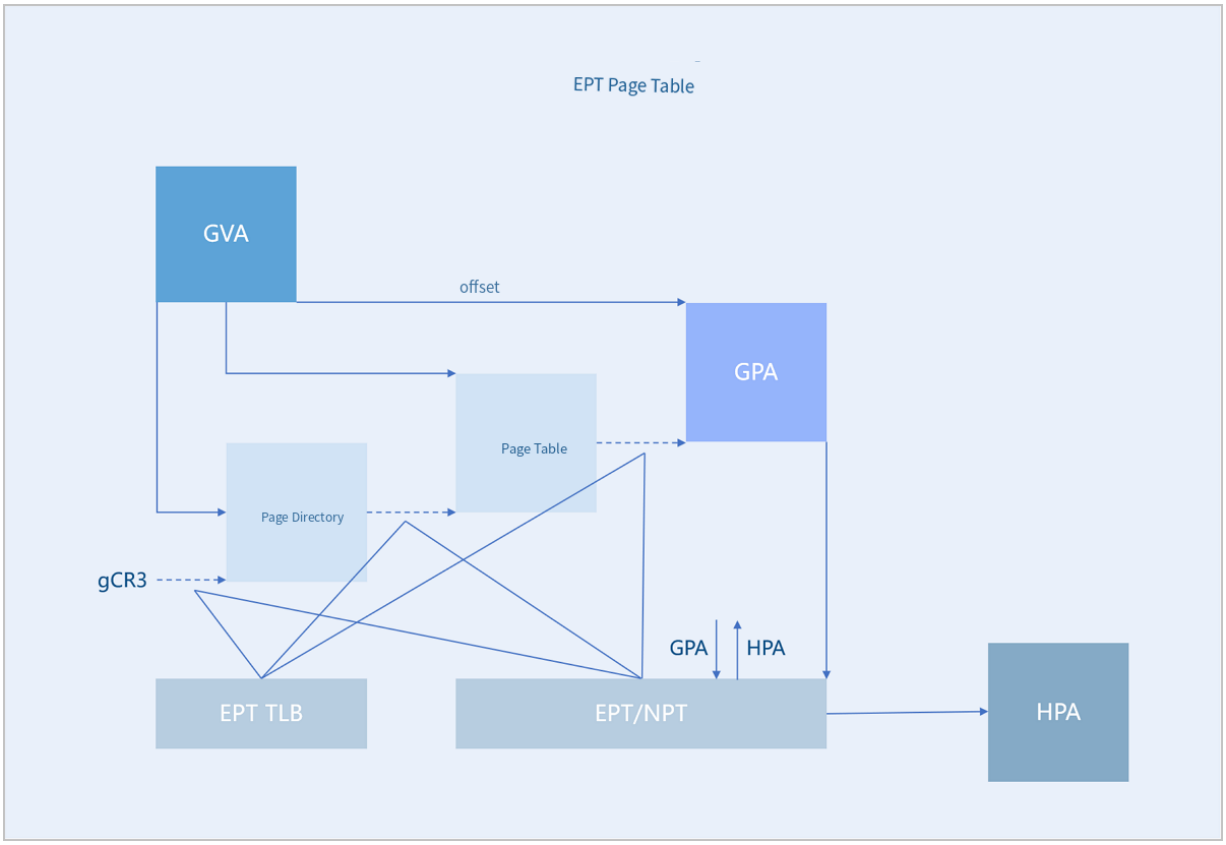
- maintaining the mapping relationship between the virtual machine physicaladdress and the host physical address.
- when a virtual machine accesses a virtual machine physical address, theaddress is converted to a host physical address based on themapping relationship.

Figure 4-5: GVA->GPA->HPA



Hardware-assisted memory virtualization uses extended page table (EPT)technology to convert virtual machine guest virtual addresses to hostphysical addresses with the help of hardware.

Figure 4-6: EPT Page Table



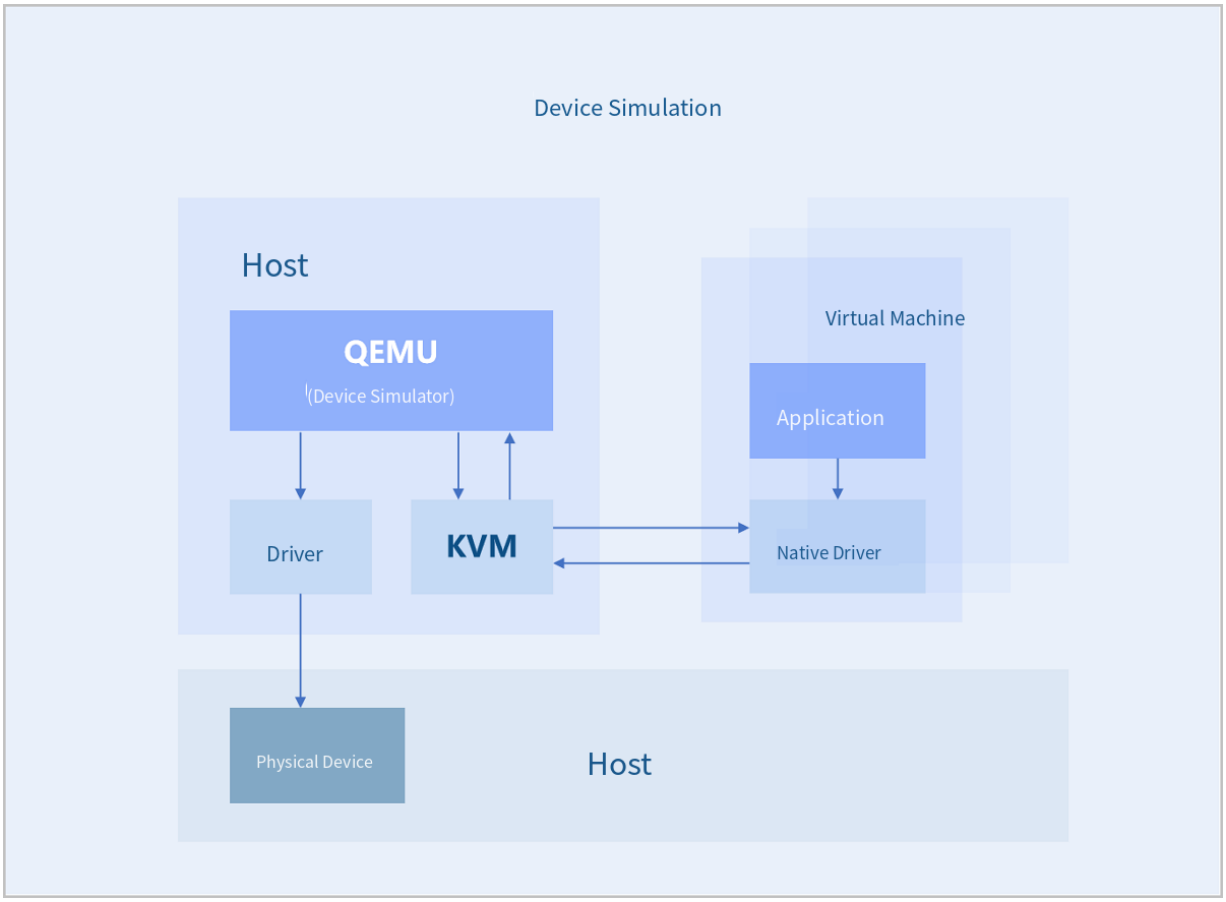
4.1.1.2.3 Device Virtualization

Three main types of device virtualization exist: Emulation, Passthrough, and Virtual Device.

Emulation

Emulation is implemented by using QEMU's emulation of devices. Emulation uses a device model provided by QEMU to simulate devices that are identical to physical devices. Therefore, in the VM operating system, you can use native drivers to use devices. However, emulation simulates only basic-function devices and does not support complex-function and complex-model devices. Fully emulated devices have good compatibility, but because they are purely software-based simulations, their performance is relatively low.

Figure 4-7: Emulation

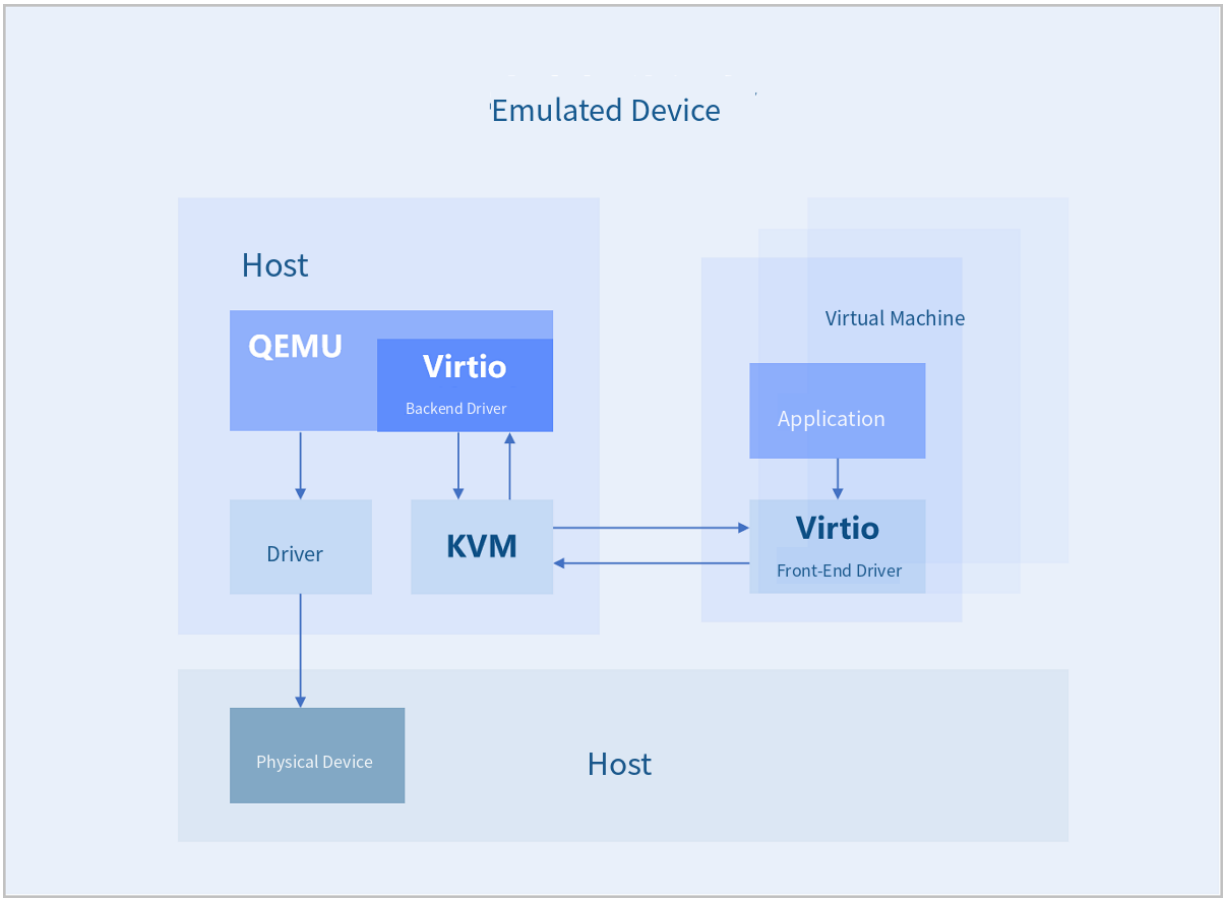


Virtual Device

A virtual device is implemented by using front-end and back-end drivers. The front-end driver is implemented in the VM and the back-end driver is implemented in the host. With a transaction-based communication mechanism, requests are sent directly to the back-end driver on the host from the front-end driver in the VM, which can greatly reduce the context-switch overhead. The performance is significantly improved compared with full emulation. However, the VirtIO back-end driver is implemented in QEMU, and the data flows multiple times between user space and kernel space during the IO processing. To further improve the performance, you can implement the VirtIO back-end driver in kernel space, which is called Vhost-kernel. With Vhost-kernel, data only needs to flow once between user space and kernel space during the IO processing, which can achieve better performance.

With the development of technology, data can be implemented in user space to obtain a more flexible form. Therefore, the Vhost architecture is improved to add the Vhost-user backend. With the DPDK and SPDK user-space function libraries, the performance is further improved.

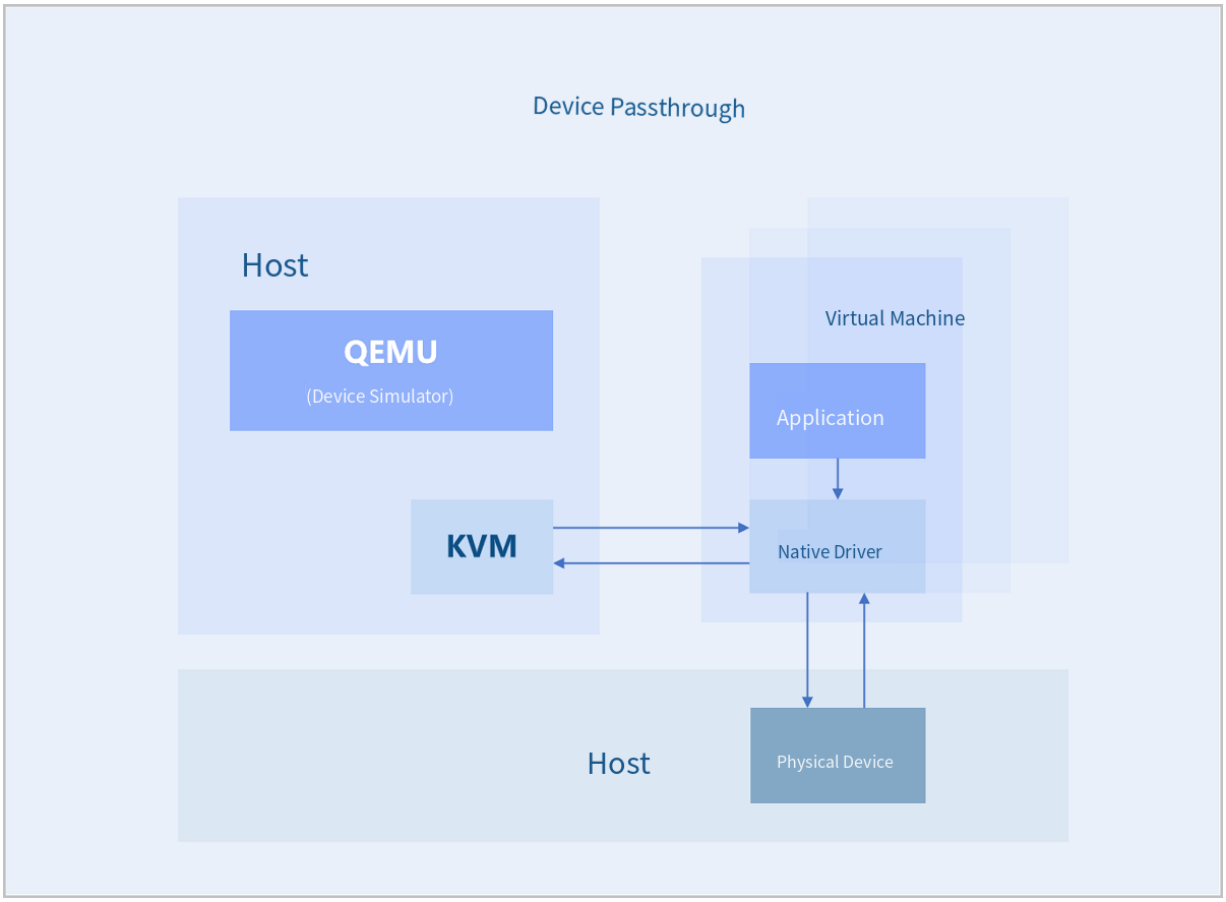
Figure 4-8: Virtual Device



Passthrough

Passthrough is a hardware-based device virtualization technology that maps PCI/PCIe physical devices to the address space of a VM. In a VM with Passthrough, you can use native device drivers to use devices directly, achieving performance close to that of physical devices. Physical devices passed through are exclusively owned by the VM and cannot be shared by other VMs.

Figure 4-9: Passthrough



4.1.2 Storage Virtualization

4.1.2.1 Overview

Storage virtualization pools and manages server storage resources, providing a variety of upper-layer storage interfaces for VM instances to flexibly allocate storage space in the resource pool according to their storage requirements. The virtualization platform supports connecting to both centralized and distributed storage.

4.1.2.2 Technical Features

4.1.2.2.1 Centralized Storage

Centralized storage refers to storing data on one or more servers in a central node. In a centralized storage, all businesses are deployed on the central node, and the central node manages data on remote nodes. Data access only needs to go through a controller.

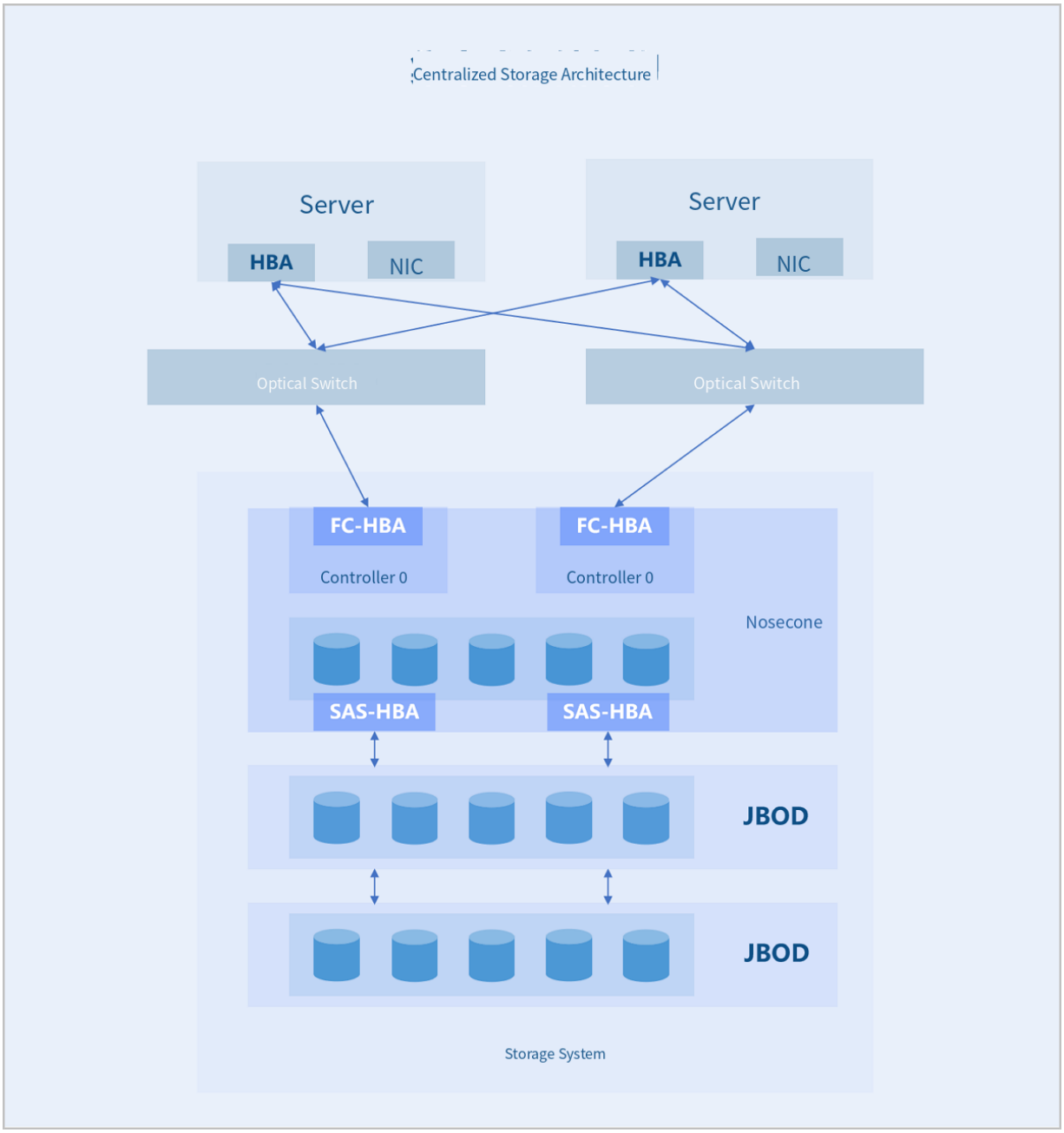
Centralized storage is classified into DAS, NAS, and SAN. You can select a storage type according to your data storage requirements.

Advantages of the centralized storage architecture:

- High performance and reliability: Data is stored on one or more servers in a central node, which avoids data dispersion risks and improves storage system performance and reliability.
- Scalability: Centralized storage can be scaled by adding network storage. The storage system can be extended to accommodate business growth.
- Security: Access control and data encryption and other security measures can be used to ensure data security and integrity.

On the centralized storage architecture, the head is the most core part. Typically, the head includes two controllers that back each other up to avoid hardware failures that make the storage system unavailable. In addition, the head includes front and back ends. The front end provides storage services for servers, while the back end is used to expand storage capacity. Through the back end, the head can connect more storage devices, thus forming a huge storage pool.

Figure 4-10: Centralized Storage Architecture



SAN Storage

The virtualization platform supports connecting multiple types of centralized storage. This topic mainly describes SAN storage. This method can directly use LUNs on SAN storage as storage pools and provide the storage pools for business virtual machines. Unlike data storage based on file systems, SAN storage has advantages such as easy deployment, flexible expansion, and excellent performance. Actual measurement data shows that SAN storage can fully leverage the

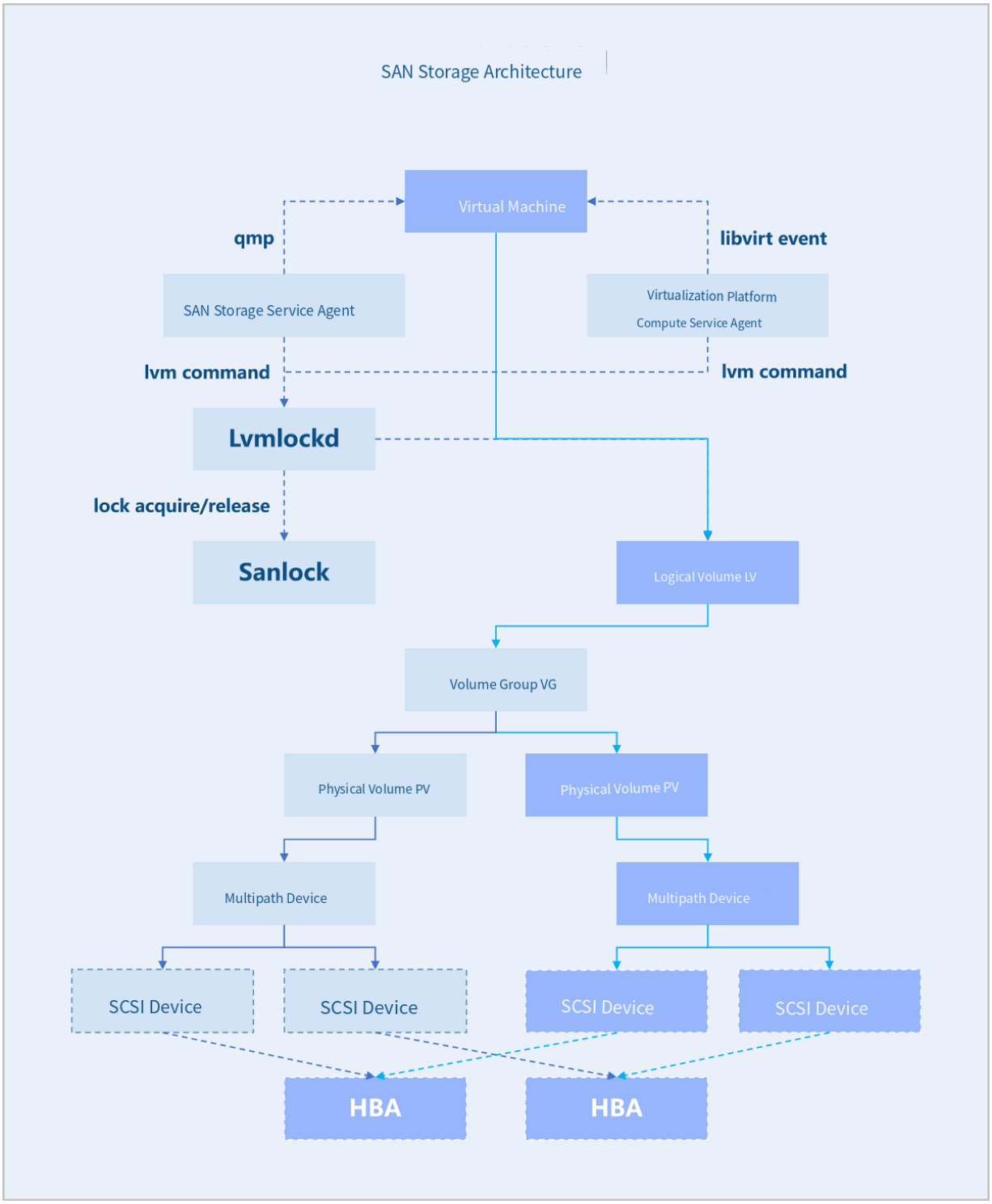
performance of physical disks. Currently, SAN storage supports iSCSI, FC, and NVMe-oF sharing access protocols.

How SAN storage works:

- Block-level access: SAN storage can divide data into blocks, each with a unique identifier. Servers can read or write specific data blocks through these identifiers.
- Parallel access: Multiple nodes can access different blocks in SAN storage simultaneously, thus achieving parallel read/write operations.
- Sharing: SAN storage can be shared across multiple nodes, allowing them to access the same data blocks simultaneously. This is important for distributed systems that need to share data.
- Fault tolerance: SAN storage usually provides data redundancy and fault recovery mechanisms to ensure data security and reliability.
- Scalability: SAN storage can be scaled by adding more storage nodes.
- Performance optimization: SAN storage usually adopts various technologies to optimize performance, such as caching and load balancing.

On the centralized storage, mapping a volume to a host. According to the connection mode of the storage controller, the multipath service on the host automatically aggregates multiple SCSI devices with the same WWID into a multipath device. The virtualization platform detects all hosts in the cluster to see whether they have multipath devices with the same WWID, and then creates a shared volume group (VG) for the selected volume in the cluster. The virtualization platform creates logical volumes (LV) based on the VG. Logical volumes actually correspond to hard disks, snapshots, and other resources in the virtualization platform. At the same time, to maintain storage cluster consistency, the virtualization platform also provides Sanlock, a shared storage lock management mechanism, for storage heartbeats, node management, and metadata management.

Figure 4-11: SAN Storage Architecture



Sanlock + Lvmlockd

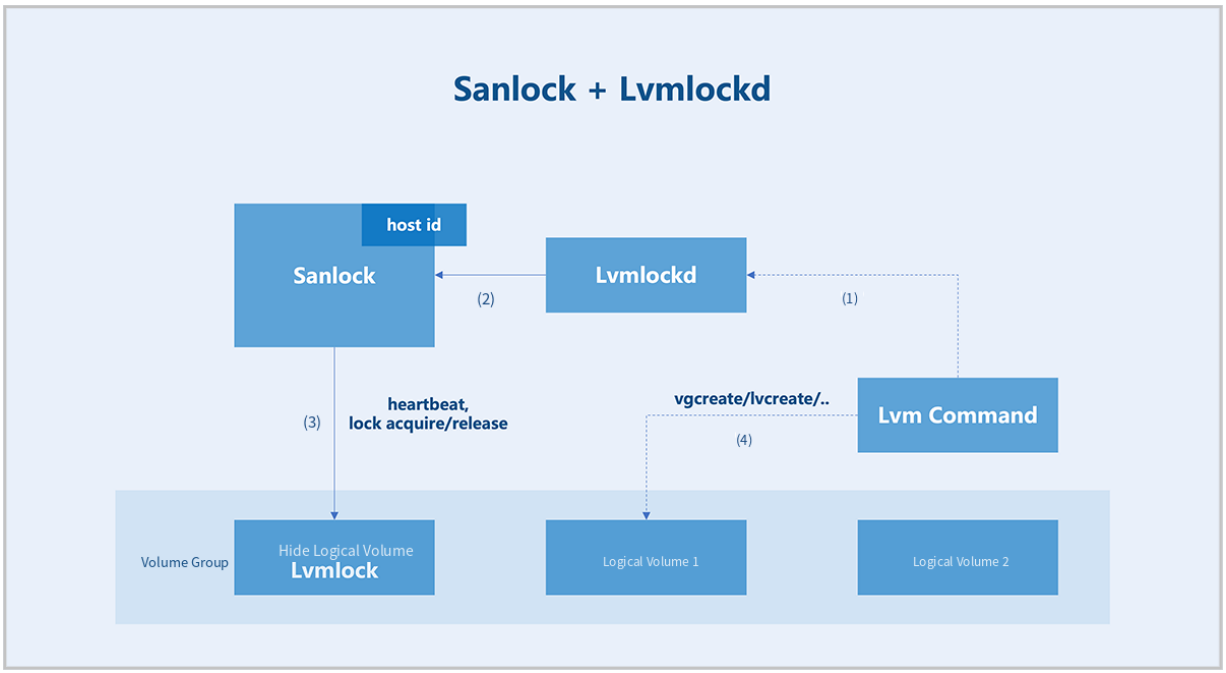
Sanlock (Shared storage lock manager) is a lease management mechanism based on several Lamport algorithms, including Delta Paxos, Disk Paxos, and others. This mechanism also implements cluster member management, heartbeat maintenance, arbitration, and informatio

n transmission. Then, the lease mechanism is converted into a lock that can be used by LVM: Lvmlockd.

After introducing Sanlock + Lvmlockd, most LVM commands executed by servers will first send requests to Lvmlockd. Lvmlockd will translate LVM operations into lock operations that Sanlock can recognize.

Different operations on different resources require different lock permissions. You can only perform operations on relevant resources after acquiring the permissions, thus ensuring metadata and data security. In addition, the entire process is distributed and does not require any intermedia te nodes to coordinate or centralized operations.

Figure 4-12: Sanlock + Lvmlockd



4.1.2.2.2 Distributed Storage

The virtualization platform supports multiple types of distributed storage. This topic describes the self-developed distributed storage that provides rich resource management features, including servers, hard disks, data disks, LUNs, storage pools, and storage buckets. By using these features, you can create storage pools with different types of data redundancy and storage buckets with specified storage policies and access permissions, thus quickly and cost-effectively building simple, stable, and efficient storage infrastructure.

Advantages of the self-developed distributed storage:

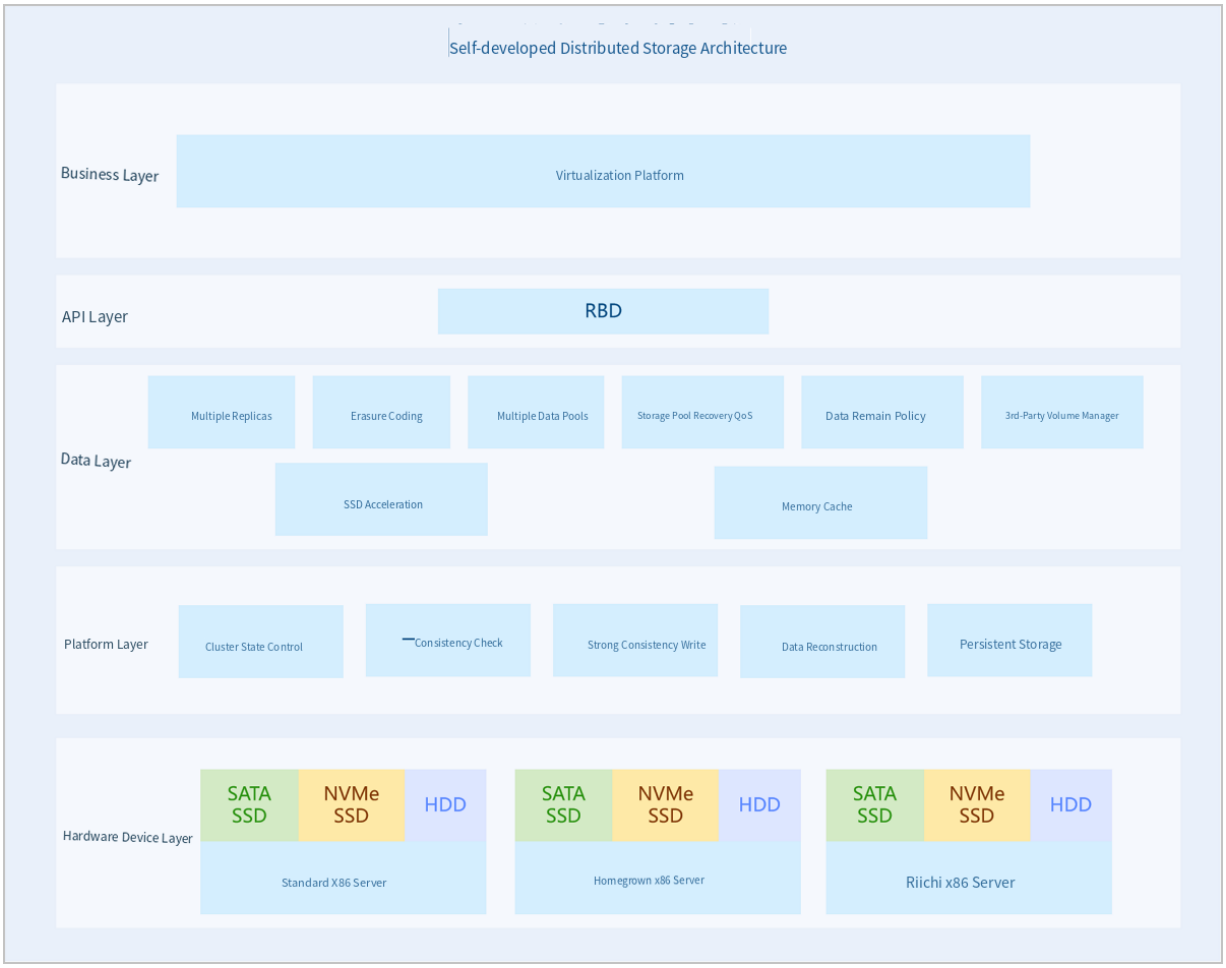
- Scale-out: Horizontally scale storage resources. In the era of Internet+, emerging businesses are thriving and data is increasing at geometric rates. Traditional centralized storage is challenged to cope with large-scale data growth. Distributed storage can be horizontally scaled by adding nodes to increase the storage cluster capacity and processing capabilities.
- High availability: Support multiple data redundancy policies, including replicas and EC. The replica policy allows online adjustment of storage pool replicas to copy data to multiple nodes . When a node or nodes fail, data can be obtained from other nodes, thus ensuring the high availability of the storage cluster. With the flexible fault domain policy, even if hardware failures, network interruptions, and other unforeseen situations occur, storage services can run normally.
- Performance optimization: Optimize data processing performance through self-developed caching acceleration technologies. By using enterprise-level NVMe SSDs and SATA SSDs to read and write to backend HDD devices, latency can be reduced and storage cluster I/O performance improved. In addition, storage I/O workloads can be evenly distributed to various nodes through load balancing technologies to ensure the full utilization of system resources.
- Ease of management: Provide a unified and convenient management interface and standard RESTful API interfaces to help administrators easily manage basic lifecycle and maintenance of physical resources and perform data operations and recovery.
- Cost-effective: Can be deployed based on generic hardware, greatly reducing hardware costs. In addition, due to its scale-out feature, you can dynamically adjust storage resources based on business needs and avoid resource waste, further reducing the cost of using the product.

The following describes the architecture of the self-developed distributed storage.

- Hardware layer: Supports standard x86 servers and domestic and independent intellectual property (IP) servers as the underlying hardware platform. At the same time, second-hand devices can be used to achieve resource pooling and protect existing IT assets.
- Platform layer: Solves the consistency hashing problem in storage services through the fault domain algorithm. In a nutshell, the fault domain algorithm adds physical deployment logic and cluster state parameters to the original hash algorithm, thereby minimizing data migration when cluster data is migrated. For example, based on the physical logical topology that you enter, the fault domain algorithm can selectively migrate data within servers or within the same rack /core switch, thus significantly reducing the resource consumption caused by the migration. In addition, data distribution based on cluster state can fully accommodate different storage devices.

- Data layer: Uses self-developed caching technologies to accelerate reads and writes to backend HDD devices with enterprise-level NVMe SSDs and SATA SSDs. Hot data is stored in high-speed SSD devices while cold data is dynamically adjusted to low-speed HDD devices based on intelligent policies. This not only improves the overall performance of the storage cluster but also provides a large storage space for upper-layer businesses.
- Interface layer: Provides virtual volume devices to operating systems, virtualization platforms, and databases through the RBD interface.
- Application layer: Connects to business virtualization platforms to provide cloud bases.

Figure 4-13: Self-developed Distributed Storage Architecture



shows the architecture of the self-developed distributed storage.

4.1.2.2.2.1 Data Storage

4.1.2.2.2.1.1 Cache Acceleration Scheme

A cache acceleration scheme generally refers to using SSD-class cache devices to accelerate HDD core devices in the data disk. The self-developed distributed storage uses the self-

developedZAS cache acceleration technology (ZAS cache module) to apply high-speed SSD devices for I/Ocaching for traditional HDD devices, caching hot data accessed frequently to the high-speed SSDdevice, which can significantly improve I/O performance, especially in scenarios with hot dataaccess features.

The ZAS cache module supports write-through and write-back caching strategies. In the write-throughmode, data is written into both the cache and the backend storage device to ensure dataconsistency, while in the write-back mode, most of the cache is used for buffering written dataand ensuring that dirty data is written in order to the backend storage device. The systemenables write-back strategy by default and allows switching the caching strategy on the fly.

The ZAS cache module has the following features:

- Flexible data migration

  The ZAS cache module automatically migrates data accessed frequently from the slow disk tothe SSD cache to improve access speed to these data. At the same time, the cache moduleautomatically migrates data accessed infrequently from the SSD cache to the slow disk torelease cache space based on the cache usage and space limit.

- Data protection

  When data I/O errors occur on the flash, the ZAS cache module first attempts to read datafrom the disk to recover the data or mark the cache item as invalid. For irrecoverableerrors, such as metadata or dirty data, the cache module automatically disables the cache.

4.1.2.2.2.1.2 Thin Provisioning

Logical volumes (LVs) provide thin-provisioned space that is greater than the actual physical storage capacity before data is written to the LVs. This mechanism makes the space more scalable and storage more efficient.

4.1.2.2.2.1.3 Set Storage Pool Recovery QoS

Quality of Service (QoS) is a technology used to solve I/O resource allocation issues. It can be used to help users limit the speed of I/O reads and writes, achieving rational resource allocation.

In the data node, there is an **op_shardedwq** queue to process various upper-level I/Os. This composite queue usually includes several sub-queues. After an I/O request is dequeued, it is interfaced with the disk through the ObjectStore interface. The main types of I/O include two categories: one is the business read/write I/O requests from the client, and the other is the I/O generated from internal activities of the storage system, including data node-to-node I/O requests, SnapTrim, Scrub, and Recovery, and so forth.

For self-developed distributed storage, a weighted priority queue (WPQ) is used. According to the above I/O classifications, each I/O is enqueued into corresponding sub-queues. When a prior queue is enqueued for the first time, it is created. When dequeuing, the priority level is determined by weight probability. The priority prior of each queue serves as its weight. The probability that a prior queue is selected is its weight over the total weight. However, even if a prior queue is selected, it does not necessarily dequeue a request. The size of the request to be dequeued also needs to be considered.

You can set the QoS for storage pool recovery. Different business types are provided with different recovery strategies, including the following three:

- Slow Speed: Slow speed prioritizes business bandwidth and has a long recovery time. During the recovery, if a hardware fault occurs again, the data security level may be reduced. We recommend that you select slow speed in production environments.
- Medium Speed: Medium speed prioritizes both business and recovery bandwidth and has moderate recovery time. In case of performance saturation, I/O latency may be increased.
- Fast Speed: Fast speed prioritizes recovery bandwidth and has the shortest recovery time. In case of performance saturation, business performance may be affected.

4.1.2.2.2.2 Data Protection

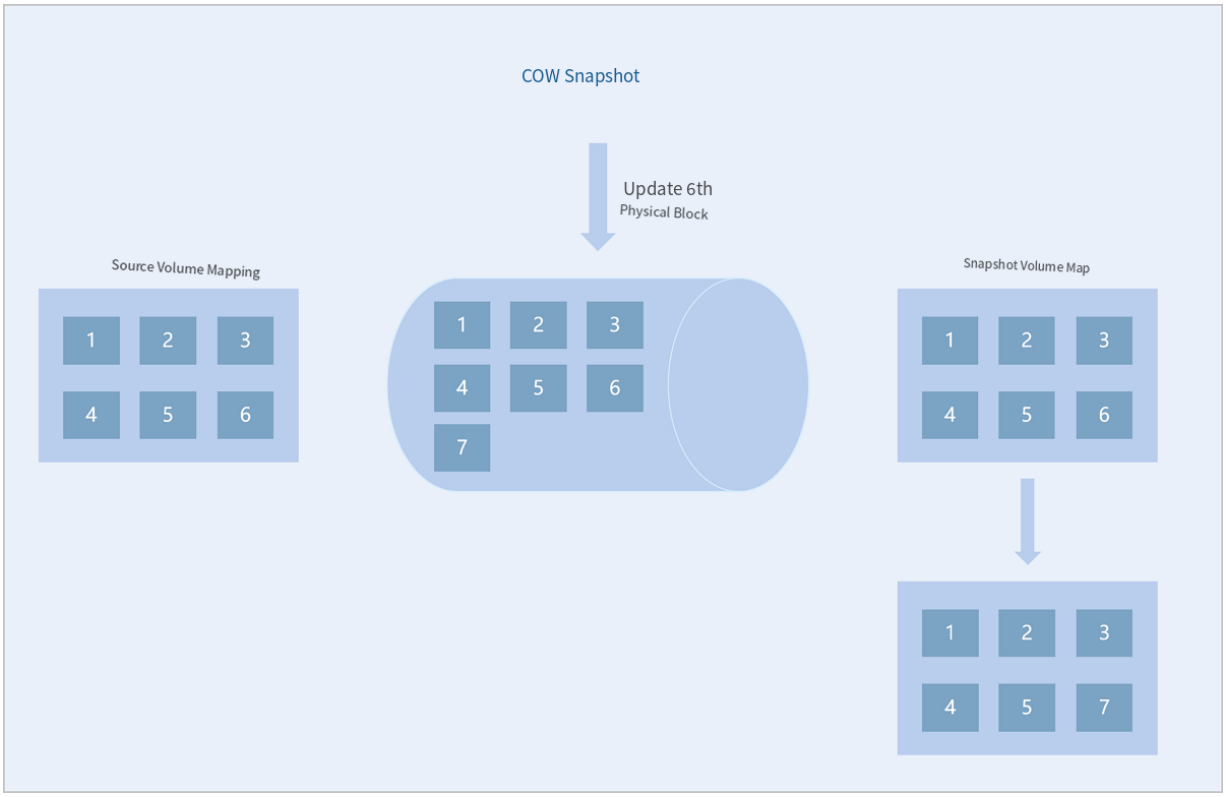
4.1.2.2.2.2.1 Volume Snapshot

A snapshot is a key feature of self-developed distributed storage that allows you tocreate point -in-time copies of block storage volumes without interrupting services orharming production environments.

Snapshot Creation

The self-developed distributed storage uses the COW (Copy-On-Write) snapshottingtechnology. When a COW snapshot is created, the Cloud creates a read-only replica ofthe original LUN in the storage cluster. Data is actually copied only when the data inthe original LUN or the snapshot is changed. This way, the original data is retainedwhile data redundancy is avoided. At the same time, the Cloud records the relationshipbetween the LUN and the snapshot as well as metadata of each snapshot, including thecreation time, size, and ID of the parent snapshot. The metadata records enable theCloud to locate the snapshots accurately when recovering data.

Figure 4-14: COW Snapshot



Data Rollback and Recovery

You can create new block storage volumes based on a snapshot of an existingvolume. In this way , the new volume is initialized with the state of the snapshot bydefault. In addition, in case of data breaches or data loss, you can also rollback avolume to the state of a specified snapshot.

4.1.2.2.2.2.2 Data Duplication

Overview

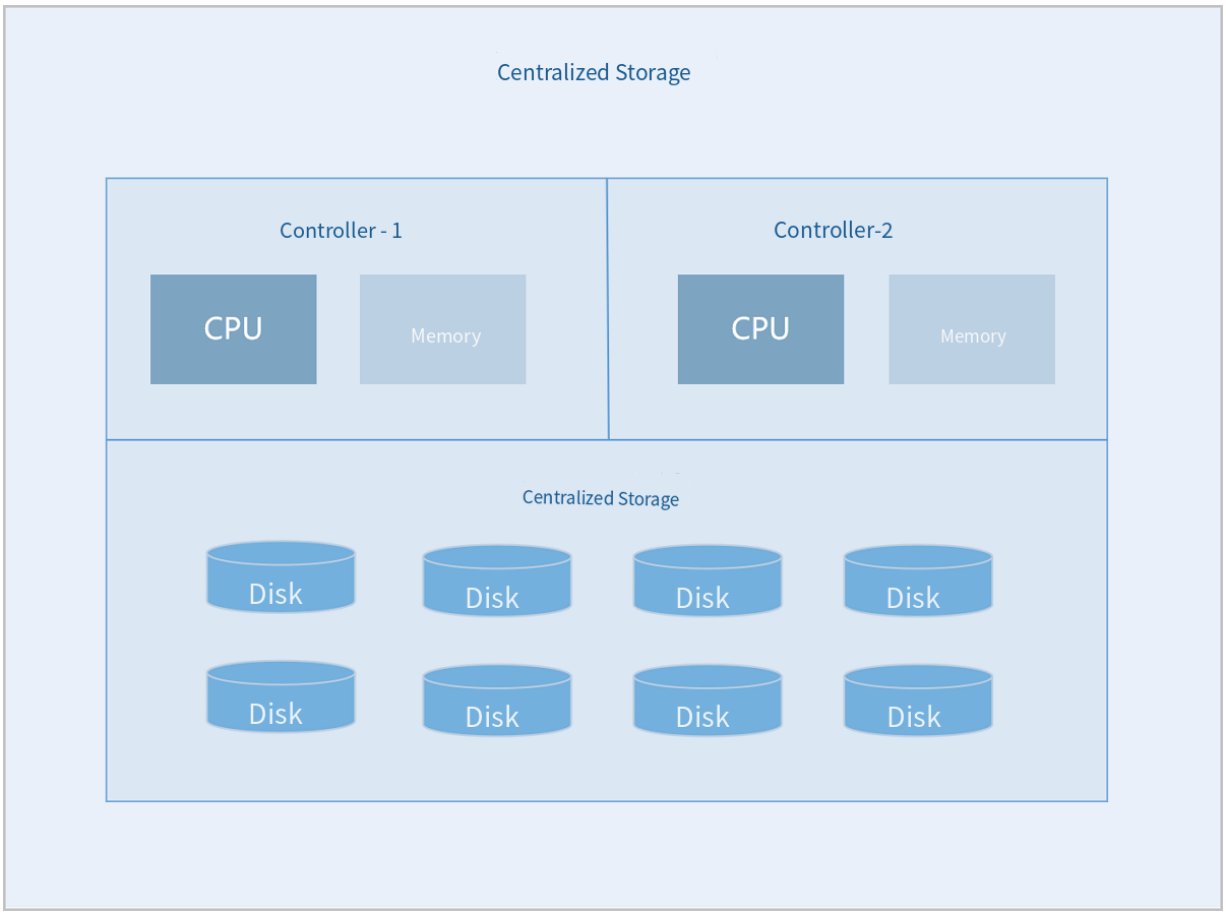
Data redundancy is the cornerstone of achieving high availability, data protection, and business continuity in storage systems. As data volumes grow, application scenarios become more complex, and potential threats such as hardware failures, network interruptions, and human errors increase, an appropriate data redundancy strategy can serve as a robust data protection barrier , ensuring business continuity and maximizing data value. This chapter briefly compares the data redundancy features of centralized and distributed storage and then elaborates on the two core data security strategies—redundant copies and erasure coding (EC)—adopted by our self-developed distributed storage solution.

Comparison between Distributed Storage and Centralized Storage

Centralized Storage

Traditional centralized storage uses controllers and disk enclosures to provide data management and read/write capabilities. Generally, dual controllers are used for redundancy, with some high-end storage systems featuring multiple controllers. Storage space can be provided through controller-native disk slots or by connecting extended disk enclosures. Traditional centralized storage typically employs RAID technologies such as RAID 5, RAID 6, and RAID 10 to protect data.

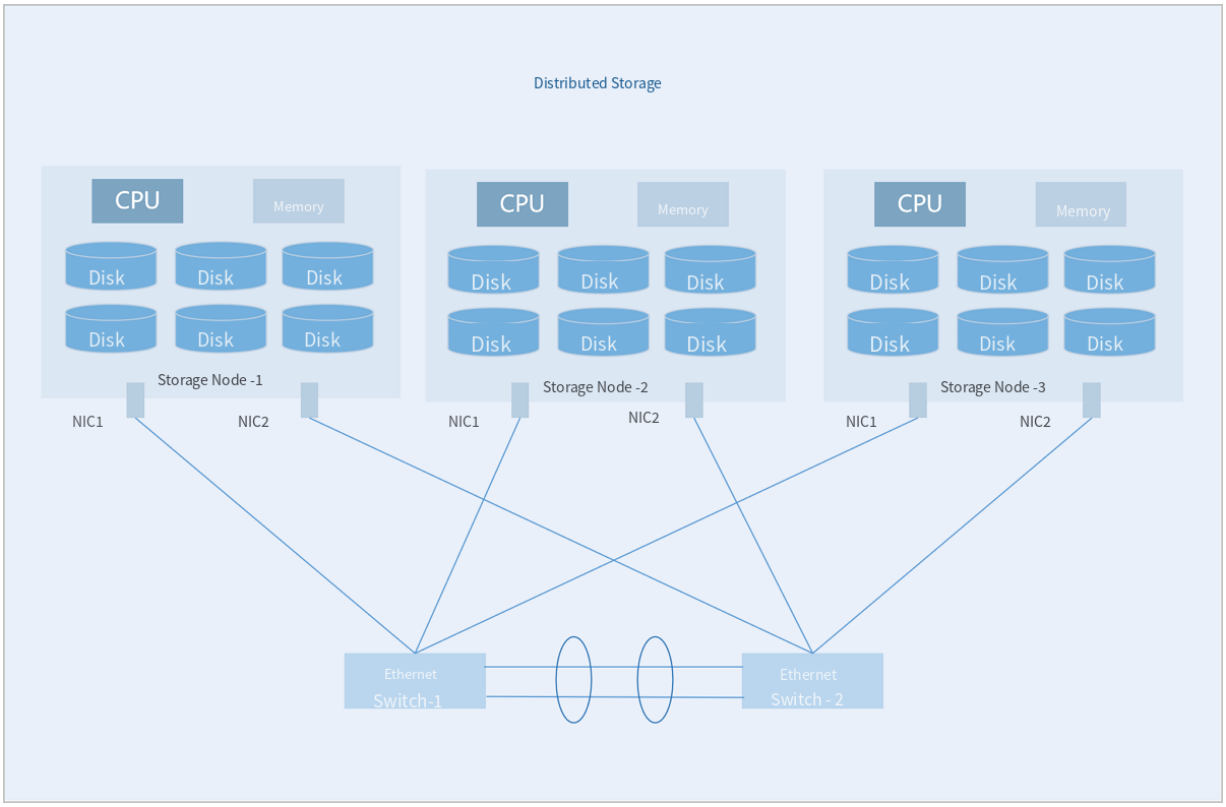
Figure 4-15: Centralized Storage



Distributed Storage

Distributed storage adopts a non-centralized networking method, with each storage node capable of providing computing and storage resources for flexible scalability and larger storage capacity. Storage nodes are interconnected via generic Ethernet switches and offer a unified storage resource pool to upper-layer businesses. Moreover, distributed storage supports horizontal scaling, allowing a single cluster to scale up to thousands of nodes for EB-level capacity, suitable for scenarios involving massive data storage.

Figure 4-16: Distributed Storage



Centralized Storage vs. Distributed Storage

- **Node-to-Node Replication**: Distributed storage supports node-to-node replication, such as 3 replicas that can tolerate the failure of two nodes without data loss, whereas RAID technologies are limited to disk redundancy within a single node.
- **Global Hot Standby and Data Recovery**: Unlike RAID, distributed storage does not require dedicated hot standby disks, as all disks participate in data recovery, offering significantly higher efficiency. Additionally, distributed storage does not require additional hardware support, unlike RAID, which necessitates independent RAID cards.

Replica

Definition

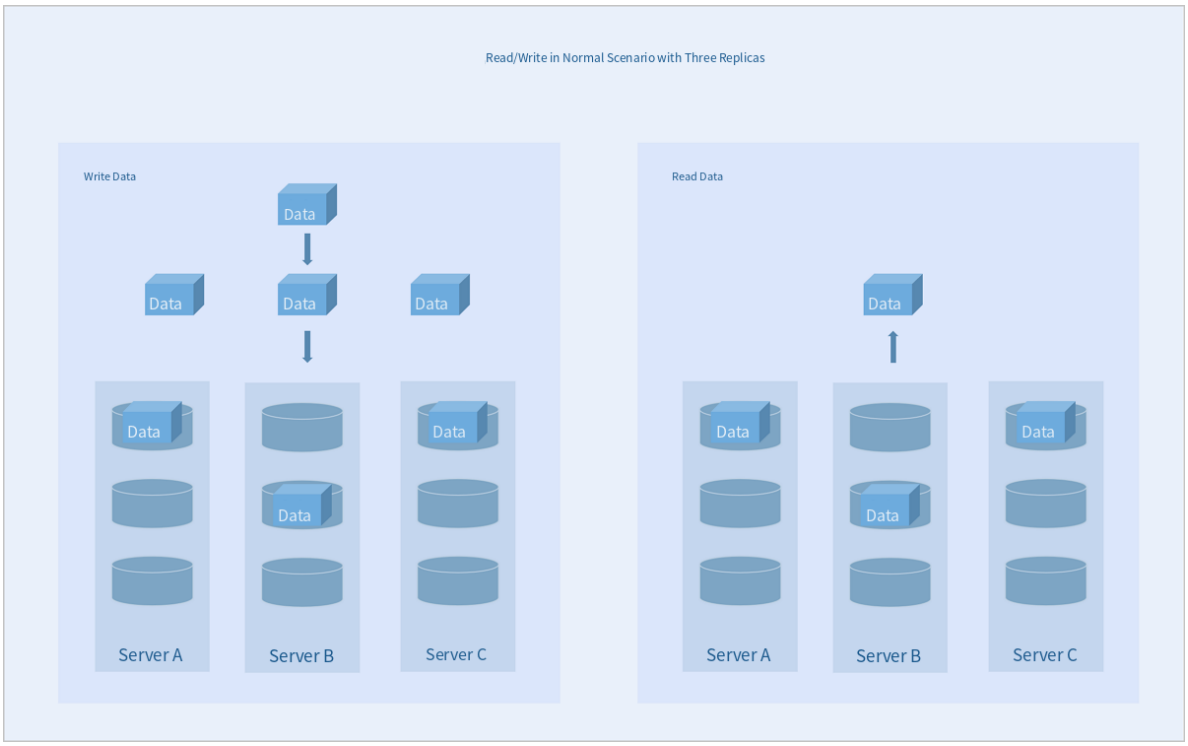
A data protection technique that creates identical data on different nodes to achieve data redundancy and high availability. When a node fails, data can be recovered from the replica on another node. You can set 2–6 replicas as needed. We recommend that you set 3 replicas in a production environment.

Read/Write Principle

- **Normal Read/Write**

  With Server-Level 3 Replica Policy, when data is written, the system duplicates the data into 3 identical replicas, which are stored on the data volumes of 3 servers respectively. When data is read, the system reads the data from one of the servers and returns the data to the user.

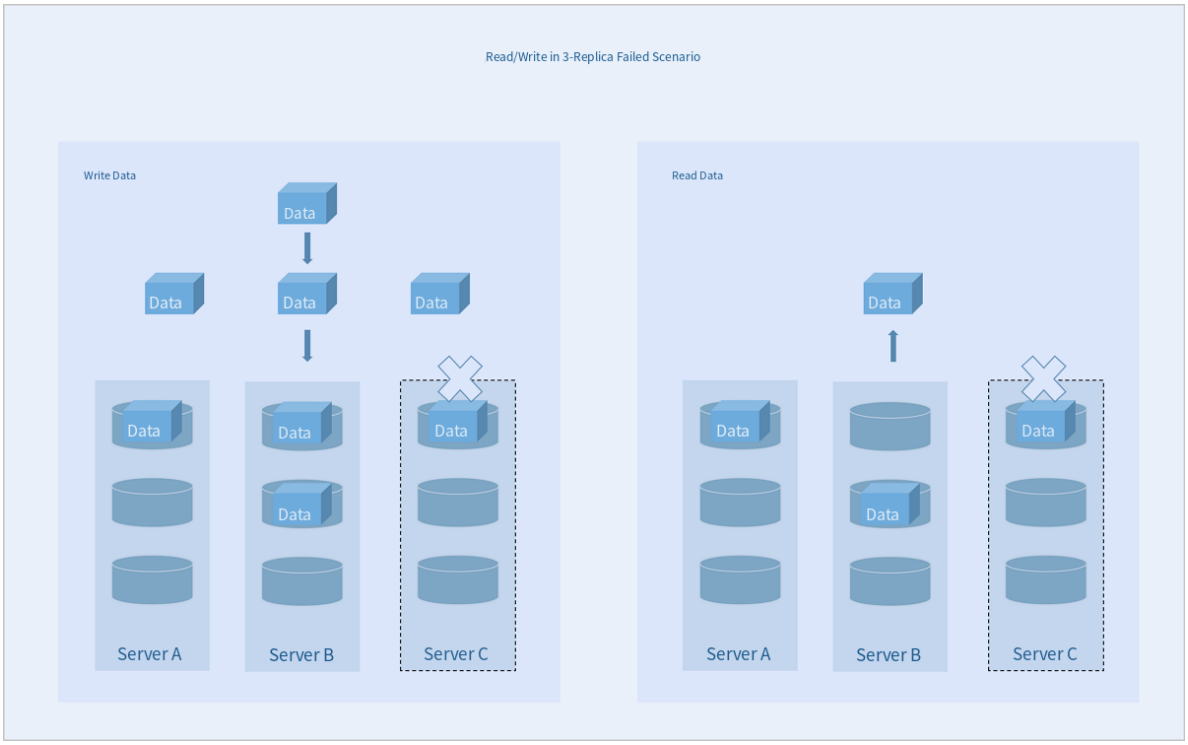
  Figure 4-17: Normal Read/Write with 3 Replicas



- **Failed Read/Write**

  With Server-Level 3 Replica Policy, after a server fails, the system stores the replicas on the remaining 2 servers. When data is read, the system reads 1 replica from the remaining 2 servers and returns the data to the user.

Figure 4-18: Failed Read/Write with 3 Replicas



Deduplication and Coocation

Overview

Erasure coding (EC) is a data protection technology that divides data into K data chunks and generates M parity chunks by using a fault-tolerant algorithm. This approach enables data recovery in case of failures.

- Standard EC (K+M): K represents the number of data chunks, and M represents the number of parity chunks. This EC strategy indicates that the data can work as expected if up to M fault domains fail.
- Folded EC (K+M:B): K represents the number of data chunks, M:B represents the number of parity chunks. This EC strategy indicates that the data can work as expected if up to M disks or B fault domains fail.

EC Policy

The following EC policies are provided:

EC Policy		Disk Utilization
Recommended	2+1	66.67%

EC Policy		Disk Utilization
	4+2	66.67%
	8+3	72.73%
	4+2:1	66.67%
	8+2:1	80.00%
	16+2:1	88.89%
Custom		K/(K+M)

Standard EC

Read/Write Principle

- Normal Scenario

  Standard EC (K+M): Using a server-level 4+2 EC strategy as an example, when writing data, the system divides the data into 4 data fragments of the same size and generates 2 parity fragments of the same size using the EC algorithm. Then, the system randomly stores these 6 fragments on 6 servers. When any 2 servers fail, the data can still be used normally. When reading data, the system reads data blocks from different disks of 4 servers and assembles these data blocks into complete data before returning the data to the user.

Figure 4-19: Standard EC (4+2) Normal Scenario Write Data

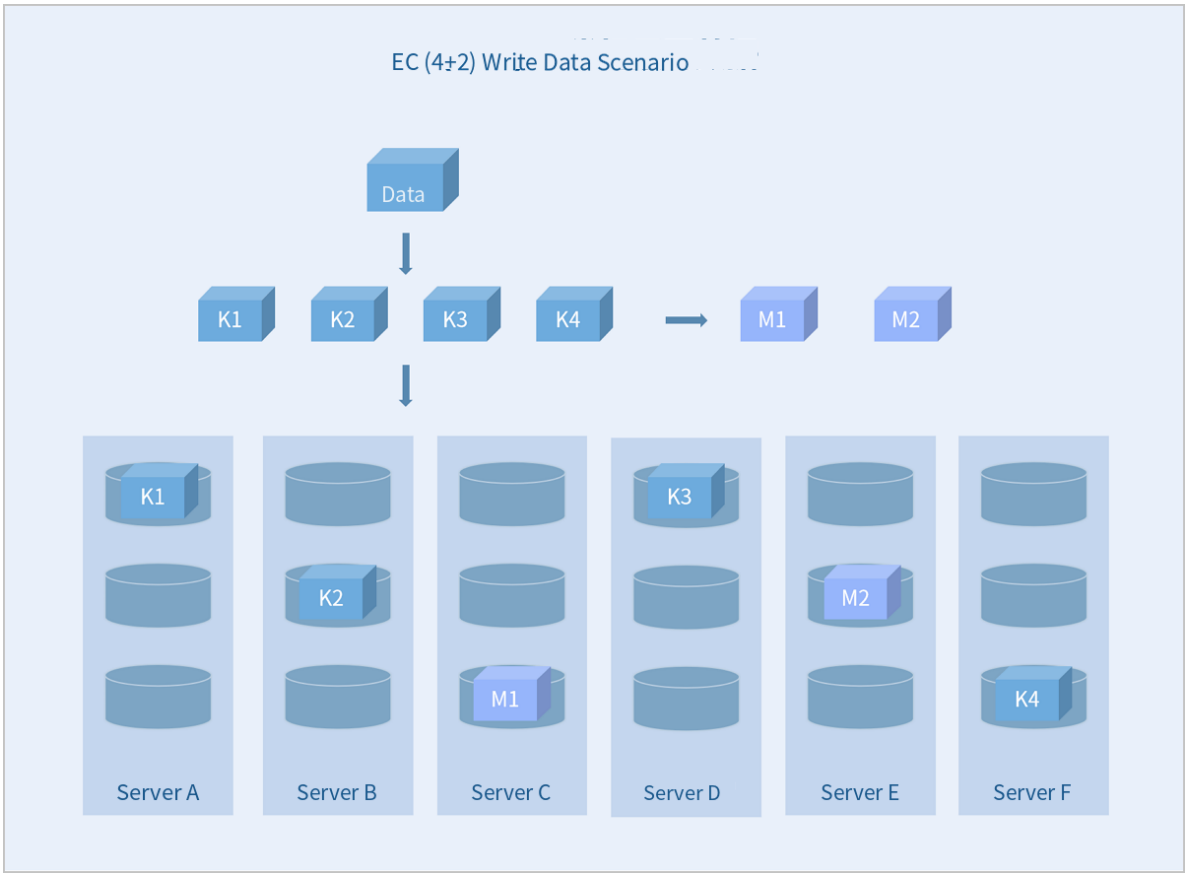
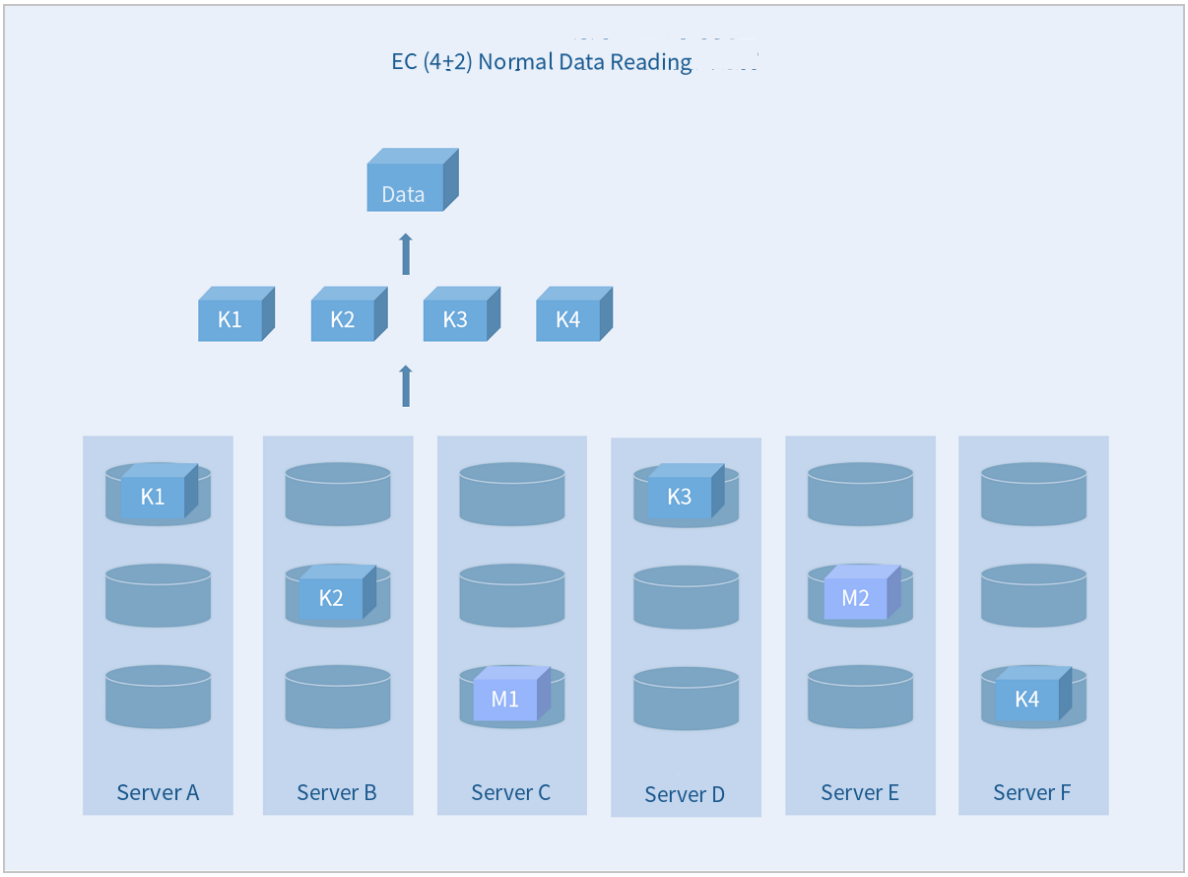


Figure 4-20: Standard EC (4+2) Normal Scenario Read Data



- Failure Scenario

  Standard EC (K+M): Using a server-level 4+2 EC strategy as an example, when the number of remaining servers after a failure is less than K+M, the system stores newly written data on the remaining servers before the failure recovery to ensure that I/O is not interrupted and the data reliability level is not reduced. After the failure recovery, the data redundancy strategy returns to K+M. When reading data, the system reads data from other normal servers and recovers the data using the EC algorithm before returning the data to the user.

Figure 4-21: Standard EC (4+2) Failure Scenario Write Data

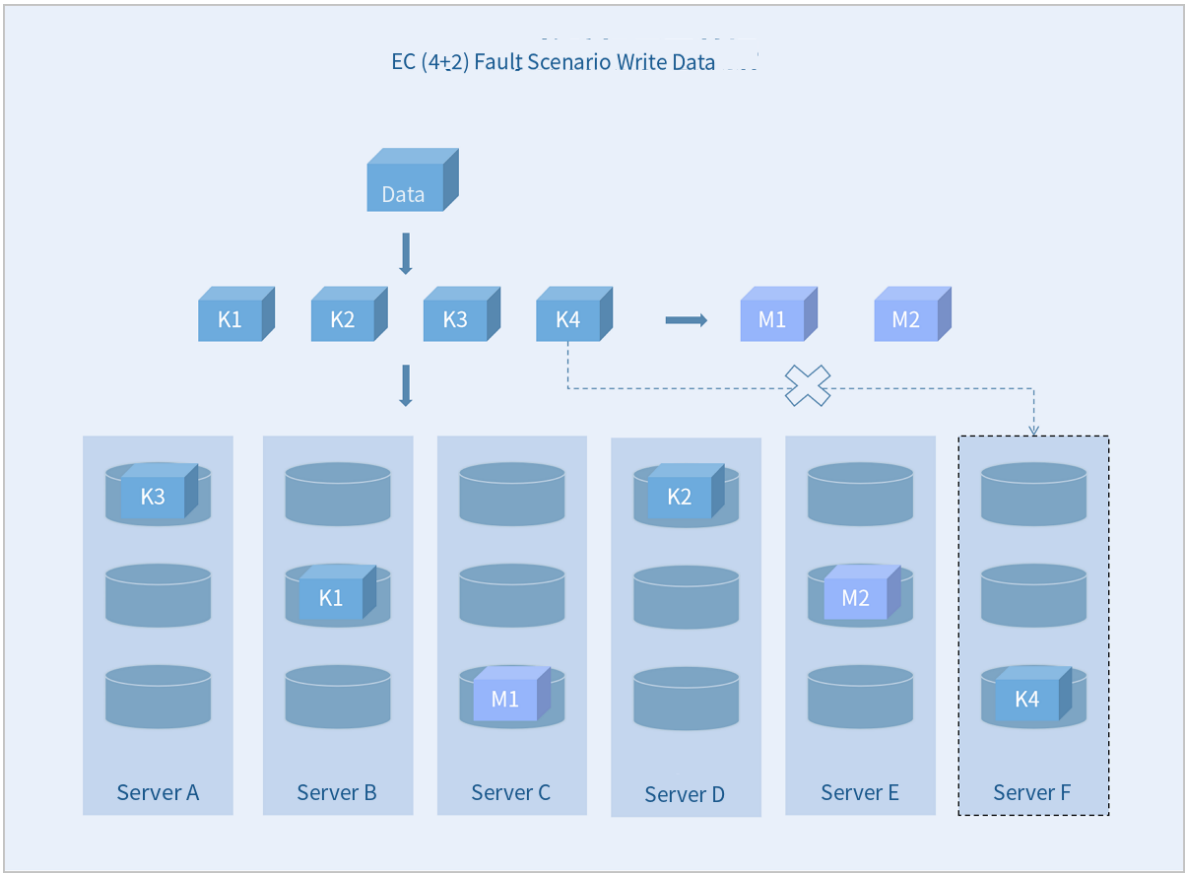
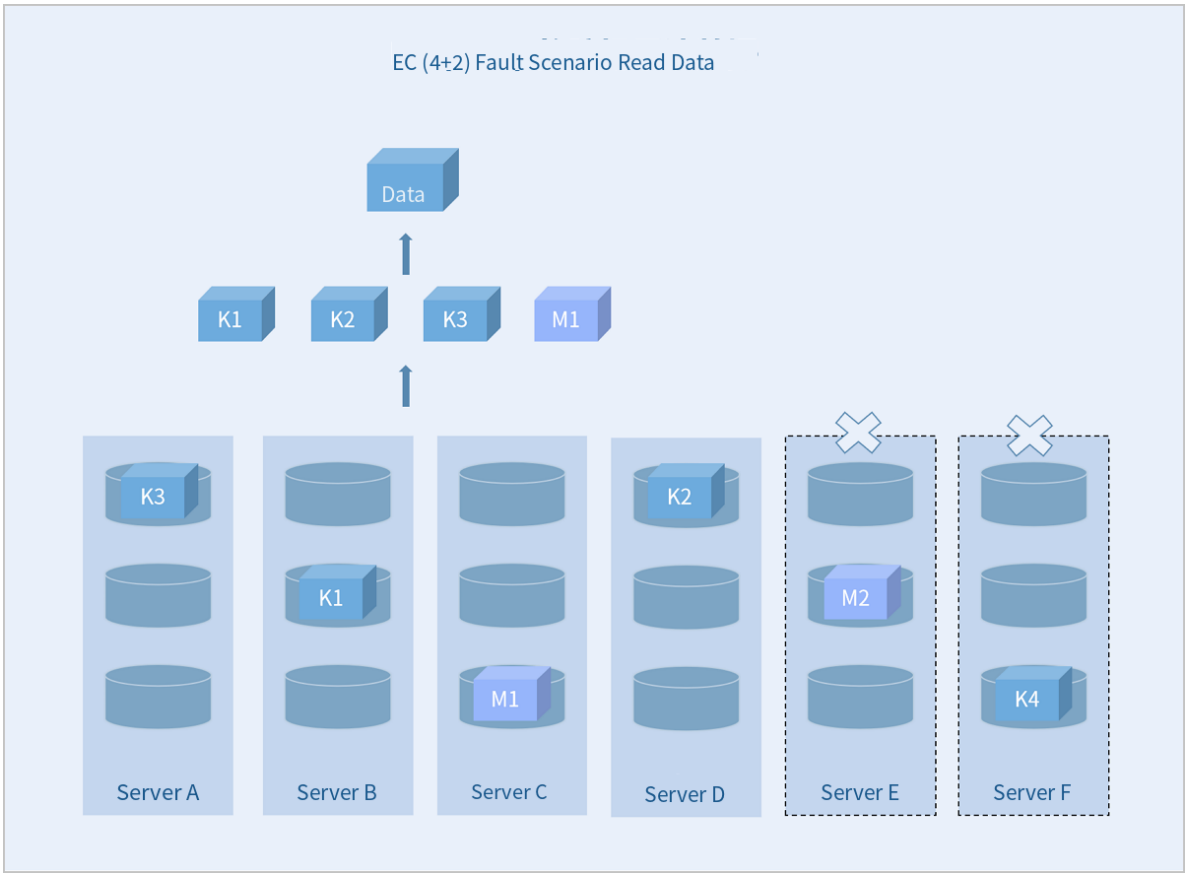


Figure 4-22: Standard EC (4+2) Failure Scenario Read Data



FEC

FEC, also known as sub-node EC, is a commonly used data redundancy technology. Unlike standard EC with a K+M ratio, the typical ratio for FEC is K+M:B, where B is usually 1. Similar to standard EC, FEC ensures high data reliability while maintaining high disk utilization.

For example, the minimum fault domain of standard EC is a storage node, so the minimum mode also requires at least 6 nodes. In contrast, the 4+2:1 ratio of FEC can meet data redundancy requirements with only 3 storage nodes.

Moreover, self-developed distributed storage allows you to scale up or down FEC, just like standard EC. You can also convert FEC to standard EC if it meets the fault domain requirement.

Write and Read Principle

- Normal Read/Write

  FEC (K+M:B): Using the server-level 4+2:1 FEC strategy as an example, when writing data, the system divides the data into 4 data slices of the same size and generates 2 parity slices

of the same size through the verification algorithm. Then, the system randomly stores these 6 slices into 5 servers. When any 1 server fails, the data can still be used normally. When reading data, the system reads data blocks from different disks of 3 servers and assembles these data blocks into complete data before returning it to the user.

Figure 4-23: FEC (4+2:1) Write Data in Normal Scenario

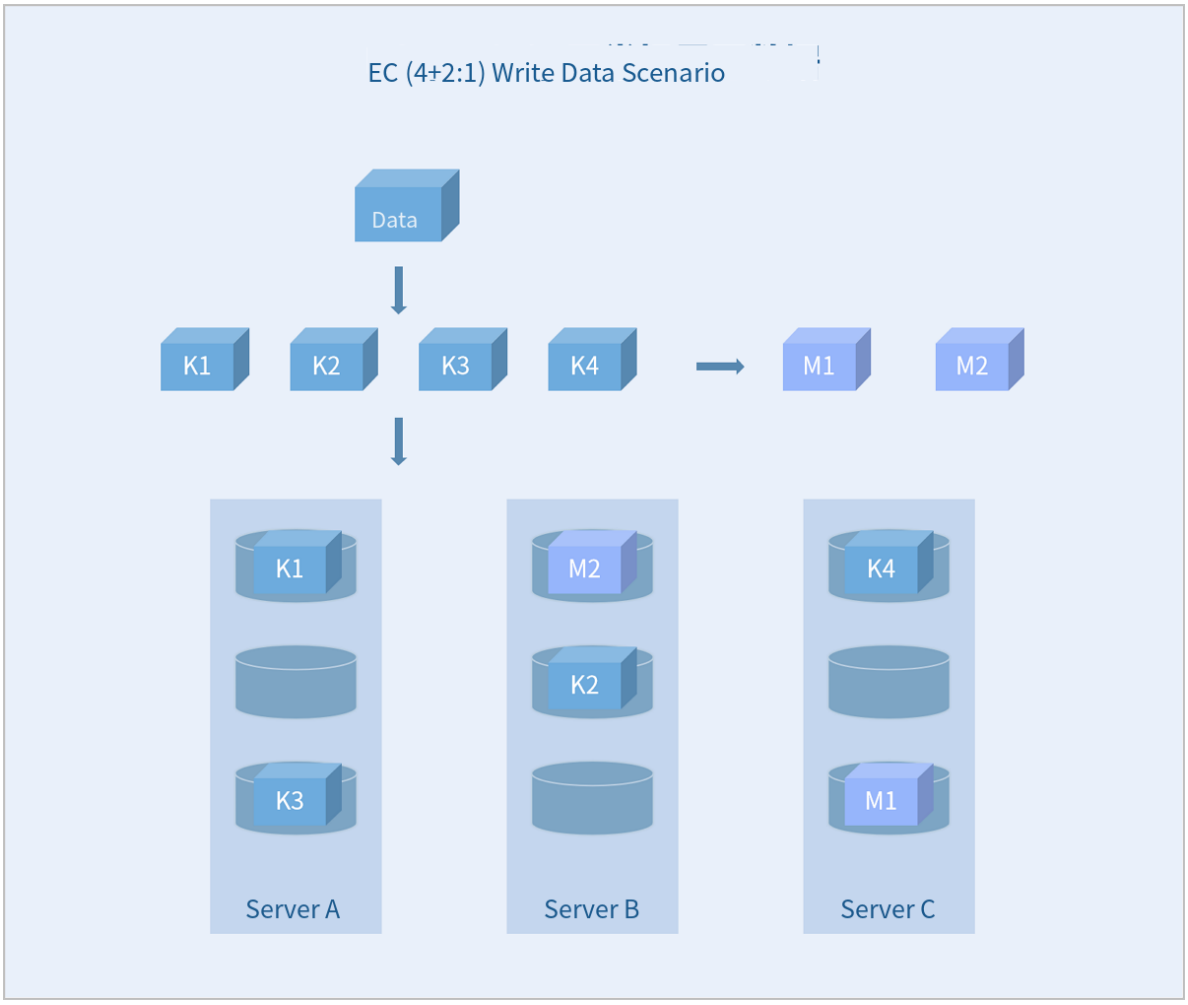
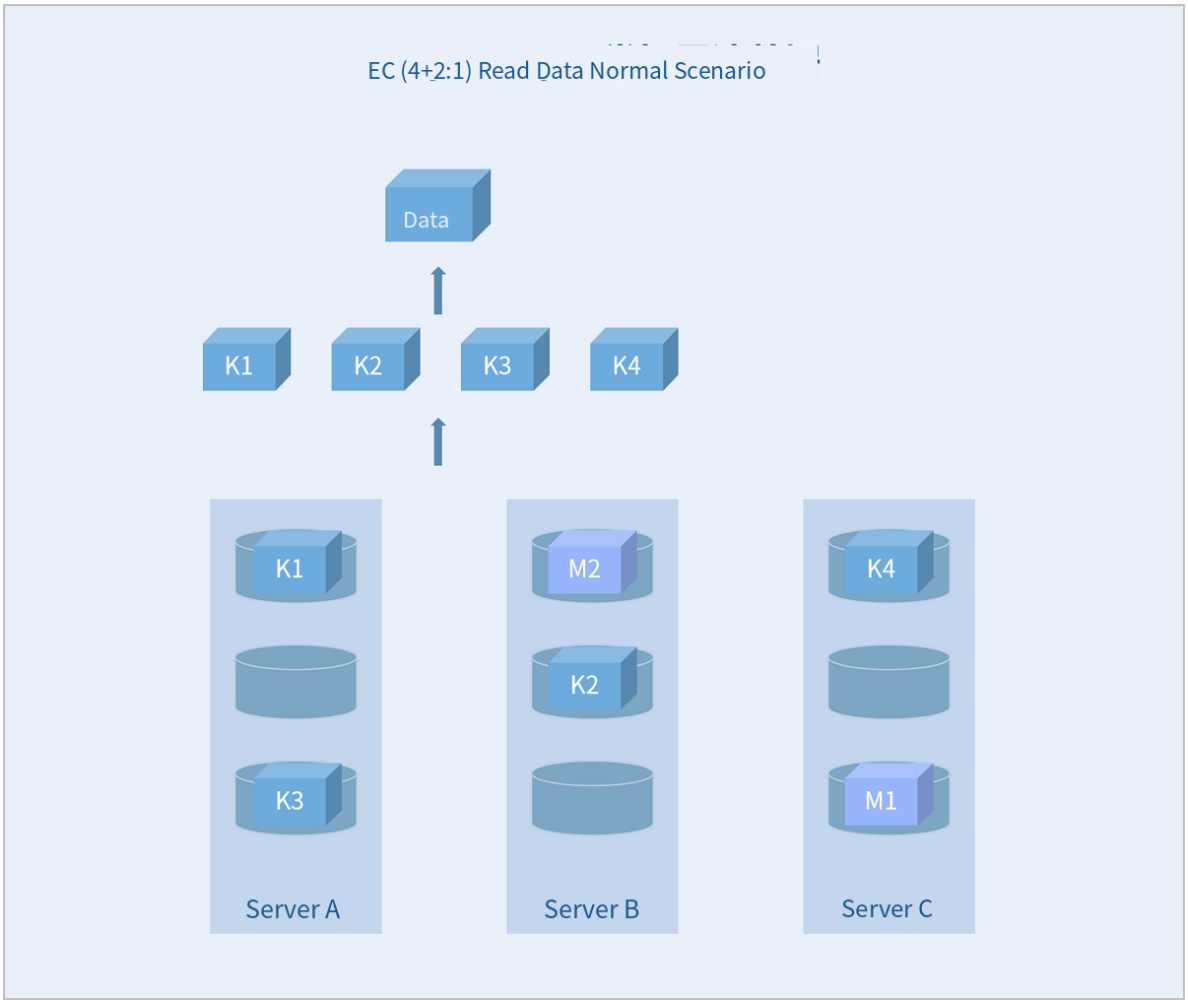


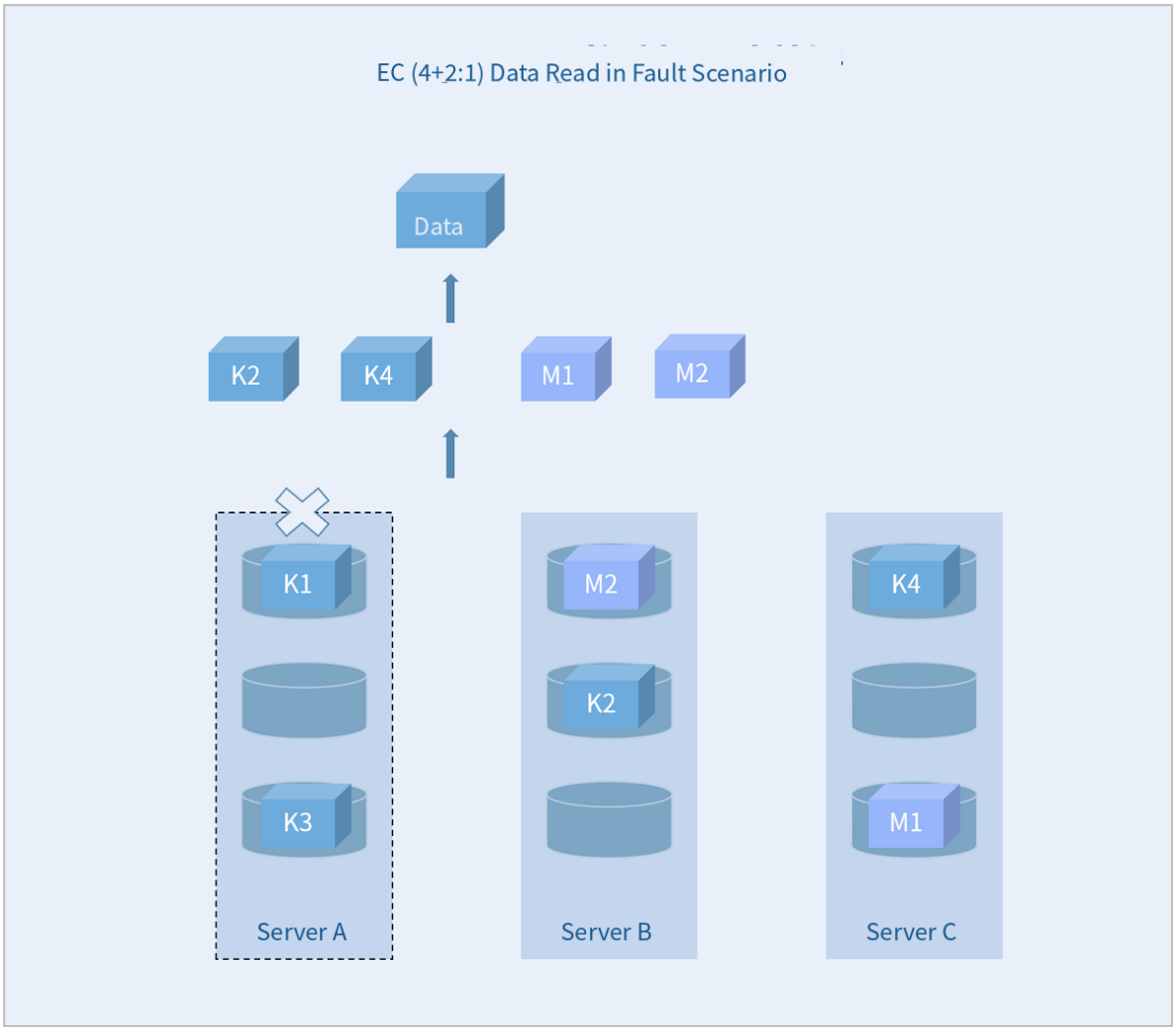
Figure 4-24: FEC (4+2:1) Read Data in Normal Scenario



- Read/Write in Fault Scenario

  FEC (K+M:B): Using the server-level 4+2:1 FEC strategy as an example, when a server or M disks fail, the system will still write the data into the remaining normal servers with K+M data slices and parity slices. When reading data, the system reads data from other normal servers and recovers the data through the verification algorithm before returning it to the user.

Figure 4-25: FEC (4+2:1) Read Data in Fault Scenario



Comparison between Replica and Erasure Code

According to your business requirements, you can choose replica or erasure code. These two strategies have their own advantages in different scenarios:

- **Space Efficiency**: The efficiency of erasure code is higher than that of replica. For example, the space efficiency of 4+2 EC strategy is about 66%, while that of 3x replica is 33.3%.
- **Read/Write Performance**: The read/write performance of replica and erasure code differs significantly under small data read/write scenarios, but the difference between these two strategies will gradually decrease under large data read/write scenarios. When data is written, erasure code involves data validation and may cause write penalties. In addition, because data is stored across multiple nodes, a high data latency of a node may seriously affect the read/write performance. While reading data, replica only needs to read one object and does not involve data splicing across nodes.

- **Reconstruction Performance**: Generally, the reconstruction performance of replica is better than that of erasure code, because replica reconstruction only involves simple data copying without data validation. While erasure code reconstruction involves a reverse validation computation, thus consuming more read/write data and CPU computing resources.
- **Tolerant Capacity**: These two strategies have their own advantages and disadvantages. Regarding replica, multiple replica technologies allow up to (replica number - 1) unmonitored node failures without data loss. While regarding erasure code, an example of 4+2 strategy allows up to 2 unmonitored node failures without data loss.

4.1.2.2.2.2.3 Failure Domain Isolation

A failure domain is the minimum unit for distributing cluster data. When storing data, different copies or slices of a file will be stored in different failure domains based on the data redundancy policy. This mechanism ensures data security by allowing a certain number of failure domains to fail without losing data. The following three data redundancy levels are supported:

- Server Level: Each server in the cluster is a failure domain. Different copies or slices of a file are stored in different servers.
- Chassis Level: Each chassis in the cluster is a failure domain. Different copies or slices of a file are stored in different chassis. We recommend that you select this level if your cluster has a large scale and many chassis.
- Facility Level: Each facility in the cluster is a failure domain. Different copies or slices of a file are stored in different facilities. We recommend that you select this level if your cluster has a large scale and many facilities.

Figure 4-26: Server Level Failure Domain

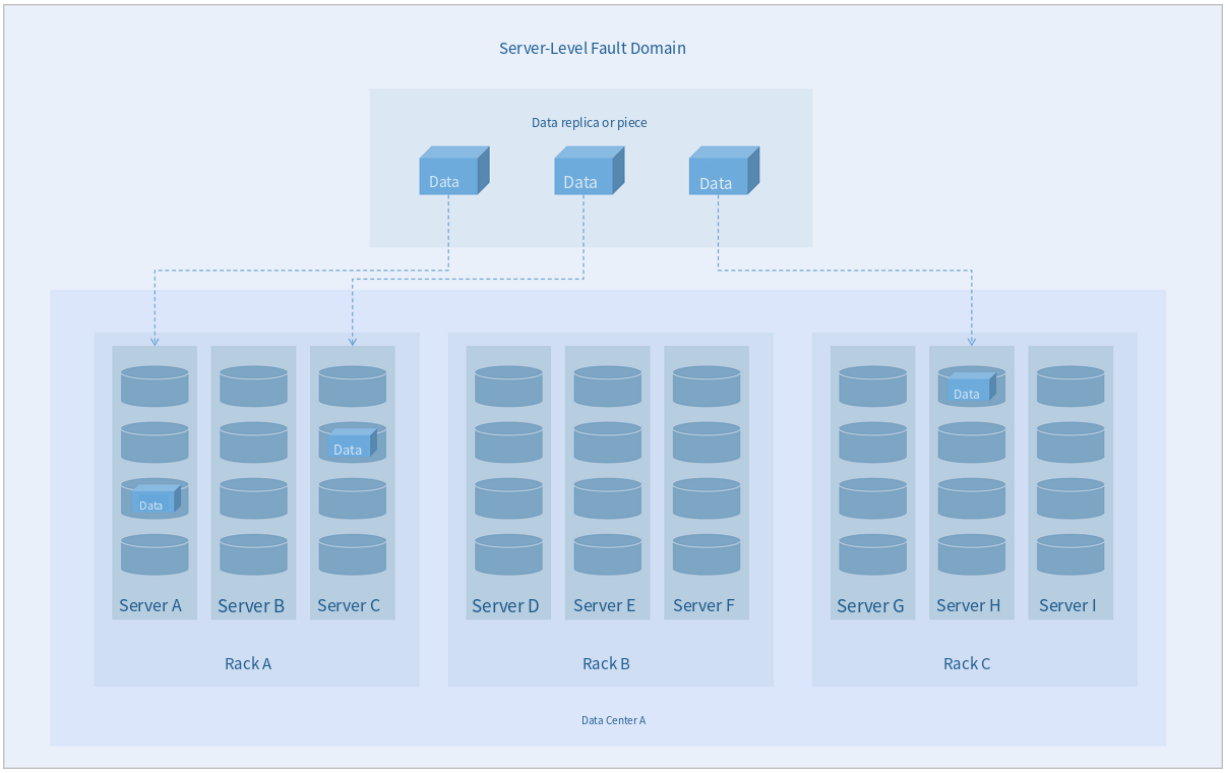


Figure 4-27: Chassis Level Failure Domain

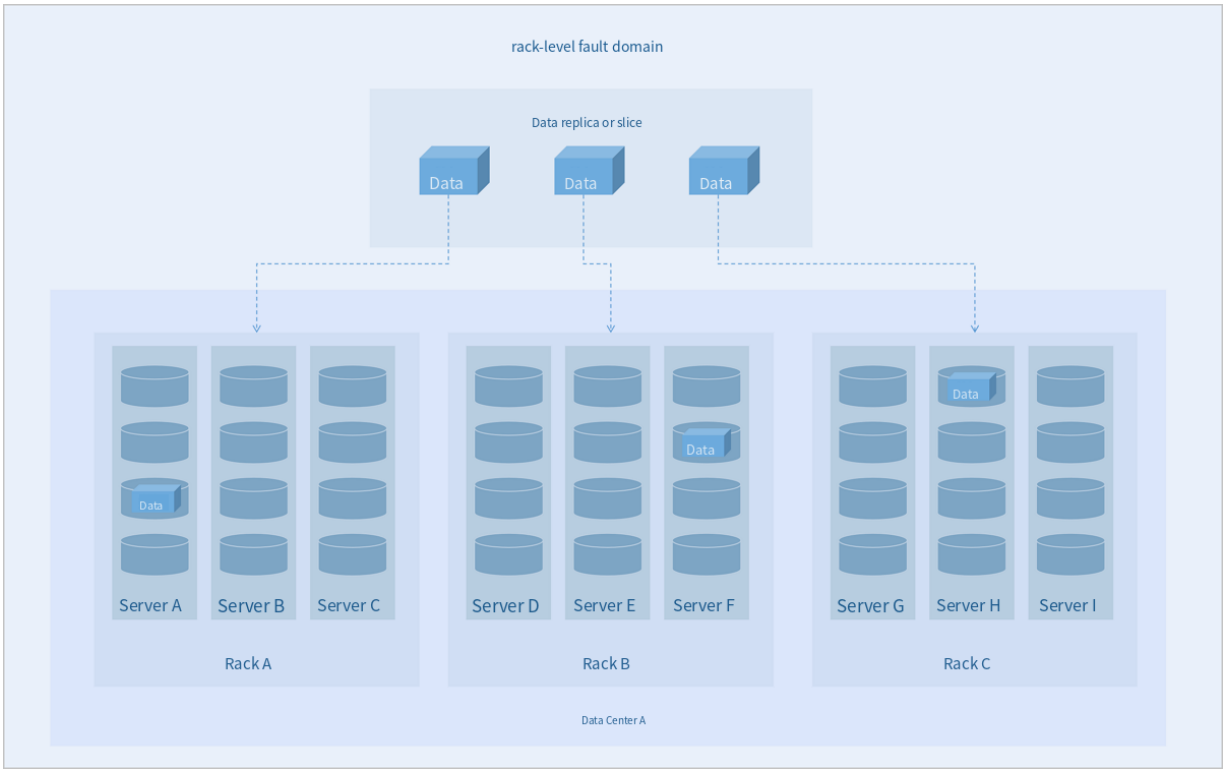
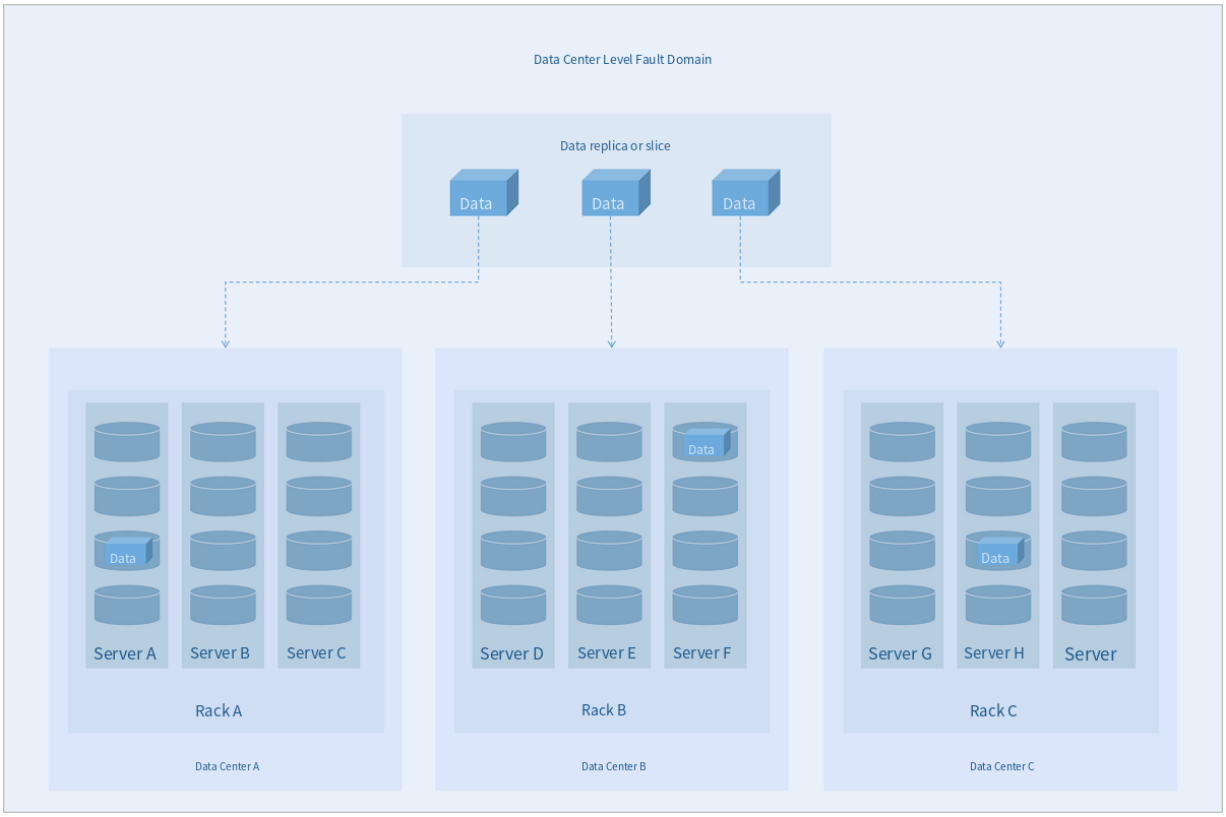


Figure 4-28: Facility Level Failure Domain



The failure domain mechanism isolates failures and confines the impact of failures to a certain range, thus avoiding the situation where a small change affects the overall system and improving business continuity.

The failure domain expansion mechanism allows you to add a new node without data migration. This mechanism achieves seamless expansion and shields application systems from the details of the underlying storage changes, avoiding the situation where both the application systems and the storage need to be changed when the storage is changed. This reduces the workload of both the maintenance and business personnel and improves system reliability and performance.

4.1.2.2.2.2.4 Data Consistency Inspection

Consistency inspections of self-developed distributed storage are implemented by theScrub mechanism, which scans data in the background to address data consistency issues. Dataconsistency inspections are periodic behaviors, divided into Scrub and Deep-Scrub.

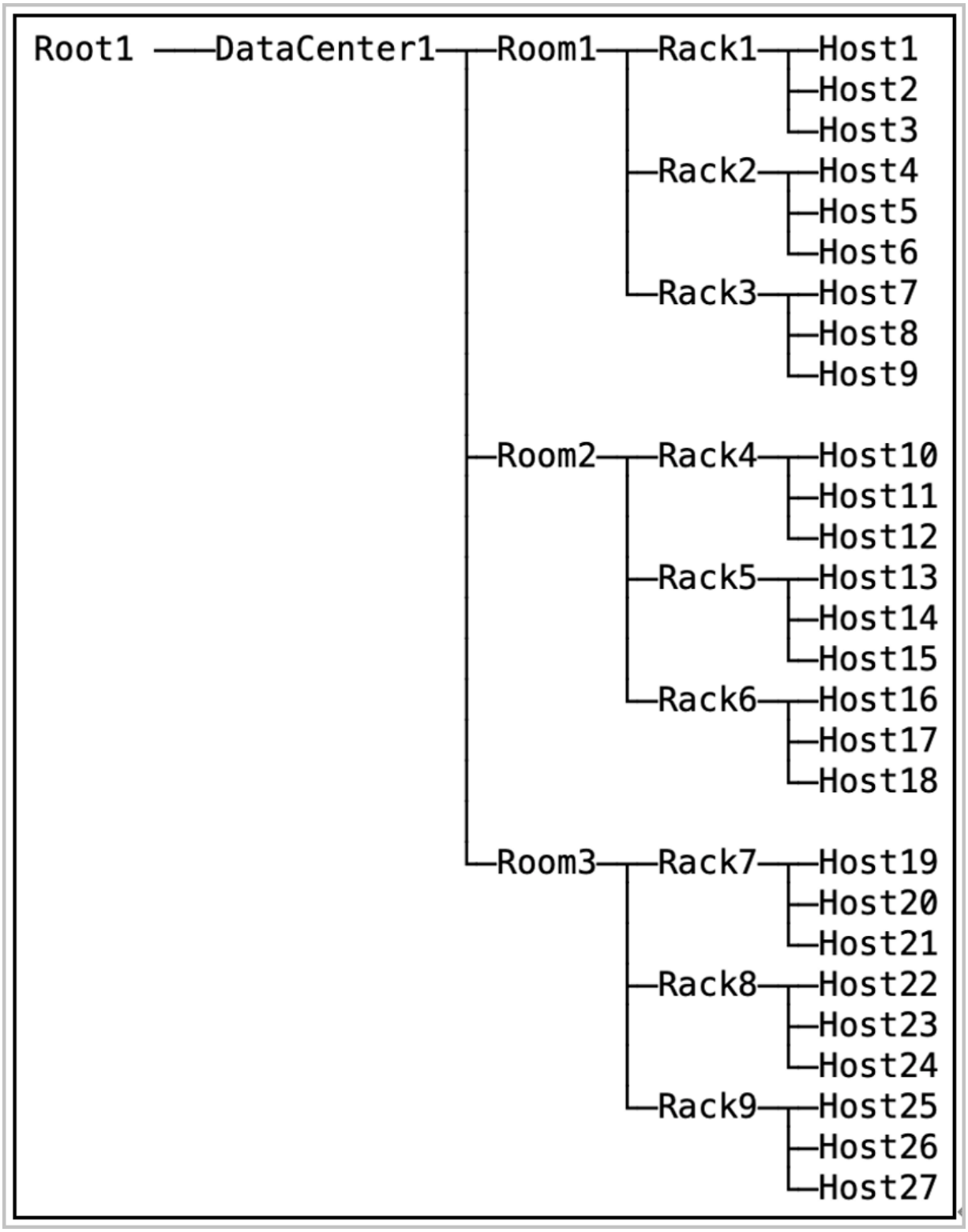
- Scrub check: This check applies to metadata. It is characterized by a shortexecution time and a frequent schedule. We recommend that you implement dataconsistency inspections daily. You can customize the time for Scrub checks.

- Deep-Scrub check: This check applies to data. It is characterized by a longexecution time and high I/O pressure. We recommend that you implement dataconsistency inspections during off-peak hours. If a Deep-Scrub check is notimplemented for more than 30 days, a warning message is displayed.

4.1.2.2.2.2.5 Cluster Hardware Topology

A cluster hardware topology is a visualized display of the actual deployment of the cluster's physical resources, including logical entities such as data centers, rooms, racks, and servers. It describes the distribution relationship from rooms to servers through tree diagrams, with each tree diagram having a root node. Note that a cluster hardware topology can have multiple root nodes. You can select a data redundancy strategy corresponding to the topology level when you create a storage pool after the topology is planned.

Figure 4-29: Topology Hierarchy



You can plan a topology for a cluster on the web interface. The following table lists the requirements on the quantity of each topology object.

Topology Object	Quantity Range
Data Center	0–2
Room	0–100 (in a data center)
Rack	0–100 (in a room)
Server	0–20 (in a rack)

4.1.2.2.2.3 Concise O&M

4.1.2.2.2.3.1 Multiple Resource Pools

Our native distributed storage supports the multiple resource pool feature that helps users achieve different performance storage media usage and fault isolation.

Each resource pool can have different attributes and performance, including the number of replicas, data redundancy level, and storage media. Based on your actual needs, you can flexibly divide and manage resources to improve storage efficiency and storage performance.

The resource pools are isolated from each other. You can manage data by using multiple resource pools. In addition, the failure of a resource pool does not affect other resource pools, which effectively ensures data security and storage reliability.

4.1.2.2.2.3.2 Hard Disk Lantern

You can use the lantern feature to locate hard disks on the UI and identify the location of a disk when performing maintenance or replacement. When a lantern is clicked on the UI, the LED on the related hard disk in the physical environment is lit. This feature makes it easier for you to locate the disk and improves the efficiency of your O&M work.

4.1.2.2.2.3.3 Disk S.M.A.R.T. Detection

Disk S.M.A.R.T. is a technology that self-developed distributed storage uses to monitor the health status, temperature, firmware, and written data volume of disks. Upper-layer businesses trigger alarms based on the I/O errors and disk status returned by S.M.A.R.T. data.

4.1.2.2.2.3.4 Data Volume Maintenance Mode

You can place a data volume into maintenance mode when you need to maintain related servers or hard disks. When a data volume is in maintenance mode, it stops serving data requests and data on the volume will not be balanced.

4.1.2.2.2.3.5 Automatic Fault Detection and Alarm

Self-developed distributed storage supports automatic fault detection and alarm mechanisms . These mechanisms monitor the storage platform system and various storage servers. Upon detecting a fault, they automatically send alarm messages to the platform. Users can add email alarmers separately to receive these alarm messages and take timely measures for fault repair.

At the same time, when a fault is detected, the system supports automatic service restart and data migration, ensuring data reliability and availability to the greatest extent possible. This forms a highly reliable and available distributed storage system.

4.1.3 Network Virtualization

4.1.3.1 Overview

Network virtualization is a virtual network method that abstracts and restructures network resources through software. It provides various network services such as data exchange, routing , and security groups, making the network experience comparable to that of physical networks. Underlying hardware only needs to provide basic packet forwarding services.

The virtualized platform abstracts the network model as distributed switches and distributed port groups. Distributed switches span multiple hosts across the data center, providing deployment , management, and monitoring capabilities for virtual networks throughout the data center. A distributed port group is a collection of virtual ports on a distributed switch, with VMs in the same port group having uniform network configurations. This ensures that VMs can migrate across hosts while maintaining consistent network settings. Additionally, the platform offers distributed DHCP services on distributed port groups and supports configuring DNS for VMs, catering to various network scenarios within the data center.

4.1.3.2 Technical Features

4.1.3.2.1 Distributed Switch

Switches identify hosts by using the hosts' MAC or IP addresses and use the addresses toforward data packets and control the data traffic. In traditional data centers, hostlocations are fixed, and the relationships between switches and hosts are also fixed, whichmeans that the configurations of switches do not need to be frequently changed.

In virtualized data centers, switches often face many new challenges, such as:

- The MAC or IP addresses of virtual machines cannot be predicted in advance.
- When virtual machines are migrated, configurations of the virtual machines on thephysical switch ports need to be migrated as well, which may cause the migration tofail.
- If the source and destination of a data packet are on the same host, the data packetmay not go through the switch, which makes it impossible for the switch to control thedata traffic.

nSSV introduces distributed switches to provide data packetforwarding and traffic control for virtual machines and for virtual machines andexternal networks. This makes the network more flexible and feature-rich.

A distributed switch is a logical switch that resides on each host in a VPC network.It consists of a virtual switch, VM NICs, and physical NICs.

- The virtual switch forwards data packets by using the MAC address.
- The VM NIC connects the virtual switch and a VM instance and serves as the datachannel for data packet forwarding.
- The physical NIC is the uplink port of the virtual switch. It forwards data packetsacross hosts to the physical switch.

4.1.3.2.2 Security Group

A security group essentially works as a distributed firewall that focuses on managing the east-west traffic and provides protection for VM NICs. Changing security group rules, associating/disassociating NICs will cause the security group rules to be updated across multiple VM instances. A VM NIC can be associated with multiple security groups. If you associate a security group with a VM NIC, the virtualization platform will automatically send iptables rules to the host where the VM instance resides, with the main use of the Filter table.

By default, four rules are configured when a security group is created, including ingress and egress rules of ALL protocol and ingress and egress rules of IPv4/IPv6. These default rules cannot be deleted but can be disabled as needed. If disabled, VM instances in the security group cannot communicate with each other. In addition, all external networks are denied from accessing VM instances in the security group, while VM instances in the security group can access all external networks freely. For example, if you create two security groups, Security Group 1 and Security Group 2, and associate vNIC 1.0, vNIC 2.0, vNIC 5.0, and vNIC 6.0 with Security Group 1, and associate vNIC 3.0, vNIC 4.0, vNIC 5.0, and vNIC 6.0 with Security Group 2.

You can configure an allow rule or a block rule for a security group based on your business requirements. For example, you can allow networks outside of the security group to access VM instances in Security Group 1, and block networks outside of the security group to access VM instances in Security Group 2.

4.1.4 Virtual Resource Management

4.1.4.1 VM Management

4.1.4.1.1 VM Scheduling Policy

A VM scheduling policy schedules host resources for VM instances to ensure business performance and high availability. You can add a VM instance to a VM scheduling group and then associate a scheduling policy with the group to schedule VM instances.

Function Principles

nSSV supports adding a virtual machine to a virtual machine scheduling group and binding scheduling policies to the group to achieve virtual machine scheduling.

- If a mutually exclusive virtual machine or aggregated virtual machine policy is bound, no host scheduling group needs to be specified, and the virtual machine is assigned to a host according to the policy and its execution mechanism.
- If a virtual machine host affinity or virtual machine mutually exclusive host scheduling policy is bound, the corresponding host scheduling group must be specified, and the virtual machine is assigned to a host according to the policy and its execution mechanism.

The following explains the working principles of the four types of scheduling policies through four scenarios:

Scenario 1: Assume there are three hosts Host1, Host2, and Host3 in the data center. Virtual machine scheduling group A is bound to the **VM Exclusive from Each Other** scheduling policy, and virtual machines VM1 and VM2 have joined this scheduling group and are running on hosts Host1 and Host2, respectively. At this point, virtual machine VM3 joins this scheduling group. Under different execution mechanisms, the behavior of virtual machine VM3 is as follows:

- Under the mandatory mechanism, virtual machine VM3 adheres to the principle of mandatory mutual exclusion with other virtual machines in the group:
  - If Host3 has sufficient resources, it can normally start and run on Host3.
  - If Host3 does not have sufficient resources, it cannot start on Host3.
- Under the preferred mechanism, virtual machine VM3 adheres to the principle of trying to mutually exclude other virtual machines in the group, prioritizing starting on Host3:
  - If Host3 has sufficient resources, it can normally start and run on Host3.

  - If Host3 does not have sufficient resources, VM3 can attempt to start on another host with sufficient resources. In this scenario, VM3 starts and runs on Host2.

Scenario 2: Assume there are two hosts Host1 and Host2 in the data center. Virtual machine scheduling group A is bound to the **VM Affinitive to Each Other** scheduling policy, and virtual machines VM1 and VM2 have joined this scheduling group and are running on host Host1. At this point, virtual machine VM3 joins this scheduling group. Under different execution mechanisms, the behavior of virtual machine VM3 is as follows:

- Under the mandatory mechanism, virtual machine VM3 adheres to the principle of mandatory aggregation with other virtual machines in the group:
  - If Host1 has sufficient resources, it can normally start and run on Host1.
  - If Host1 does not have sufficient resources, it cannot start on Host1.
- Under the preferred mechanism, virtual machine VM3 adheres to the principle of trying to aggregate with other virtual machines in the group, prioritizing starting on Host1:
  - If Host1 has sufficient resources, it can normally start and run on Host1.
  - If Host1 does not have sufficient resources, VM3 can attempt to start on another host with sufficient resources. In this scenario, VM3 starts and runs on Host2.

Scenario 3: Assume there are three hosts Host1, Host2, and Host3 in the data center. Virtual machine scheduling group A is bound to the **VMs Exclusive from Hosts** scheduling policy, and virtual machines VM1 and VM2 have joined this scheduling group and are running on host Host1. Host scheduling group A is also bound to the **VMs Exclusive from Hosts** scheduling policy, and hosts Host2 and Host3 have joined this scheduling group, each running two virtual machines. At this point, virtual machine VM3 joins virtual machine scheduling group A. Under different execution mechanisms, the behavior of virtual machine VM3 is as follows:

- Under the mandatory mechanism, virtual machine VM3 adheres to the principle of mandatory mutual exclusion with hosts in host scheduling group A:
  - If Host1 has sufficient resources, it can normally start and run on Host1.
  - If Host1 does not have sufficient resources, it cannot start on Host1.
- Under the preferred mechanism, virtual machine VM3 adheres to the principle of trying to mutually exclude hosts in host scheduling group A, prioritizing starting on Host1:
  - If Host1 has sufficient resources, it can normally start and run on Host1.

- If Host1 does not have sufficient resources, VM3 can attempt to start on another host with sufficient resources. In this scenario, VM3 starts and runs on Host2.

Scenario 4: Assume there are three hosts Host1, Host2, and Host3 in the data center. Virtual machine scheduling group A is bound to the **VMs Affinitive to Hosts** scheduling policy, and virtual machines VM1 through VM5 have joined this scheduling group and are running on hosts Host2 and Host3. Host scheduling group A is also bound to the **VMs Affinitive to Hosts** scheduling policy, and hosts Host2 and Host3 have joined this scheduling group. At this point, virtual machine VM6 joins virtual machine scheduling group A. Under different execution mechanisms, the behavior of virtual machine VM6 is as follows:

- Under the mandatory mechanism, virtual machine VM6 adheres to the principle of mandatory aggregation with hosts in host scheduling group A:
  - If Host2 or Host3 has sufficient resources, it can normally start and run on Host2 or Host3.
  - If Host2 and Host3 do not have sufficient resources, it cannot start on Host2 or Host3.
- Under the preferred mechanism, virtual machine VM6 adheres to the principle of trying to aggregate with hosts in host scheduling group A, prioritizing starting on Host2 or Host3:
  - If Host2 or Host3 has sufficient resources, it can normally start and run on Host2 or Host3.
  - If Host2 and Host3 do not have sufficient resources, VM6 can attempt to start on another host with sufficient resources. In this scenario, VM6 starts and runs on Host1.

4.1.4.1.2 VM Cloning

The virtualization platform provides two VM cloning methods to meet different business scenario requirements: full cloning and rapid full cloning.

Full Clone

Full cloning creates a new VM image that is not shared with the source VM. The source VM image is a complete image containing all files and configurations required by the VM, which is saved in the image storage. Then, the image is pushed to a data storage to serve as the image cache for the cloned VM, thereby creating a new VM.

The cloned VM created by full cloning is completely isolated and independent from the source VM and is not dependent on the source VM. Therefore, the VM performance is not affected. Full cloning usually consumes more storage space, butnSSV has implemented storage optimization so that the image cache of the new VM only occupies a share of the data storage, which avoids consuming more storage space and speeds up the startup of cloned VMs.

Rapid Full Clone

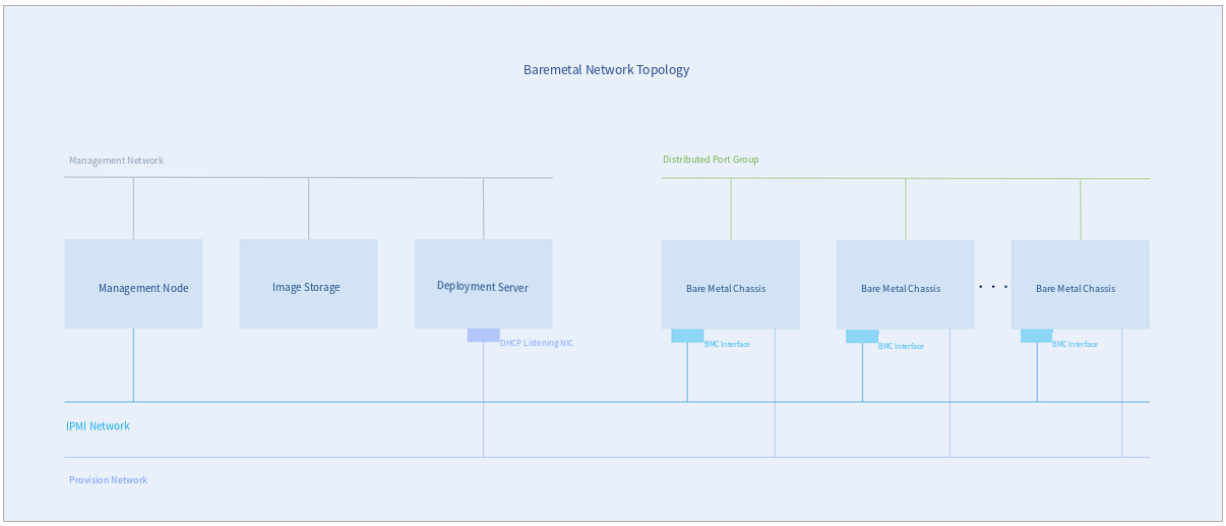
A rapid full clone works by using linked cloning to create a new VM that is dependent on the source VM for fast VM startup. Then, a job is triggered in the background to merge the cloned VM . Finally, the data of the cloned VM is completely independent from the source VM. Therefore, VMs cloned by using rapid full cloning start quickly and are independent from the source VM. In addition, the VM performance is not affected after the cloning is completed.

4.1.4.2 Baremetal Management

The network traffic flow of baremetal management includes the management network, distributed port group, IPMI network, and provision network.

- Management network: Used for managing related hardware resources of the platform.
- Distributed port group: Used for the business network of baremetal instances, providing external application services.
- IPMI network: Used for management nodes to remotely power on/off, reboot, and obtain hardware information of baremetal chassis and baremetal instances.
- Provision network: Used for the DHCP service of the PXE server to assign IP addresses to baremetal instances and for the TFTP service to send images.

Figure 4-30: Baremetal Network Topology



shows the network topology of baremetal management.

Baremetal management core: Obtaining hardware information of baremetal chassis and implementing unattended deployment of baremetal instances.

Baremetal management mechanism:

1. Management nodes specify baremetal chassis to boot from a PXE server.
2. The PXE server encapsulates DHCP, TFTP, and image storage services. After a baremetal chassis boots from the PXE server, it obtains an IP address by using the DHCP service, and then downloads **pxelinux.0** and boot files from the TFTP server. The kernel is loaded into memory and executed, and the LiveCD system is entered.
3. Detection scripts are executed in the LiveCD system to send the hardware information of baremetal chassis to management nodes.
4. Based on the returned hardware information, preconfigured templates are set for baremetal chassis, including partition information, NIC bonding, and IP addresses.
5. Select an operating system ISO file to deploy baremetal instances.
6. Restart baremetal chassis to boot from the PXE server. The PXE server will pre-download the operating system ISO file specified in the unattended deployment configuration. The baremetal chassis will be deployed based on the configured template. After the deployment is completed , the network configurations in the preconfigured template will be applied automatically, and the baremetal instance is created.
7. To better manage baremetal instances, the PXE server sends the agent to the baremetal chassis to monitor the real-time running status of the baremetal instance, including CPU, memory, disk usage, disk I/O, NIC, and other information.

4.2 Data Protection

4.2.1 Snapshot Management

The virtualization platform supports ROW (Redirect-On-Write, write-through) and COW (Copy-On-Write, write-around) snapshot mechanisms.

- Local/NFS/SAN storage uses QCOW2 external snapshot, which is a type of ROW snapshot.
- Distributed storage uses ROW and COW snapshot. Enterprise Edition distributed storage uses ROW snapshot. Self-developed distributed storage uses COW snapshot.

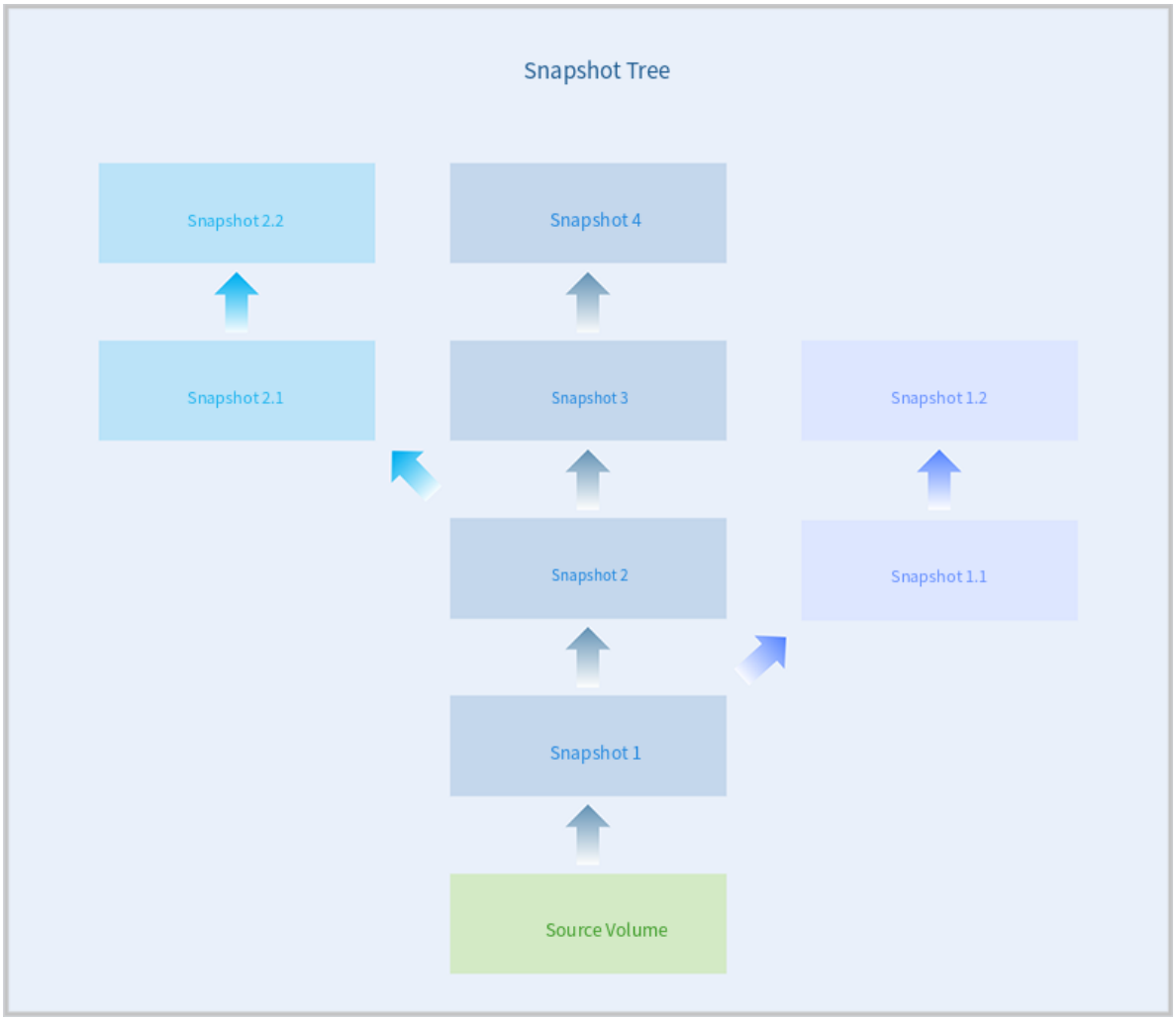
Snapshotting with a Distributed Storage

Describes QCOW2 external snapshot.

1. **Snapshot Chain and Snapshot Tree**

   Typically, a disk has a snapshot chain. You can create a snapshot tree for a disk, with each branch of the tree a snapshot chain.

Figure 4-31: Snapshot Tree



A snapshot tree includes the following information:

- Snapshot chain: A chain of snapshots of a disk. Each branch of a snapshot tree is a snapshot chain.
- Snapshot node: A node of a snapshot chain, representing a snapshot of a disk.
- Snapshot size: The storage space a snapshot occupies. You can view the total size of all snapshots in a snapshot tree and the size of each snapshot node.

Note:

- For non-distributed storage, the system defaults a snapshot chain with a maximum of 128 nodes.For distributed storage, a disk can have a maximum of 32 snapshots, including both manually and automatically created snapshots.
- When the length of a snapshot chain reaches the upper limit:

- If you continue to create auto-snapshots, the Cloud automatically deletes the earliest auto-snapshot.
- If you continue to create manual snapshots, you need to delete unnecessary snapshots manually.

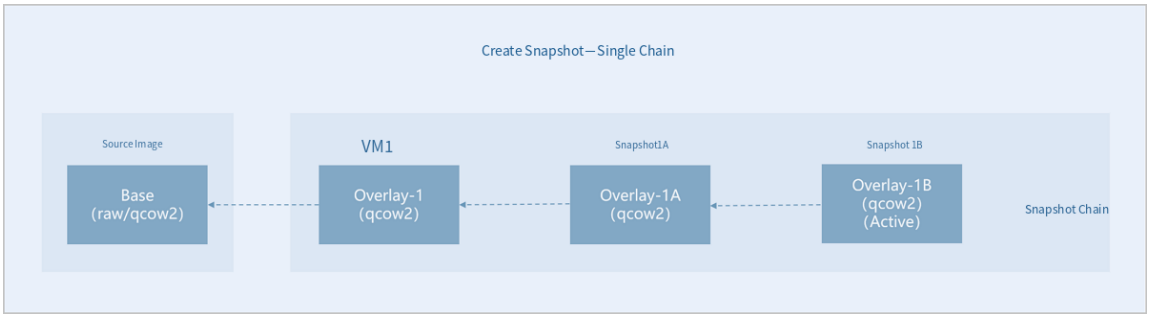
- In a production environment, we recommend that you do not have more than 5 snapshots for a VM to avoid a negative impact on VM performance, data security, and storage space.

2. Create Snapshot

When an external snapshot is created, in fact, a new qcow2 file is created, which serves as a blank file. The backing file of this blank file points to the old qcow2 file, and the old qcow2 file is set to read-only. Thus, the old qcow2 file becomes a snapshot, and data is written only to the new qcow2 file in the future.

- Create a single snapshot chain based on the backing file.

Figure 4-32: Create Snapshot Single Chain



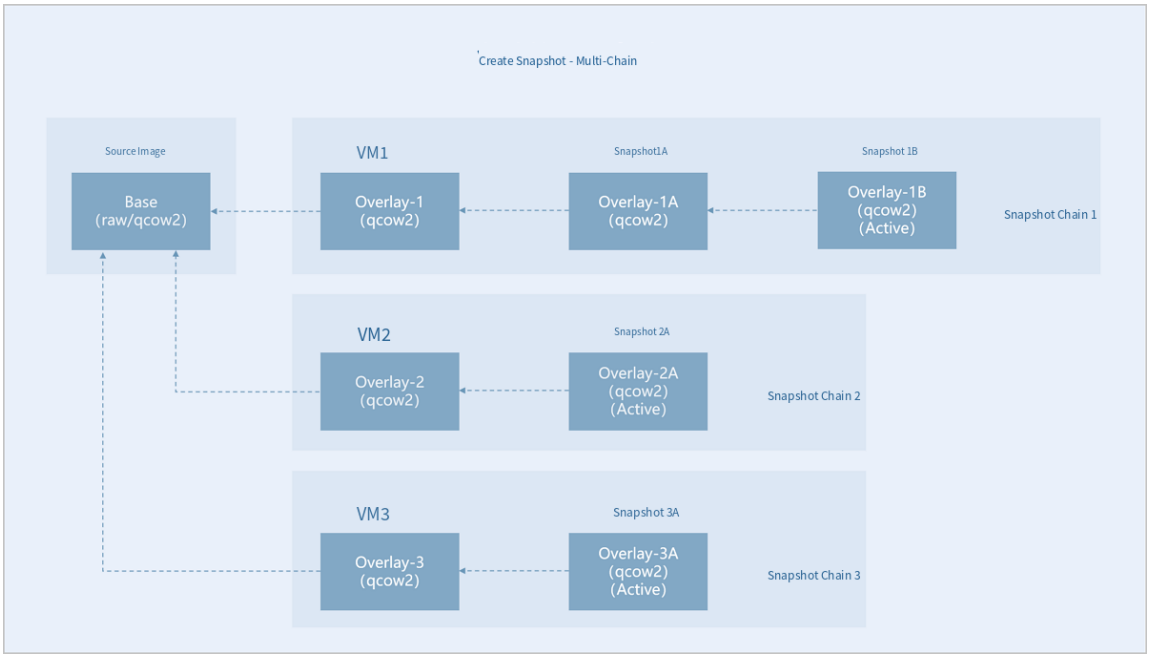
Assume that there is an initial image (Base) and a virtual machine 1 is created by using this initial image as a template. Then, you can create snapshot 1A, 1B, and so forth for virtual machine 1.

- Base: A disk image file that is pre-made and contains a complete operating system and boot program. It is used as a read-only image.
- Virtual machine 1: A new blank file Overlay-1 is created. The backing file of this new blank file points to Base, and Base is set to read-only. Thus, Base becomes a snapshot. Data is written only to Overlay-1 in the future.
- Snapshot 1A: A new blank file Overlay-1A is created. The backing file of this new blank file points to Overlay-1, and Overlay-1 is set to read-only. Thus, Overlay-1 becomes a snapshot. Data is written only to Overlay-1A in the future.

- Snapshot 1B: A new blank file Overlay-1B is created. The backing file of this new blank file points to Overlay-1A, and Overlay-1A is set to read-only. Thus, Overlay-1A becomes a snapshot. Data is written only to Overlay-1B in the future. Virtual machine 1 uses the last snapshot 1B in the snapshot chain as its disk file. Snapshot 1B is in the Active state.

- Create multiple snapshot chains based on the backing file.

Figure 4-33: Create Snapshot Multiple Chains



Assume that there is an initial image (Base) and virtual machines 1, 2, and 3 are created by using this initial image as a template. Then, you can create snapshot 1A, 1B for virtual machine 1, snapshot 2A for virtual machine 2, and snapshot 3A for virtual machine 3.

- Base: A disk image file that is pre-made and contains a complete operating system and boot program. It is used as a read-only image.
- Snapshot Chain 1:
  - Virtual machine 1: A new blank file Overlay-1 is created. The backing file of this new blank file points to Base, and Base is set to read-only. Thus, Base becomes a snapshot. Data is written only to Overlay-1 in the future.
  - Snapshot 1A: A new blank file Overlay-1A is created. The backing file of this new blank file points to Overlay-1, and Overlay-1 is set to read-only. Thus, Overlay-1 becomes a snapshot. Data is written only to Overlay-1A in the future.

- Snapshot 1B: A new blank file Overlay-1B is created. The backing file of this new blank file points to Overlay-1A, and Overlay-1A is set to read-only. Thus, Overlay-1A becomes a snapshot. Data is written only to Overlay-1B in the future. Virtual machine 1 uses the last snapshot 1B in Snapshot Chain 1 as its disk file. Snapshot 1B is in the Active state.

- Snapshot Chain 2:
  - Virtual machine 2: A new blank file Overlay-2 is created. The backing file of this new blank file points to Base, and Base is set to read-only. Data is written only to Overlay-2 in the future.
  - Snapshot 2A: A new blank file Overlay-2A is created. The backing file of this new blank file points to Overlay-2, and Overlay-2 is set to read-only. Thus, Overlay-2 becomes a snapshot. Data is written only to Overlay-2A in the future. Virtual machine 2 uses the last snapshot 2A in Snapshot Chain 2 as its disk file. Snapshot 2A is in the Active state.
- Snapshot Chain 3:
  - Virtual machine 3: A new blank file Overlay-3 is created. The backing file of this new blank file points to Base, and Base is set to read-only. Data is written only to Overlay-3 in the future.
  - Snapshot 3A: A new blank file Overlay-3A is created. The backing file of this new blank file points to Overlay-3, and Overlay-3 is set to read-only. Thus, Overlay-3 becomes a snapshot. Data is written only to Overlay-3A in the future. Virtual machine 3 uses the last snapshot 3A in Snapshot Chain 3 as its disk file. Snapshot 3A is in the Active state.

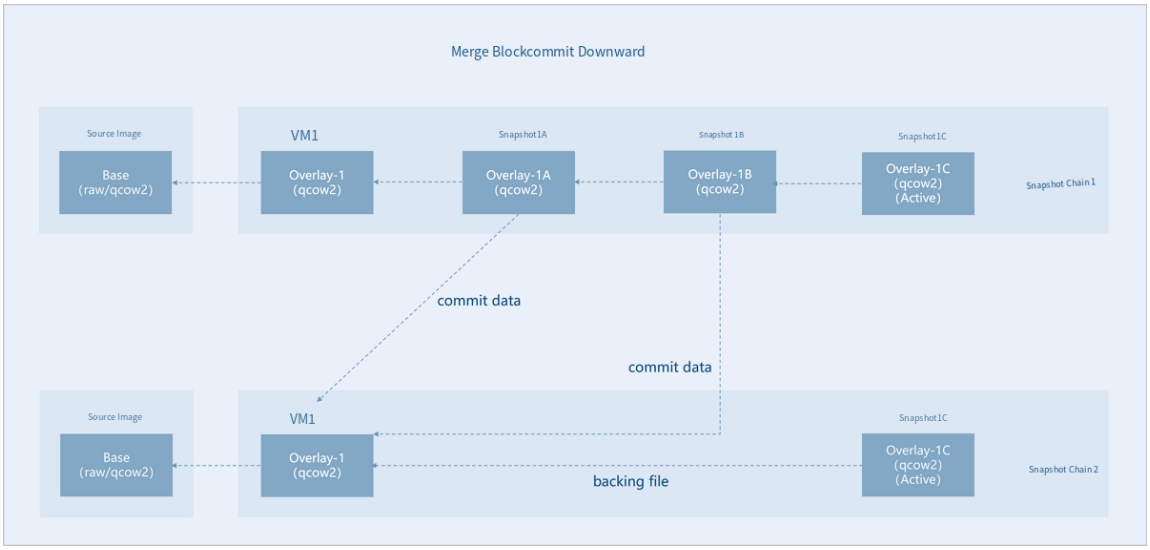
3. Merge Snapshot

Dependent snapshots are interdependent on each other, with each overlay relying on its backing file. Each snapshot saves corresponding data and cannot be directly deleted to shorten the snapshot chain. You can use two methods, block commit and block pull, to shorten the snapshot chain.

- Block Commit

  On the same snapshot chain, you can commit overlays to their respective backing files.

Figure 4-34: Block Commit



Assume that you have a base image, VM1 is created from the base image, and three dependent external snapshots, Snapshot1A, Snapshot1B, and Snapshot1C, are created for VM1. Committing Snapshot1A and Snapshot1B to VM1 makes the backing file of Snapshot1C (Active) point directly to VM1, thus shortening the snapshot chain. You can delete Snapshot1A and Snapshot1B because they are no longer needed.

- Block Pull

  On the same snapshot chain, you can pull the backing files to the corresponding overlays.

  Assume that you have a base image, VM1 is created from the base image, and three dependent external snapshots, Snapshot1A, Snapshot1B, and Snapshot1C, are created for VM1. Pulling Snapshot1A and Snapshot1B to Snapshot1C (Active) makes the backing file of Snapshot1C (Active) point directly to VM1, thus shortening the snapshot chain. You can delete Snapshot1A and Snapshot1B because they are no longer needed.

Distributed Storage Snapshot Mechanism

Enterprise Edition distributed storage uses the ROW snapshot technology. Self-developed distributed storage uses the COW snapshot technology. For more information, see *Volume Snapshot*.

4.2.2 Backup Management

4.2.2.1 Data Backup

Backup of data based on the QEMU block device layer is supported. You can backup virtual machines on all types of data storage. Two backup types are supported: full backup and incremental backup. A full backup includes all data sets. An incremental backup includes only data sets updated after the last backup. Note that both full and incremental backup backup only actual data.

By default, the full backup is auto-triggered when it is executed 63 times after the first incremental backup. This is because to restore VM instances from backup, the corresponding backup data, including the full backup and all incremental backups before the backup to be used for the recovery, need to be merged. To ensure data security, the maximum length of the backup chain is 64. In fact, the system internally applies a smarter and more flexible strategy to decide on an appropriate backup mode to ensure the security and reliability of the backup data.

Data backup consists of three parts: data duplication, data transfer, and data retention.

4.2.2.1.1 Data Replication

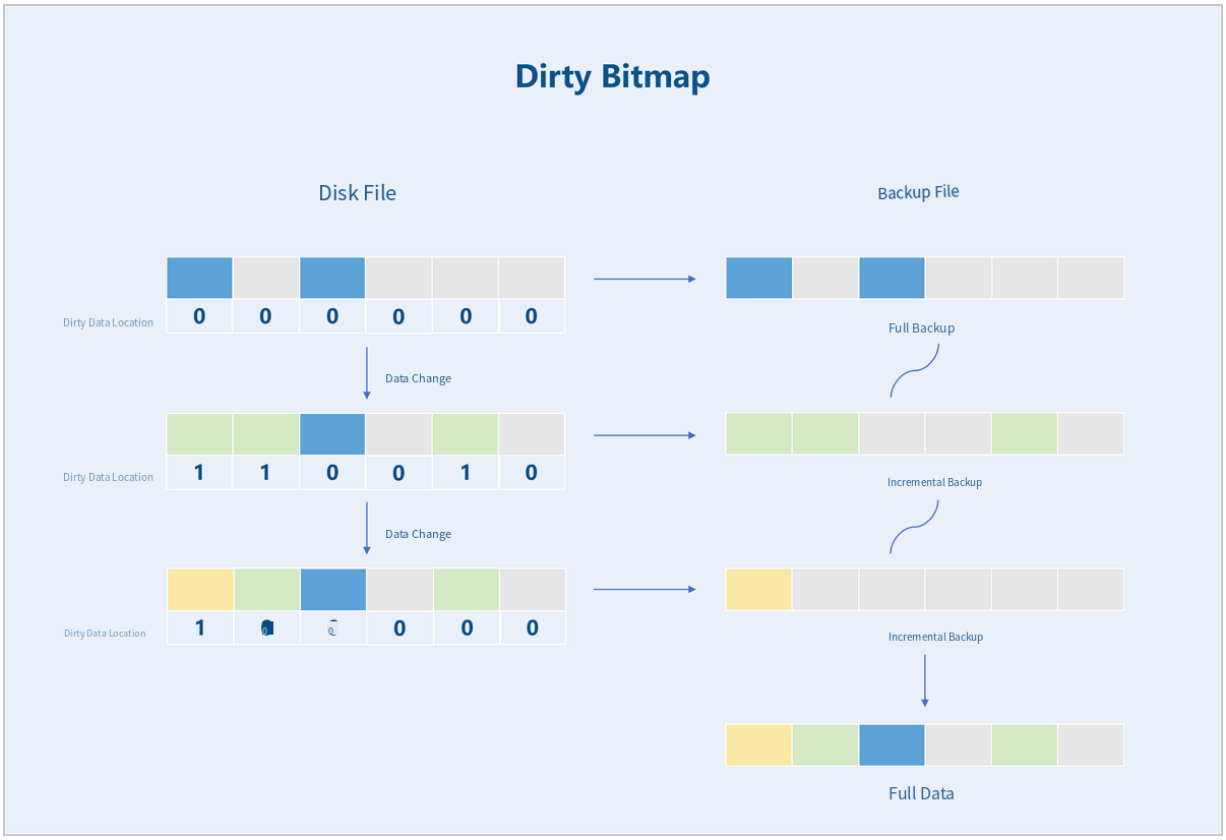
The backup module utilizes the dirty data tracking feature (Dirty Bitmap) of the QEMU block device layer to export backup data.

The locations where changes occur in the virtual machine disk file data are referred to as dirty data locations. The Dirty Bitmap records all dirty data locations on the virtual disk file since the last backup. Based on these location records, all modified data since the last backup—i.e., incremental backup data—can be exported.

Ultimately, the full backup file and various incremental backup files form a complete backup chain, preserving all data. The Dirty Bitmap resides in the memory of the QEMU process and is lost after a virtual machine reboot. Therefore, the system automatically performs a full backup the next time a backup is initiated after a reboot.

As shown in *Figure 4-35: Dirty Bitmap*:

Figure 4-35: Dirty Bitmap

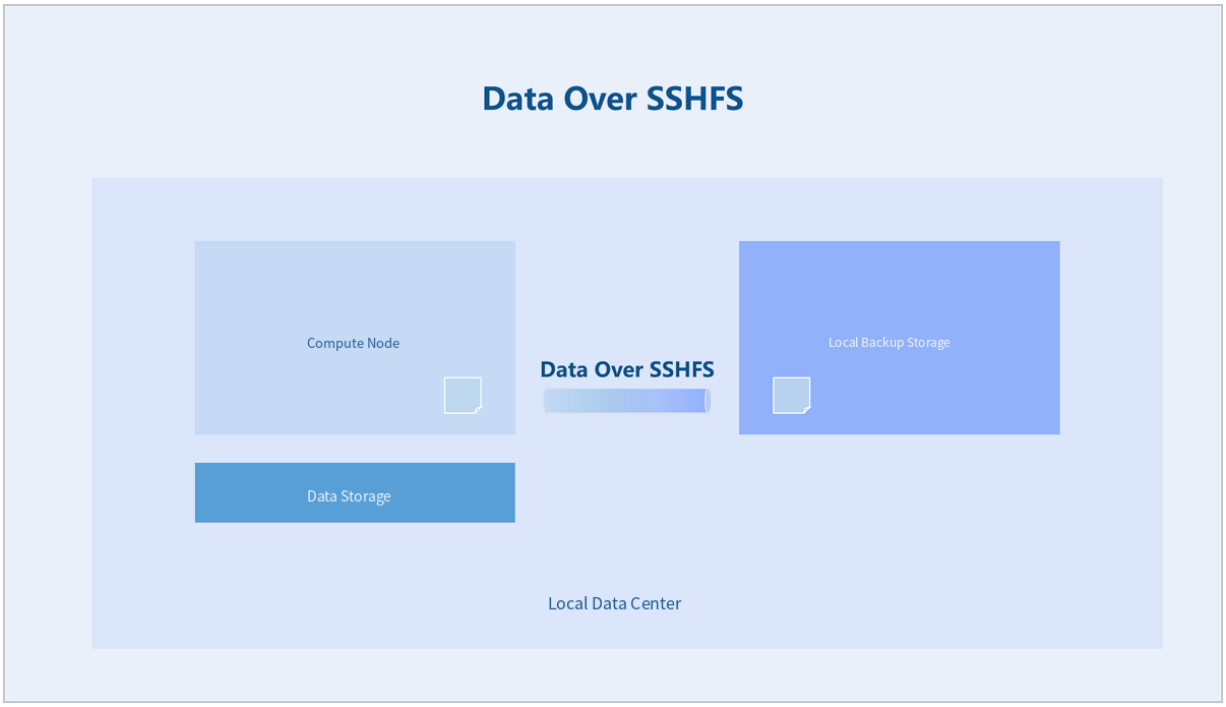


4.2.2.1.2 Data Transfer

Two implementation methods are supported based on the virtualization component version. The main difference between these two methods lies in the data transfer method.

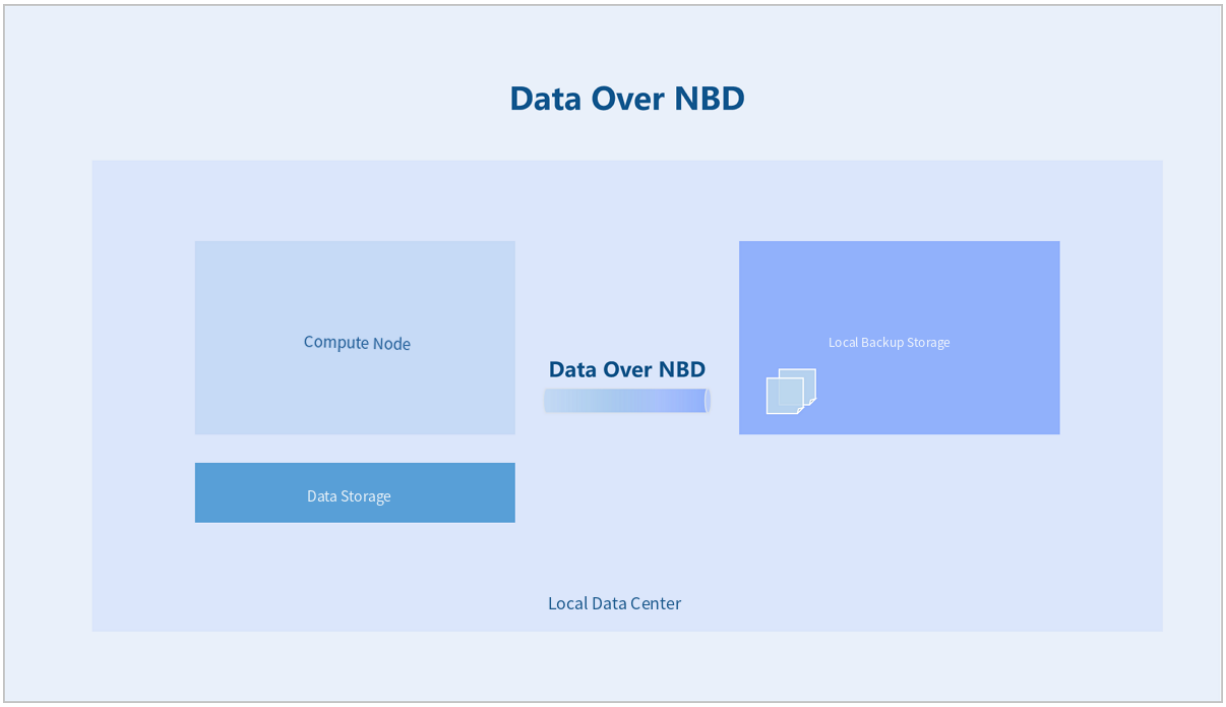
The first method uses SSHFS to mount the backup directory on a remote, distant backup storage to a compute node, and then imports the backup data into the storage. SSHFS is an SSH-based file system implementation. The data transfer is encrypted by an SSH session, and each backup plan uses an independent SSHFS link.

Figure 4-36: Data Over SSHFS



The second method uses the NBD module to export a backup disk on the distant backup storage, and then uses a QEMU job to import the backup data into the disk on the compute node.

Figure 4-37: Data Over NBD



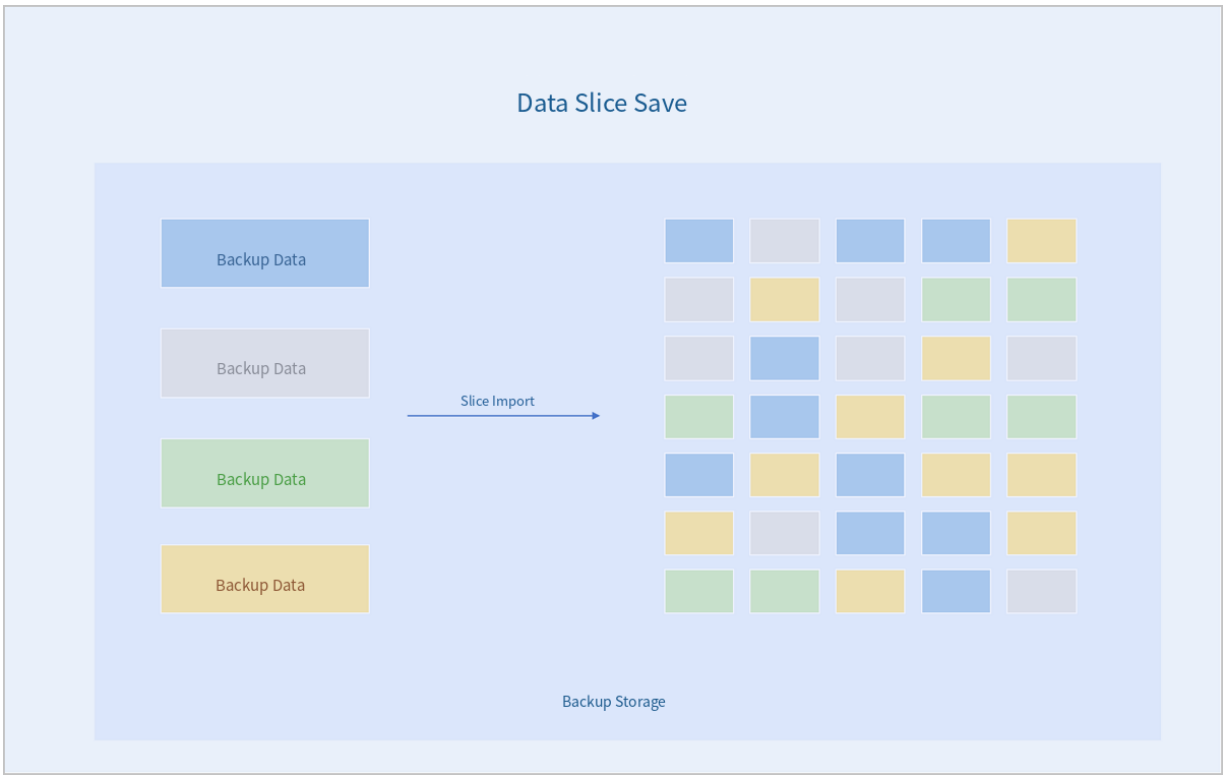
4.2.2.1.3 Data Storage

The backup storage supports multiple storage media types, including: SAN, NAS, disk arrays, and tape libraries.

Backup data is stored in deduplicated slices within the backup storage. The backup data is divided into 64MB data chunks, with hash values calculated and indexes established. Data chunks with identical hash values are not stored redundantly.

As shown in *Figure 4-38: Data Slice Storage*:

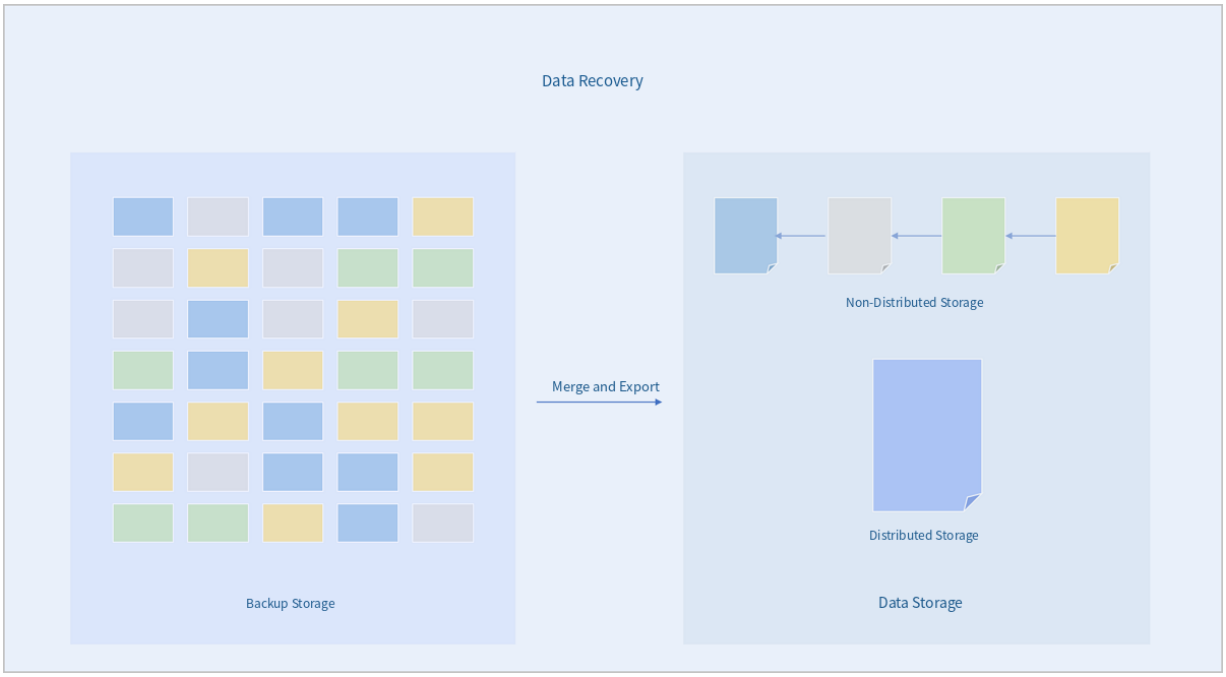
Figure 4-38: Data Slice Storage



4.2.2.2 Data Recovery

Recovering a VM from local backup data will merge the slice data on the backup storage and import it into the data storage. The recovered data will be stored in non-distributed storage in the form of a disk chain, while in distributed storage, it will be stored as an individual disk file. If you select New Recovery, the recovered disk data will be treated as image caches for creating new VMs. If you select Overwrite Recovery, the path of the recovered disk will be updated in the current VM database record, and then the old VM files will be deleted.

Figure 4-39: Data Recovery



4.3 Operate and Manage

4.3.1 Management Node Monitoring

On the HA scenario with multiple management nodes, you can intuitively view the health status of each management node.

The management node monitoring displays the management node IP, node status, VIP, and management service status of multiple management nodes, including the following management services:

- Arbitration IP Reachability: Monitors the reachability of the arbitration IP addresses of the primary and secondary management nodes. An unreacheable arbitration IP may cause the HA feature of the management nodes to fail.
- Reachability of Secondary Management Node: Monitors whether the secondary management node is reachable. If the secondary management node is unreachable, you cannot communicat e with it.
- VIP Reachability: Monitors whether the VIP is reachable. An unreacheable VIP may cause the primary management node to fail to access the UI interface via the VIP.
- Database Status: Monitors the database status. An abnormal database or unsynchronized database of multiple management nodes may cause data loss. Please restore the fault as soon as possible.

4.3.2 Monitoring and Alarm

Monitoring and alarm enable you to monitor the resource performance and the storage space of resources and to detect pre-defined events occurring in the Cloud. You can configure an SNS to receive alarm messages when an alarm is triggered. Two types of alarmers, resource alarmers and event alarmers, are supported. Besides, multiple endpoints, including System, DingTalk, HTTP, and Microsoft Teams, are supported. However, some resource alarmers require installing agents first.

Monitoring data is provided by Prometheus. When you monitor business data, you need to summarize the data and make it available to Prometheus.

By default, a Prometheus server is used to monitor specific targets. The server mainly collects , stores, and queries data. To monitor sample data, for example, the CPU utilization of a host, an exporter is used to periodically collect monitoring samples. The virtualized platform uses pull and push modes to collect monitoring data for different targets. When a host or VM instance is used as a monitoring target, the Prometheus server uses the pull mode to periodically obtain data collected by the exporter on the host. In addition, due to network or security issues, the Prometheus server cannot access the internal of a VM instance or baremetal instance. In this case, a pushgateway is used as a transit to forward data collected by the exporter on the VM instance or baremetal instance to the Prometheus server. Then, the Prometheus server obtains data from the pushgateway. This way, monitoring data is collected in a unified manner.

5 Reliability

5.1 Compute HA

Virtualization platforms using KVM-based hardware virtualization technologies virtualize a group of hardware servers into a logical resource pool and divide the pool into clusters for management . The platforms continuously detect the running status of all hosts and virtual machines in a cluster . If a host fails, the manangement node of the virtualization platform continuously detects the host until it is determined to be crashed. Then, the platform restarts all VM instances running on other hosts in the cluster to ensure business continuity.

5.2 High Availability of Distributed Storage

This topic describes the high availability of self-developed distributed storage. For more information, see the following documents:

- *Data Duplication*
- *Failure Domain Isolation*
- *Data Consistency Inspection*
- *Cluster Hardware Topology*
- *Multiple Resource Pools*
- *Hard Disk Lantern*
- *Disk S.M.A.R.T. Detection*
- *Data Volume Maintenance Mode*
- *Automatic Fault Detection and Alarm*
- *#unique_60*

5.3 Network HA

Virtual Network HA

A NIC bond is a logical NIC interface that ties two or more physical NICs and is used as a single NIC. The virtualization platform supports two NIC bonding modes:

- LACP mode (mode 4): You can add 1 to 8 physical NIC ports to an LACP bond. The NIC ports in an LACP bond share the same speed rate and duplex setting. Network flows are evenly sent to the NIC ports for load balancing.

An LACP bond uses hash calculations to determine the network exit. Three hash calculations are supported:

- layer2+3: Hash calculations are based on the source MAC address, destination MAC address, and IP address to determine the NIC port for data packet transmission.
- layer3+4: Hash calculations are based on the IP address and port to determine the NIC port for data packet transmission. TCP/IP stacks are supported.
- layer2: Hash calculations are based on the source MAC address and destination MAC address to determine the NIC port for data packet transmission.

- Active-backup mode (mode 1): You can add 1 to 8 physical NIC ports to an active-backup bond. After a bond is created, one NIC port is set as the active NIC port and the rest are set as backup NIC ports. Network flows are handled by the active NIC port by default. If the active NIC port fails, the backup NIC port is activated to handle network flows instantly to avoid business interruptions.

5.4 Manage HA

A management node is in charge of resource management, monitoring, scheduling, allocation , and reclaim of the entire resources of a virtualized platform. An HA management node is unavailable if a management node crashes, which in turn affects the management, monitoring, access, and auto-tasks execution of the virtualized platform and may have a great impact on your platform or business continuity.

Dual-Management Node HA

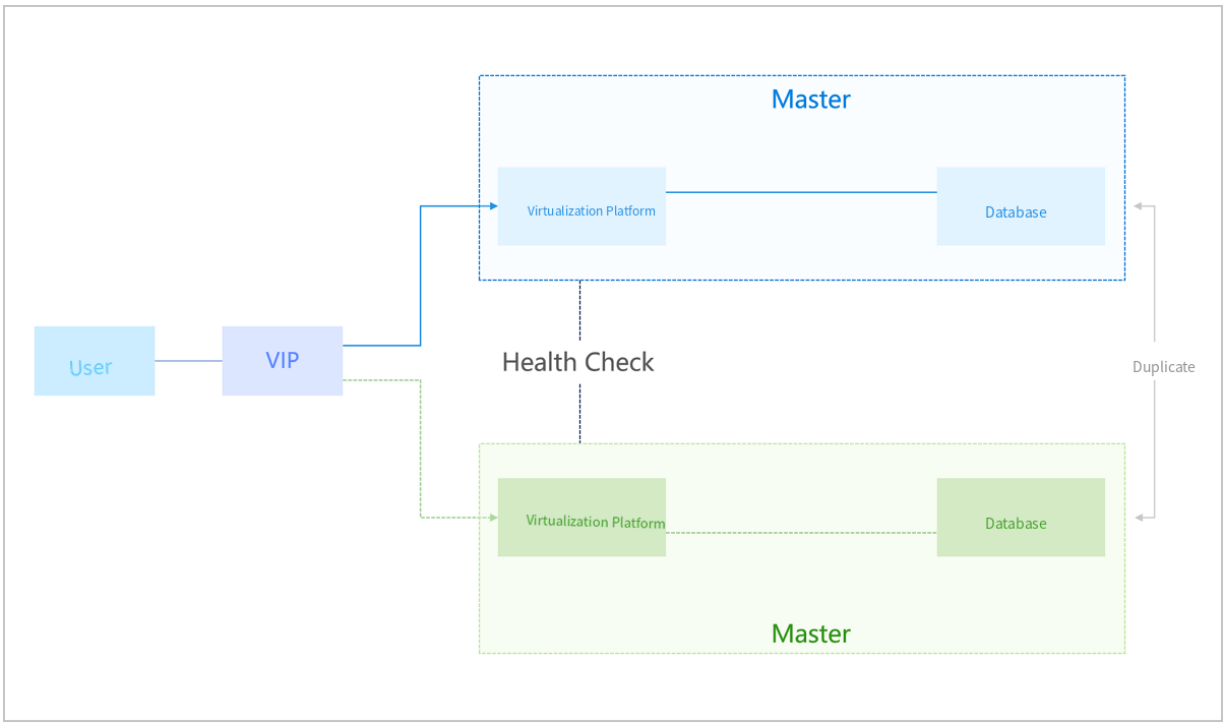
The dual-management-node HA scheme provides a highly available environment for two management nodes.

- Two management nodes are installed on the platform. A crash of one management node does not affect the work of the other management node and the UI work normally and can be used to execute various tasks.
- Databases, including MySQL and monitoring databases, can automatically synchronize to the other management node if the other management node crashes. You do not need to manually perform this operation.
- The platform can work properly for a long time if a management node crashes.
- The deployment and management of a dual-management-node environment are as simple as that of a single-management-node environment.

To address the HA issue of management nodes, we provide an HA process that runs on a management node. The HA process is in charge of the initialization, configuration, maintenance, watchdog, and other management tasks of the entire management node environment.

- The HA process provides system configurations and a CLI that calls HA commands to configure the system for an HA environment.
- The HA process monitors key services, including management node processes, UI processes , and MySQL, running on a management node. If a service crashes, the HA process triggers a VIP migration by using a Keepalived process and attempts to restart the crashed service.
- The HA process monitors the Keepalived process to ensure its continuous running.
- The HA process provides commands to print the health status of a cluster.
- An arbitration gateway is introduced to avoid the brain split issue of a dual-management-node environment.

Figure 5-1: Dual-Management Node HA Mechanism



Management Node Monitoring

You can view the management node IP, node status, VIP, and management node service status of multiple management nodes on the management node monitoring page. For details, see *Management Node Monitoring*.

Management Node Monitoring Data Collection

The mechanism how the Cloud collects the monitoring data of management nodes. For details, see *Monitoring and Alarm*.

6 Security

6.1 Compute Security

6.1.1 HTTPS UI Login Interface

By default, the UI login interface uses the HTTPS protocol to improve system security.

- **HTTPS Protocol**: The virtualization platform forwards requests to port 443 of the HTTPS address. You can enter the management node IP address in the browser to access the UI login interface, for example, *https://management_node_ip*.
- **HTTP Protocol**: By default, the HTTP protocol is not used. You can configure whether to use this protocol.
- The system supports certificates in the default**PKCS12**format. Currently, only certificates in the PKCS12 or JKS format are supported. If you use certificates in other formats, format conversion is required before you can use these certificates.

6.1.2 VM Console

A VM console provides a quick way to monitor and manage VM instances. To open a VM console, you need to have the corresponding permissions. Two authentication methods are supported: SSH Key Pair and Username/Password.

- SSH Key Pair
  - You can log into a Linux VM instance with an SSH key pair.
  - An SSH key pair consists of a public key and a private key. The public key isgenerated by using an encryption algorithm. The public key is made available tothe public, while the private key is reserved by you.
  - After a VM instance is associated with a public key, you can use the privatekey of another VM instance to SSH into the VM instance with the associatedpublic key without a password.
  - If you associate a public key with a VM instance when the VM instance iscreated, make sure that the VM image used for the creation has cloud-initinstalled in advance. Supported cloud-init versions: 0.7.9, 17.1, 19.4,19.4, and later.
  - If you associate a public key with a VM instance after the VM instance iscreated, make sure that the VM instance is in the running state and has QemuGuest Agent (QGA) installed. In addition, make sure that QGA is in the runningstate. You can install QGA through VMTools. If you install QGA through othermeans, make sure that QGA is 2.5 or later.

- Username/Password
  - You can log into a VM instance with a username and password.
  - The default username for a Linux VM instance is **root**. The defaultusername for a Windows VM instance is **administrator**.
  - If a VM instance has a password injected, you can use a username andpassword to SSH into the VM instance.
  - Make sure that the VM image used for the VM instance has cloud-initinstalled in advance. Supported cloud-init versions: 0.7.9, 17.1, 19.4,19.4, and later.

6.1.3 High Availability

VM HA

You can enable HA for a VM. When the VM is unexpectedly or planned stopped due toerrors or maintenance, the HA mechanism will automatically start the VM to improveits availability.

NeverStop HA works as follows:

- The VM status is constantly monitored by using a polling or trigger mechanism.If the VM is found to be stopped, the VM will be automatically started.
- The VM status is constantly monitored by using a polling or trigger mechanism.If the VM status cannot be determined, the following procedure is executed:
  1. The existing network configurations are used to detect the host where the VMresides as accurately as possible.
  2. If the host is found to be in abnormal status, the VM will be automaticallystarted.

6.1.4 Defend against IP/MAC/ARP Spoofing

IP/MAC/ARP spoofing is a common attack in traditional networks. With IP/MAC/ARP spoofing, hackers can disrupt network communications and steal network secrets.

On the data link layer, the host isolates abnormal protocol accesses from VMs and blocks MAC/ ARP spoofing attacks. On the network layer, the host prevents IP spoofing.

6.1.5 Image/Snapshot

Image

Images of VM instances can be created. The data of the VM instance is completely contained in an image, and you can duplicate the image to achieve fast duplication of resources.

The virtualization platform protects the integrity and security of images in the following ways:

- Images are protected from being tampered with by using an encryption algorithm. When an image is downloaded from the image storage, the encryption algorithm is used for authentica tion. If the authentication succeeds, the image is downloaded.
- Image files are stored in slices in the image storage. Before you can read the image data, the virtualization platform assembles the slice files. This implementation protects the security of image data.

Snapshot

Snapshots are essentially the state files of the data on a disk volume at a certain time point. Before you perform a business-sensitive operation, you can create a snapshot for the VM instance . This allows rollback to the snapshot when a fault occurs. To back up data for the long term, we recommend that you use a backup service instead.

The snapshot feature applies to the following scenarios:

- Rapid recovery from faults: If an exception occurs in the production environment, you can use the snapshot rollback feature to recover the VM instance to its normal state quickly.
- Data development: You can create snapshots for the data on the VM instance and use the snapshot data to implement data mining, report queries, and application development and testing.
- Increased tolerance to errors: Before you perform a major operation such as system upgrade or data migration, we recommend that you create one or more snapshots. If a problem occurs during the upgrade or migration, you can use the snapshots to revert the VM instance to its original state.

6.1.6 Cryptography for HSM

Encrypts all plain texts of passwords for the virtualized platform to protect yourdata privacy and autonomy.

The following scenarios are implemented.

- Host password: Non-clear text display.
- Data storage password: Non-clear text display.
- Database password: Encrypted by using an HSM and is not displayed to users.
- Job password: Non-clear text display or not displayed to users for all jobsrunning on the Cloud.

6.1.7 Resource Delete Protection

Delete Policy Control

Controls can be set for vital resources to prevent accidental deletions.

Currently, three types of delete policies are supported: immediate deletion, delayed deletion, and never delete.

- Immediate deletion: The resources are physically deleted and removed from the database, and cannot be recovered.
- Delayed deletion: The resources are marked as deleted in the database but are not physically deleted. Within a certain period of time, you can recover the resources by using the recycle bin on the UI or through API calls. During this period, the resources still exist physically and occupy physical storage space, such as disk space. After the period, the resources are physically deleted and cannot be recovered.
- Never delete: The resources are marked as deleted in the database but are never physically deleted. The resources occupy physical storage space permanently.

The following resources can have delete policies set:

- VM instance: Immediate deletion, delayed deletion, and never delete. Default: delayed deletion.
- Disk: Immediate deletion, delayed deletion, and never delete. Default: delayed deletion.
- Image: Immediate deletion, delayed deletion, and never delete. Default: delayed deletion.

UI Delete Reminder

The UI provides a mechanism to protect vital resources from being accidentally deleted. When a resource is deleted, the system prompts the potential consequences of the deletion and displays the number of VM instances and disks directly associated with the resource. You need to confirm the deletion before you can proceed, which reduces the risk of accidental deletions.

6.1.8 Monitoring and Alarm

Primarily, the monitoring system and the alarm system provide monitoring and alarm services. The monitoring system monitors time-series data and events, while the alarm system pushes alarm messages to the specified endpoints.

The monitoring system provides monitoring data indicators including system performance and resource usage, which are displayed through big screens, dashboards, visual charts, and banners to give you a global view of the resource usage, system running state, and health degree of the

virtualized platform. You can also customize alarm rules and endpoints to achieve fine-grained monitoring and timely discovery and diagnosis of related issues.

Key features of the monitoring system:

- Time-series monitoring: Currently, two types of time-series data are supported.
  - Resource workloads: for example, the VM CPU utilization and host memory utilization.
  - Resource capacities: for example, the number of available IPs and running VMs.
- Event collection: Predefined events occurred in the virtualized platform are collected, such as host disconnection and HA startup of VMs.
- Alarm: Alarms are triggered when time-series data or events occur, and global prompts are made for key resources, such as the physical storage available capacity.
- Event: All operations are recorded and can be searched for.
- Customization: You can customize alarm rules and alarm message templates.

Key features of the alarm system:

- Push alarm messages to specified endpoints.
- The system provides a system endpoint by default, and you can set a DingTalk, HTTP application, or Microsoft Teams endpoint as needed.

6.2 Network Security

6.2.1 Security Group

A security group controls VM security and effectively filters TCP/UDP/ICMP data packets and controls the traffic of a specified NIC based on a security rule.

6.3 Permission Management Security

6.3.1 Two-Factor Authentication

Two-factor authentication is an additional layer of security on the static passwordauthentication. With two-factor authentication enabled, you need to enter the 6-digitsecurity code generated by the authentication app each time you log in to theplatform.

After you enable two-factor authentication and log in successfully, the QR code is no longerdisp layed. This effectively prevents malicious login and improves the system security.

6.3.2 AccessKey Authentication

Supports AccessKey authentication.

Local AccessKey: Consists of AccessKey ID and AccessKey Secret, serving as a security credential that authorizes third-party users to call virtualized platform APIs and access the platform 's cloud resources. Keep it confidential.

6.3.3 Task Event

Provides unified management for task logs and records the login and resource operations of accounts in the virtualized platform, including description of operation, task result, operator, login IP, and time when the task is created or completed, and the details of the operation. By using task events, you can meet the requirements of security analysis, intrusion detection, resource change tracking, and compliance checking.

7 Open Compatibility

7.1 Open API

nSSV provides a comprehensive and developer-friendly API support. In addition to native RESTful API support, where developers can design APIs based on the architecture principles and constraints defined in REST and program using HTTP-based languages, we also provide Java SDK and Python SDK for API calls to achieve corresponding functions.

The Development Manual accompanying the product describes the usage norms of RESTful APIs and SDKs and provides detailed definitions of all APIs. To lower the entry threshold for developers , we provide the API Inspector feature, where developers can quickly view the calling situations of query APIs and action APIs on the UI interface, making API information at hand. At the same time , developers can one-click copy the calling statements to third-party tools for debugging, which greatly improves efficiency and enhances developer experience.

7.2 Hardware Compatibility

nSSV supports software-hardware decoupled deployments. You canconfigure a list of hardware configurations and a list of virtualization platformsoftware according to your requirements. Regarding specific hardware brands, nSSV has no specific requirements or restrictions. If yourorganization has existing servers that you want to reuse, nSSVsupports this scenario and manages all your servers through the virtualization platform.