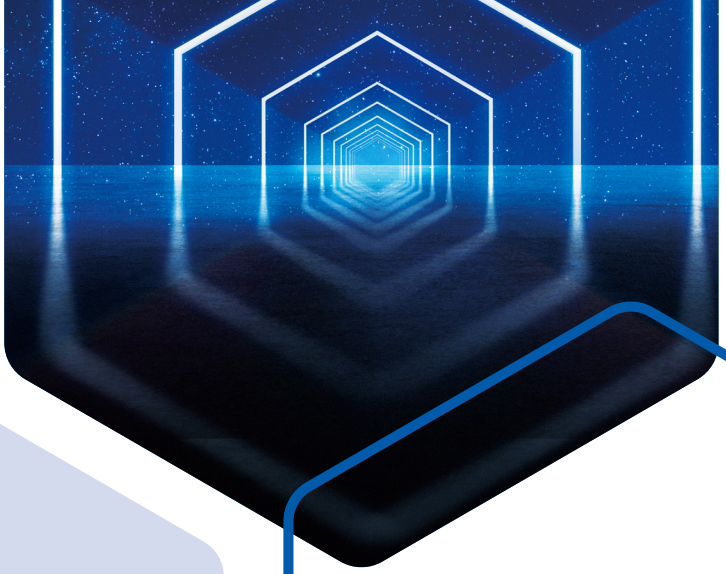
openEuler 22.03 LTS SP3

Technical White Paper



CONTENTS

01 Introduction

The openEuler open source community is incubated and operated by the OpenAtom Foundation.

openEuler is a digital infrastructure OS that fits into any server, cloud computing, edge computing, and embedded deployment. This secure, stable, and easy-to-use open source OS is compatible with multiple computing architectures. openEuler suits operational technology (OT) applications and enables the convergence of OT and information and communications technology (ICT).

The openEuler open source community is a portal available to global developers, with the goal of building an open, diversified, and architecture-inclusive software ecosystem for all digital infrastructure scenarios. It has a rich history of helping enterprises develop their software, hardware, and applications.

The openEuler open source community was officially established on December 31, 2019, with the original focus of innovating diversified computing architectures.

On March 30, 2020, the Long Term Support (LTS) version openEuler 20.03 was officially released, which was a new Linux distribution with independent technology evolution.

Later in 2020, on September 30, the innovative openEuler 20.09 version was released through the collaboration efforts of multiple companies, teams, and independent developers in the openEuler community. The release of openEuler 20.09 marked a milestone not only in the growth of the openEuler community, but also in the history of open sourced software in China.

On March 31, 2021, the innovative kernel version openEuler 21.03 was released. This version is enhanced in line with Linux kernel 5.10 and also incorporates multiple new features, such as live kernel upgrade and tiered memory expansion. These features improve multi-core performance and deliver the computing power of one thousand cores.

Fast forward to September 30, 2021, openEuler 21.09 was released. This premium version is designed to supercharge all scenarios, including edge and embedded devices. It enhances server and cloud computing features, and incorporates key technologies including cloud-native CPU scheduling algorithms for hybrid service deployments and KubeOS for containers.

On March 30, 2022, openEuler 22.03 LTS was released based on Linux kernel 5.10. Designed to meet all server, cloud, edge computing, and embedded workloads, openEuler 22.03 LTS is an all-scenario digital infrastructure OS that unleashes premium computing power and resource utilization.

On September 30, 2022, openEuler 22.09 was released to further enhance all-scenario innovations.

On December 30, 2022, openEuler 22.03 LTS SP1 was released, which is designed for hitless porting with best-of-breed tools.

On March 30, 2023, openEuler 23.03 was released. Running on Linux kernel 6.1, it streamlines technical readiness for Linux kernel 6.x and facilitates innovations for hardware adaptation and other technologies.

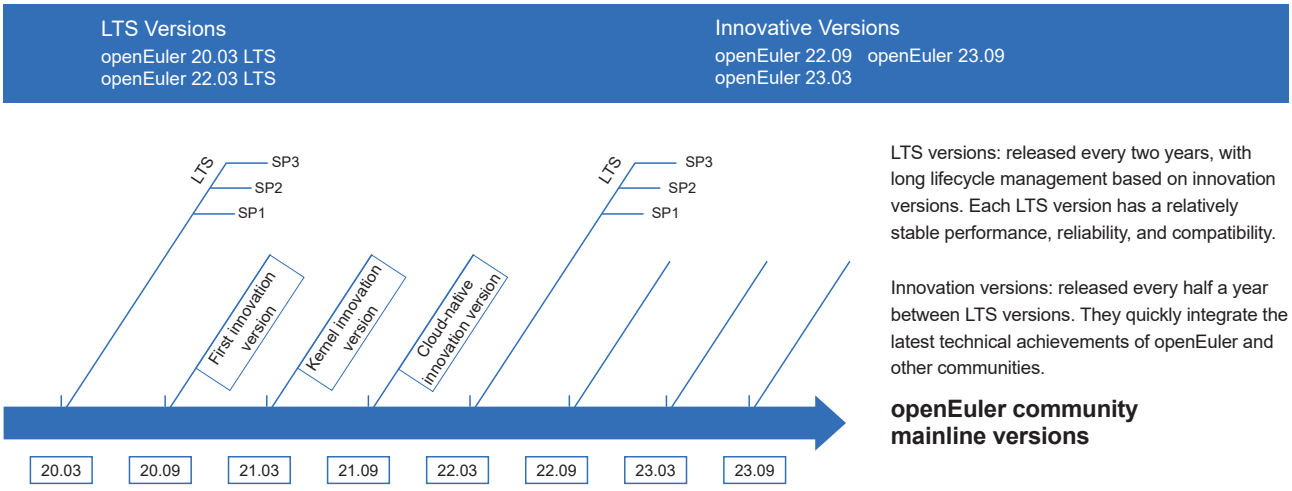
On June 30, 2023, openEuler 22.03 LTS SP2 was released, which enhances scenario-specific features and increases system performance to a higher level.

On September 30, 2023, the openEuler 23.09 innovation version was released based on Linux kernel 6.4. It provides a series of brand-new features to further enhance developer and user experience.

On November 30, 2023, openEuler 20.03 LTS SP4 was released, an enhanced extension of openEuler 20.03 LTS. openEuler 20.03 LTS SP4 provides excellent support for server, cloud native, and edge computing scenarios.

On December 30, 2023, openEuler 22.03 LTS SP3 was released. Designed to improve developer efficiency, it extends features for server, cloud native, edge computing, and embedded scenarios.

openEuler Version Management



LTS versions: released every two years, with long lifecycle management based on innovation versions. Each LTS version has a relatively stable performance, reliability, and compatibility.

Innovation versions: released every half a year between LTS versions. They quickly integrate the latest technical achievements of openEuler and other communities.

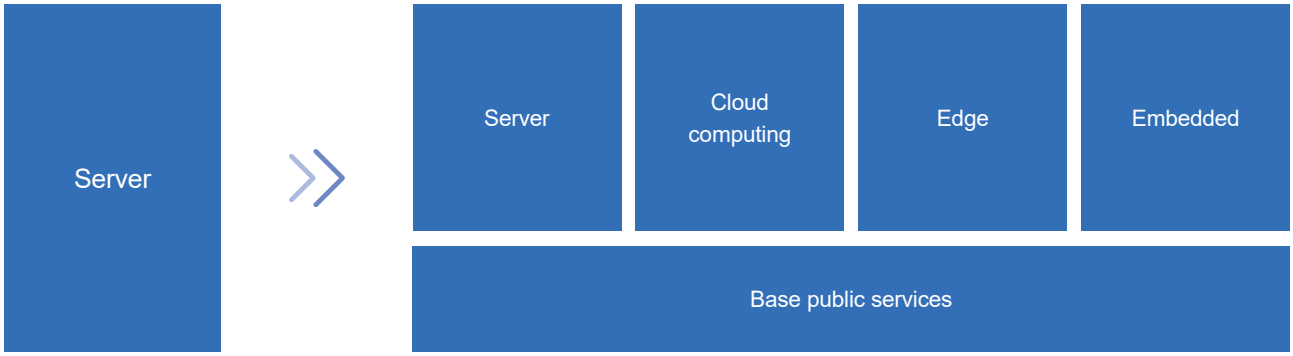
openEuler community mainline versions

As an OS platform, openEuler releases an LTS version every two years. Each LTS version provides enhanced specifications and a secure, stable, and reliable OS for enterprise users.

openEuler is built on tried-and-tested technologies. A new openEuler innovative version is released every 6 months to quickly integrate the latest technical achievements of openEuler and other communities. The innovative tech is first verified in the openEuler open source community as a single open source project, and then these features are added to each new release, enabling community developers to obtain the source code.

Technical capabilities are first tested in the open source community, and continuously incorporated into each openEuler release. In addition, each release is built on feedback given by community users to bridge the gap between innovation and the community, as well as improve existing technologies. openEuler is both a release platform and incubator of new technologies, working in a symbiotic relationship that drives the evolution of new versions.

Innovative Platform for All Scenarios



openEuler supports multiple processor architectures (x86, Arm, SW64, RISC-V, and LoongArch) and will support other brands (such as PowerPC) in the future, as part of a focus to continuously improve the ecosystem of diversified computing power.

The openEuler community is home to an increasing number of special interest groups (SIGs), which are dedicated teams that help extend the OS features from server to cloud computing, edge computing, and embedded scenarios. openEuler is built to

be used in any scenario, and comprises openEuler Edge and openEuler Embedded that are designed for edge computing and embedded deployments, respectively.

The OS is a perfect choice for ecosystem partners, users, and developers who plan to enhance scenario-specific capabilities. By creating a unified OS that supports multiple devices, openEuler hopes to enable a single application development for all scenarios.

Open and Transparent: The Open Source Software Supply Chain

The process of building an open source OS relies on supply chain aggregation and optimization. To ensure reliable open source software or a large-scale commercial OS, openEuler comprises a complete lifecycle management that covers building, verification, and distribution. The brand regularly reviews its software dependencies based on user scenarios, organizes the upstream community addresses of all the software packages, and verifies its source code by comparing it to that of the upstream communities. The build, runtime dependencies, and upstream communities of the open source software form a closed loop, realizing a complete, transparent software supply chain management.

Platform Architecture 02

System Framework

openEuler is an innovative open source OS platform built on kernel innovations and a solid cloud base to cover all scenarios. It is built on the latest trends of interconnect buses and storage media, and offers a distributed, real-time acceleration engine and base services. It provides competitive advantages in edge and embedded scenarios, and is the first step to building an all-scenario digital infrastructure OS.

openEuler 22.03 LTS SP3 runs on Linux kernel 5.10 and provides POSIX-compliant APIs and OS releases for server, cloud native, edge, and embedded environments. It is a solid foundation for intelligent collaboration across hybrid and heterogeneous deployments. openEuler 22.03 LTS SP3 is equipped with a distributed soft bus and KubeEdge+ edge-cloud collaboration framework, among other premium features, making it a perfect choice for collaboration over digital infrastructure and everything connected models.

In the future, the openEuler open source community will continue to innovate, aiming to promote the ecosystem and consolidate the digital infrastructure.

Cloud base:

- **KubeOS for containers**: In cloud native scenarios, the OS is deployed and maintained in containers, allowing the OS to be managed based on Kubernetes, just as service containers.
- **Secure container solution**: Compared with the traditional Docker+QEMU solution, the iSulad+shimv2+StratoVirt secure container solution reduces the memory overhead and boot time by 40%.
- **Dual-plane deployment tool eggo**: OSs can be installed with one click for Arm and x86 hybrid clusters, while deployment of a 100-node cluster is possible within just 15 minutes.

New scenarios:

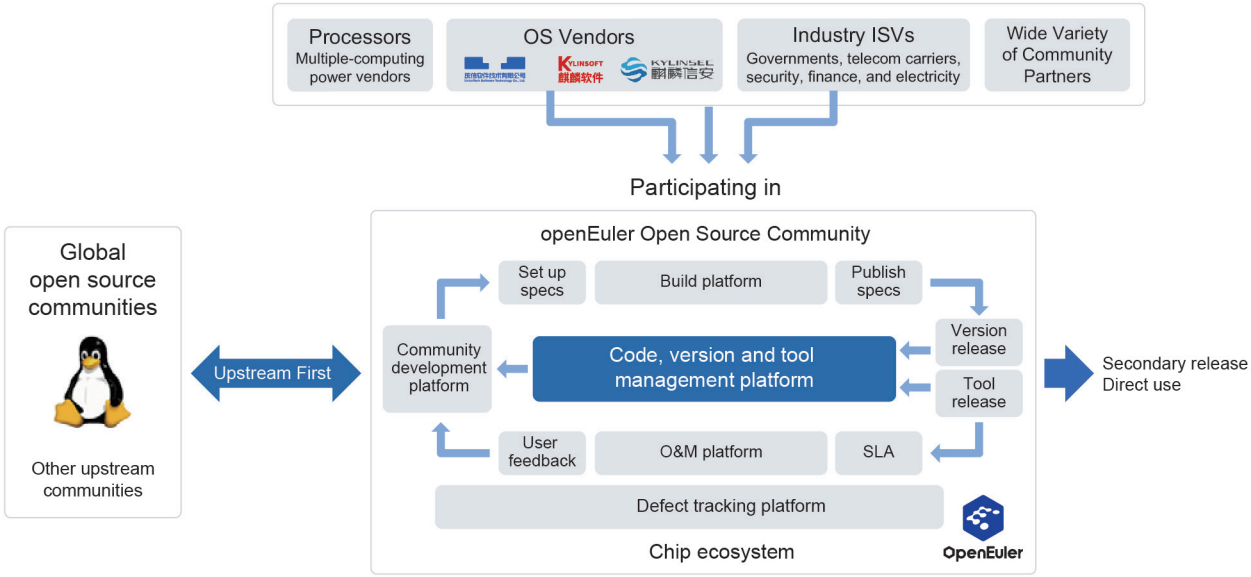
- **Edge computing**: openEuler 22.03 LTS SP3 Edge is released for edge computing scenarios. It integrates the KubeEdge+ edge-cloud collaboration framework to provide unified management, provisioning of edge and cloud applications, and other capabilities.
- **Embedded**: openEuler 22.03 LTS SP3 Embedded is released for embedded scenarios, helping compress images to under 5 MB and shorten image loading time to under 5 seconds.

Flourishing community ecosystem:

- **Desktop environments**: UKUI, DDE, Xfce, Kiran-desktop, and GNOME.
- **openEuler DevKit**: Supports OS migration, compatibility assessment, and various development tools such as secPaver which simplifies security configuration.

Platform Framework

The openEuler open source community partners with upstream and downstream communities to advance the evolution of openEuler versions.



Hardware Support

The openEuler open source community works with multiple vendors to build a vibrant southbound ecosystem. With participation of major chip vendors including Intel and AMD, all openEuler versions support x86, Arm, ShenWei, Loongson, and RISC-V CPU architectures, and a wide range of CPU chips, such as Loongson 3 series, Zhaoxin KaiXian and KaiSheng, Intel Ice Lake and Sapphire Rapids, and AMD EPYC Milan and Genoa. openEuler can run on servers from multiple hardware vendors and is compatible with NIC, RAID, Fibre Channel, GPU & AI, DPU, SSD, and security cards.

openEuler supports the following CPU architectures:

Hardware Type	x86	Arm
CPU	Intel, AMD, Zhaoxin, Hygon	Kunpeng, Phytium

openEuler supports the following servers:

Hardware Type	x86	Arm
Server	**Intel**: xFusion, Supermicro	**Kunpeng**: TaiShan
	AMD: Supermicro, H3C	**Phytium**: QS, PowerLeader, H3C
	Hygon: Sugon/Suma	
	Zhaoxin: Zhaoxin	

openEuler supports the following cards:

Hardware Type	x86	Arm
NIC	Huawei, Mellanox, Intel, NebulaMatrix, 3SNIC, NetSwift, Yunsilicon, Mucse	Huawei, Mellanox, Intel, NebulaMatrix, 3SNIC, NetSwift, Yunsilicon, Mucse
RAID	Avago, 3SNIC, PMC	Avago, 3SNIC, PMC
Fibre Channel	Marvell, Qlogic, Emulex	Marvell, Qlogic, Emulex
GPU & AI	NVIDIA	NVIDIA
SSD	Huawei	Huawei

Operating Environments 03

Servers

To install openEuler on a physical machine, check that the physical machine meets the compatibility and hardware requirements. For a full list, visit https://openeuler.org/en/compatibility/.

Item	Configuration Requirement
Architecture	AArch64, x86_64
Memory	≥ 4 GB
Drive	≥ 20 GB

VMs

Verify VM compatibility when installing openEuler.

Hosts running on openEuler 22.03 LTS SP3 support the following software package versions:

- libvirt-6.2.0-60.oe2203sp3
- libvirt-client-6.2.0-60.oe2203sp3
- libvirt-daemon-6.2.0-60.oe2203sp3
- qemu-6.2.0-86.oe2203sp3
- qemu-img-6.2.0-86.oe2203sp3

openEuler 22.03 LTS SP3 is compatible with the following guest OSs for VMs:

Host OS	Guest OS	Architecture
openEuler 22.03 LTS SP3	CentOS 6	x86_64
openEuler 22.03 LTS SP3	CentOS 7	AArch64
openEuler 22.03 LTS SP3	CentOS 7	x86_64
openEuler 22.03 LTS SP3	CentOS 8	AArch64
openEuler 22.03 LTS SP3	CentOS 8	x86_64
openEuler 22.03 LTS SP3	Windows Server 2016	AArch64
openEuler 22.03 LTS SP3	Windows Server 2016	x86_64
openEuler 22.03 LTS SP3	Windows Server 2019	AArch64
openEuler 22.03 LTS SP3	Windows Server 2019	x86_64

Item	Configuration Requirement
Architecture	AArch64, x86_64
CPU	≥ 2 CPUs
Memory	≥ 4 GB
Drive	≥ 20 GB

Edge Devices

To install openEuler on an edge device, check that the edge device meets the compatibility and minimum hardware requirements.

Item	Configuration Requirement
Architecture	AArch64, x86_64
Memory	≥ 4 GB
Drive	≥ 20 GB

Embedded Devices

To install openEuler on an embedded device, check that the embedded device meets the compatibility and minimum hardware requirements.

Item	Configuration Requirement
Architecture	AArch64, AArch32
Memory	≥ 512 MB
Drive	≥ 256 MB

04 Scenario-specific Innovations

AI

AI is redefining OSs by powering intelligent development, deployment, and O&M. openEuler supports general-purpose architectures like Arm, x86, and RISC-V, and next-gen AI processors like NVIDIA and Ascend. Further, openEuler is equipped with extensive AI capabilities that have made it a preferred choice for diversified computing power.

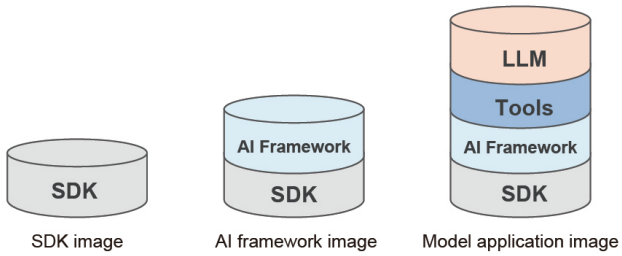
openEuler for AI

openEuler offers an efficient development and runtime environment that containerizes software stacks of AI platforms with out-of-the-box availability.

Feature Description

(1) openEuler supports TensorFlow and PyTorch frameworks and software development kits (SDKs) of major computing architectures, such as Compute Architecture for Neural Networks (CANN) and Compute Unified Architecture (CUDA), to make it easy to develop and run AI applications.

(2) Environment setup is further simplified by containerizing software stacks. openEuler provides three types of container images:



- **SDK images**: Use openEuler as the base image and install the SDK of a computing architecture, for example, Ascend CANN and NVIDIA CUDA.
- **AI framework images**: Use the SDK image as the base and install AI framework software, such as PyTorch and TensorFlow.
- **Model application images**: Provide a complete set of toolchains and model applications.

For more information, see the openEuler AI Container Image User Guide.

Application Scenarios

openEuler uses AI container images to simplify deployment of runtime environments. You can select the container image that best suits your requirements and complete the deployment in a few simple steps.

- **SDK images**: You can develop and debug Ascend CANN or NVIDIA CUDA applications using an SDK image, which provides a compute acceleration toolkit and a development environment. These containers offer an easy way to perform high-performance computing (HPC) tasks, such as large-scale data processing and parallel computing.
- **AI framework images**: This type of containers is designed to support AI model development, training, and inference.
- **Model application images**: Such an image contains a complete AI software stack and purpose-built models for model inference and fine-tuning.

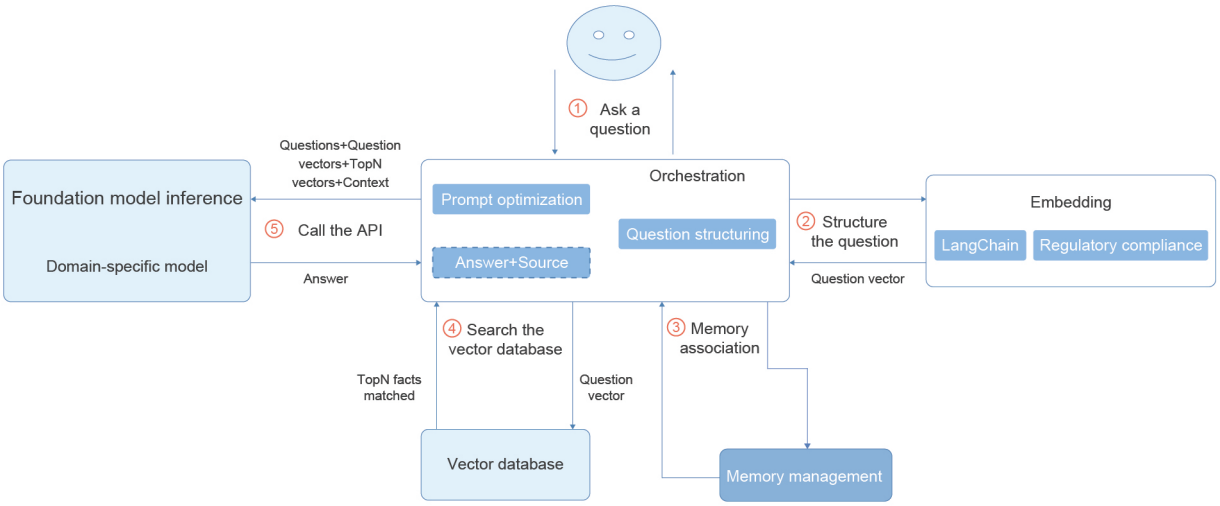
AI for openEuler

AI makes openEuler more intelligent. EulerCopilot, an intelligent Q&A platform, is developed using foundation models and openEuler data. It assists in code generation, problem analysis, and system O&M.

Feature Description

EulerCopilot is accessible via web or shell.

- **Web**: Provides basic OS knowledge, openEuler data, O&M methods, and project introduction and usage guidance.
- **Shell**: Delivers user-friendly experience using natural languages.



Application Scenarios

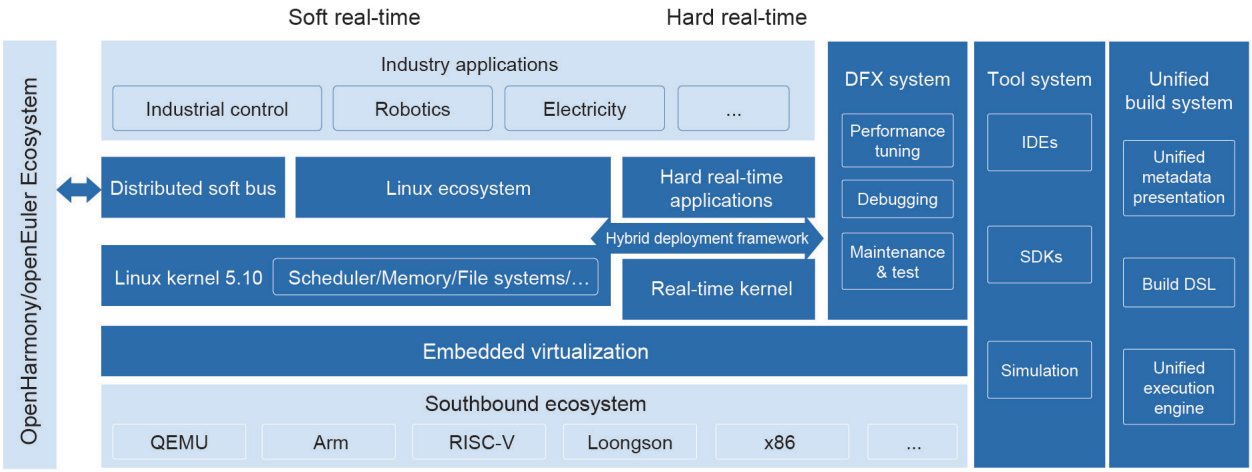
- **Common users**: Best practices for openEuler operations, such as porting applications to openEuler.
- **Developers**: Extensive knowledge of openEuler contribution processes, key features, and project development.
- **O&M personnel**: Efficient system management to quickly resolve common problems.

Embedded

openEuler 22.03 LTS SP3 Embedded offers a distributed soft bus and software package build capabilities to allow for mixed criticality deployment of real-time and non-real-time planes.

openEuler Embedded helps build applications for industrial control and robotics in a closed loop design, whereby innovations help optimize its embedded technology stack and ecosystem. openEuler 22.03 LTS SP3 Embedded is equipped with an embedded virtualization base that is available in the Jailhouse virtualization solution or the OpenAMP lightweight hybrid deployment solution. You can select the most appropriate solution to suit your services. openEuler 22.03 LTS SP3 Embedded supports the Robot Operating System (ROS) Humble version, which integrates core software packages such as ros-core, ros-base, and simultaneous localization and mapping (SLAM) to meet the ROS 2 runtime requirements. Future versions will utilize contributions from ecosystem partners, users, and community developers, to increase support for chip architectures such as RISC-V and LoongArch, and add capabilities such as industrial middleware, ROS middleware, and simulation systems.

System Architecture



Southbound Ecosystem

openEuler Embedded Linux supports AArch64 and x86-64 chip architectures. openEuler 22.03 LTS SP3 Embedded currently supports the RK3588 chip and will support Loongson and Phytium chips.

Embedded Virtualization Base

openEuler Embedded uses an elastic virtualization base that enables multiple OSs to run on a system-on-a-chip (SoC). The base incorporates a series of technologies including bare metal, embedded virtualization, lightweight containers, LibOS, trusted execution environment (TEE), and heterogeneous deployment.

- The bare metal hybrid deployment solution runs on OpenAMP to manage peripherals by partition at a high performance level; however, it delivers poor isolation and flexibility. This solution supports the hybrid deployment of UniProton/Zephyr/RT-Thread and openEuler Embedded Linux.
- Partitioning-based virtualization is an industrial-grade hardware partition virtualization solution that runs on Jailhouse. It offers superior performance and isolation but inferior flexibility. This solution supports the hybrid deployment of FreeRTOS and openEuler Embedded Linux.

- Real-time virtualization is a solution originating from the openEuler community that balances performance, isolation, and flexibility. This solution supports the hybrid deployment of Zephyr and openEuler Embedded Linux.

MICA Deployment Framework

The mixed-criticality (MICA) deployment framework is a unified environment that masks the differences between technologies that comprise the embedded elastic virtualization base. The multi-core capability of hardware combines the universal Linux OS and a dedicated real-time operating system (RTOS) to make full use of all OSs.

The MICA deployment framework covers lifecycle management, cross-OS communication, service-oriented framework, and multi-OS infrastructure.

- Lifecycle management provides operations to load, start, suspend, and stop the client OS.
- Cross-OS communication uses a set of communication mechanisms between different OSs based on shared memory.
- Service-oriented framework enables different OSs to provide their own services. For example, Linux provides common file system and network services, and the RTOS provides real-time control and computing.
- Multi-OS infrastructure integrates OSs through a series of mechanisms, covering resource expression and allocation and unified build.

The MICA deployment framework provides the following functions:

- Lifecycle management and cross-OS communication for openEuler Embedded Linux and the RTOS (Zephyr or UniProton) in bare metal mode
- Lifecycle management and cross-OS communication for openEuler Embedded Linux and the RTOS (FreeRTOS) in partitioning-based virtualization mode

Northbound Ecosystem

- **Northbound software packages**: Over 350 common embedded software packages can be built using openEuler.
- **ROS runtime**: openEuler 22.03 LTS SP3 Embedded supports the ROS 2 Humble version, which contains core software packages such as ros-core, ros-base, and SLAM. It also provides the ROS SDK that simplifies embedded ROS development.
- **Soft real-time kernel**: This capability helps respond to soft real-time interrupts within microseconds.
- **DSoftBus**: The distributed soft bus system (DSoftBus) of openEuler 22.03 LTS SP3 Embedded integrates the DSoftBus and point-to-point authentication module of OpenHarmony. It implements interconnection between openEuler-based embedded devices and OpenHarmony-based devices as well as between openEuler-based embedded devices.
- **iSula containers**: openEuler or other OS containers can be deployed on embedded devices to simplify application porting and deployment.

UniProton

UniProton is an RTOS that features ultra-low latency and flexible MICA deployments. It is suited for industrial control because it supports both microcontroller units and multi-core CPUs. UniProton provides the following capabilities:

- Compatible with CPU architectures like Cortex-M, AArch64, and x86_64, and supports M4, RK3568, x86_64, Hi3093, and Raspberry Pi 4B.
- Connects with openEuler Embedded Linux on Raspberry Pi 4B, Hi3093, and x86_64 devices in bare metal mode.
- Can be debugged using GDB on openEuler Embedded Linux.
- Over 890 POSIX APIs available for file systems, device management, shell consoles, and networks.

Application Scenarios

Embedded systems help supercharge computing performance in a wide range of industries and fields, including industrial and power control, robotics, aerospace, automobiles, and healthcare.

05 Kernel Innovations

What's New in the openEuler Kernel

openEuler 22.03 LTS SP3 runs on Linux kernel 5.10 and inherits the competitive advantages of community versions and innovative features released in the openEuler community.

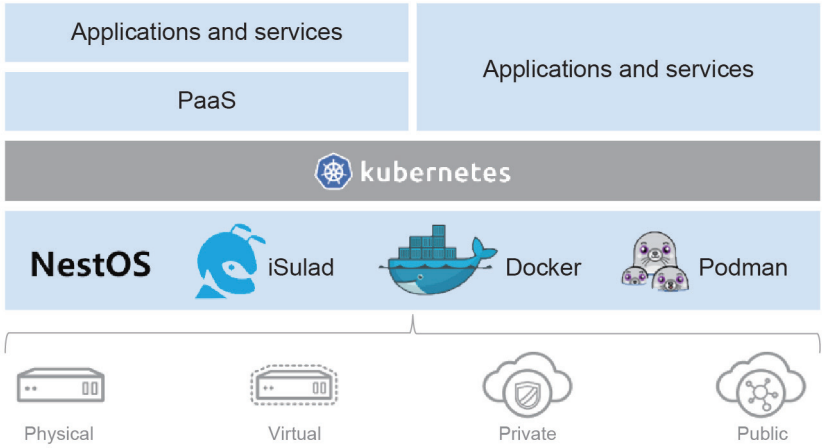
- **Dynamic memory isolation and release**: Memory pages are dynamically isolated and de-isolated. When isolated, the original memory is migrated and the to-be-isolated and de-isolated memory pages are recorded for kernel and page reliability. When data is replicated on isolated memory pages with uncorrectable errors, the machine check safe (MCS) technology prevents the kernel from resetting.
- **Online CPU inspection**: To avoid silent data corruption that is a common cause of data loss, faulty cores are detected and isolated to prevent faults before they are exacerbated.
- **Adaptive provisioning of computing power**: To ensure consistency and reliability of certain applications (such as cloud desktop systems) running on multi-core servers, computing power is dynamically provisioned based on load changes. When the service load is heavy, a lightweight task search algorithm triggers idle CPUs to execute tasks more quickly. This mechanism implements fast load balancing between cores and maximizes CPU utilization. In the event of resource contention, services are processed by priority and computing power is provisioned preferentially to foreground tasks with high priorities over background tasks with low priorities.
- **Power-aware scheduling**: At the service layer, memory access bandwidth, CPU load, and other information are collected to ensure sufficient resources for critical threads. A physical topology is introduced so that the P-state control mechanism extends to new dimensions, further reducing power consumption beyond the limits of single-die frequency and voltage regulation. This feature minimizes power consumption when the service load is low.
  - An escape channel runs on the physical topology. The power consumption automatically adjusts to the CPU load and memory access bandwidth. Specifically, light-load services are assigned to fewer CPUs to reduce the power consumption, whereas heavy-load services are assigned to more CPUs to ensure the quality of service.
  - The timer collects load and bandwidth information and detects the memory access bandwidth to prevent cross-die access.
  - In the OS, the new mechanisms such as P-state control, static power awareness, and service labels help enable smart frequency control, set the CPU idle time management, and implement dynamic voltage and frequency scaling (DVFS).
- **Enhanced core isolation**: CPUs are classified into housekeeping and non-housekeeping. The former executes background processes such as periodic system clock maintenance, while the latter executes service processes. Background processes and interrupts are all allocated to housekeeping CPUs to prevent noise from affecting service process. This enhanced core isolation improves service performance, especially needed for HPC workloads.
- **Performance monitor unit (PMU) indicators**: Pressure detection of CPUs, I/Os, memory, and interrupt requests (Pressure Stall Information, or PSI) can be performed globally or in cgroup v1 only. When multiple services share node resources, indicators such as PSI are used to measure system contention, service throughput, and delay. These indicators are essential to locating system resource bottlenecks, understanding the resource demand of specific service processes, and dynamically adjusting resource allocation. CPU and memory pressure sources are classified to quickly identify problems and improve the quality of online services and system health.
- **KVM TDP MMU**: In Linux kernel 5.10 and later, KVM can scale to match demand for memory virtualization. This feature is contributed by Intel to the openEuler community. Compared with the traditional KVM memory management unit (MMU), the two dimensional paging MMU, or TDP MMU, offers more efficient handling of concurrent page faults and better support for large-scale VM deployments, such as those with multiple vCPUs and large memory. In addition, the new Extended Page Tables (EPT) and Nested Page Tables (NPT) traversal interface boosts host memory utilization by removing the dependency on the rmap data structure that is typical in traditional memory virtualization solutions. openEuler 22.03 LTS SP3 provides KVM TDP MMU support for Linux kernel 5.10 to 5.15. Further optimizations will be available in the SP4 version.

06 Cloud Base

NestOS

NestOS is a cloud OS incubated in the openEuler community. It runs rpm-ostree and Ignition technologies over a dual rootfs and atomic update design, and uses nestos-assembler for quick integration and build. NestOS is compatible with Kubernetes and OpenStack, and reduces container overheads and provides extensive cluster components in large-scale containerized environments.

Feature Description



- **Out-of-the-box availability**: integrates popular container engines such as iSulad, Docker, and Podman to provide lightweight and tailored OSs for the cloud.
- **Easy configuration**: uses the Ignition utility to install and configure a large number of cluster nodes with a single configuration.
- **Secure management**: runs rpm-ostree to manage software packages and works with the openEuler software package source to ensure secure and stable atomic updates.
- **Hitless node updating**: uses Zincati to provide automatic node updates and reboot without interrupting services.
- **Dual rootfs**: executes dual rootfs for active/standby switchovers, to ensure integrity and security during system running.

Application Scenarios

NestOS aims to satisfy the demands of containerized cloud applications by resolving problems such as inconsistent and repeated O&M operations of stacks and platforms. These problems are typically caused by decoupling of containers and underlying environments when using container and container orchestration technologies for rollout and O&M. NestOS resolves these problems to ensure consistency between services and the base OS.

07 Enhanced Features

SysCare

In the world of Linux development, there is a long-standing problem: how to quickly and reliably fix vulnerabilities and rectify faults without causing interruptions to online services.

A common solution to this problem is hot patching. During service running, code-level repair is directly performed on faulty components without affecting services. However, hot patches are complex to make because they must match the source code, and are also difficult to manage. There was no simple and unified patching mechanism for user-mode components that need to adapt to different file forms, programming languages, compilation methods, and running modes.

To solve this problem, SysCare was developed.

SysCare is a system-level hotfix software that provides security patches and hot fixing for OSs. It can fix system errors without restarting hosts. SysCare combines kernel-mode and user-mode hot patching to take over system repair, saving time for users to focus on other aspects of their business. In the future, live OS updates will be provided to improve O&M efficiency.

Feature Description

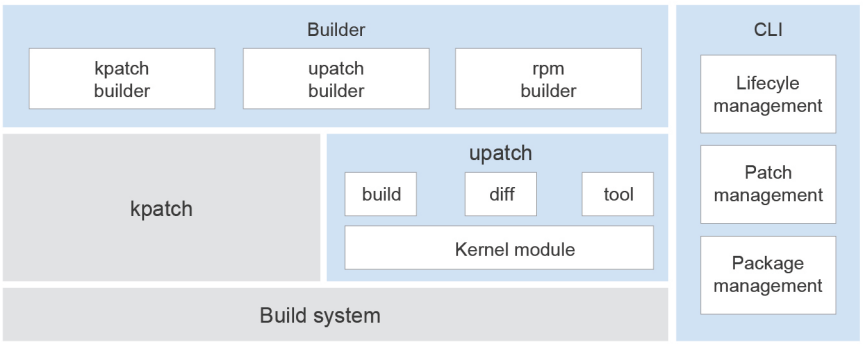
- **Hot patch making**

  To generate a hot patch RPM package, users only need to input the paths to the source RPM package, debuginfo RPM package, and patches to be installed of the target software without modifying the software source code.

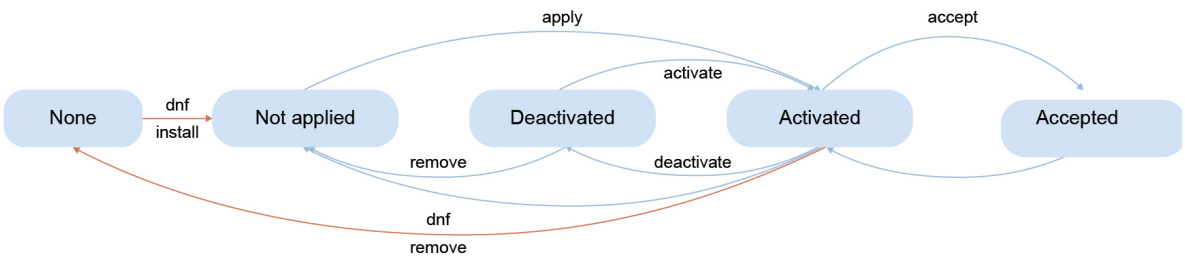
- **Patch lifecycle management**

  SysCare provides a complete and easy-to-use patch lifecycle management method to simplify usage. Users can manage hot patches by running a command. By utilizing the RPM system, SysCare can build hot patches with complete dependencies, so that hot patches can be integrated into the software repository, without the need for special processing for distribution, installation, update, and uninstallation.

SysCare architecture



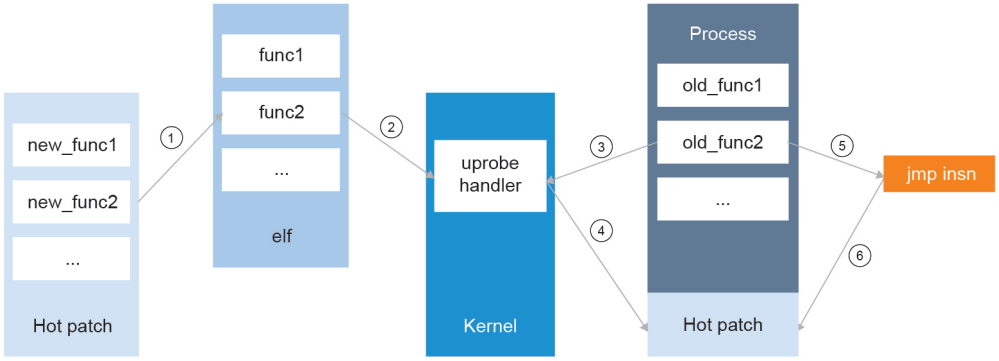
Lifecycle of a hot patch



- **User-mode hot patches for ELF files (program executable files)**

  SysCare uses the uprobe technology to bind hot patches to ELF files. When ELF files are running, uprobe can make the patches take effect. In this way, the patching process does not need to be monitored. The patches can take effect after being installed or when a new process is running, regardless of whether the user process is running. Apart from that, this technology can also install hot patches for dynamic libraries. The following figure shows how a patch takes effect.

Process of making a patch take effect

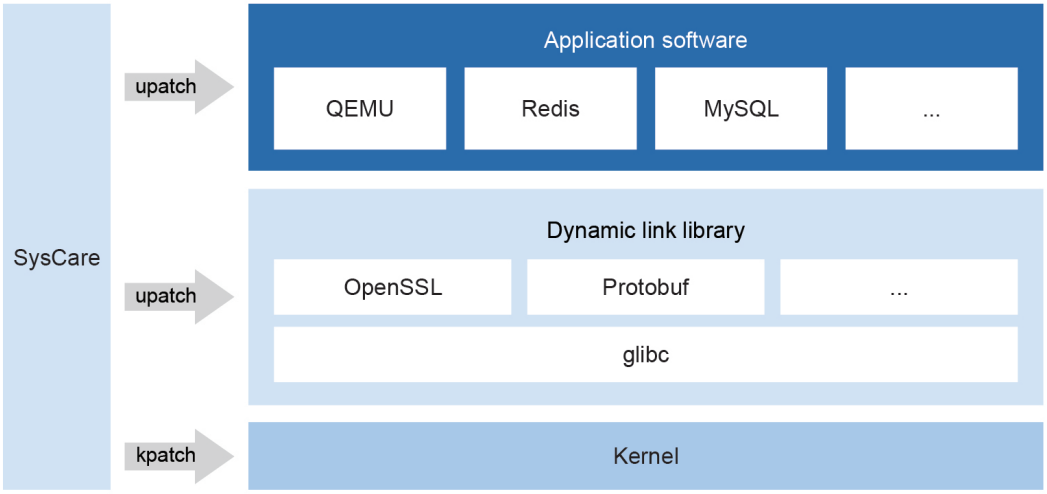


1. Execute the uprobe system call and add a uprobe breakpoint at the function to be modified.
2. Register a uprobe handler.
3. Call the uprobe handler when the function is running.
4. Map the patch to the address space of the current process.
5. Perform security check and change the first instruction of the function to a JUMP instruction, pointing to the patch address.
6. Jump to the patch address for execution.

- **Integrating kernel-mode and user-mode hot patches**

  By utilizing the upatch and kpatch technologies, SysCare streamlines the hot patch software stack from top to bottom for applications, dynamic libraries, and kernels, and provides seamless full-stack hot fixing.

Application scope of hot patches



- **New features**
  - Configures the dependencies of hot patches when they are created.
  - Manages multiple user-mode patches.
  - Detects conflicts between user-mode hot patches.
  - Forcibly overwrites user-mode hot patches when conflicts occur.
- **Constraints**
  - Only available for 64-bit OSs
  - Only available for ELF files, with no support for interpreted languages and pure assembly modification
  - Only available for the GCC and G++ compilers, and cross compilation not supported
  - LTO optimization not supported

Application Scenarios

Scenario 1: quick fix of common vulnerabilities and exposures (CVEs)

Scenario 2: temporary locating for live-network issues

GCC for openEuler

GCC for openEuler is a high-performance compiler oriented to the openEuler ecosystem for various scenarios. It is developed on the open source GNU Compiler Collection (GCC) and inherits the capabilities of the open source GCC. GCC for openEuler optimizes C, C++, and Fortran deployments in terms of instructions, memory, and automatic vectorization, to adapt to and unleash the compute of hardware platforms, such as Kunpeng, Phytium, and LoongArch. By supporting the Plug-IN (PIN) framework, it can obtain universal plug-ins, support multi-architecture computing and micro-architectures, and tap into the performance, security, reliability, and O&M capabilities of openEuler projects. Also, the continuous FGO (CFGO) design helps improve application performance of databases.

- **Ultimate performance**: GCC for openEuler provides multiple compilation optimizations across structure, Kunpeng affinity, FDO, and pipeline. In the intrate performance test of SPEC CPU 2017, GCC for openEuler achieved 20% higher performance over GCC 10.3 of the upstream community. For typical cloud applications like MySQL, Redis, Nginx, and Ceph, the feature helps improve the scenario-specific performance by 5% to 10% compared with GCC 10.3.
- **Ease of use**: The cross compilation toolchain of GCC for openEuler has been used to build the openEuler Embedded system. Thanks to the PIN framework, developers do not need to modify the internal logic of compilers when developing plug-ins. The framework helps developers implement independent compilation optimizations and develop compilation tools.
- **Mature ecosystem**: By supporting programming languages such as C/C++, Fortran, and Objective-C/C++ and enhancing distributed storage, database, web, and core network workloads, the feature improves application performance by 15%. It supports Arm, x86, RISC-V, SW-64, and LoongArch and other mainstream architectures, as well as Kunpeng, Phytium, Zhaoxin, Hygon, Loongson, Sunway, Intel, and AMD processors.
- **Reliable design**: GCC for openEuler has passed multiple tests of the open source ecosystem, including basic function, reliability, and compatibility tests with millions of test cases generated to ensure quality protection. The openEuler community vulnerability handling ensures over 90% of public vulnerabilities are detected within 12 hours to prevent vicious attacks.

New capabilities:

- Multiple GCC versions now support OpenMP, including the gcc-toolset-12 package series that run on GCC 12.3.0. Fortran supports OpenMP 4.5, while C/C++ supports some OpenMP 5.0 specifications.
- Last-level cache (LLC) allocation is optimized. By analyzing memory multiplexing on the main execution paths in a program, GCC for openEuler determines and sorts hot data. Then, prefetch instructions are inserted to pre-allocate data to the LLC, reducing LLC misses.
- Optimizations of CPUBench help intelligently identify and reduce instructions while boosting performance.

Feature Description

Support for GCC of multiple versions:

- **LLC allocation optimization**
  - Hot data can be prefetched and allocated to the LLC to reduce LLC misses and improve performance.
  - The -**fllc-allocate** option is supported.
- **Base performance optimizations**
  - **CRC**: Cyclic redundancy check (CRC) code is identified to generate efficient hardware instructions.- Only available for ELF files, with no support for interpreted languages and pure assembly modification.

- **IF-conversion**: Enhanced to use more registers and reduce conflicts.
- **Multiplication**: Arm instructions are combined to convert low-order multiplications into high-order multiplication instructions.
- **CMLT instruction generation**: CMLT instructions are generated for some elementary arithmetic operations to reduce the number of instructions.
- **Vectorization**: Redundant instructions generated during vectorization are identified and simplified, and shorter arrays can be vectorized.
- **maxmin and UZP1/UZP2 instructions**: The maxmin and UZP1/UZP2 instructions are optimized to reduce the total instructions and improve performance.
- **LDP and STP instructions**: Each LDP or STP instruction with poor performance is split into two LDR and two STR instructions.
- **AES instructions**: The AES algorithm code is identified to accelerate instructions using hardware.

Application Scenarios

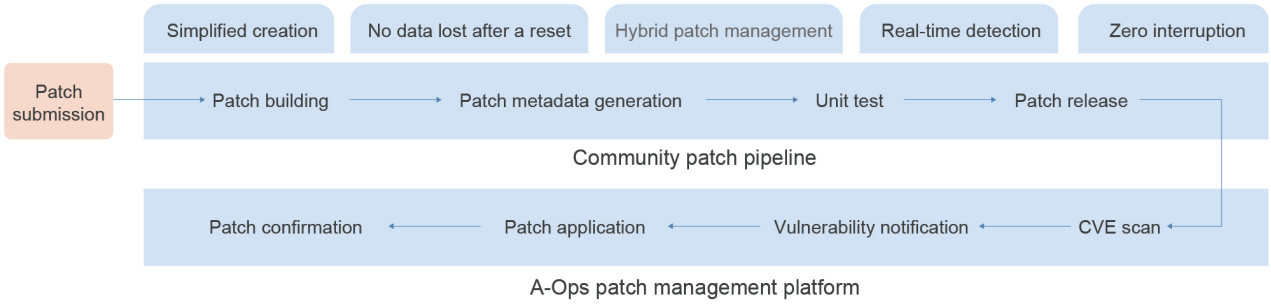
In the general-purpose computing field, GCC for openEuler reported 20% and 10% performance gains over GCC 10.3 in the SPEC CPU 2017 and CPUBench tests, respectively.

In the high-performance computing (HPC) field, tests relating to Weather Research and Forecasting (WRF) and Microsystems Engineering and Materials Science (NEMO) revealed that GCC for openEuler delivers 10% higher performance than GCC 10.3 of the upstream community. Further, when testing the hot data prefetch in OpenFOAM, WRF, and RELION applications, hot data from the hotspot function is preferentially prefetched and allocated to the LLC to reduce LLC misses and increase system speed.

A-Ops

A-Ops is an intelligent O&M platform that covers data collection, health check, and fault diagnosis and rectification. In addition to ensuring stable system runtime, A-Ops can help O&M personnel fix vulnerabilities or roll back fixes. The A-Ops project includes the following sub-projects: fault detection (Gala), fault locating (X-diagnosis), and vulnerability rectification (Apollo).

Released with openEuler 22.03 LTS SP3, Apollo is an intelligent patch management framework that integrates core functions such as vulnerability scanning, CVE fixing (with cold/hot patches), and hot patching rollback. Apollo periodically downloads and synchronizes security advisories and sets scheduled tasks to scan for vulnerabilities.



Feature Description

Apollo enables the intelligent management of kernel patches.

- **Hot patch source management**: When openEuler vulnerabilities are released through a security advisory, the software package used for fixing the vulnerabilities is also released in the update repository. By default, after openEuler is installed, the cold patch update repository of the corresponding OS version is provided. Users can also configure the update repository of cold or hot patches.
- **Vulnerability scanning**: Manual or periodic cluster scans can be performed to check the impact of CVEs on a cluster and cold or hot patches are provided for repair.
- **Hybrid patch management**: Cold and hot patches can be applied independently or together to implement silent incorporation of hot patches on the live network and reduce hot patch maintenance costs.
- **Hot patch lifecycle management**: hot patch removal, rollback, and query.

Application Scenarios

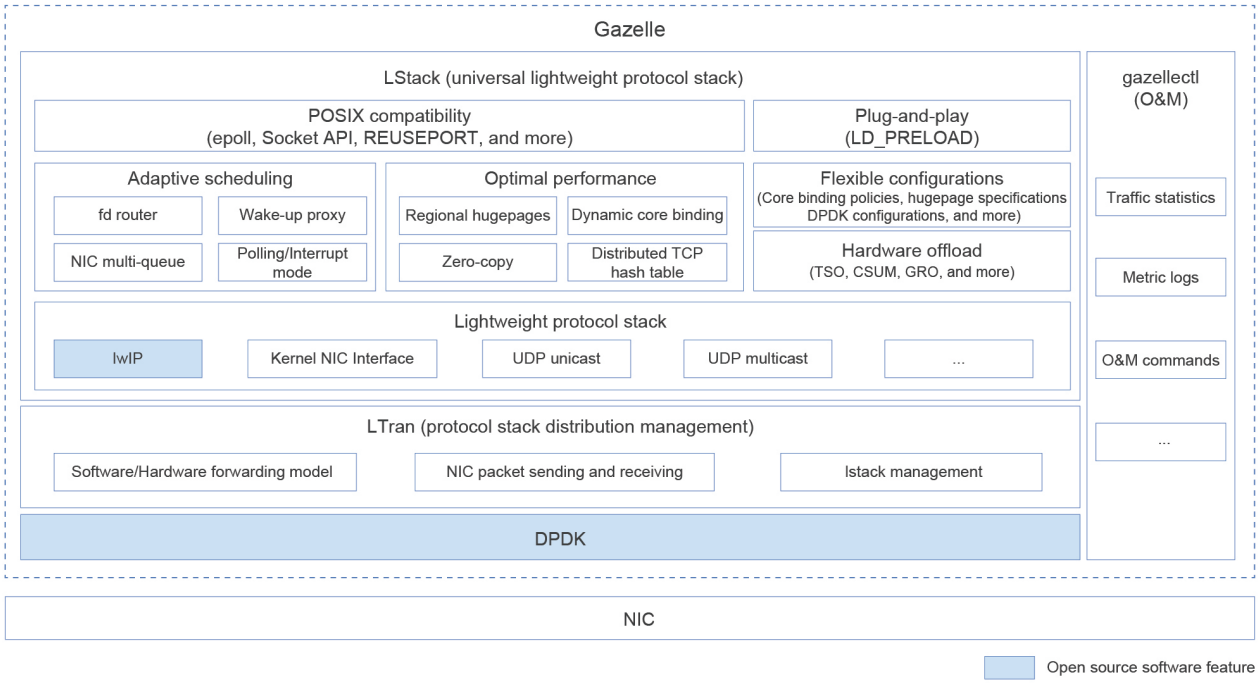
Apollo can help system administrators or developers who need to perform O&M and want to quickly detect and fix kernel vulnerabilities in single-node systems or clusters, or even roll back cold and hot patching. This greatly reduces patch management costs, ensures cluster security, and improves troubleshooting efficiency.

Gazelle

Gazelle is a high-performance user-mode protocol stack. It directly reads and writes NIC packets in user mode based on the Data Plane Development Kit (DPDK), transmits the packets through shared hugepage memory, and uses the LwIP protocol stack, thereby greatly improving the network I/O throughput of applications and accelerating the network for databases. With Gazelle, high performance and universality can be achieved at the same time. In openEuler 22.03 LTS SP3, support for the UDP protocol and related interfaces is added for Gazelle to enrich the user-mode protocol stack.

Feature Description

Gazelle architecture



High performance (ultra-lightweight): High-performance lightweight protocol stack capabilities are implemented based on DPDK and LwIP.

- **Ultimate performance**: A highly linearizable concurrent protocol stack is implemented based on technologies such as regional hugepage splitting, dynamic core binding, and full-path zero-copy.
- **Hardware acceleration**: TCP Segmentation Offload (TSO), checksum (CSUM) offload, Generic Receive Offload (GRO), and other offload technologies streamline the vertical acceleration of software and hardware.
- **Universality (POSIX compatibility)**: Full compatibility with POSIX APIs eliminates the need to modify applications. The recvfrom and sendto interfaces of UDP are supported.
- **General networking model**: Adaptive scheduling of the networking model is implemented based on mechanisms such as fd router and wake-up proxy. The UDP multi-node multicast model meets the requirements of any network application scenario.
- **Usability (plug-and-play)**: LD_PRELOAD enables zero-cost deployment by removing the requirement for service adaptation.

- **Easy O&M (O&M tool)**: Complete O&M methods, such as traffic statistics, metric logs, and CLI commands, are provided.

New features

- Available in single VLAN, bond4, and bond6 modes, and supports NIC self-healing after network cables are reinstalled.
- A single-instance Redis application on Kunpeng 920 VMs supports over 5,000 connections, improving performance by more than 30%.
- The TCP_STREAM and TCP_RR tests of netperf (packet length less than 1,463 bytes) are supported.
- Logs of the LStack, lwIP, and gazellectl modules of Gazelle are refined for more accurate fault locating.

Verified NICs

Mellanox ConnectX 4/ConnectX 5

Application Scenarios

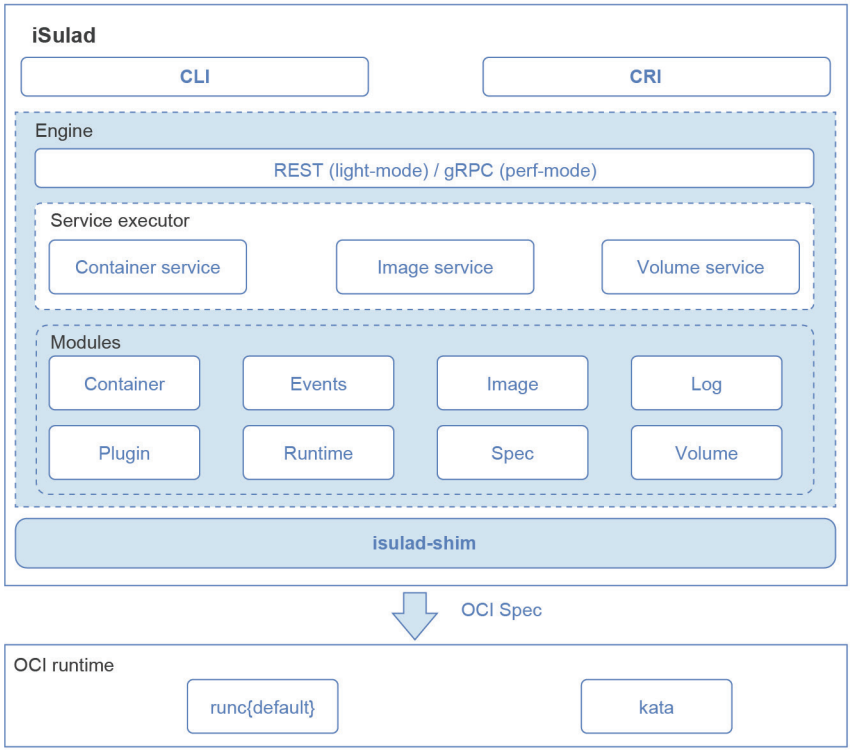
Gazelle improves service performance for applications where the network protocol stack is a performance bottleneck.

OCI Runtime for iSulad

Open Container Initiative (OCI) is a lightweight and open governance project dedicated to creating an open industry standard for container formats and runtimes. Developed with the support of the Linux Foundation, it aims to let any container runtimes that support OCI Runtime use OCI images to run containers. iSulad is a lightweight container engine compatible with mainstream container ecosystems, and supports standard southbound OCI APIs and can connect to multiple OCI runtimes, such as runc and kata.

Feature Description

As OCI has matured dramatically in the last few years, container runtimes that comply with OCI Runtime have been fitting into an expanding scope of application scenarios. runc is the first reference implementation of OCI Runtime. The current openEuler version optimizes the interconnection between iSulad and OCI Runtime. It rectifies known defects and adds the isula top and isula attach APIs, and sets runc as the default runtime for iSulad. After the default runtime is switched to runc, the dependency library of isulad-shim connected to OCI Runtime is changed to an independent and tailored static tool library. The switchover avoids existing process breakdowns caused by tool library upgrades, and reduces the memory overhead of containers.



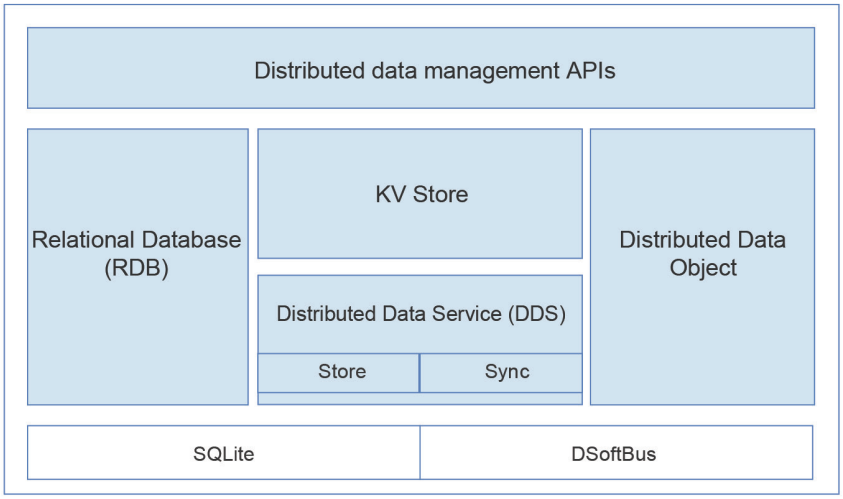
Application Scenarios

This feature is designed for container engines that comply with OCI Runtime. After the default container runtime is changed to runc, this feature can also be used in scenarios with lower memory overhead and live upgrades.

Distributed Data Management

The distributed data management system is a data management capability ported from the OpenHarmony community. This system encapsulates over 100 universal APIs that adopt DSoftBus dynamic networking to provide a range of data synchronization, such as strong and weak consistency, for each device node on the network.

Feature Description



- **RDB**: manages data based on a relational model. It uses SQLite as the underlying persistent storage engine and supports all SQLite features.
- **KV Store**: a key–value (KV) database that runs on SQLite. It manages KV pairs and distributes data across multiple devices and applications.
- **Distributed Data Object**: an object-oriented in-memory data management framework that implements data object collaboration for the same application among multiple devices.
- **DDS**: synchronizes data between trusted devices, delivering a consistent access experience on different devices.
- **DSoftBus**: discovers and connects devices at the network link layer.
- **SQLite**: an open source component that provides native SQLite capabilities.

Containerized DSoftBus

Migrating legacy service software to containers can remove the barriers to modernization. In openEuler 22.03 LTS SP3, DSoftBus can be deployed as a container with its dependencies and multi-client support is enabled, to greatly simplify service installation, deployment, and testing and improve compatibility with service software.

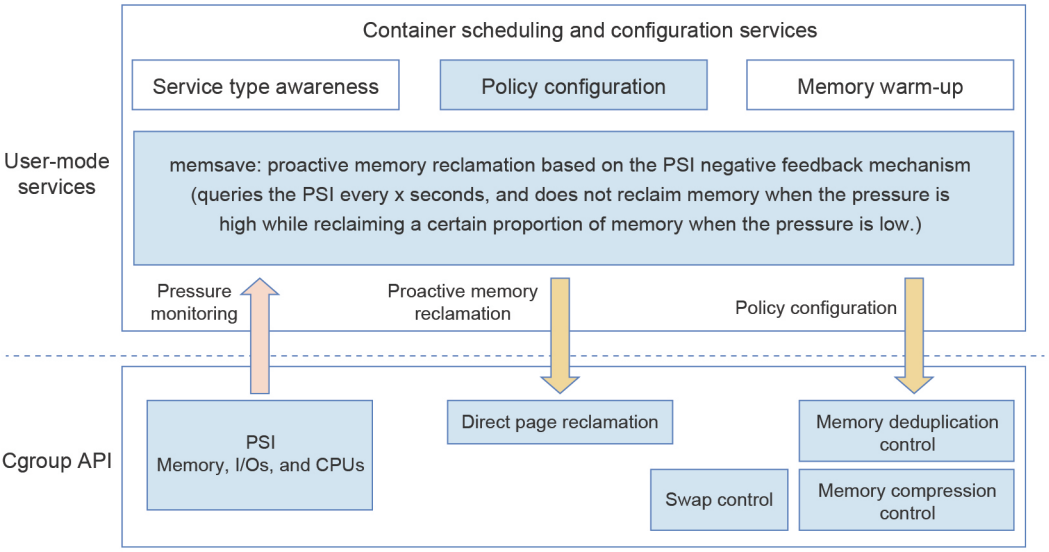
Application Scenarios

The openEuler distributed data management system enables data synchronization and cross-platform collaboration between openEuler Embedded devices, edge servers, and native OpenHarmony devices with less code.

Memory Overcommitment

Memory overcommitment is an efficient method to increase the available memory space for cloud native containers. This feature supports cgroup-specific memory deduplication, proactive memory reclamation, memory compression, swap usage control, and swap device control. In addition, kernel performance is improved for key modules like memory reclamation and compression, providing secondary compression, translation look aside buffer (TLB) batch refresh, and transparent huge page swap. This feature also provides memsave, a sample service program used for proactive reclamation in user mode. memsave periodically triggers container-level memory reclamation through the proactive reclamation API, and dynamically adapts the reclamation amount to the PSI mechanism. It limits the amount of memory used by containers and subsequent impact on container performance.

Feature Description



Cgroup memory policies

- **Proactive memory reclamation**: The type of reclaimed memory pages can be specified, for example, file pages and anonymous pages.
- **Watermark-based reclamation**: Minimum, low, and high watermarks can be configured for passive reclamation. Asynchronous reclamation can be performed in the background to avoid impact on existing services.
- **Memory deduplication**: All the memory space used by processes in a container can be included in KSM deduplication, without requiring applications to call the madvise API to mark memory areas beforehand.
- **Swap space**: For each independent container, you can configure the swap backend devices (such as zram and storage devices), maximum swap space, proactive swap-in, and enable or disable swap.

Basic optimizations

- **Memory compression**: Secondary compression with zram leverages multiple compression algorithms to increase the compression ratio and compression/decompression speed.
- **Memory reclamation**: TLB refresh is optimized in unmap and migration processes to accelerate memory reclamation and reduce lock conflicts. Transparent huge page swap is optimized as well.

Optimal decision-making based on the PSI mechanism

- PSI is available in cgroup v1 and v2.
- Memory is proactively reclaimed using the PSI negative feedback mechanism, to improve decisions that are based on service load and cluster information. This design maintains service performance and reliability during memory overcommitment.

Application Scenarios

Suited to cloud-native deployments that involve hybrid containers or serverless services, this feature boosts service quality at the same memory cost, that is, more services can be deployed or less memory space is required while ensuring low impact on performance.

Memory overcommitment adversely affects services in certain scenarios. In hybrid container deployments, you can reclaim the memory from offline tasks to prioritize memory of online, memory access–sensitive tasks. For online tasks, you can reduce the impact of overcommitment on system performance by configuring zram devices of memory compression as the swap backend, or even disable memory reclamation for the container. For offline tasks, you can enable proactive memory reclamation.

DIM

Dynamic Integrity Measurement (DIM) enables timely detection and troubleshooting measures to handle attacks. It measures key memory data like code segments during program running and compares the results with the reference values to determine data tampering in the memory.

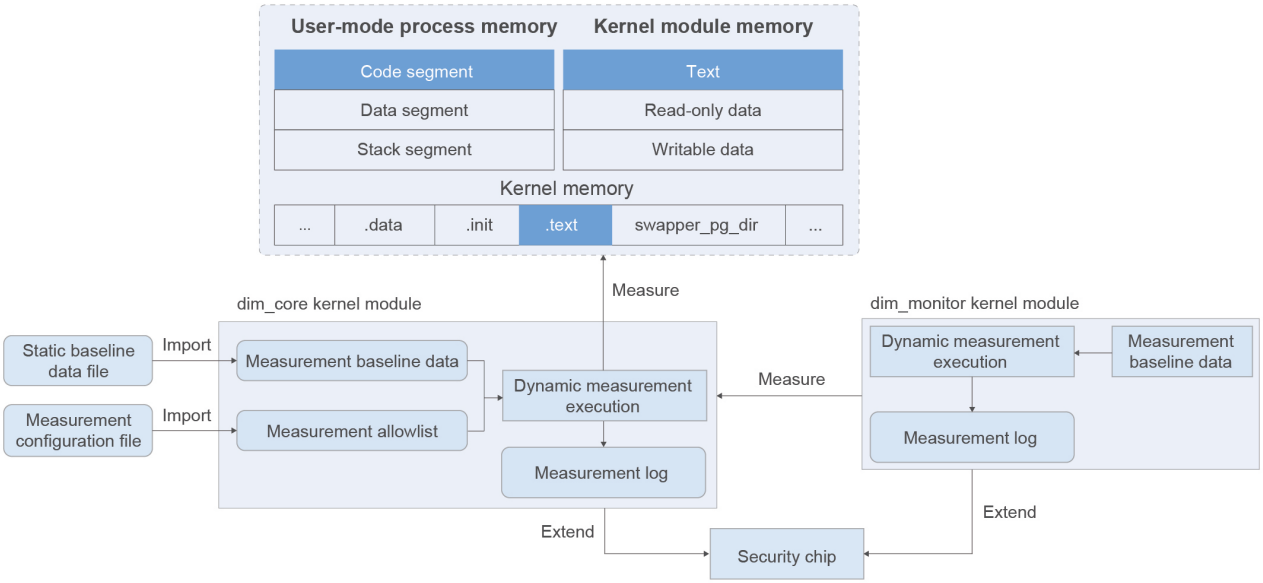
Feature Description

DIM provides the following features:

- Measures user-mode processes, kernel modules, and code segment in the kernel memory.
- Extends measurements to the PCR register of the TPM 2.0 chip for remote attestation.
- Configures measurements and verifies measurement signatures.
- Generates and imports measurement baseline data using tools, and verifies baseline data signatures.
- Supports SM3 algorithms.

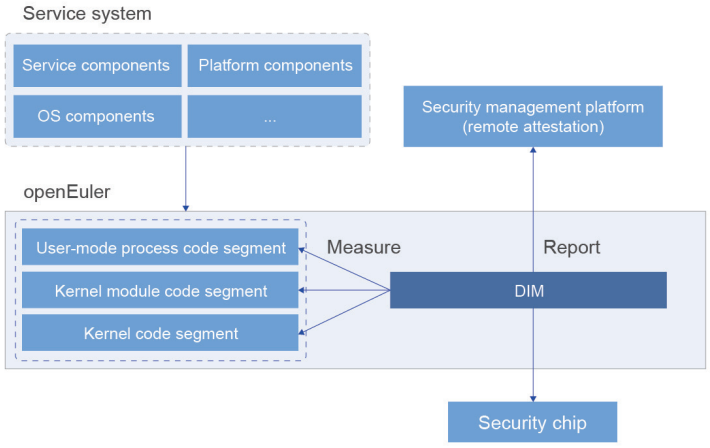
DIM consists of two software packages: dim_tools and dim.

- dim_tools: provides the dim_gen_baseline command-line tool, which generates code segment measurement baseline in a specified format by parsing the Executable and Linkable Format (ELF) binary file.
- dim: provides the dim_core and dim_monitor kernel modules. The former is the core module that parses and imports measurements and baselines configured by users, obtains target measurement data from memory, and performs measurement. The latter protects code segments and key data in dim_core to prevent invalid measurement due to dim_core tampering.



Application Scenarios

DIM works as a base security mechanism for the OS to protect memory data integrity for each system component. A typical application scenario is as follows:



DIM can be used to set dynamic measurement for key program data, after which it sends results to the security management platform, which can inform users of the security situation. For high security requirements, users can connect to the trusted computing mechanism for remote attestation, to confirm the integrity of measurement results using the Trusted Platform Module (TPM).

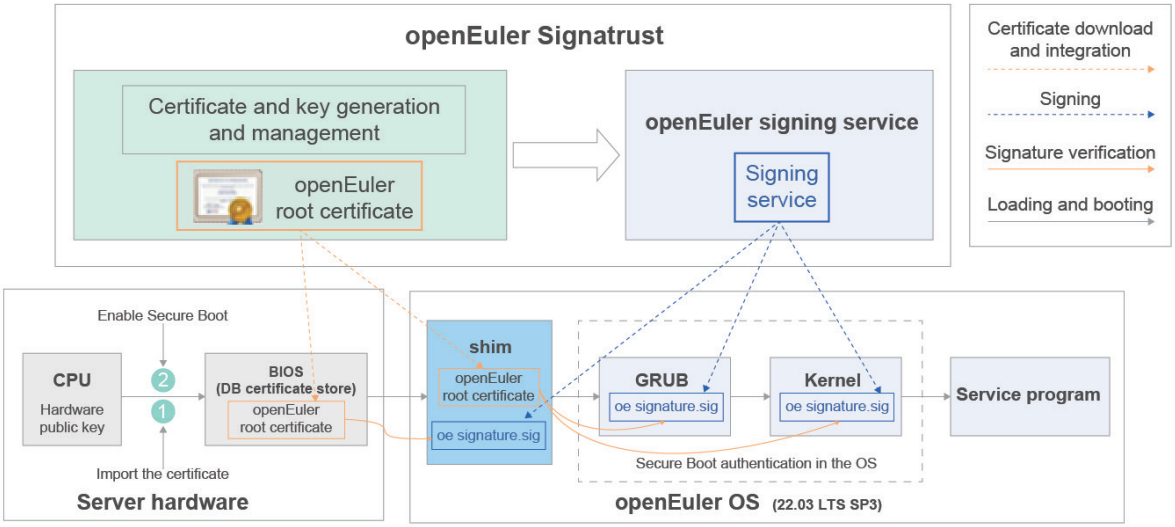
Secure Boot

Secure Boot relies on public and private key pairs to sign and verify components in the boot process. A typical boot process uses the previous component to verify the digital signature of the next component. If the verification is successful, the next component runs; if the verification fails, the boot stops. Secure Boot ensures the integrity of each component during system boot and prevents unverified components from being loaded and running, mitigating security threats to the system and user data.

In Secure Boot, the order of components to be verified are: BIOS, shim, GRUB, and vmlinuz (kernel image).

Feature Description

The openEuler Signatrust platform provides the signing service to simplify private key signature management, solving the signature protection problem of the community Secure Boot. This platform implements unified management of keys for Secure Boot, and enhances system integrity protection.



1. The Signatrust platform generates and manages public and private key pairs and certificates, and provides the signing service for EulerMaker to build openEuler software packages.
2. The Signatrust platform signs code of the EFI components (shim, GRUB, vmlinux) for Secure Boot when the software packages are built by EulerMaker.
3. Signature verification is performed during system boot to ensure system components are safe and secure.

Constraints

- The Signatrust platform can only sign components built in the openEuler community, but cannot sign files developed by external projects or custom user files.
- The Signatrust platform supports only the RSA algorithm.

Application Scenarios

openEuler 22.03 LTS SP3 supports Secure Boot (disabled by default). You can enable Secure Boot as required by activating the Secure Boot function and importing the root certificate. The procedure is as follows:

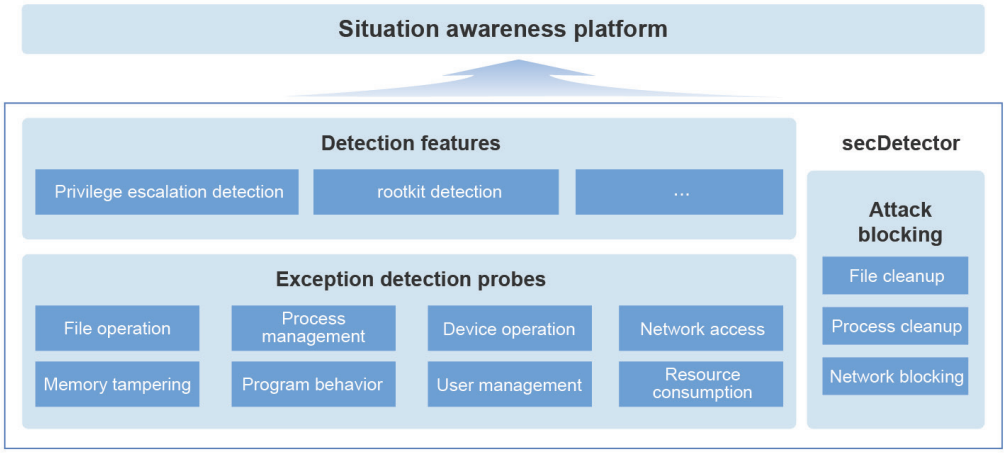
1. Obtain the root certificate named **openEuler Shim Default CA** from the openEuler certificate center at https://www.openeuler.org/en/security/security-bulletins/.
2. After the device is started, enter BIOS setup to import the openEuler root certificate to the DB certificate store and enable Secure Boot. Then, save the settings and restart the device.

Note: The BIOS operations vary according to the BIOS version. For details, see the documents provided by the BIOS vendor.

secDetector

secDetector is an intrusion detection system designed for OSs. It provides intrusion detection and response for critical infrastructure and reduces development costs while enhancing detection and response for third-party security tools. Based on ATT&CK attack patterns, secDetector provides real-time blocking and flexible responses in addition to a high volume of security primitives.

Feature Description



secDetector consists of the detection feature cases, exception detection probes, and attack blocking module. The exception detection probes collect OS attack events that match the MITRE ATT&CK patterns. There are eight types of exception detection probes that can detect advanced persistent threats (APTs): file operation, process management, network access, program behavior, memory tampering, resource consumption, account management, and device operation. The attack blocking module provides real-time blocking measures during information collection and detection based on the preset security rules, including clearing malicious objects (processes, files, networks, modules, and scripts) and blocking actions (process privilege escalation and file modification). Based on the capabilities of exception detection probes and the attack blocking module, advantageous attack detection features are provided, such as process privilege escalation and kernel rootkit detection.

The detection feature cases, exception detection probes, and attack blocking module can be extended. Users can add or enhance the detection and response capabilities based on application scenarios. The extensibility is attributed to the technical implementation architecture of secDetector, which consists of the SDK, service, detection feature cases, and detection framework (core).

- The secDetector SDK is provided as a user-mode dynamic link library (DLL) deployed in the security awareness services that require secDetector. The SDK communicates with the secDetector service to complete related operations (such as subscription, unsubscription, and message reading). The exception information provided by secDetector is defined as different cases. The security awareness services can subscribe to the cases as required.
- The secDetector service is a user-mode service application. It manages and maintains the subscriptions of the security awareness services and maintains the probe running statuses. On a unified environment, it gathers and forwards the information collected by the detection framework (core) and detection feature cases to different security awareness services, and manages and forwards the configurations and management requirements of the security awareness services on the detection feature cases and detection framework (core). Because multiple security awareness services may require the same case, the secDetector service finds and registers the intersection of cases of all security awareness services.

- The detection feature cases correspond to a series of exception detection probes, which are in different forms. For example, each probe for detecting kernel exceptions is available in a kernel module (.ko file). A case represents a probe, which usually covers a type of exceptions or exception events. For example, the process probes are for creation, exit, and property modification events of all processes, whereas the memory modification probes collect information such as the kernel module list and security switches. A probe may monitor multiple events, but the monitoring logic may not be deployed in the same execution flow. Workflows are introduced to represent the scope of a probe in the same execution flow, whereby a probe contains one or more workflows. For example, the process probe manages the creation and property modification in different workflows. A workflow consists of four types of functional units: event generator, information collector, event analyzer, and response unit.
- The detection framework (core) is the base framework for case management, and provides common functional units required by workflows. The kernel exception detection framework is carried by a kernel module (.ko file), in which a detection feature case can register itself with or deregister itself from the detection framework (core). The detection framework (core) also provides specific interaction APIs to handle external dynamic requests.

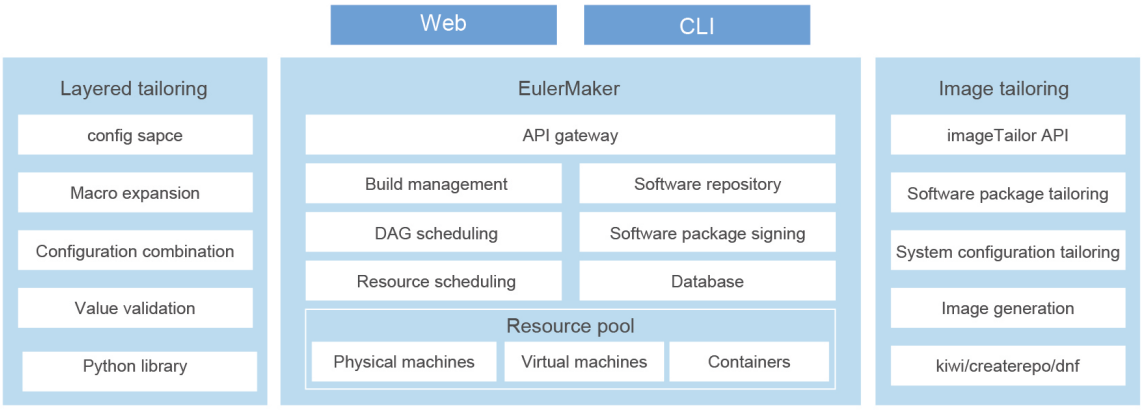
Application Scenarios

As a flexible OS intrusion detection system, secDetector can be used in three modes:

- Enabled by system users to generate alarms for and handle abnormal events.
- Integrated with security awareness services to collect system information for analysis of complex security threats (such as APTs) and real-time blocking of critical events.
- Used to build accurate, efficient, and timely intrusion detection and response capabilities based on the extensible framework under secondary development of security practitioners or security awareness service providers.

EulerMaker

EulerMaker is a package build system that converts source code into binary packages. It enables developers to assemble and tailor scenario-specific OSs thanks to incremental/full build, gated build, layer tailoring, and image tailoring capabilities.



Feature Description

- **Incremental/Full build**: Analyzes the impact of the changes to software and dependencies, obtains the list of packages to be built, and delivers parallel build tasks based on the dependency sequence.
- **Build dependency query**: Provides a software package build dependency table in a project, and collects statistics on software package dependencies.
- **Layered tailoring**: Overlays configuration layer models based on SPEC or YAML to tailor the software package version, patches, build and installation dependencies, compilation options, and build process to your project.
- **Image tailoring**: Developers can configure the repository source to generate ISO, QCOW2, and container OS images, and tailor the list of software packages for the images.
- **Local task reproduction**: Reproduces a build task locally using commands, facilitating build problem locating.
- **Easy project creation**: Creates projects based on YAML configurations, and packages can be added in batches, greatly simplifying user operations.

Application Scenarios

Community developers and partners build core OS repositories and OSs tailored to their own needs.

DPUDirect

DPUDirect creates a collaborative operating environment for services, enabling them to be easily offloaded and ported between hosts and data processing units (DPUs). The feature includes a process-level seamless offload function and a cross-host and -DPU collaboration framework. This allows management-plane processes to be split and offloaded to the DPU without requiring reconstruction. Once offloaded, these processes can continue managing processes on the host side. The DPUDirect feature significantly reduces service offload costs in DPU scenarios, simplifies O&M, and significantly reduces subsequent maintenance costs.

Challenges

Intelligent NICs are gradually evolving into DPUs/IPUs, becoming an increasingly important part of cloud and data center infrastructure. In addition to accelerating I/Os on the data plane, DPUs/IPUs are also supporting the offload of management- and control-plane components. This means that all management and control components of the data center infrastructure can be offloaded to the DPUs, resulting in better architecture and more flexible deployments. Most mainstream offload solutions involve splitting components. However, this approach has several drawbacks.

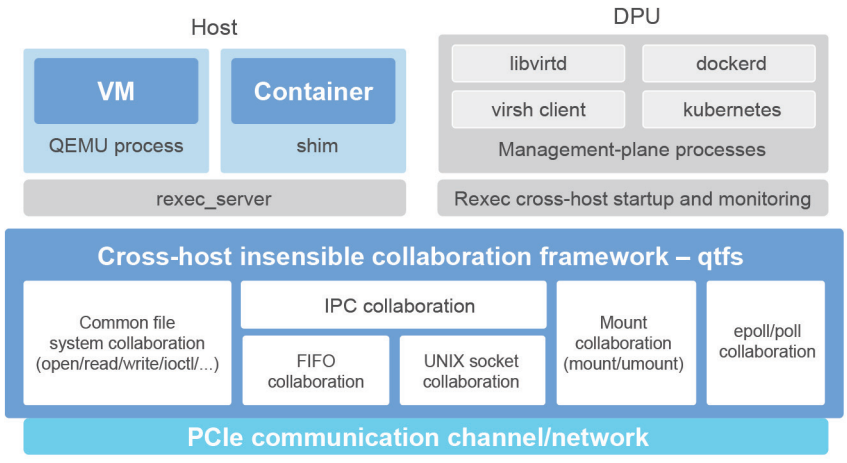
- Developers must understand the code of the components being offloaded.
- Cloud vendors maintain a large number of components, making offload a burden.
- The code of offloaded components is difficult to inherit between upgrades and must adapt to the newest version, which is complicated and expensive.

Project Introduction

DPUDirect builds a cross-host collaboration framework at the OS layer of the host and DPU, providing a consistent runtime view for the management-plane processes offloaded to the DPU and the service processes on the host. In this way, applications are unaware of offload. Only a small amount of service code on the management plane needs to be adapted to ensure software compatibility and evolution, as well as reducing component maintenance costs.

Feature Description

The figure illustrates the DPUDirect architecture. qtfs, a cross-host collaboration framework is established at the host and DPU OS layers to provide a consistent runtime view for the management-plane processes offloaded to the DPU and the service processes on the host. This approach ensures that applications remain unaware of the offload process.



DPUDirect allows you to combine the following collaboration mechanisms to achieve seamless offload in various scenarios.

- File system collaboration supports cross-host file system access and provides a consistent file system view for host and DPU processes. It also supports special file systems such as proc, sys, and dev.
- IPC collaboration enables imperceptible communication between host and DPU processes. It supports FIFO and UNIX domain sockets for cross-host communication.
- Mounting collaboration performs the mount operation in a specific directory on the host, which can adapt to the container image overlay scenario. The offloaded management-plane process can construct a working directory for the service process on the host, providing a unified cross-node file system view.
- epoll collaboration supports epoll operations for cross-host access of remote common files and FIFO files, and supports read and write blocking operations.
- Process collaboration uses the rexec tool to remotely start executable files. The rexec tool takes over the input and output streams of the remotely started processes and monitor the status to ensure the lifecycle consistency of the processes at both ends.

By combining these mechanisms, policies can be tailored for different scenarios to fulfill the service requirements of the management-plane processes, eliminating the need to split and reconstruct too many services.

Application Scenarios

DPUDirect facilitates the complete offload of the container management plane such as kubelet and dockerd, as well as the virtualization management plane libvirtd. It eliminates the need of splitting over 10,000 lines of code, thereby reducing the workload of adapting and maintaining to almost 1/20 of the original. Furthermore, the service logic of the management plane remains unaltered, ensuring service software compatibility and evolution.

Repository

https://gitee.com/openeuler/dpu-utilities/blob/master/README.en.md

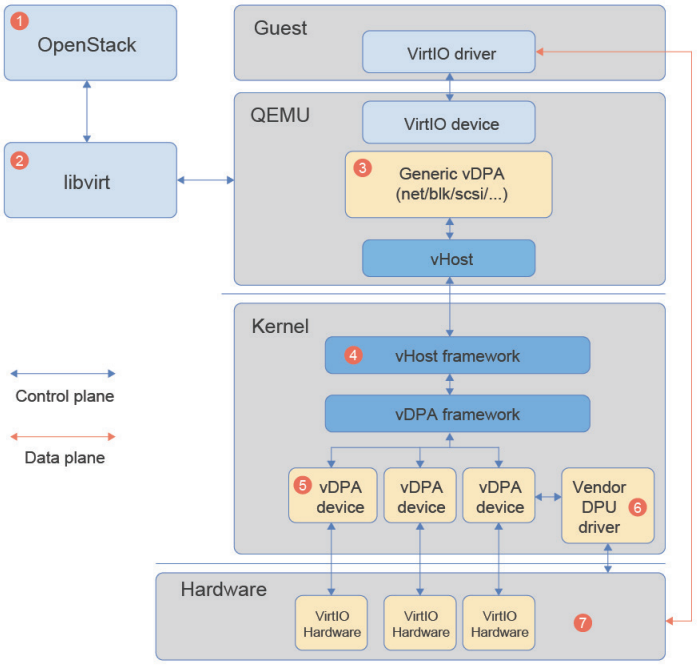
Live VM Migration with vDPA NIC Passthrough

The kernel-mode vHost Data Path Acceleration (vDPA) framework provides a device virtualization solution that performs equivalently to passthrough. The vDPA framework unifies the architecture for diverse hardware forms, such as intelligent NICs and DPUs, and supports live migration across different hardware vendors.

Extended vDPA and vHost APIs are used for live migrating VMs across vDPA devices from the same vendor, addressing the basic live migration requirements of vDPA passthrough VMs. Further, cross-vendor live migration uses embedded code to meet future requirements.

Feature Description

The following figure shows the overall architecture of kernel-mode vDPA. This feature enables libvirt, QEMU, and kernel modules to support the generic vDPA framework and the live migration capability.



No.	Component	Description
1	OpenStack	Manages devices that support generic vDPA.
2	libvirt	Supports vDPA device resource query and device lifecycle management.
3	Generic vDPA	Implements the generic vDPA device model and supports multiple VirtIO device types.
4	vHost-vDPA	Connects to the vHost subsystem and streamlines the management path of vDPA devices.
5	vDPA device	Connects to the vDPA framework to live migrate VirtIO devices and manage their lifecycles.
6	Vendor DPU driver	Connects to the vDPA framework.
7	VirtIO hardware	Hardware that supports the VirtIO protocol.

The vDPA API in the kernel modules supports generic vDPA. To support live migration of vDPA devices, the APIs of the open source vDPA framework are extended as follows:

API for marking dirty pages

Kernel-mode vDPA live migration depends on hardware capabilities to mark dirty pages.

Two optimizations are added to this API: (1) The capability to mark dirty pages can be enabled or disabled after VirtIO negotiation completes. (2) Dirty page addresses and sizes can be set and dirty pages can be synchronized.

The process of marking dirty pages is as follows:

1. The libvirt and QEMU components on the source hardware device initiate a live VM migration.

2. The QEMU component on the source hardware device uses the new ioctl interface function to set the addresses and sizes of the marked dirty pages.

3. The QEMU component on the source hardware device enables the function of marking dirty pages.

4. The QEMU component on the source hardware device obtains the dirty pages through the dirty page synchronization API.

5. The QEMU component on the source hardware device merges and iteratively copies the dirty pages to the QEMU component on the destination hardware device.

Device status API

Live migrating a VM sends both memory and status information to ensure consistency between the source and destination devices. A new API is used to save and restore the vDPA device status.

The overall process of saving and restoring the vDPA device status is as follows:

1. The QEMU component on the source hardware device completes migrating dirty pages and stops the VM.

2. The QEMU component on the source hardware device calls the vDPA device status saving API to query the vDPA device status.

3. The QEMU component on the source hardware device sends the vDPA device status to the QEMU component on the destination hardware device.

4. The QEMU component on the destination hardware device calls the vDPA device status restoration API to set the vDPA device status.

5. The QEMU component on the destination hardware device restores the VM.

Migration status API

Different hardware vendors may require operations such as hardware resource reclamation for certain live migration states. For this requirement, a new API is added for QEMU to notify vDPA hardware of different live migration states.

Application Scenarios

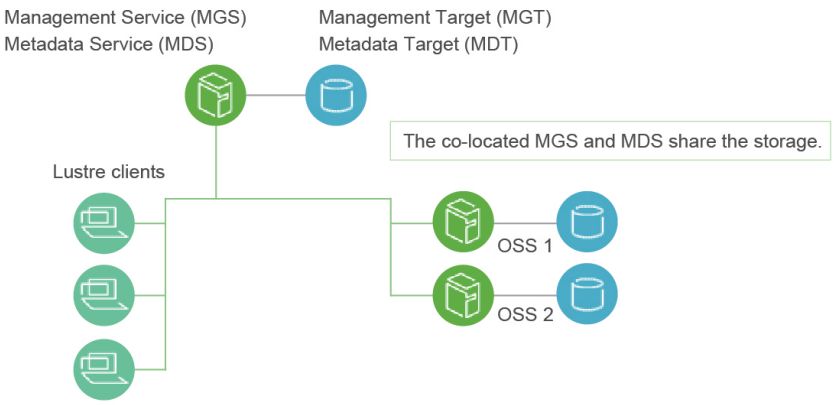
Kernel-mode vDPA is positioned for deployments of intelligent NICs and DPUs that support the VirtIO protocol. This solution provides I/O performance equivalent to non-virtualized hardware, and supports live VM migration across vDPA devices from the same vendor.

In terms of DPUs, one framework supports all VirtIO devices, such as VirtIO-net and VirtIO-scsi, reducing the development workload of OpenStack, libvirt, and QEMU device models while ensuring solution compatibility and evolution.

Lustre Server Software Package

Lustre is an open source parallel file system designed for high scalability, performance, and availability. Lustre runs on Linux and provides POSIX-compliant UNIX file system interfaces.

A Lustre cluster contains a Management Service (MGS) and at least one Lustre file system instance, connected over the Lustre Network (LNet).



- MGS: The MGS stores and gives access to the file system configurations to Lustre file systems and Lustre clients.
- Management Target (MGT): MGTs, each corresponding to a drive, are entities that store configuration information. Generally, the MGT and Metadata Target (MDT) share the same drive.
- LNet: LNet is the network interface abstraction of Lustre, providing the infrastructure for communication. Each network protocol, such as TCP and InfiniBand, needs to implement an LNet driver.

Lustre file system components

A Lustre file system comprises the following components:

- Metadata Service (MDS): The MDS manages file system metadata (such as file names, directories, permissions, and file layouts) stored in the MDTs for Lustre client access.
- MDT: The MDT is used to store cloud data. Generally, each MDT corresponds to a drive.
- Object Storage Service (OSS): Files of users are split and stored in one or more Object Storage Targets (OSTs). The OSS manages these OSTs.
- OST: The OST stores file data objects. Each OST corresponds to a drive.
- Lustre client: A client that mounts and uses the Lustre file systems. Generally, a Lustre client is a compute node.

Feature Description

High scalability and performance

A Lustre system is scalable in terms of the number of client nodes, drive storage capacity, and bandwidth. The scalability and performance depend on the available drives, network bandwidth, and server throughput. The following lists the main features.

- **Client scalability**: Up to 100,000 clients are supported. A typical production environment usually has 10,000 to 20,000 clients.
- **Client performance**: The I/O performance of a single client is 90% of the network bandwidth. The aggregated I/O performance reaches 50 million IOPS, with an I/O bandwidth of up to 50 TB/s.
- **OSS scalability**: A single OSS can manage up to 32 OSTs, each capable of storing 500 million objects, or 1,024 TB. A maximum of 1,000 OSSs and 4,000 OSTs are supported in a Lustre system.
- **OSS performance**: A single OSS can deliver 1.5 million IOPS, with an I/O bandwidth of 15 GB/s. The aggregated I/O performance reaches 50 million IOPS, with an I/O bandwidth of up to 50 TB/s.
- **MDS scalability**: A single MDS can manage up to four MDTs. A single MDT supports 4 billion files of up to 16 TB when LDISKFS is used as the backend file system, or 64 billion files of up to 64 TB when ZFS is used as the backend file system.
- **MDS performance**: 1 million creation operations or 2 million metadata **stat** operations can be performed within a second.
- **File system scalability**: The maximum size of a single file in the LDISKFS backend is 32 PB. An aggregated Lustre system can contain up to 1 trillion files, or 512 PB.

Other features

- **Performance-enhanced ext4 file system**: LDISKFS is an improved version of ext4 built for the Lustre backend file system, and provides optimized distributed file performance. Alternatively, ZFS is available to ensure better scalability and data integrity.
- **POSIX compliance**: Lustre passes the full POSIX test suite on the clients with few exceptions, just like a local ext4 file system, and also supports **mmap()** file I/O operations.
- **High-performance heterogeneous networking**: Lustre supports various high-performance and low-latency networks through remote direct memory access (RDMA), including InfiniBand and Intel OmniPath.
- **High availability**: Lustre supports active/active failover. It can work with various high-availability managers to provide automated failover and eliminate single points of failure.
- **Security**: By default, Lustre only allows connections from privileged ports. UNIX group membership is verified on the MDS. File encryption is also supported.
- **Access control list (ACL) and extended attributes**: Lustre follows the UNIX file system security model and is enhanced with POSIX ACLs. Additional access control, such as root squash, is also supported.
- **Interoperability**: The Lustre file system runs on a variety of CPU architectures and mixed-endian clusters and is interoperable between successive Lustre releases.
- **Object-based architecture**: Clients are isolated from the file structure on drives. This allows the storage architecture to be upgraded without affecting the clients.
- **Quota**: User and group quotas can be configured.
- **Capacity growth**: By adding OSTs and MDTs, the size of a Lustre file system and aggregate bandwidth can be increased without interrupting services.
- **Controlled file layout**: The layout of files across OSTs can be configured on a per-file, -directory, or -file system basis, to better tune file I/Os to specific application requirements within a single file system. Lustre uses RAID 0 striping and balances space usage across OSTs.
- **Network data integrity protection**: All data sent from the client to the OSS has verification values, preventing data corruption during transmission.
- **MPI I/O**: Lustre has a dedicated MPI ADIO layer that matches parallel I/Os to the underlying file system architecture.
- **NFS and CIFS export**: Files in the Lustre file system can be re-exported using NFS (Linux knfsd or Ganesha) or CIFS (Samba), enabling sharing with non-Linux clients such as Windows and OS X.
- **Disaster recovery tool**: LFSCK is an online distributed file system check tool that restores consistency between storage

components in case of a major file system error. The Lustre file system can operate even if there are inconsistencies.

- **Performance monitoring**: The Lustre file system offers a variety of mechanisms to monitor and tune performance.

Application Scenarios

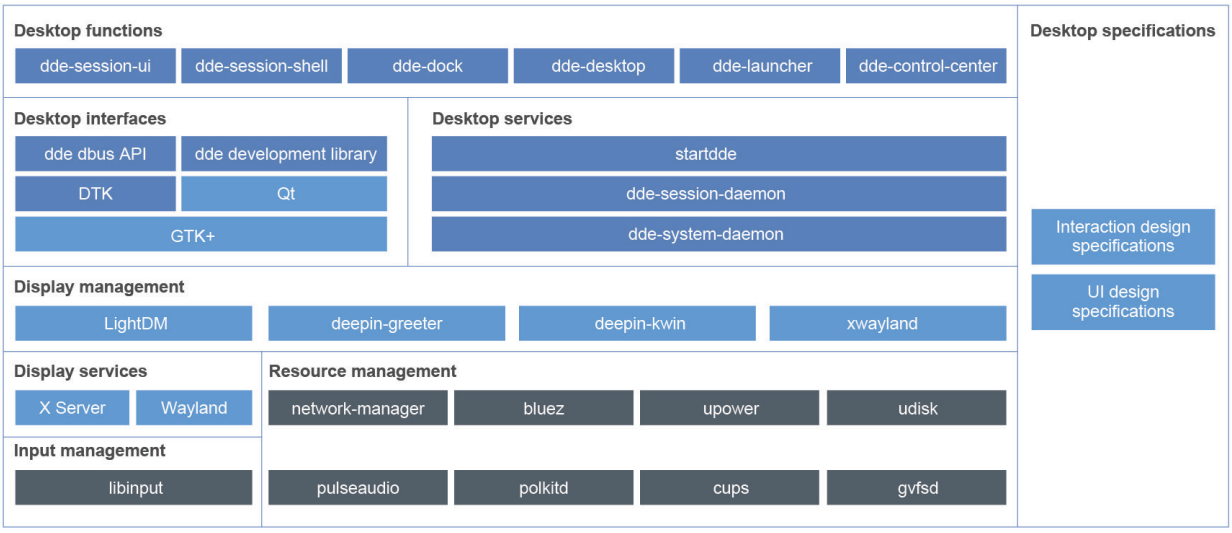
The Lustre parallel file system is designed for systems that require massive and high-performance file storage. A survey shows that Lustre is widely used in HPC and AI computing scenarios in industries such as academic research, scientific computing, media, manufacturing, finance, and education.

DDE

Deepin Desktop Environment (DDE) was originally developed for Uniontech OS and has been used in the desktop, server, and dedicated device versions of Uniontech OS.

Feature Description

DDE focuses on delivering high quality user interactions and visual design. DDE is powered by independently developed core technologies for desktop environments and provides login, screen locking, desktop, file manager, launcher, dock, window manager, control center, and additional functions. Due to its user-friendly interface, excellent interactivity, high reliability, and strong privacy protection, it is one of the most popular desktop environments among users. DDE helps you boost your creativity and efficiency at work, keep in touch with friends, effortlessly browse the Internet, and enjoy music and videos.



DDE adopts the DTK framework with unified UI elements and excellent UX design, and integrated third-party graphics libraries such as Qt and GTK+. Display services, input management, and resource management are at the bottom layer and are generally backend services written in Go. They provide interfaces required by upper-layer GUI programs to implement desktop functions such as user creation, screen brightness setting, device volume setting, and network connection management. Display management, desktop interfaces, and desktop services are at the shell layer and communicate with backend services through the D-Bus protocol. They provide support for UI definition and interactions, such as the login screen, window appearance, and GUI application controls.

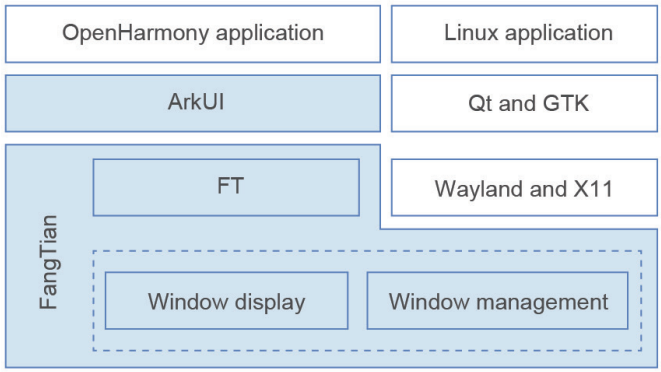
Application Scenarios

Desktop functions are at the application layer and are generally functional interfaces that can be operated by users, such as the launcher and dock.

FangTian Window Engine

The FangTian window engine delivers fundamental display technologies to build a foundation for openEuler's desktop environments. FangTian hosts display services such as window management, graphic drawing and compositing, and screen delivery.

Feature Description



- **Window display** provides capabilities such as buffer allocation and swapping, vertical synchronization, rendering, compositing, and screen display. The data-driven interfaces and unified architecture realize high performance and low memory usage.
- **Window management** creates, moves, zooms, arranges, and destroys windows. An independent window policy module is used to support various scenarios on multiple device types, such as mobile phones and PCs.
- **FT** is a display protocol that enables the ArkUI framework to interact with FangTian. It provides unified rendering and data-driven interfaces to lower rendering load, reduce data from cross-process interactions, and enhance application animation performance.
- **ArkUI** is a declarative UI development framework for OpenHarmony applications. It is derived from OpenHarmony and has been adapted to openEuler, allowing ArkUI-based OpenHarmony applications to run on openEuler as well.

Highlights

- **Linux application support**: Native Wayland and OpenHarmony applications can run simultaneously.
- **High-performance display of OpenHarmony applications**: 50 application windows can be displayed at 60 FPS.

Constraints

- Only x86_64 applications are supported. The functions of some ArkUI controls are not enabled.
- Wayland protocol compatibility does not apply to protocol extensions.

Application Scenarios

The FangTian window engine builds a high-performance foundation for openEuler's desktop environments, allowing various application ecosystems to be integrated into openEuler.

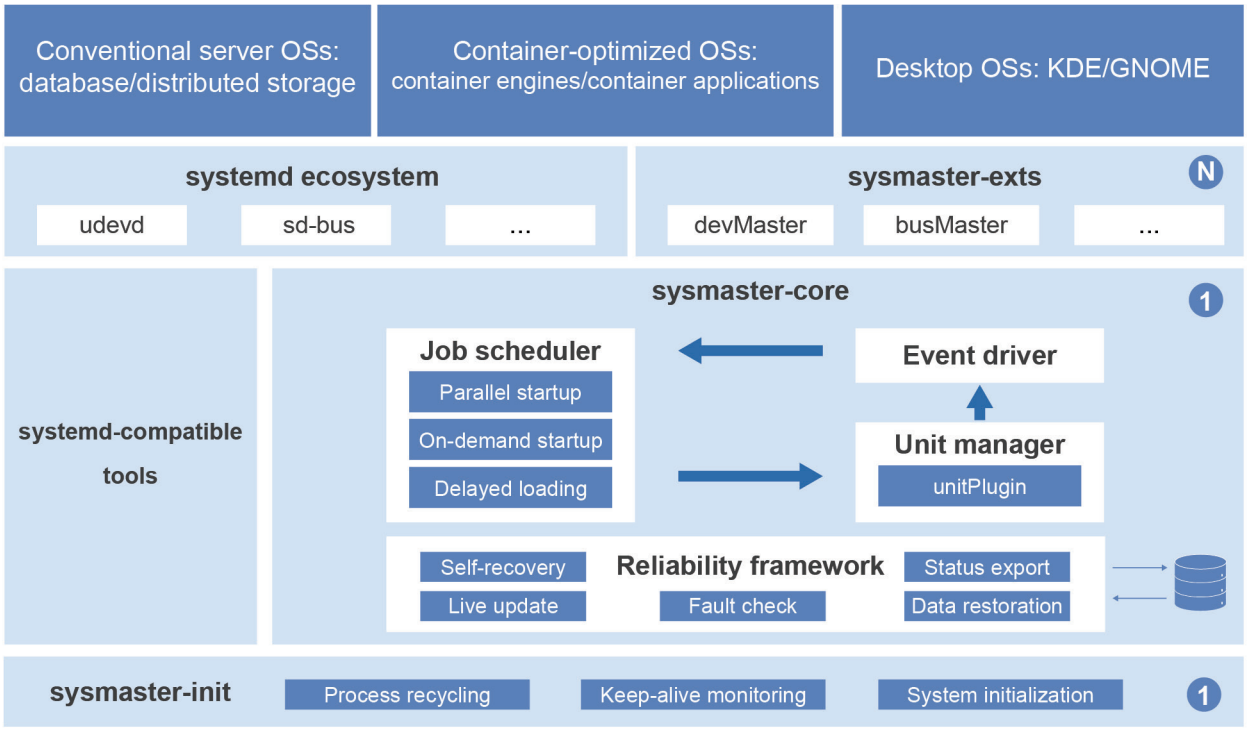
sysMaster

sysMaster is a collection of ultra-lightweight and highly reliable service management programs. It provides an innovative implementation of PID 1 to replace the conventional init process. Written in Rust, sysMaster is equipped with fault monitoring, second-level self-recovery, and quick startup capabilities, which help improve OS reliability and service availability. sysMaster manages processes, containers, and VMs centrally and provides fault monitoring and self-healing mechanisms to help deal with Linux initialization and service management challenges. All these features make sysMaster an excellent choice for server, cloud computing, and embedded scenarios.

Feature Description

sysMaster divides the functions of traditional PID 1 into a 1+1+*N* architecture based on application scenarios. As shown in the figure, sysMaster consists of three components:

- **sysmaster-init**, which is a new implementation of PID 1, features simplified functions, a thousand lines of code (KLOC), and ultimate reliability. It is applicable to embedded systems with functions such as system initialization, zombie process recycling, and keep-alive monitoring.
- **sysmaster-core** undertakes the core service management functions and incorporates the reliability framework to enable live updates and quick self-recovery in the event of crashes, ensuring 24/7 service availability.
- **sysmaster-exts** offers a collection of components (such as devMaster for device management) that deliver key system functions, which are coupled in traditional PID 1. You can choose the components to use as required.



Featuring a simple component architecture, sysMaster improves the scalability and adaptability of the overall system architecture while reducing development and maintenance costs. sysMaster provides the following advantages:

Featuring a simple component architecture, sysMaster improves the scalability and adaptability of the overall system architecture while reducing development and maintenance costs. sysMaster provides the following advantages:

- Service management, device management, live updates, and self-recovery in seconds in the event of crashes
- Faster startup speed with lower memory overhead
- Migration tools that provide seamless migration from systemd to sysMaster
- Unified interfaces that work with the iSulad container engine and QEMU for management of container and virtualization instances

sysMaster 0.5.1 released with openEuler 22.03 LTS SP3 supports system service management in the container and VM scenarios.

New features:

- devMaster component to manage device hot swap.
- Live updates and hot reboot operations.
- VMs now support PID 1.

Constraints:

- Only available for 64-bit OSs.
- sysMaster configuration files must be in TOML format.
- sysMaster can run only in system containers and VMs.

In the future, sysMaster will extend to more scenarios and have its architecture and performance further optimized for higher scalability and adaptability. In addition, new features and components will be developed to meet the requirements of container, virtualization, and edge computing scenarios. These features will make sysMaster a powerful, efficient, and user-friendly system management framework.

Application Scenarios

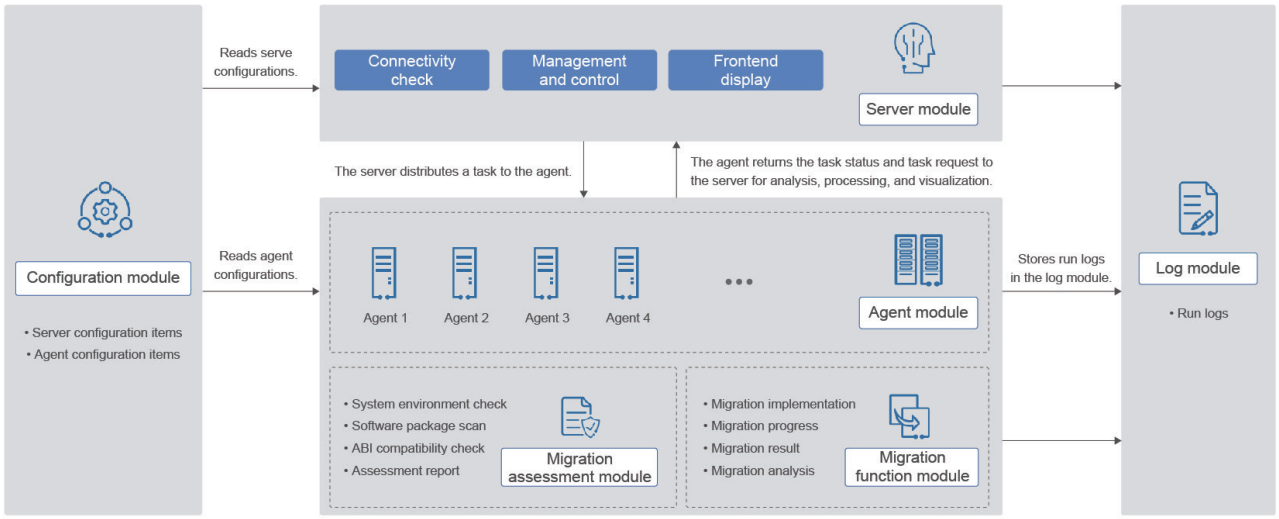
sysMaster implements PID 1 in containers, VMs, servers, and edge devices.

Code repository: https://gitee.com/openeuler/sysmaster

migration-tools

migration-tools, developed by UnionTech Software Technology Co., Ltd., is positioned to meet demand for smooth, stable, and secure migration to the openEuler OS.

Feature Description



migration-tools comprises modules for the server, agent, configurations, logs, migration assessment, and migration function.

- **Server module**: the core of migration-tools. This module is developed on the Python Flask Web framework. It receives task requests, processes execution instructions, and distributes the instructions to each Agent.
- **Agent module**: installed in the OS to be migrated to receive task requests from the Server module and perform migration.
- **Configuration module**: reads configuration files for the Server and Agent modules.
- **Log module**: records logs during migration.
- **Migration assessment module**: provides assessment reports such as basic environment check, software package comparison and analysis, and pre-migration compatibility checks.
- **Migration function module**: provides quick migration, displays the migration progress, and checks the migration result.

Application Scenarios

Finance, telecom, energy, and other industries often need to replace the OSs on existing hardware, such as the x86_64 architecture. migration-tools is a good choice for migrating applications and system components from other OSs to openEuler.

utshell

utshell is a new shell that introduces new features and inherits the usability of Bash. It enables interaction through command lines, such as responding to user operations to execute commands and providing feedback, and can execute automated scripts to facilitate O&M.

Feature Description

utshell provides the following features:

- **Command execution**: Runs and sends return values from commands executed on user machines.
- **Batch processing**: Automates task execution using scripts.
- **Job control**: Executes, manages, and controls multiple user commands as background jobs.
- **Historical records**: Records the commands entered by users.
- **Command aliases**: Allows users to create aliases for commands to customize their operations.

Application Scenarios

utshell is well suited to conventional server, cloud-native environments, and attended or unattended production sites where automated O&M scripts are executed, as well as the demands of individual users.

utsudo

sudo is one of the commonly used utilities for Unix-like and Linux OSs. It enables users to run specific commands with the privileges of the super user. utsudo is developed to address issues of security and reliability common in sudo. utsudo uses Rust to deliver more efficient, secure, and flexible privilege escalation. The tool uses modules such as common utility, overall framework, and function plugins.

Feature Description

- **Access control**: Limits the commands that can be executed by users, and specifies the required authentication method.
- **Audit log**: Records and traces all commands and tasks executed by each user.
- **Temporary privilege escalation**: Allows common users to temporarily escalate to a super user for executing privileged commands or tasks.
- **Flexible configuration**: Allows users to set arguments such as command aliases, environment variables, and execution parameters to meet system requirements.

Application Scenarios

utsudo reduces security risks by enabling administrators to better manage user privileges and authorization, and prevent unauthorized privileged operations. It is suitable for management and maintenance, user permission control, and multiuser environments.

i3

i3 is a tiling window manager that enables the keyboard to manage the window layouts in a session or across multiple monitors. Now, you can use i3 in openEuler 22.03 LTS SP3.

Feature Description

You can press the following shortcut keys for some basic i3 operations. (**Mod** is usually mapped to the Windows key on a Windows-compatible keyboard.)

- **Mod+d**: Open dmenu for quickly starting processes.
- **Mod+Enter**: Open a terminal.
- **Mod+** ↑ / ↓ / ← / → : Move focus between windows.
- **Mod+Shift+q**: Close the window of focus.
- **Mod+Shift+r**: Hot load the configuration file.
- **Mod+Shift+e**: Exit i3.
- **Mod+Shift+l**: Lock the screen.

For more details, see the upstream document.

Trusted Platform Control Module

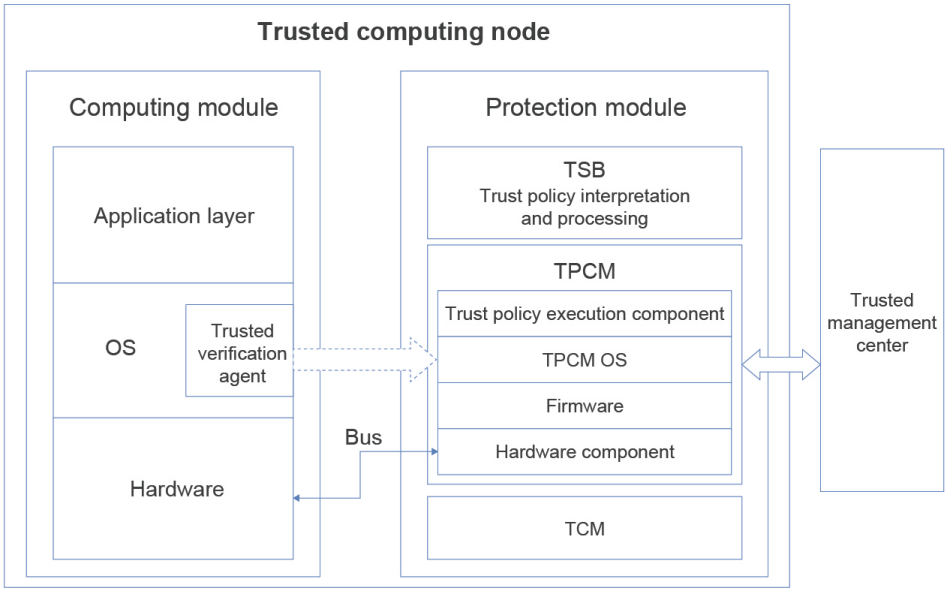
Trusted computing has undergone continuous development and improvement in the past 40 years and has become an important branch of information security. Trusted computing technologies have developed rapidly in recent years and have solved the challenges in Trusted Computing 2.0—integration of trusted systems and existing systems, trusted management, and simplification of trusted application development. These technical breakthroughs form Trusted Computing 3.0, that is, trusted computing based on an active immune system. Compared with the passive plug-in architecture of the previous generation, Trusted Computing 3.0 proposes a new trusted system framework based on self-controlled cryptography algorithms, control chips, trusted software, trusted connections, policy management, and secure and trusted protection applications, implementing trust across the networks.

The trusted platform control module (TPCM) is a base and core module that can be integrated into a trusted computing platform to establish and ensure a trust source. As one of the innovations in Trusted Computing 3.0 and the core of active immunity, TPCM implements active control over the entire platform.

The TPCM-based Trusted Computing 3.0 architecture consists of the protection module and the computing module. On the one hand, based on the Trusted Cryptography Module (TPM), the TPCM main control firmware measures the reliability of the protection and computing modules, as well as their firmware. On the other hand, the Trusted Software Base (TSB) measures the reliability of system software and application software. In addition, the TPCM management platform verifies the reliability measurement and synchronizes and manages the trust policies.

Feature Description

The overall system design consists of the protection module, computing module, and trusted management center software, as shown in the following figure.



- **Trusted management center**: This centralized management platform, provided by a third-party vendor, formulates, delivers, maintains, and stores protection policies and reference values for trusted computing nodes.
- **Protection module**: This module operates independently of the computing module and provides trusted computing protection functions that feature active measurement and active control to implement security protection during

computing. The protection module consists of the TPCM main control firmware, TCB, and TCM. As a key module for implementing trust protection in a trusted computing node, the TPCM can be implemented in multiple forms, such as cards, chips, and IP cores. It contains a CPU and memory, firmware, and software such as an OS and trusted function components. The TPCM operates alongside the computing module and works according to the built-in protection policy to monitor the trust of protected resources, such as hardware, firmware, and software of the computing module. The TPCM is the Root of Trust in a trusted computing node. The TPCM interacts with other components as follows:

1. The TPCM hardware, firmware, and software provide an operating environment for the TSB. The trusted function components of the TPCM provide support for the TSB to implement measurement, control, support, and decision-making based on the policy library interpretation requirements.
2. The TPCM accesses the TCM for trusted cryptography functions to complete computing tasks such as trusted verification, measurement, and confidential storage, and provides services for TCM access.
3. The TPCM connects to the trusted management center through the management interface to implement protection policy management and trusted report processing.
4. The TPCM uses the built-in controller and I/O port to interact with the controller of the computing module through the bus to actively monitor the computing module.
5. The built-in protection agent in the OS of the computing module obtains the code and data related to the preset protection object and provides them to the TPCM. The TPCM forwards the monitoring information to the TSB, and the TSB analyzes and processes the information according to the policy library.

- **Computing module**: This module includes hardware, an OS, and application layer software. The running of the OS can be divided into the boot phase and the running phase. In the boot phase, GRUB2 and shim of openEuler support the reliability measurement capability, which protects boot files such as shim, GRUB2, kernel, and initramfs. In the running phase, openEuler supports the deployment of the trusted verification agent (provided by third-party vendor HTTC). The agent sends data to the TPCM for trusted measurement and protection in the running phase.

Constraints

- Supported server: TaiShan 200 server (model 2280)
- Supported BMC card: BC83SMMC

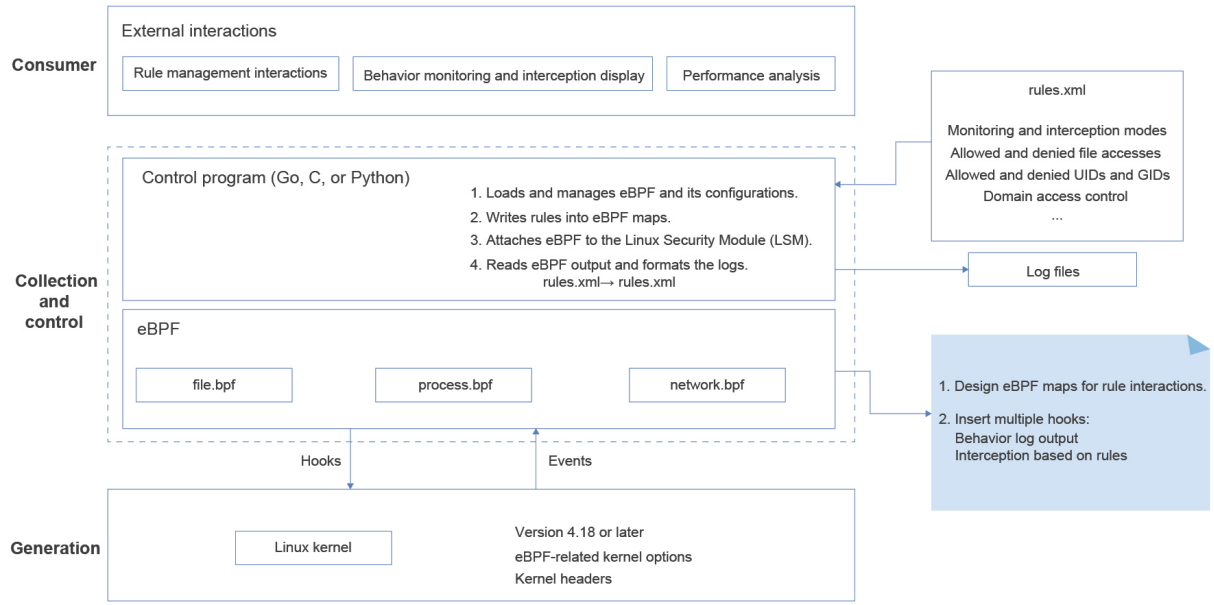
Application Scenarios

The TPCM enables a complete trust chain to ensure that the OS boots into a trusted computing environment.

safeguard

safeguard helps protect the Linux kernel and the OS based on eBPF by intercepting and auditing security operations. It uses the libbpfgo library and the Go language to implement top-level control.

Feature Description



File safeguarding

- Traces file system activities, including file open, close, read, write, and delete.
- Modifies the behavior of file systems through the interception of certain file operations and custom security policies.

  Security policies

  - Operations on files can be intercepted or redirected through eBPF. For example, read and write operations on sensitive files can be intercepted, and access to certain files can be redirected.
  - Access control can be customized. eBPF checks the identity, permissions, and environment of a user who requests access to a file, and allows or denies the request based on custom rules.
  - Audit and monitoring can be customized. For example, eBPF records the information about operations on certain files, such as the operator, time, and action, and outputs the information to the logs.

Process safeguarding

- Traces process life cycles, such as process creation and termination.
- Modifies the behavior of processes, such as injecting or modifying some system calls or implementing custom scheduling policies.

Network safeguarding

- Traces network activities, such as sending, receiving, forwarding, and discarding network packets.
- Modifies the behavior of networks through filtering and rewriting of network packets and custom routing policies.

Application Scenarios

safeguard is a Linux security audit and control solution based on kernel runtime security instrumentation (KRSI), which is a combination of the eBPF and LSM. safeguard ensures OS-wide comprehensive protection and monitoring. While able to fit into a wide range of scenarios, it mainly streamlines the following areas:

- **Container security**: Audits and controls behavior inside a container. For example, safeguard records the container process, file, and network activities, limits resources or ports available to the container, and detects abnormal container behavior. In this way, safeguard effectively protects the container against attacks or abuse, thereby improving the security and stability of the container.
- **Cloud service security**: Audits and controls the clients of cloud service providers. For example, safeguard records OSs, applications, and users of the clients, limits the commands and system calls available to the clients, and detects malicious behavior or exploitation of vulnerabilities. This helps cloud service providers protect resources and reputations and prevents clients from being intruded upon or damaged.
- **Security compliance**: Audits and controls system security compliance. For example, safeguard records information about system configurations, permissions, and logs, prevents the system from modifying specific settings or files, and detects violations and abnormal events in the system. This ensures compliance with security standards and regulations and improves the credibility and legitimacy of the system.

Copyright Statement 08

09 Trademark

All trademarks and logos used and displayed on this document are all owned by the openEuler community, except for trademarks, logos, and trade names that are owned by other parties. Without the written permission of the openEuler community or other parties, any content in this document shall not be deemed as granting the permission or right to use any of the aforementioned trademarks and logos by implication, no objection, or other means. Without prior written consent, no one is allowed to use the name, trademark, or logo of the openEuler community in any form.

10 Appendixes

Appendix 1: Setting Up the Development Environment

Environment Setup	URL
Downloading and installing openEuler	https://openeuler.org/en/download/
Preparing the development environment	https://gitee.com/openeuler/community/blob/master/en/contributors/prepare-environment.md
Building a software package	https://gitee.com/openeuler/community/blob/master/en/contributors/package-install.md

Appendix 2: Security Handling Process and Disclosure

Security Issue Disclosure	URL
Security handling process	https://gitee.com/openeuler/security-committee/blob/master/security-process-en.md
Security disclosure	https://gitee.com/openeuler/security-committee/blob/master/security-disclosure-en.md