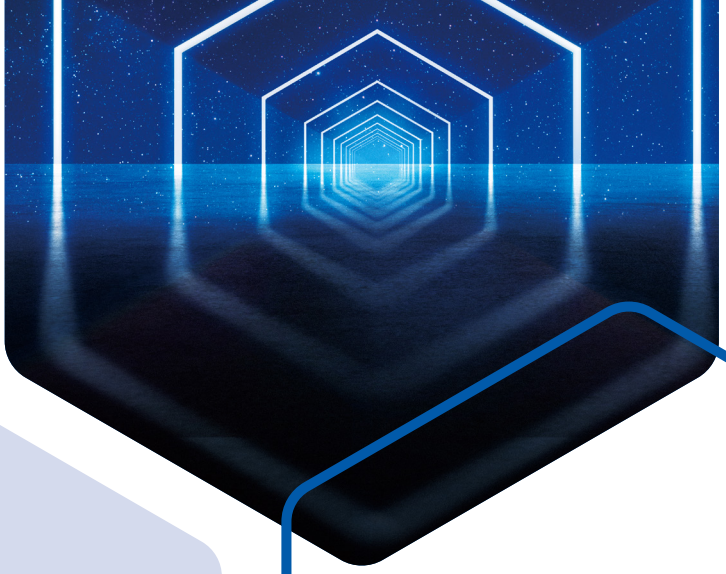
openEuler 23.09

Technical White Paper



CONTENTS

01 Introduction

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.

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 building an OS oriented to 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 innovation version 20.09 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 kernel innovation 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.

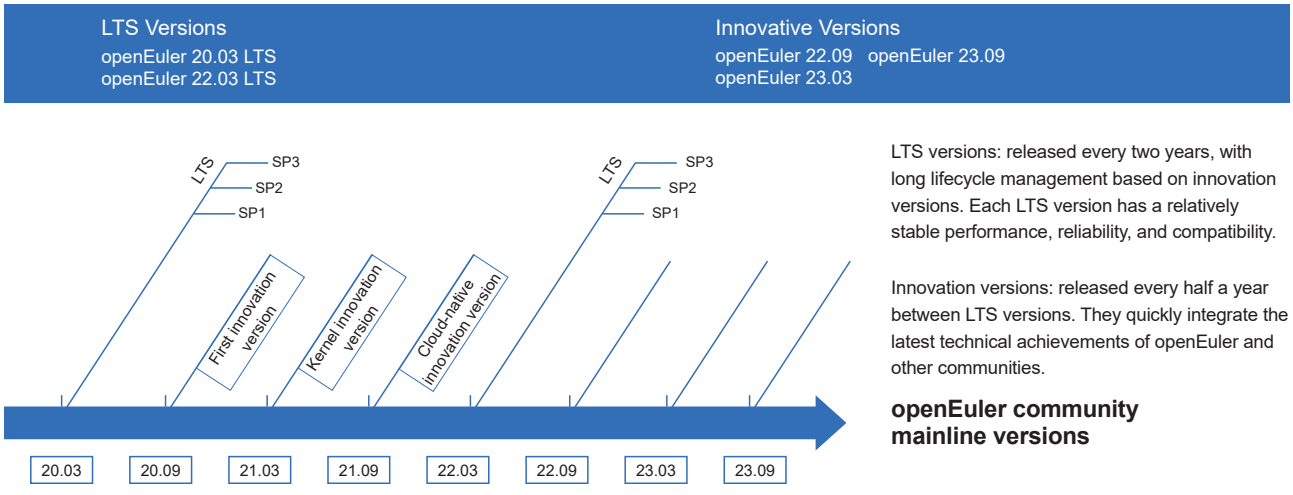
On December 30, 2022, openEuler 22.03 LTS SP1 was released, which is designed for hitless porting with best-of-breed tools.

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.

Later 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.

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.

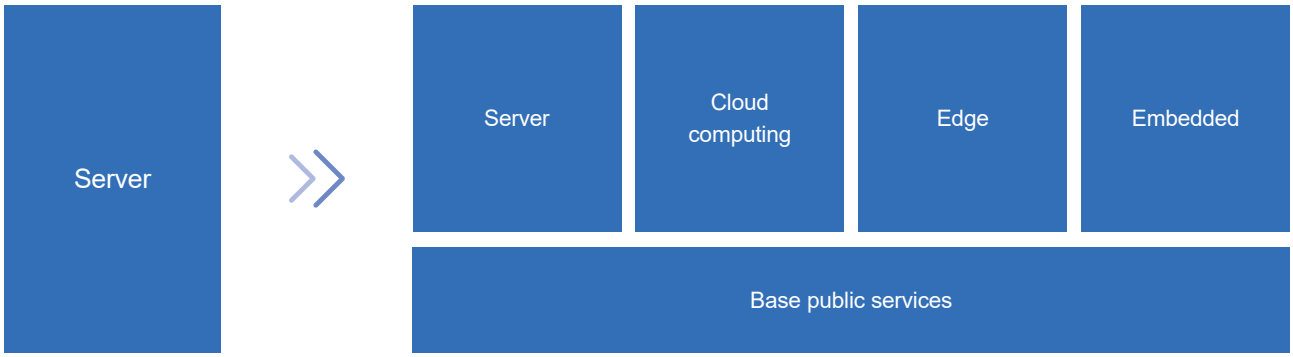
openEuler community mainline versions

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 innovation 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.

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 CPU 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 fit into any scenario.

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.

02 Platform Architecture

System Framework

openEuler is an innovative open source OS platform built on kernel innovations and a solid cloud base to cover all scenarios. It incorporates 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 23.09 runs on Linux kernel 6.4 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 23.09 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.

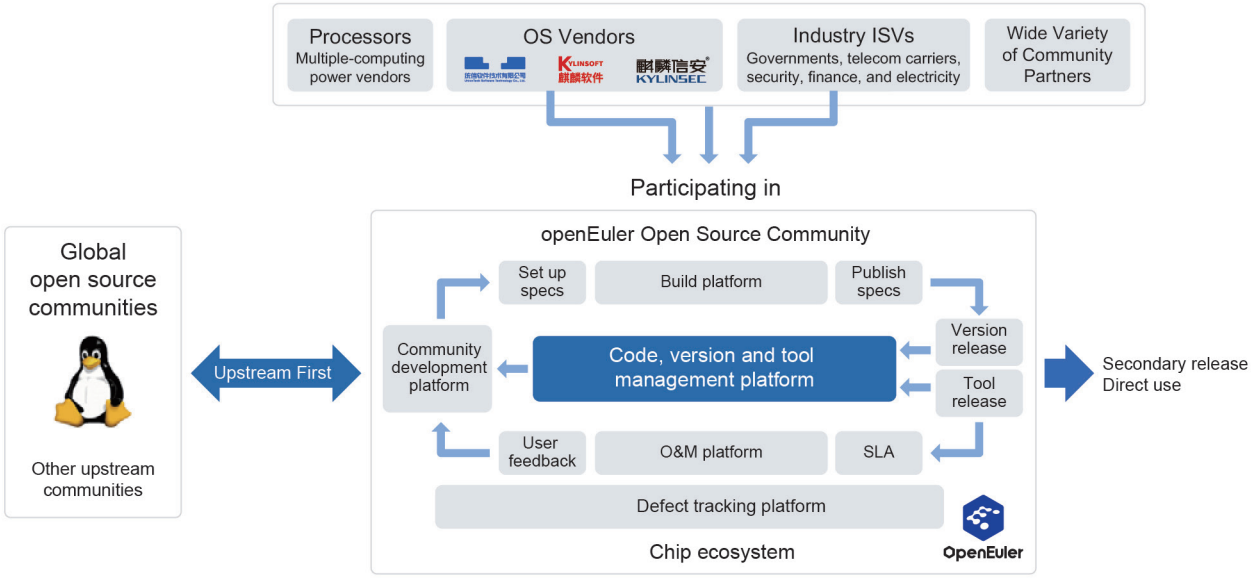
In the future, the openEuler open source community will continue to innovate, aiming to promote the ecosystem and consolidate the digital infrastructure.

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 AMD: Supermicro Hygon: Sugon/Suma Zhaoxin: Zhaoxin	Kunpeng: TaiShan Phytium: QS, PowerLeader

openEuler supports the following cards:

Hardware Type	x86	Arm
NIC	Huawei, Mellanox, Intel	Huawei, Mellanox, Intel
RAID	Avago	Avago
Fibre Channel	Marvell, Emulex	Marvell, Emulex
GPU & AI	NVIDIA	NVIDIA
SSD	Huawei	Huawei

For the complete compatibility list, visit https://www.openeuler.org/en/compatibility/.

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	At least 4 GB
Drive	At least 20 GB

VMs

openEuler supports the following virtual machines (VMs):

- centos-6 qemu 6.2.0-79.oe2309 libvirt 6.2.0-59.oe2309 virt-manager 4.1.0.2-oe2309
- centos-7 qemu 6.2.0-79.oe2309 libvirt 6.2.0-59.oe2309 virt-manager 4.1.0.2-oe2309
- centos-8 qemu 6.2.0-79.oe2309 libvirt 6.2.0-59.oe2309 virt-manager 4.1.0.2-oe2309
- windows2016 qemu 6.2.0-79.oe2309 libvirt 6.2.0-59.oe2309 virt-manager 4.1.0.2-oe2309 qemu 8.1.0
- windows2019 qemu 6.2.0-79.oe2309 libvirt 6.2.0-59.oe2309 virt-manager 4.1.0.2-oe2309 qemu 8.1.0

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

Embedded 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, AArch32, x86_64
Memory	At least 512 MB
Drive	At least 256 MB

Scenario-specific Innovations 04

GMEM

In the post-Moore era, there have been breakthroughs in GPUs, TPUs, and FPGAs and other dedicated heterogeneous accelerators. Similar to CPUs, these devices increase computing speeds by storing data in local memory (such as LPDDR SDRAM or HBM), but such design catalyzes more complicated memory systems.

Modern memory systems have the following defects:

- Memory management is divided between the CPU and accelerator sides. Explicit data migration makes it difficult to balance the usability and performance of the accelerator memory.
- High bandwidth memory (HBM) of accelerator devices is typically insufficient for foundation models, and manual swap operations can be performed only in certain situations and even then cause large performance loss.
- A large number of invalid data migrations occur in search & recommendation and big data scenarios, and no efficient memory pooling solution is available.

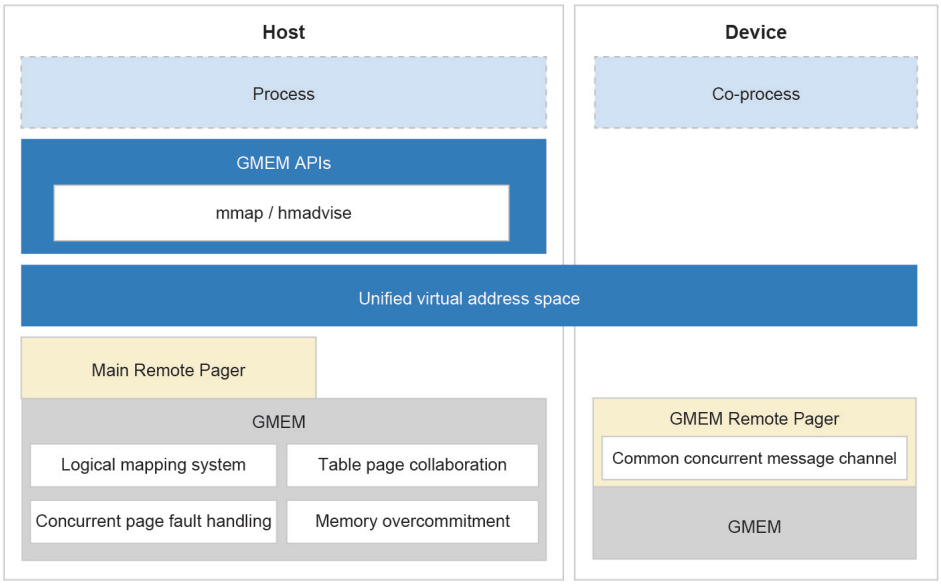
Heterogeneous Memory Management (HMM) is a Linux feature that is plagued by issues of poor programming, performance, and portability, while also relying heavily on manual tuning. As such, it is unfavored by most OS communities, and has fueled demand for an efficient solution for heterogeneous accelerators.

Generalized Memory Management (GMEM) is one new option, which offers a centralized management mechanism for heterogeneous memory connections. GMEM APIs are compatible with native Linux APIs, and feature high usability, performance, and portability.

After an accelerator calls GMEM APIs to connect its memory to the unified address space, the accelerator automatically obtains the programming optimization capability for heterogeneous memory, and does not need to execute the memory management framework multiple times. This greatly reduces development and maintenance costs.

Developers can apply for and release a unified set of APIs to achieve heterogeneous memory programming without memory migrations. If HBM of an accelerator is insufficient, GMEM can use the CPU memory as the accelerator cache to transparently over-allocate the HBM without manual swap. GMEM offers an efficient memory pooling solution thanks to a shared memory pool that eliminates the need for duplicate migrations.

Feature Description



GMEM enhances memory management in the Linux kernel. Its logical mapping system masks the differences between the ways how the CPU and accelerator access memory addresses. The Remote Pager memory message interaction framework provides the device access abstraction layer. In the unified address space, GMEM automatically migrates data to the OS or accelerator when data is to be accessed or paged.

- **GMEM**

  GMEM leverages the general-purpose computing capabilities of CPUs and accelerators. It combines the two independent address spaces of the OS and accelerator into a unified virtual memory address space, to realize unified memory management and transparent memory access.

  GMEM uses a collection of new logical page tables to manage the unified virtual memory address space, ensuring data consistency between the page tables of different CPUs and micro architectures. Based on the memory access consistency mechanism of the logical page tables, the target memory space can be migrated between the host and accelerator using a kernel page fault handling procedure. If the accelerator's memory space is insufficient, the accelerator can borrow memory from the host and reclaim its cold memory to achieve memory overcommitment. This practice removes issues in which model parameters are restricted by the accelerator's memory, and reduces the cost of foundation model training.

  GMEM high-level APIs in the kernel allow the accelerator driver to directly use memory management functions by registering the MMU functions defined in the GMEM specification. With memory management functions, the accelerator can create logical page tables and perform memory overcommitment. The logical page tables decouple the high-layer logic of memory management from the CPU's hardware layer, so as to abstract the high-layer memory management logic that can be reused by different accelerators. This design only requires the accelerator to register bottom-layer functions, and does not need high-layer logic for the unified address space.

- **Remote Pager**

  Remote Pager is a memory message interaction framework that adopts message channel, process management, memory swap, and memory prefetch modules for the interaction between the host and accelerator. It is enabled by the independent driver **remote_pager.ko**. The Remote Pager abstraction layer simplifies device adaptation by enabling third-party accelerators to easily access the GMEM system.

- **User APIs**

  To allocate the unified virtual memory, you can directly use the memory map (mmap) of the OS. GMEM adds the flag (MMAP_PEER_SHARED) of allocating the unified virtual memory to the mmap system call.

  The libgmem user-mode library provides the hmadvise API of memory prefetch semantics that helps optimize the accelerator memory access. For details, see https://gitee.com/openeuler/libgmem/blob/master/README.md.

Constraints

- GMEM supports 2 MB huge pages only. Therefore, transparent OS huge pages must be enabled on hosts and NPUs to use GMEM.
- The heterogeneous memory obtained using **MAP_PEER_SHARED** cannot be inherited during fork.

Application Scenarios

- **Unified heterogeneous memory programming**

  To simplify heterogeneous memory programming, GMEM can allocate a unified virtual memory to the CPU and accelerator, which then share the same pointer. For example, an NPU memory management system can be connected to GMEM with just 100 lines of modified code in the NPU driver, a huge reduction on the original 4,000 lines in the memory management framework.

- **Automatic memory overcommitment of an accelerator**

  When a GMEM API is used to allocate memory to an application, the application is not limited by the physical memory capacity

of the accelerator. Instead, the application can transparently over-allocate memory until the CPU's DRAM capacity is exhausted. GMEM swaps cold device memory pages out to the CPU memory to achieve high-performance and easy-to-start training and inference. This design makes GMEM deliver 1.6 times the performance of NVIDIA-UVM in ultra-large model training when the overcommitment ratio is 200%. (Test results compared the Ascend 910 and NVIDIA A100 NPUs under the same HBM conditions.)

Native Support for Open Source Large Language Models (LLaMA and ChatGLM)

Despite the widespread interest in large language models, there are still some difficulties for common users to use these models. The two model inference frameworks, llama.cpp and chatglm-cpp, are implemented based on C/C++, to facilitate model quantization and sequent deployment and use of open source large language models on CPUs.

Feature Description

llama.cpp can be used to deploy multiple open source English large language models, such as LLaMa, LLaMa2, and Vicuna. chatglm-cpp can be used to deploy multiple open source Chinese large language models, such as ChatGLM-6B, ChatGLM2-6B, and Baichuan-13B. Developers can use llama.cpp or chatglm-cpp to select an open source model that best suits their needs.

The two model inference frameworks are implemented based on ggml C/C++, and accelerate memory to realize efficient CPU inference through int4/int8 quantization, optimized KV cache, and parallel computing.

Application Scenarios

Llama.cpp and chatglm-cpp can be deployed on a computer that supports only CPUs to enable intelligent functions such as Q&A and AI dialog.

05 Kernel Innovations

New and Inherited Features in the openEuler Kernel 6.4

openEuler 23.09 runs on Linux kernel 6.4 and inherits the advantages of openEuler community versions and other innovative features.

- **Tidal affinity scheduling**: The system dynamically adjusts CPU affinity based on the service load. When the service load is heavy, the system adds new CPU cores to improve the quality of service (QoS); otherwise, the system uses preferred CPUs to enhance resource locality.
- **Priority-based load balancing**: CPU QoS isolation is enhanced in online and offline hybrid deployments, and QoS load balancing across CPUs is supported to further reduce QoS interference from offline services.
- **SMT expeller free of priority inversion**: This feature resolves the priority inversion problem in the simultaneous multithreading (SMT) expeller feature and reduces the impact of offline tasks on the QoS of online tasks.
- **Multiple priorities in a hybrid deployment**: Each cgroup can have a **cpu.qos_level** that ranges from –2 to 2, and the value of **qos_level_weight** relates to the priority of each cgroup, which allocates CPU resources based on the CPU usage. This feature is also capable of wakeup preemption. It ensures that high-priority and latency-sensitive online services are not affected by offline services while improving resource utilization.
- **Programmable scheduling**: The eBPF-based programmable scheduling framework extends the kernel scheduler policies to meet varying loads. It has the following features:
  - » Tag management mechanism: The open capability of tagging tasks and task groups allows users and kernel subsystems to tag specific workloads by calling interfaces. The scheduler can detect tasks of specific workloads by tag.
  - » Policy extension: The programmable scheduling framework adds new extension points and various auxiliary methods to support policy extension for completely fair scheduling (CFS) preemption, core selection, and task execution.
- **NUMA-aware spinlock**: The lock transfer algorithm offers increased lock throughput for the multi-NUMA system based on the MCS spinlock. The lock is preferentially transferred within the local NUMA node, greatly reducing cross-NUMA cache synchronization and ping-pong.
- **TCP compression**: Because network bandwidth is a typical bottleneck in big data and other distributed scenarios involving a large amount of data transfer, the data of specified ports is compressed at the TCP layer before transfer and, once received, the data is decompressed and transferred to the user mode. TCP compression accelerates data transfer between nodes.
- **Hot patches**: Kernel hot patches are used to fix bugs in kernel function implementation without a system restart, and dynamically replace functions when the system is running. Hot patches on openEuler ensure high efficiency by modifying instructions, that is, they directly jump to new functions without search and transfer, whereas hot patches on the Linux mainline run based on ftrace utility.
- **Shared pool**: A shared pool is used to share data between processes. Multiple processes read and write the same memory area concurrently to avoid multiple copy operations, improving data access efficiency and reducing the communication overhead between processes.
- **Memcg asynchronous reclamation**: Memcg is a kernel mechanism that controls the memory usage of process groups. When the memory used by a process group exceeds the specified limit, memcg triggers memory reclamation to ensure system stability and reliability. Memcg asynchronous reclamation is an optimized mechanism that asynchronously reclaims memory in the memcg instance when the system load is low, to avoid memory reclamation impact when the system load is high, helping improve system reliability and performance.
- **Cgroup files**: The cgroup file subsystem manages the number of files (that is, the number of handles) opened by a collection of processes. It provides easy-to-call APIs that, compared with the rlimit method, better control the number of file handles used for resource application and release, resource usage, and group control. The cgroup file subsystem prevents a process from opening too many files to ensure that system resources are not overused.
- **Cgroup writeback for cgroup v1**: Cgroup writeback provides a flexible method to manage the writeback behavior of the file system cache. It optimizes I/Os and enhances resource management. The main functions of cgroup writeback include

cache writeback control, I/O priority control, and writeback policy adjustment.

- **Core suspension detection**: If the performance monitor unit (PMU) stops counting, the hard lockup cannot detect system suspension. The core suspension detection feature enables each CPU to confirm adjacent CPUs that have been suspended. It ensures that self-healing can be performed even when interrupts are disabled on part of CPUs.

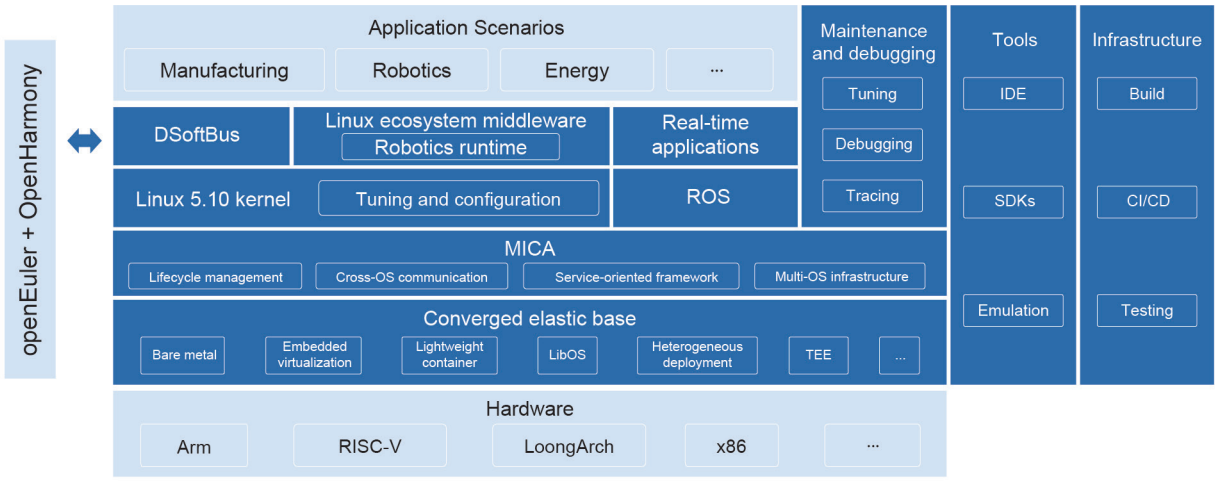
Enhanced Features 06

Embedded

openEuler 23.09 Embedded is developed based on Linux kernel 5.10. It 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 streamlines development of applications built for industrial control and robotics, whereby innovations help optimize its embedded technology stack and ecosystem. openEuler 23.09 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 suite your services. openEuler 23.09 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 requirements of ROS 2 runtime. 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.

Feature Description



System architecture

Southbound ecosystem

openEuler Embedded Linux supports AArch64 and x86-64 chip architectures and related hardware such as RK3568, Hi3093, Raspberry Pi 4B, and x86-64 industrial computers. openEuler 23.09 Embedded supports RK3399 and RK3588 chips, and preliminarily supports AArch32 and RISC-V chip architectures but only based on QEMU simulation. In the future, openEuler 23.09 Embedded will support Loongson and Phytium chips.

Embedded elastic virtualization base

openEuler Embedded uses a converged elastic 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. These technologies bring many benefits like high performance (bare metal servers), isolation and protection (embedded virtualization), and usability and flexibility (lightweight containers).

- The bare metal hybrid deployment solution runs on OpenAMP and supports the hybrid deployment of UniProton/Zephyr/RT-Thread and openEuler Embedded Linux. It manages peripherals by partition at the highest performance level but provides inadequate isolation and flexibility.
- Partitioning-based virtualization is an industrial-class hardware 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
- Debugging the RTOS (UniProton) using the GNU debugger (GDB) on openEuler Embedded Linux in bare metal mode

Northbound ecosystem

- **Northbound software packages**: Over 350 common embedded software packages can be built using openEuler.
- **ROS runtime**: openEuler Embedded 23.09 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 is inherited from Linux kernel 5.10, and helps respond to soft real-time interrupts within microseconds.
- **Distributed soft bus**: The distributed soft bus system of openEuler Embedded 23.09 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.

UniProton

UniProton is a hard 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 chips and boards of 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 the GDB on openEuler Embedded Linux.
- Over 360 POSIX APIs available for file systems, device management, and shell consoles.

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.

GCC for openEuler

The baseline version of GCC for openEuler has been upgraded from open source GCC 10.3 to GCC 12.3 and supports features such as automatic feedback-directed optimization (AutoFDO), software and hardware collaboration, memory optimization, Scalable Vector Extension (SVE), and vectorized math libraries.

- The default language of GCC for openEuler has been upgraded from C14/C++14 to C17/C++17, enabling GCC for openEuler to support more hardware features like Armv9-A and x86 AVX512-FP16.

Item	GCC 10.3.0	GCC 11.3.0	GCC 12.3.0
Release date	2021-04-08	2022-04-21	2023-05-08
C standard	C17 by default C2x supported	C17 by default C2x supported	C17 by default C2x supported
C++ standard	C++ 14 by default C++ 17 supported C++ 2a experimental optimization C++ 20 partially supported	C++ 17 supported C++ 2a experimental optimization C++ 20 partially supported	C++ 17 supported C++ 2a experimental optimization C++ 20 partially supported
New architecture features	Armv8.6-a (bfloat16 Extension/Matrix Multiply Extension) SVE2 Cortex-A77 Cortex-A76AE Cortex-A65 Cortex-A65E Cortex-A34	Armv8.6-a, +bf16, +i8mm Armv8.6-r Cortex-A78 Cortex-A78AE Cortex-A78C Cortex-X1	Armv8.6-a, +ls64 atomic load and store Armv8.8-a, +mop, accelerate memory operations Armv9-a Ampere-1 Cortex-A710 Cortex-x2 AVX512-FP16 SSE2-FP16

- GCC for openEuler supports structure optimization and instruction selection optimization, leveraging Arm hardware features to improve system running efficiency. In the benchmark tests such as SPEC CPU 2017, GCC for openEuler has proven to deliver higher performance than GCC 10.3 of the upstream community.
- Further, it fuels AutoFDO to improve the performance of the MySQL database at the application layer.

Feature Description

- **SVE vectorization**: Significantly improves program running performance for Arm-based machines that support SVE instructions.
- **Memory layout**: Rearranges the positions of structure members so that frequently accessed structure members are placed in continuous memory space, increasing the cache hit ratio and improving program performance.
- **Redundant member elimination**: Eliminates structure members that are never read and deletes redundant write statements, which in turn reduces the memory occupied by the structure and alleviates subsequent bandwidth pressure, while improving performance.

- **Array comparison**: Implements parallel comparison of array elements to improve execution efficiency.
- **Arm instructions**: Simplifies the pipeline of ccmp instructions for a wide range of deployments.
- **AutoFDO**: Uses perf to collect and parse program information and optimizes feedback across the compilation and binary phases, boosting mainstream applications such as MySQL databases.

Application Scenarios

In the general-purpose computing test of SPEC CPU 2017, GCC for openEuler is proven to improve performance by 20% when compared with GCC 10.3 of the upstream community.

With AutoFDO enabled, the MySQL performance is improved by more than 15%, and with kernel-mode PGO enabled, the UnixBench performance is improved by more than 3%.

Kmesh

The boom of cloud-based AI and live streaming applications has seen data centers expand to connect with more cluster services. It is a big challenge to boost communication between cluster services while meeting SLA requirements.

Popular infrastructures, like Kubernetes, enable agile application deployment and management on the cloud, though they cannot effectively orchestrate application traffic. To compensate for this, service meshes are introduced. However, proxies in a service mesh generally incur extra latencies and overheads in the data plane.

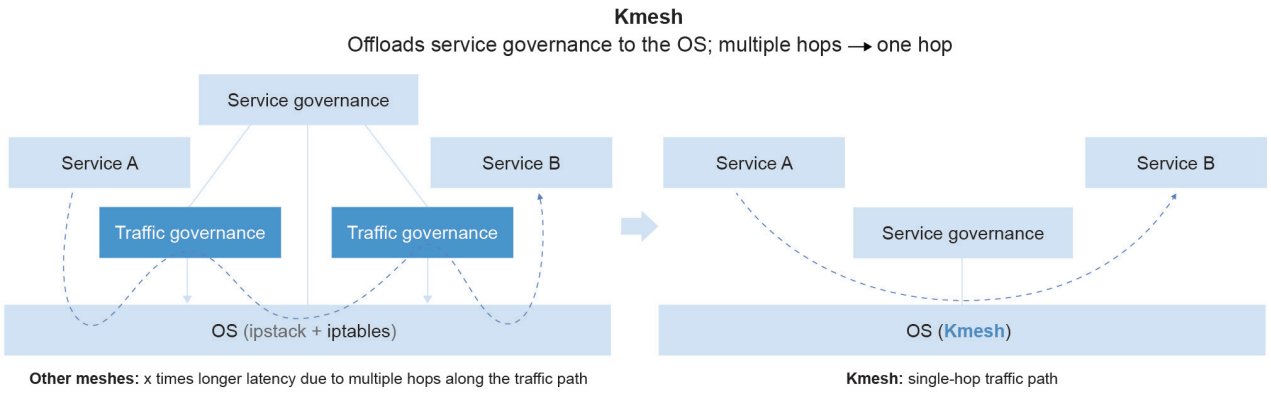
- **Latency**

  Taking the typical service mesh Istio as an example, the single-hop access latency increases by 2.65 ms after meshing, which is too high for core applications.

- **Resource overhead**

  By default, each Istio sidecar occupies at least 50 MB of memory and two CPU cores. For large clusters, such overhead is too high to deploy service containers.

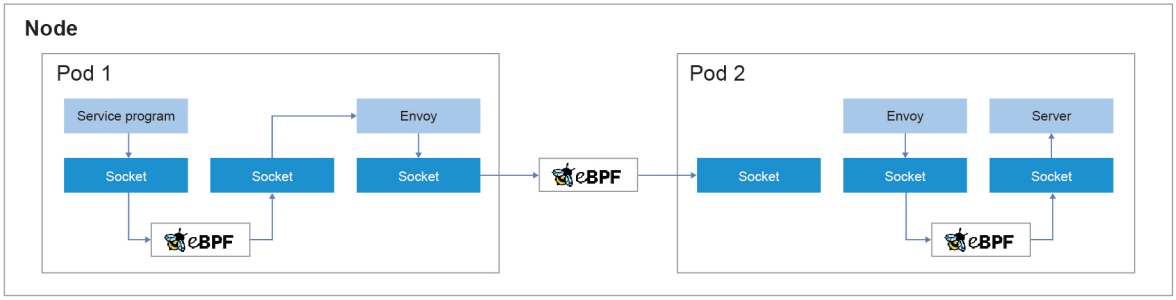
Kmesh runs on a programmable kernel to offload service governance to the OS, which shortens the communication latency between services to one-fifth the industry average.



Feature Description

- Kmesh can connect to a mesh control plane (such as Istio) that complies with the Dynamic Resource Discovery (xDS) protocol.
- Kmesh orchestrates application traffic in the following ways:
  - » Load balancing: Various load balancing policies such as polling.
  - » Routing: L4 and L7 routing support.
  - » Gray: Backend service policies available in percentage mode.
- Sockmap for mesh acceleration

  When a sockmap is used in a service mesh, the eBPF program takes over the communication between service containers and Envoy containers. As a result, the communication path is shortened to achieve mesh acceleration. The eBPF program can also accelerate the communication between pods on the same node.



Note:

- Only data connections created after a sockmap is used can be accelerated. Connections established before that will not be accelerated.
- Only IPv4 TCP connections on the same node can be accelerated. Nodes using IPv4 TCP will experience a latency loss of 10% to 20%.

Application Scenarios

Optimized communication for cloud-native service meshes to meet latency-sensitive needs of e-commerce, billing, finance, logistics, short video, online conference, and cloud gaming deployments.

sysMaster

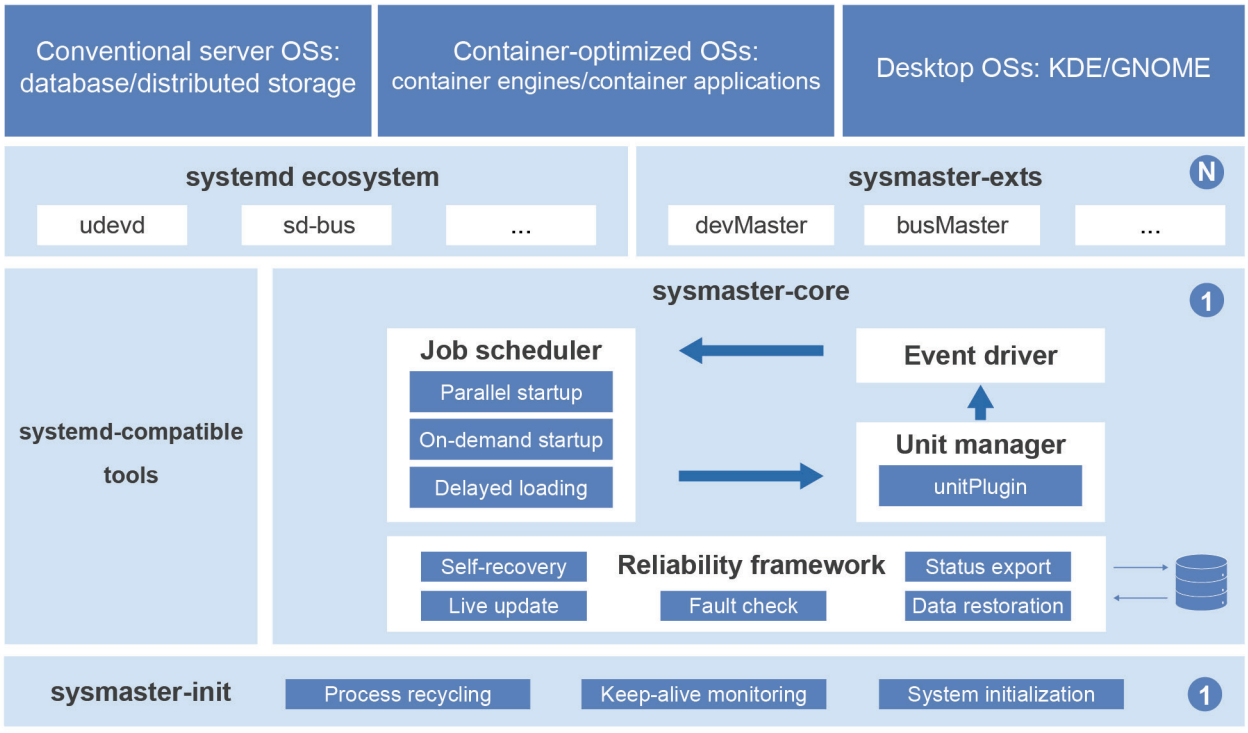
sysMaster is a collection of ultra-lightweight and highly reliable service management programs. It provides an innovative implementation of PID 1 to replace the conventional init process. Written in Rust, sysMaster is equipped with fault monitoring, second-level self-recovery, and quick startup capabilities, 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:

- 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 0.5.0 released with openEuler 23.09 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

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

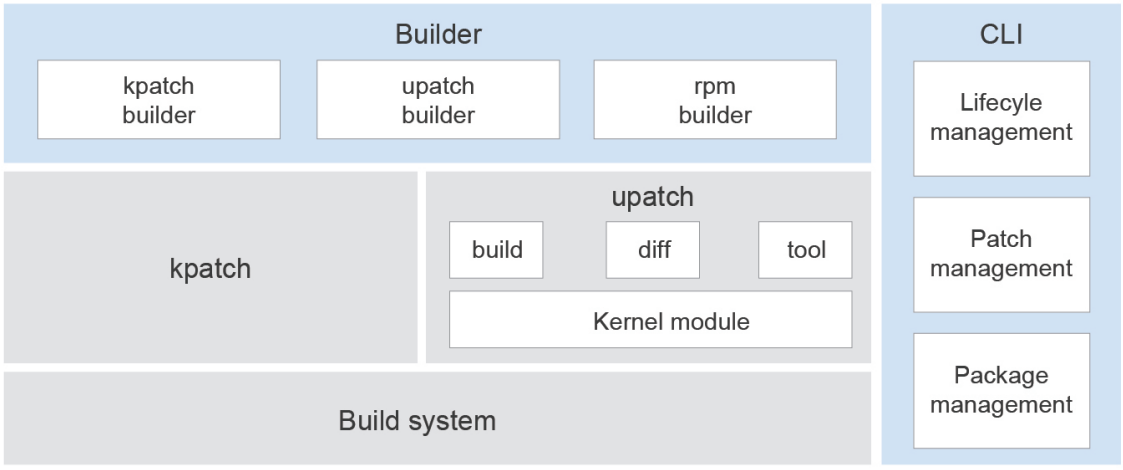
- **Hot patch making**

  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.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.

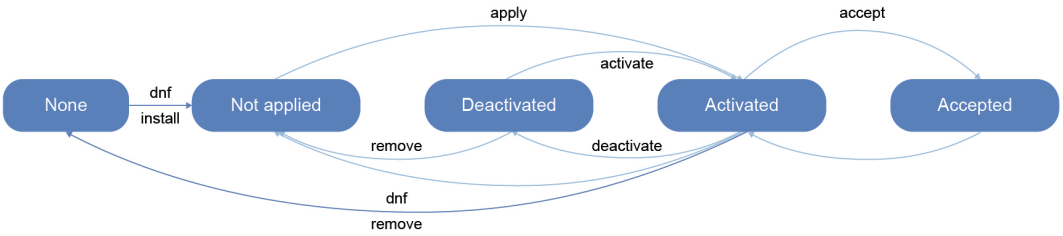
- **Patch lifecycle management**

  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



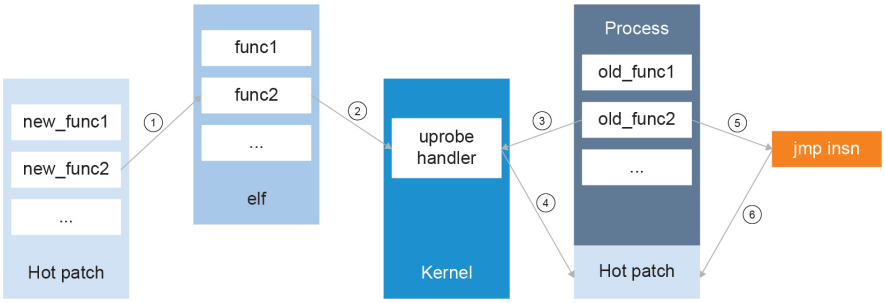
Lifecycle of a hot patch



- **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

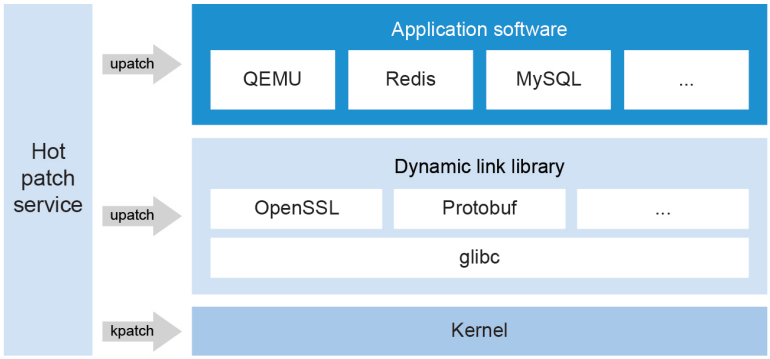


1. Execute the uprobe system call and add a uprobe breakpoint at the function to be modified.
2. Register a uprobe handler.
3. Call the uprobe handler when the function is running.
4. Map the patch to the address space of the current process.
5. Perform security check and change the first instruction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

  Patches can be built in containers.

  » eBPF is used to monitor the compiler process. Now, changes in hot patches can be obtained in pure user mode without creating character devices, and users can compile hot patches in multiple containers concurrently.

  » Users can install an RPM package (syscare-build-kmod or syscare-build-ebpf) to use ko or eBPF. The syscare-build process automatically adapts to underlying implementation.

- **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

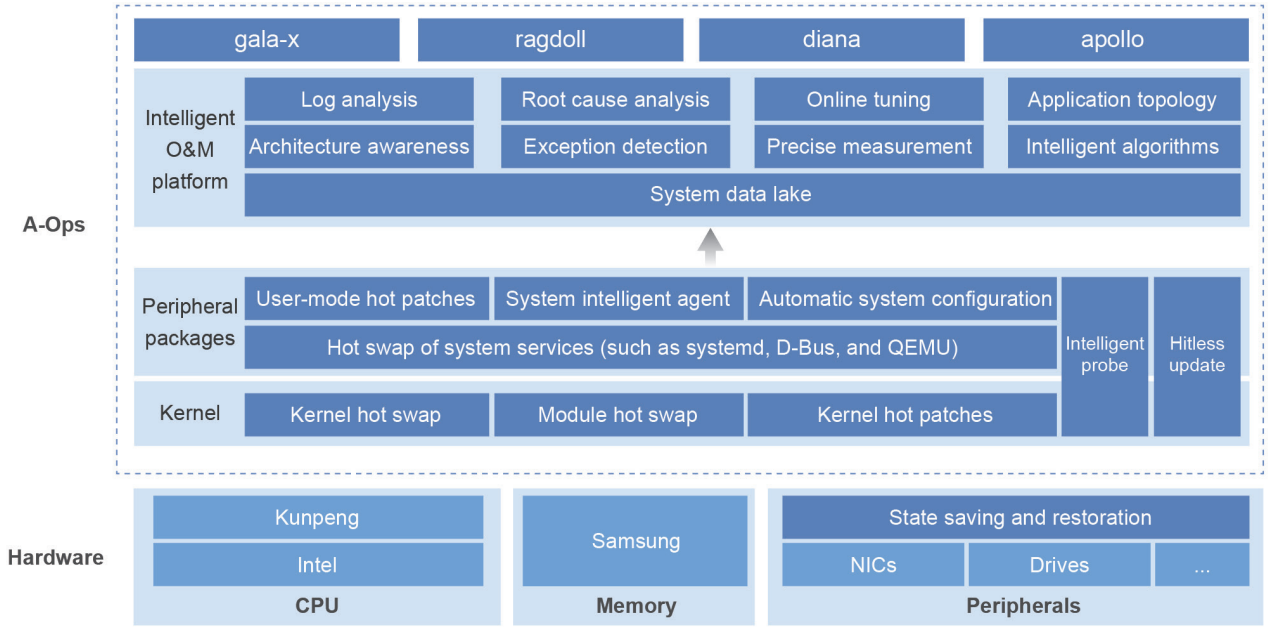
Scenario 1: quick fix of common vulnerabilities and exposures (CVEs)

Scenario 2: temporary locating for live-network issues

A-Ops

Big data and machine learning technologies are generating huge amounts of data at a 2- to 3-fold increase every year. This is driving demands for intelligent O&M systems that support huge capacities while reducing costs. A-Ops is such an intelligent O&M framework that offers base O&M capabilities of CVE management, exception detection (in databases), and quick troubleshooting.

Feature Description



A-Ops software architecture

- **Intelligent patch management**
  - » Patching service: Releases cold and hot patches for each CVE.
  - » Kernel hot fixing: Fixes defects in the kernel without restarting any process or the computer.
  - » Intelligent patch inspection: Provides CVE inspection and notification for standalone nodes and node clusters, and supports quick repair and rollback, slashing patch management costs and improving vulnerability repair efficiency and cluster security.
  - » Hybrid management of cold and hot patches: All patches available for the system are automatically incorporated to reduce the number of patches on the live network and alleviate cluster O&M pressure.
- **Exception detection**

  Intelligent fault diagnosis: Detects network I/O delays, packet loss, interruption, and high disk I/O loads in MySQL and openGauss environments.
- **Configuration tracing**
  - » Configuration baseline: Collects cluster configuration information and provides baseline capabilities to facilitate configuration management.
  - » Configuration exception check: Checks the entire cluster and compares the configuration with the baseline in real time to identify unauthorized configuration changes and locate faults.

Application Scenarios

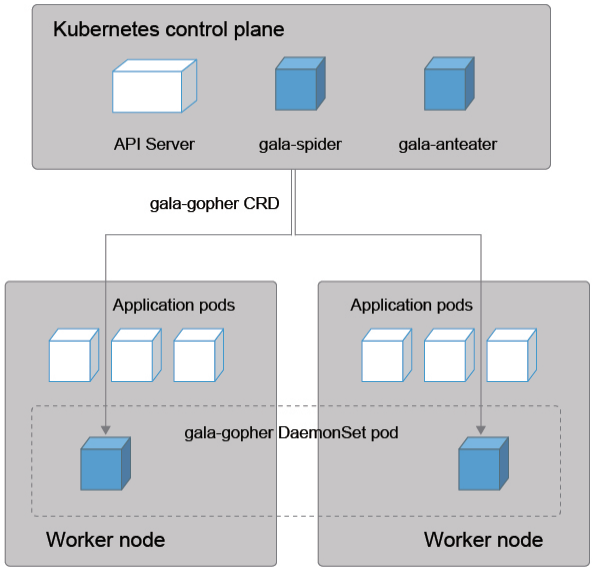
A-Ops mounts the community CVE repo source, inspects CVEs, and uses cold and hot patch releases (RPM packages) to fix, roll back, and incorporate CVEs.

A-Ops gala

The gala project is designed to cover all Kubernetes fault diagnosis scenarios, including application drill-down analysis, observable performance across microservices and databases, cloud-native monitoring and profiling, process diagnosis, and minute-level diagnosis of five types of OS issues (network, drives, processes, memory, and scheduling).

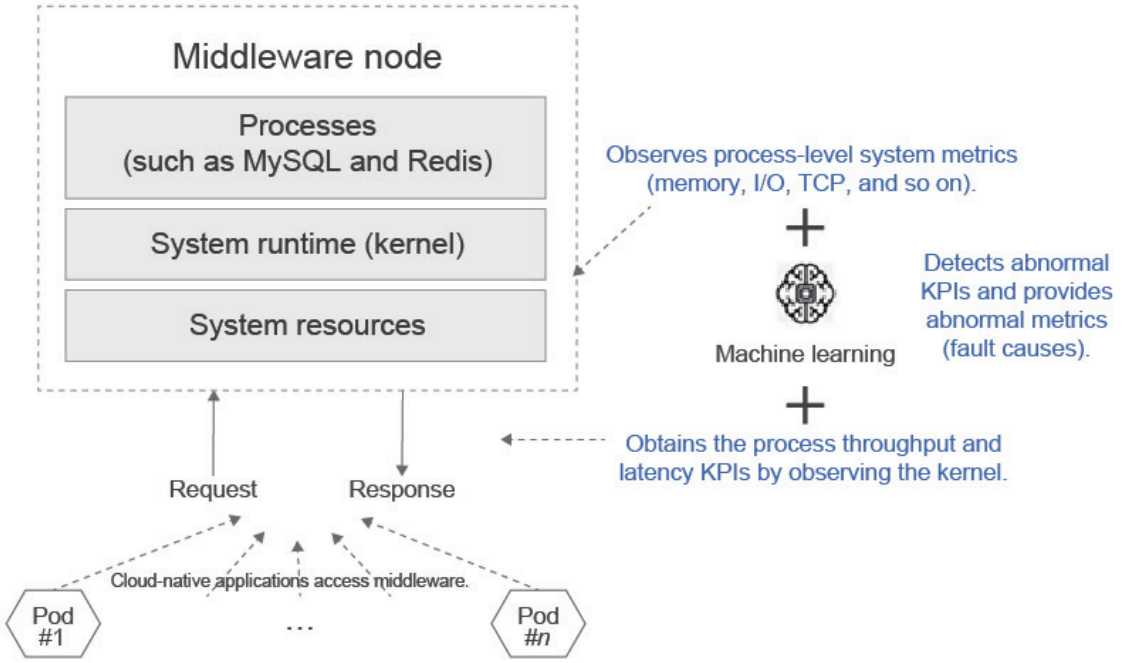
Feature Description

- **Easy deployment on Kubernetes**: gala-gopher can be deployed as a DaemonSet, with a gala-gopher instance deployed on each worker node. gala-spider and gala-anteater are deployed as containers on the Kubernetes master node.



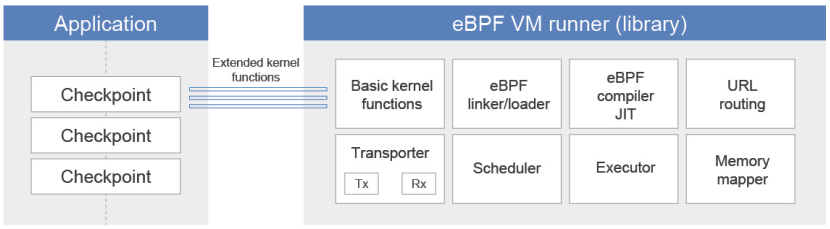
- **Application drill-down analysis** : Diagnoses and demarcates subhealth status in cloud-native environments and applications in minutes.
- **Full-stack monitoring**: Provides application-oriented refined monitoring across software stacks, including the language runtime (JVM), glibc, system call, and kernel (TCP, I/O, and scheduling), and provides insight into the real-time impact of system resources on applications.
- **Full-link monitoring**: Provides network traffic (TCP and RPC) and software topology information on a 3D system display to accurately show the resource scope and identify the fault radius.
- **Causal AI model**: Demarcates faults in minutes with visualized root cause derivation.
- **Observable microservice and database performance**: Provides non-intrusive observation of microservices, databases, and HTTP1.x performance, covering the throughput, latency, and error rate; and supports refined API observation and HTTP TRACE to view abnormal HTTP requests.
- **Observable PostgreSQL access performance**: Observes the throughput, latency, and error rate and supports refined observation for access and slow trace to display SQL statements of slow SQL queries.
- **Cloud-native application performance profiling**: Provides a non-intrusive cross-stack profiling analysis tool that avoids the need for modification and connects to the common UI front end of Pyroscope. The technical features are as follows:

  - » Low memory overhead, with < 2% interference in benchmark tests.
  - » Wide support for mainstream languages like C/C++, Go, Rust, and Java.
  - » Multi-instance profiling of multiple processes or containers. The UI front end can compare and analyze problem causes.
  - » Comprehensive and custom profiling across processes, containers, and PoDs.
  - » Multi-dimensional profiling, including **OnCPU**, **OffCPU**, and **MemAlloc**, for applications.
- **Cloud-native monitoring**: Provides TCP, socket, and DNS monitoring for networks of Kubernetes environments.
- **Process performance diagnosis**: Diagnoses issues for middleware (such as MySQL and Redis) in cloud-native environments, monitors process performance KPIs and related system-layer metrics (I/O, memory, and TCP), and detects KPI exceptions and system-layer metrics that affect the KPIs (causes of process performance issues).



CTinspector

CTinspector is a language VM framework developed by China Telecom e-Cloud Technology Co., Ltd. based on the eBPF instruction set. CTinspector enables quick expansion of application instances to diagnose network performance bottlenecks, storage I/O hotspots, and load balancing issues, ensuring stable and timely diagnosis during system running.

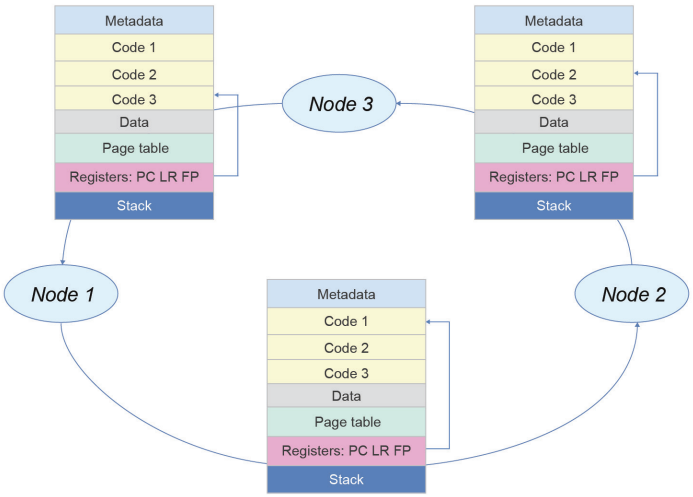


The CTinspector framework comprises the following components:

- **eBPF compiler/JIT**: The eBPF compiler compiles C code into eBPF binary code, and JIT compiles eBPF instructions into machine code.
- **eBPF linker/loader**: loads and links library functions, that is, kernel functions.
- **Runner**: executes the eBPF VM, including loading registers, code segments, and stacks, and mapping data segments.
- **Scheduler**: determines when to execute the eBPF VM, including determining the VM status and dependency wait conditions.
- **Basic kernel functions**: basic library functions, such as transporter, memory mapper, fork, and join_meeting.
- **Extended kernel functions**: custom library functions provided by each hook point in addition to the core functions provided by the eBPF VM runner.
- **Memory mapper**: maps application data to the eBPF VM to ensure the eBPF program can read and write application data.

Feature Description

CTinspector uses a packet VM of the eBPF instruction set. The minimum size of the packet VM is 256 bytes, covering registers, segments (stack, code, and data), and page tables. The packet VM supports independent migration, in which the packet VM code can invoke the migrate kernel function to migrate to a specified node. It also supports resumable execution, that is, once migrated, the packet VM continues to execute the next instruction from the position where it has been interrupted on the previous node. This function plays an important role in running stateful commands on different nodes in the network topology.



Application Scenarios

Up to 10 MB CT flow tables can be used for Open vSwitch connection tracking, and each flow table can contain 20 or more fields. Typically, O&M personnel dump Open vSwitch flow tables to locate a network fault, such this can take a long time due to the large number of flow tables. It is necessary to filter the flow tables, and in such a scenario, CTinspector can analyze Open vSwitch flow tables of virtual networks.

Traditional Access Control Lists (ACLs) require complex configurations and are not intuitive in that packets are accepted or dropped only after all chain rules have been checked. Its implementation, especially range comparison, mask matching, and command line parsing, is also complex. CTinspector is introduced to help O&M personnel develop application instances and deliver ACL rules.

CTinspector supports resumable execution of commands, which can be used to test network connectivity and diagnose network usage performance metrics.

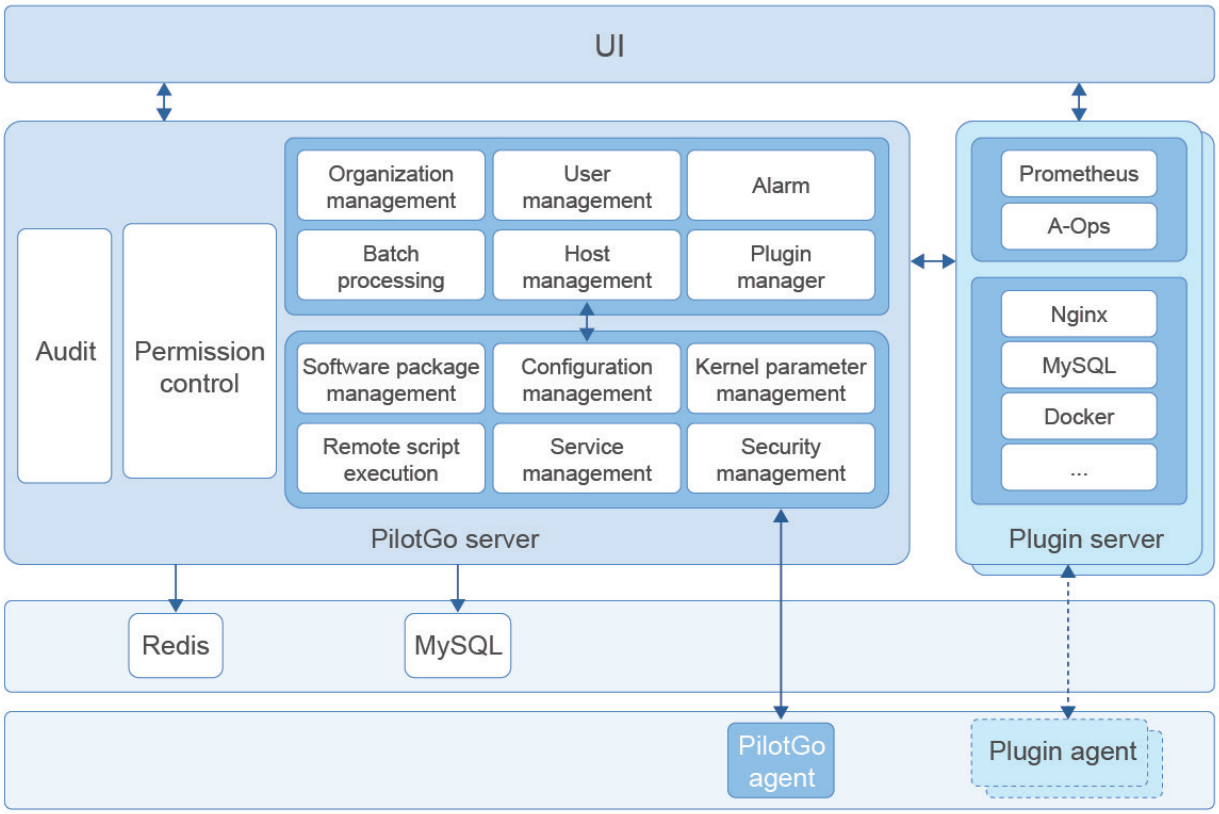
PilotGo

PilotGo is a plugin-based O&M platform developed by the openEuler community. It adopts a lightweight modular design of functional modules which each can be iterated and evolved independently, while ensuring stability of core functions. Plugins are used to enhance platform functions and remove differences between O&M components, implementing global situation awareness and automation.

Feature Description

PilotGo has the following core functional modules:

- **User management**: Manages users by group based on the organizational structure, and imports existing platform accounts for streamlined migration.
- **Permission management**: Supports role-based access control, which is flexible and reliable permission management.
- **Host management**: Offers visualized front-end status, software package and service management, and kernel parameter optimization.
- **Batch management**: Performs concurrent O&M operations.
- **Log audit**: Traces and records user and plugin change operations, facilitating issue backtracking and security audit.
- **Alarm management**: Detects platform exceptions in real time.
- **Plugins**: Extends platform functions and associates plugins to realize automation and reduce manual intervention.



openEuler 23.09 also integrates the following plugins:

- **Prometheus**: Hosts Prometheus monitoring components, automatically collects node-exporter monitoring data, and connects to the platform's alarm component.
- **Grafana**: Integrates the Grafana visualization platform to provide easy-to-use metric monitoring dashboard operations.

Application Scenarios

PilotGo is used to manage and monitor a large number of server clusters. It integrates service function plugins into a single unified platform, such as the MySQL database cluster, Redis data cache cluster, and Nginx gateway cluster.

CPDS

Cloud-native technologies empower organizations to build and run scalable applications in modern, dynamic environments. Containers deliver excellent flexibility and convenience, but further complicate IT monitoring and troubleshooting. Container Problem Detect System (CPDS) is developed to bolster the reliability and stability of containerized applications.

Feature Description

- **Cluster information collection**

  Node agents run on host machines to monitor key container services using systemd, initv, eBPF, and other technologies. Cross-NS agents are configured on nodes and containers in non-intrusive mode to track the application situation, resources, system function execution, and I/Os. The collected information covers the node network, kernel, and drive LVM.

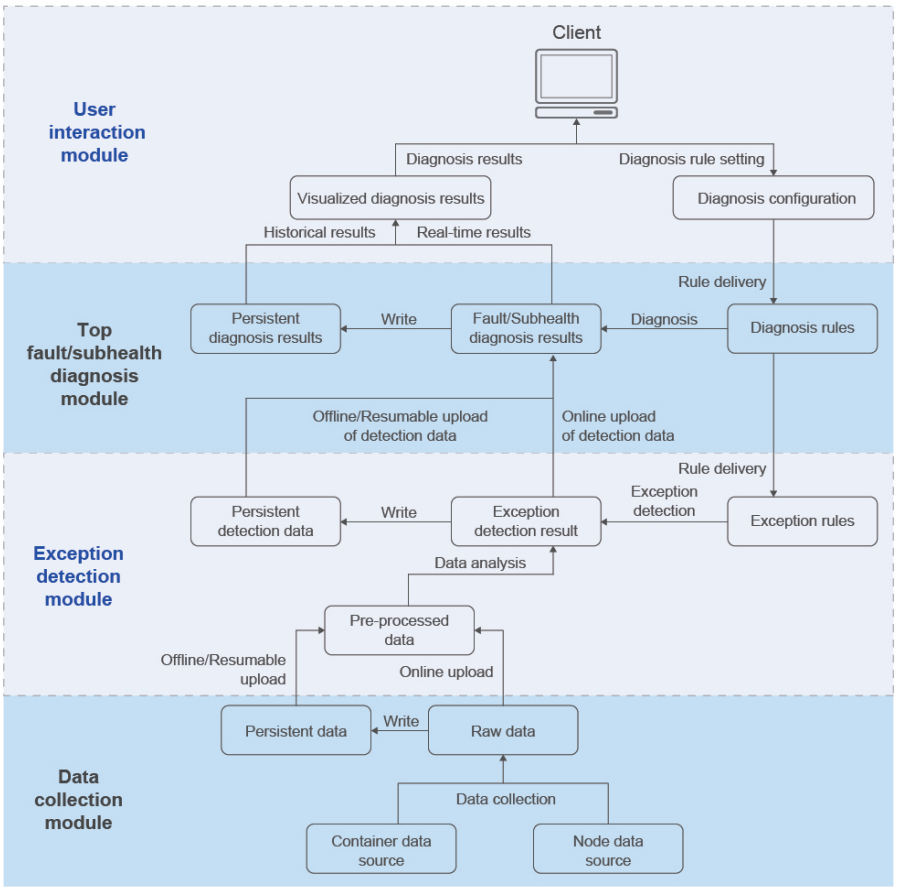
- **Cluster-level exception detection**

  Raw data is collected from each node to detect exceptions based on exception rules and extract key information, and the detection results and raw data are uploaded online and saved permanently.

- **Node and container health diagnosis**

  Nodes and service containers are diagnosed based on exception detection data, and the real-time and historical results are saved permanently and displayed on the UI.

The following figure shows the system architecture.



Application Scenarios

Cloud-native technologies deliver excellent latency and high-concurrency specifications needed in modern operations. Innovative technologies, like containers, are popular, offering a lightweight and high-elasticity computing carrier in artificial intelligence, big data, and edge computing.

Businesses and container clusters continue to expand, causing extra pressure on IT O&M systems. Service interruptions caused by software and hardware faults have become the major bottleneck in stability issues. Mainstream fault detection solutions for container clusters use inadequate techniques like cluster component status detection, service entry monitoring, and custom liveness probes, but these cannot detect or identify the causes of subhealth status issues, nor deliver fault diagnosis or execution policies, preventing key faults from being handled.

CPDS is an essential managing, monitoring, and troubleshooting tool for containerized applications and infrastructure. It monitors node and container situations, including resource usage and performance metrics, to guide troubleshooting and handling of potential faults, improving the stability and availability of the system, and ensuring reliable running of applications.

CPDS can detect the following faults.

No.	Fault Detection Item
1	Checks whether a container service is normal.
2	Checks whether a container node agent is normal.
3	Checks whether a container group is normal.
4	Checks whether the node health is normal.
5	Checks whether log collection is normal.
6	Checks whether the disk usage has reached 85% of the total capacity.
7	Detects network faults.
8	Detects kernel crashes.
9	Detects residual LVM disks.
10	Checks whether the CPU usage exceeds 85%.
11	Checks whether node monitoring is normal.
12	Detects container memory allocation failures.
13	Detects container memory application timeouts.
14	Detects container network response timeouts.
15	Detects slow disk read/write of containers.
16	Detects zombie subprocesses of containerized applications.
17	Detects subprocess and thread failures of containerized applications.

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 multi-user environments.

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.

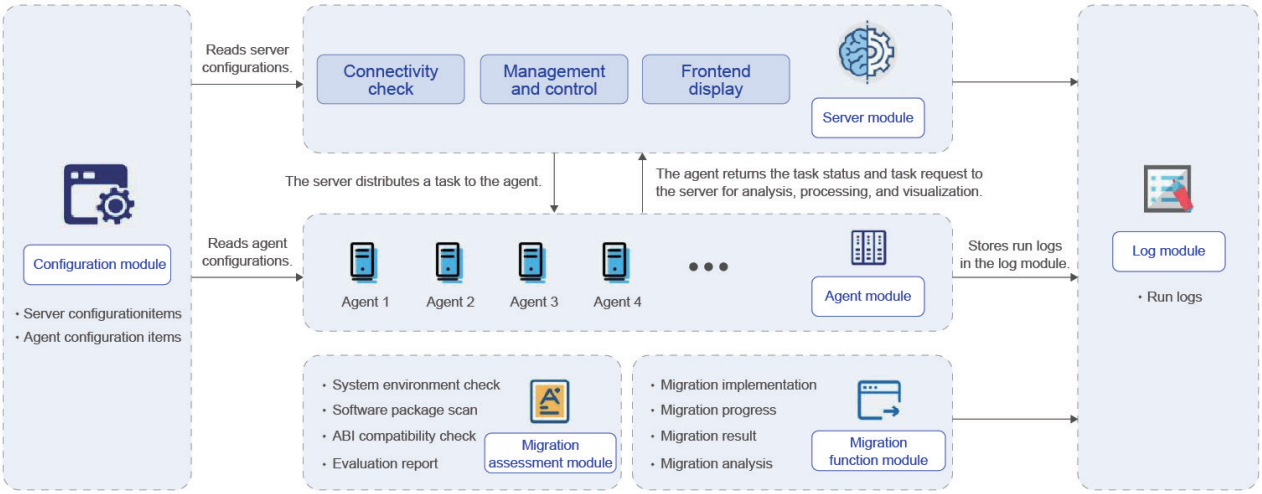
Application Scenarios

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.

migration-tools (UnionTech)

The 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

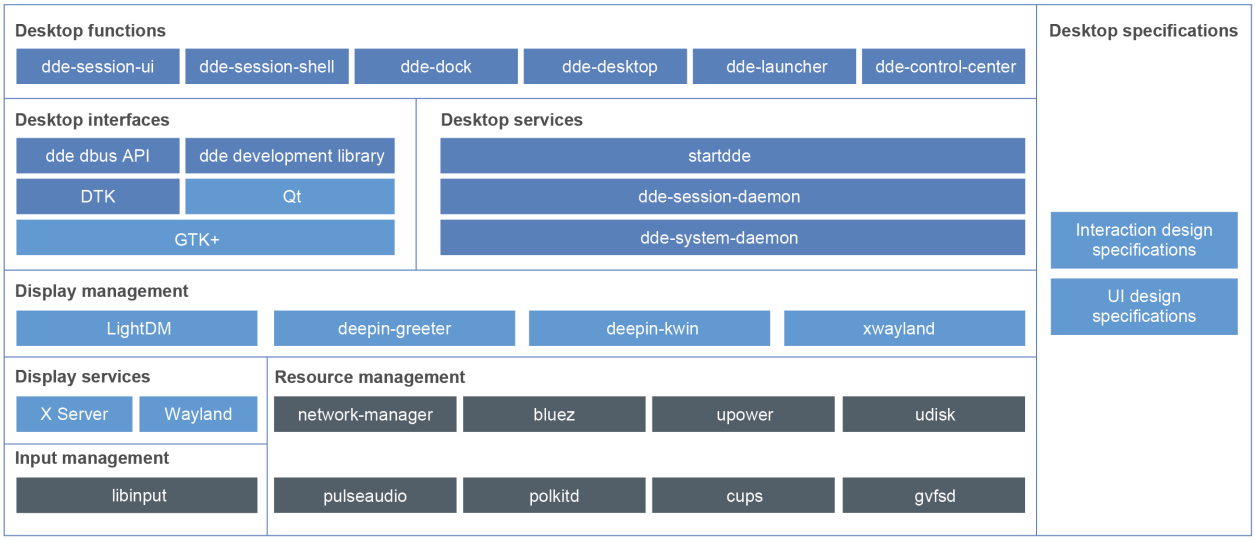
Finance, telecom, energy, and other industries often need to replace the OSs on existing hardware, such as the x86_64 architecture. migration-tools is a good choice for migrating applications and system components from other OSs to openEuler.

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.

RISC-V QEMU

openEuler provides base images for the RISC-V architecture, an open, free, and customizable instruction set architecture. RISC-V features simplified design with easy customization and portability, and has expanded its application scope and ecosystem in recent years, to the extent it is now widely used in embedded systems, high-performance computing (HPC), cloud computing, and academic research.

openEuler has been adapting to the RISC-V architecture since April 2020, and now openEuler 23.09 is officially released with support for the RISC-V architecture for upper-layer installation and verification applications. The 23.09 version is customizable, flexible, and secure, and provides a stable and reliable operating environment for RISC-V-based computing platforms, while also harnessing the RISC-V ecosystem.

Feature Description

openEuler 23.09 on RISC-V QEMU provides the following features:

- The OS kernel is updated to version 6.4.0, which is consistent with mainstream architectures.
- It features a stable base, including core functions such as processor and memory management, task scheduling, and device drivers, as well as common utilities.

Application Scenarios

openEuler on RISC-V QEMU is perfectly suited to the following scenarios:

- **Development and testing**: openEuler RISC-V images provide a stable and controllable base to build and verify applications on the RISC-V architecture using compatibility, performance, and functionality tests.
- **Research and education**: openEuler RISC-V images provide a platform for researchers and educators to explore and study the RISC-V architecture. It enables them to develop and conduct experiments on new algorithms, and teach courses on RISC-V-based systems and software development.
- **Software ecosystem development**: openEuler RISC-V images contribute to the growth of the RISC-V software ecosystem, helping developers and organizations create and optimize software solutions for the RISC-V platform.
- **System customization**: Custom operations of openEuler RISC-V images let users add or remove components, optimize configurations, and meet application requirements or use cases.
- **Exploration and innovation**: The openEuler RISC-V image fuels innovations and new ideas, and equips users to develop new applications, and contribute to the RISC-V ecosystem.

In a word, the openEuler RISC-V image is a base OS that promotes development, research, and innovation for application, system, and ecosystem growth in the RISC-V community. In the future, openEuler will incorporate more and open applications and features into the mainline version after testing and verification. Users can download the preview version to try more functions.

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.

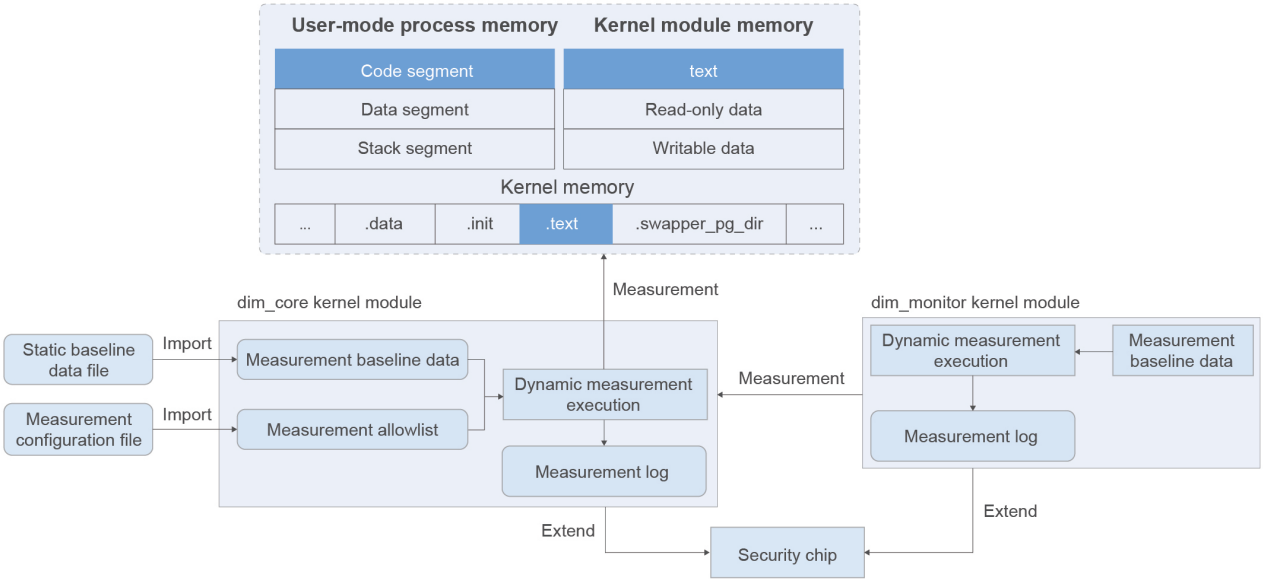
Feature Description

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.

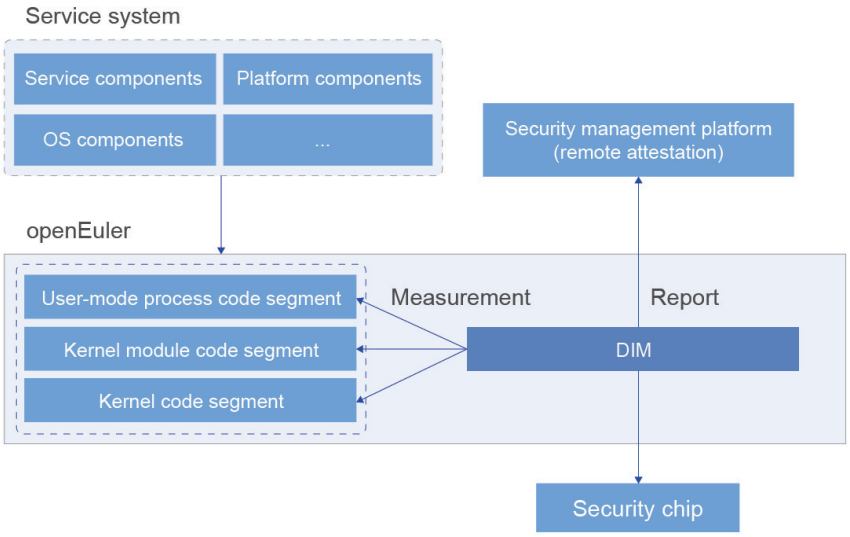
DIM consists of two software packages: dim_tools and dim.

- 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:



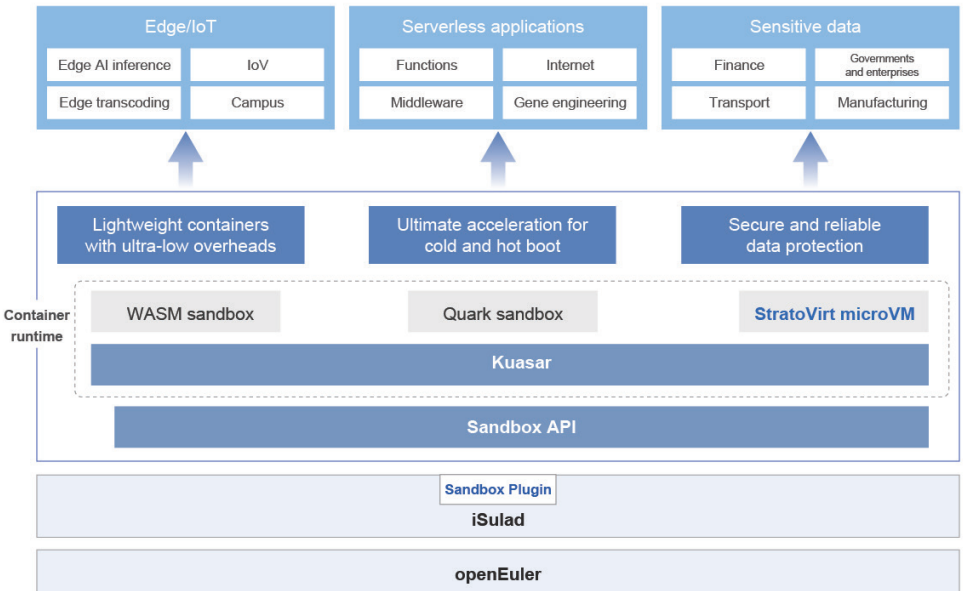
DIM can be used to set dynamic measurement for key program data, after which it sends results to the security management platform, which can inform users of the security situation. For high security requirements, users can connect to the trusted computing mechanism for remote attestation, to confirm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

Kuasar

Kuasar is a container runtime that provides a unified management platform for multiple types of sandboxes. It supports mainstream sandbox isolation technologies like the kernel-based native container sandbox, the microVM sandbox based on lightweight virtualization, the application kernel sandbox based on process-level virtualization, and the WebAssembly sandbox.

openEuler uses the Kuasar container runtime alongside the iSulad container engine and StratoVirt virtualization engine to build lightweight full-stack secure cloud containers. It delivers key competitiveness of ultra-low overhead and ultra-fast startup.

Feature Description



Kuasar makes the sandbox a first-class citizen in the container runtime interface (CRI) based entirely on the sandbox API. It provides native support for PodSandbox in the Kubernetes CRI, and supports multiple sandbox forms, including the user-mode Quark, microVM, and WASM language runtime, meeting diverse cloud sandbox requirements.

Kuasar 0.1.0 supports the StratoVirt lightweight VM sandbox and StratoVirt secure container instances created from Kubernetes+iSulad.

It provides the following features:

- Compatible with the Kubernetes ecosystem when the iSulad container engine connects to the Kuasar container.
- Provides secure container sandboxes based on the StratoVirt lightweight VM sandbox.
- Delivers precise resource control using StratoVirt containers.

Constraints:

Currently, Kuasar 0.1.0 supports only APIs related to sandbox and container lifecycle in Kubernetes CRI v1. The following lists the supported APIs:

rpcVersion(VersionRequest)returns(VersionResponse){}

rpcRunPodSandbox(RunPodSandboxRequest)returns(RunPodSandboxResponse){}

rpcStopPodSandbox(StopPodSandboxRequest)returns(StopPodSandboxResponse){}

rpcRemovePodSandbox(RemovePodSandboxRequest)returns(RemovePodSandboxResponse){}

rpcPodSandboxStatus(PodSandboxStatusRequest)returns(PodSandboxStatusResponse){}

rpcListPodSandbox(ListPodSandboxRequest)returns(ListPodSandboxResponse){}

rpcCreateContainer(CreateContainerRequest)returns(CreateContainerResponse){}

rpcStartContainer(StartContainerRequest)returns(StartContainerResponse){}

rpcStopContainer(StopContainerRequest)returns(StopContainerResponse){}

rpcRemoveContainer(RemoveContainerRequest)returns(RemoveContainerResponse){}

rpcListContainers(ListContainersRequest)returns(ListContainersResponse){}

rpcContainerStatus(ContainerStatusRequest)returns(ContainerStatusResponse){}

rpcExec(ExecRequest)returns(ExecResponse){}

rpcStatus(StatusRequest)returns(StatusResponse){}

Application Scenarios

Single-node multi-tenant sharing

- Serverless services are provided as containers, which consume fewer resources, causing density containers on a single node and fast startup.
- Multiple containers from different tenants can be deployed on a physical machine, which requires strong isolation between containers.
- Achieves auto scaling for containers that need to be quickly started and destroyed.

Hybrid deployment of trusted and untrusted applications

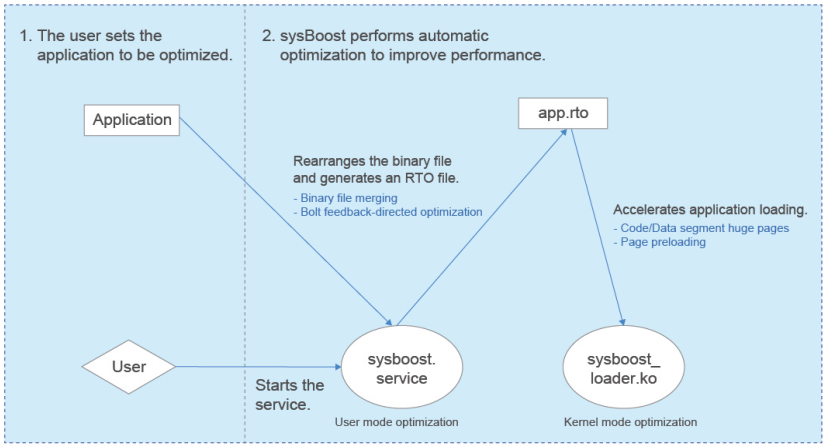
Trusted applications developed by users and untrusted applications obtained from third parties are deployed on the same physical machine. To ensure that third-party applications do not access or attack other applications, containers must be isolated.

sysBoost

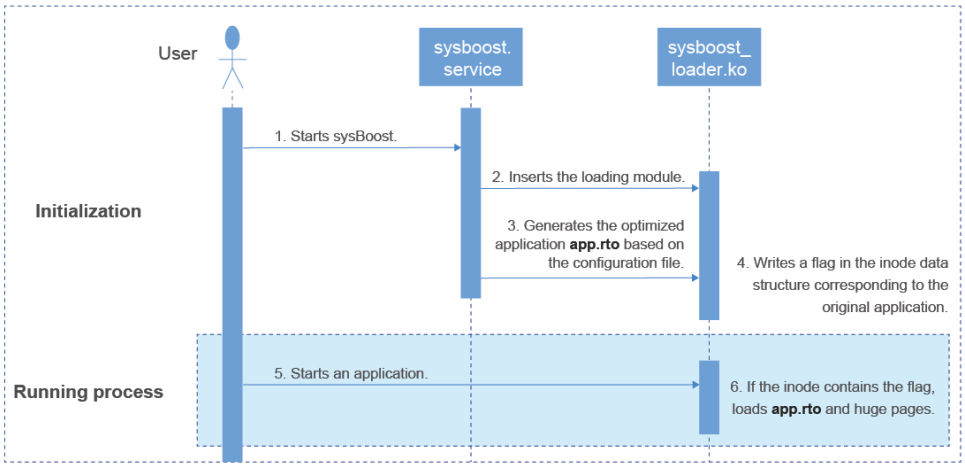
The sysBoost tool is used to optimize the application microarchitectures, covering assembly instructions, code layout, data layout, memory huge pages, and system calls.

Feature Description

Deployment view



Process design



- **Binary file merging**

  Applications and their dependent dynamic libraries are merged into one binary file, and segment rearrangement is performed (only full static merging is supported). Multiple discrete code or data segments are merged to improve application performance.

  In the future, the following two methods will be provided:

  » Merging dynamic libraries for runtimes

  » Partial merging. For example, the LGPL open source libraries cannot be merged into binary files of closed source products.

- **sysBoost daemon**

  To boost performance, sysBoost automatically optimizes binary files and registers with systemd, so that systemd starts the sysBoost daemon after the system is started.

  Then, the sysBoost daemon reads the configuration file to obtain the target binary files and corresponding methods, then optimizes the binary files to meet user requirements, before finally storing the optimized binary files in RTO files.

  sysBoost now supports only Bash optimization, which is enabled by default. More applications will be available in the future.

- **RTO binary file loading module in the kernel**

  For operations such as application update and abnormal rollback, the original file cannot be replaced with the optimized RTO file. Instead, a binary loading module is added to automatically load the optimized binary file when the kernel loads binary files.

  When loading binary files, the kernel matches a proper processing function according to the binary data types. For example, the binary files contained in openEuler are in ELF format, and in such cases, the ELF loading function is used. sysBoost registers the ELF loading function with the kernel. This function checks whether the inodes of the target binary files contain a flag written by sysBoost, and if so, sysBoost loads the optimized RTO file; otherwise, binary files are loaded according to the original process.

  The loading mechanism uses a flag in the inode to identify whether an application is optimized by sysBoost. When sysBoost generates an RTO file, a device file is added as the flag through this kernel module. In user mode, the ioctl system call is used to instruct the kernel module to set the flag.

- **Huge page preloading for binary code or data segments**

  When the user-mode page table is mapped to the physical memory, huge page (2 MB) mapping can improve performance. However, openEuler does not support huge page mapping of file pages. sysBoost was introduced to provide the huge page preloading function. Once binary optimization is complete, sysBoost immediately loads the content to the kernel as a huge page. When an application is started, sysBoost maps the pre-loaded content to the user-mode page table in batches to reduce page faults and memory access delay of the application, which improves application startup speed and running stability.

Application Scenarios

sysBoost is equipped with system microarchitecture optimization that is suited to user-mode programs, such as those that frequently invoke the dynamic library **/itlbmiss**.

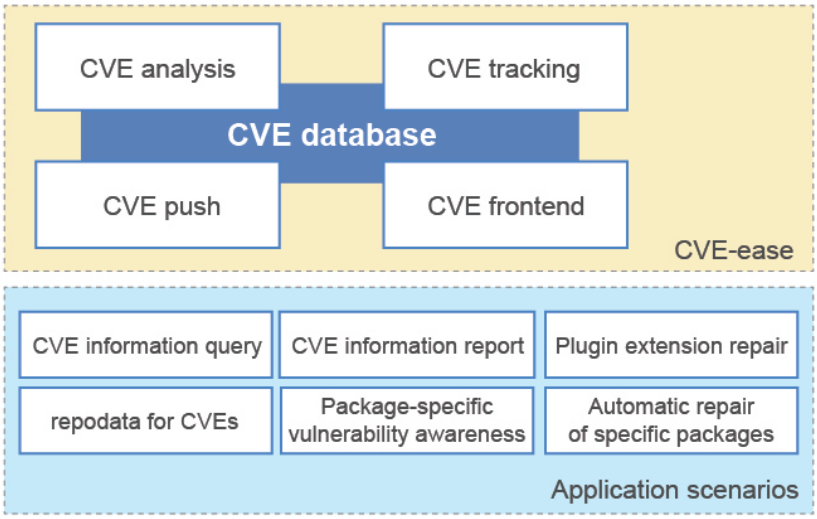
CVE-ease

CVE-ease is an innovative Common Vulnerabilities and Exposures (CVE) platform developed by China Telecom e-Cloud Technology Co., Ltd. It collects various CVE information released by multiple security platforms and notifies users through multiple channels, such as email, WeCom, and DingTalk.

Feature Description

The CVE-ease platform is designed to help users quickly learn about and handle vulnerabilities in their systems. In addition to improving system security and stability, the platform displays CVE details such as the vulnerability description, impact scope, and handling suggestions, and selects a fixing solution as required. CVE-ease has the following capabilities:

- repodata catering to the needs of operating system vendors
- motd login broadcast
- DNF plugin extension repair
- Automatic repair for specific packages
- Awareness of specific packages



Technical challenges: Our world is increasingly faced with complex computer security situations. The question of how to sort valid CVE information from huge volumes in real time and notify users in a timely manner must be answered to improve security assurance of the OS.

CVE-ease delivers the following features:

- Real-time tracking of CVEs on multiple platforms and integration into the CVE database.
- Extracts key inform ation from the collected CVE information and updates changed CVE information in real time.
- Automatically maintains and manages the CVE database.
- Queries historical CVE information based on various conditions in interactive mode.
- Reports historical CVE information in real time through WeCom, DingTalk, and email.

Application Scenarios

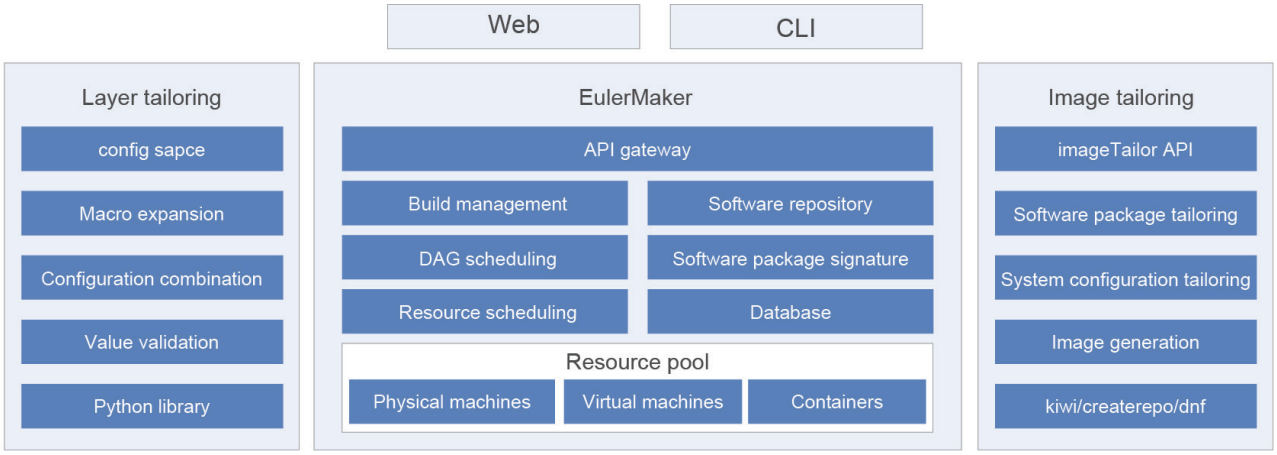
CVE-ease can be used to detect, manage, and fix OS security vulnerabilities.

Code repository: https://gitee.com/openeuler/cve-ease

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**: Customizes build projects by layer models to create patches, build and installation dependencies, and compilation options for software packages.
- **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.

Application Scenarios

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

Copyright Statement 07

08 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.

09 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


OpenEuler