openEuler 22.03 LTS SP4
Technical White Paper

Contents

1 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.

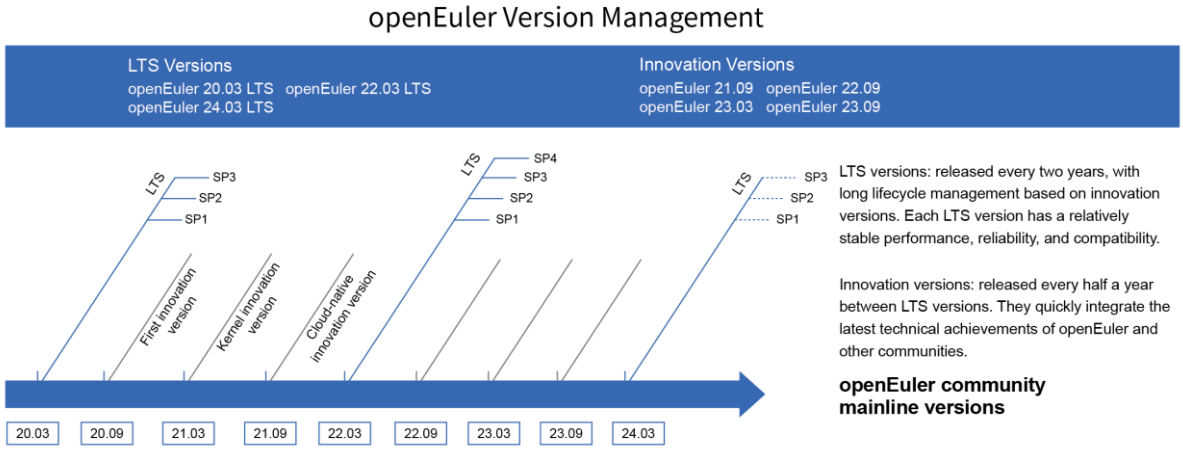
Later on September 30, 2023, the openEuler 23.09 innovation version was released based on Linux kernel 6.4. It provides a series of brand-new features to further enhance developer and user experience.

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 features for server, cloud native, edge computing, and embedded scenarios.

On May 30, 2024, openEuler 24.03 LTS was released. This version is built on Linux kernel 6.6 and brings new features for server, cloud, edge computing, AI, and embedded deployments to deliver enhanced developer and user experience.

Lately on June 30, 2024, openEuler 22.03 LTS SP4 was released. Designed to improve developer efficiency, it further extends features for server, cloud native, edge computing, and embedded scenarios.

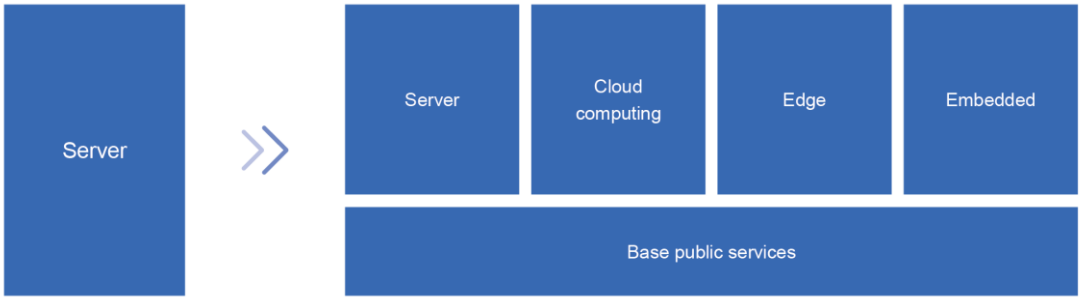


As an OS platform, openEuler releases an LTS version every two years. Each LTS version provides enhanced specifications and a secure, stable, and reliable OS for enterprise users.

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.

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: 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.

2 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 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 SP4 runs on Linux kernel 6.6 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 SP4 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:

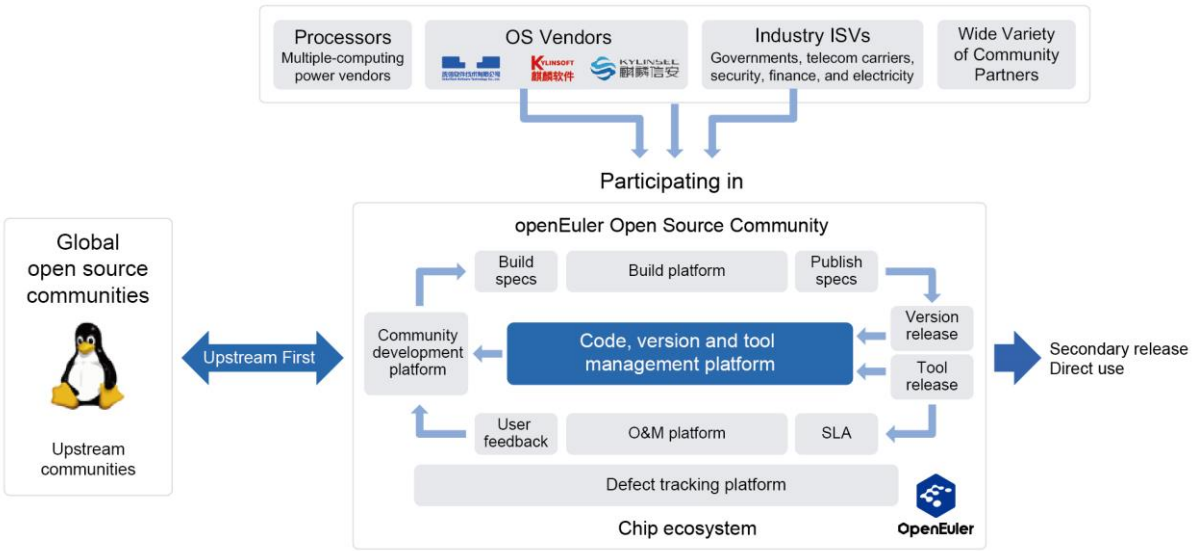
- **Edge computing**: openEuler 22.03 LTS SP4 Edge is released for edge computing scenarios. It integrates the KubeEdge+ edge-cloud collaboration framework to provide unified management, provisioning of edge and cloud applications, and other capabilities.
- **Embedded**: openEuler 24.03 LTS Embedded 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 processor architectures, and a wide range of CPU chips, such as Loongson 3 series, Zhaoxin KaiXian and KaiSheng, Intel Sapphire Rapids and Emerald 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 processor architectures:

Hardware Type	x86	Arm	LoongArch	SW64	RISC-V
CPU	Intel, AMD, Zhaoxin, Hygon	Kunpeng, Phytium	Loongson	ShenWei	Sophgo, T-Head

openEuler supports the following servers:

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

openEuler supports the following cards:

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

For a full hardware list, visit https://www.openeuler.org/en/compatibility/.

3 Operating Environments

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 SP4 support the following software packages:

- libvirt-6.2.0-63.oe2203sp4
- libvirt-client-6.2.0-63.oe2203sp4
- libvirt-daemon-6.2.0-63.oe2203sp4
- qemu-6.2.0-91.oe2203sp4
- qemu-img-6.2.0-91.oe2203sp4

openEuler 22.03 LTS SP4 is compatible with the following guest OSs for VMs:

Guest OS	Architecture
CentOS 6	x86_64
CentOS 7	AArch64
CentOS 7	x86_64
CentOS 8	AArch64
CentOS 8	x86_64
Windows Server 2016	x86_64
Windows Server 2019	x86_64

Item	Configuration Requirement
Architecture	AArch64, x86_64
CPU	≥ 2 CPUs

Item	Configuration Requirement
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

4 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.

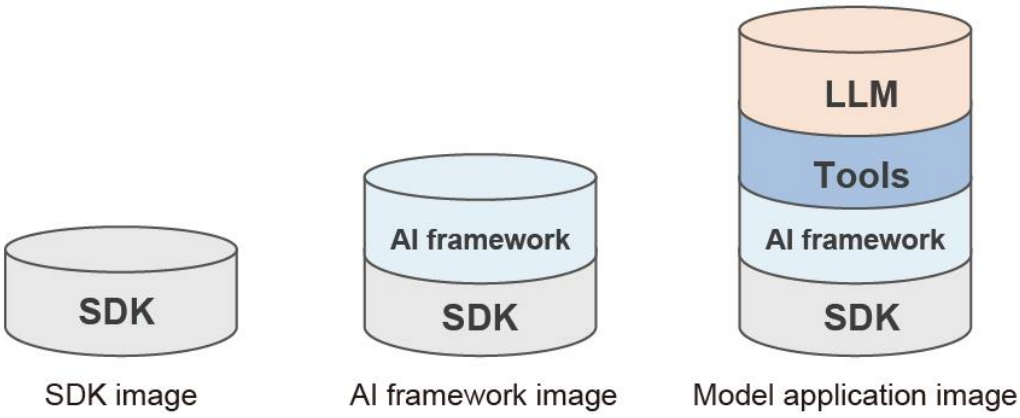
OS for AI

openEuler offers an efficient development and runtime environment that containerizes software stacks of AI platforms with out-of-the-box availability. It also provides various AI frameworks to facilitate AI development.

Feature Description

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.

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 details, 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.

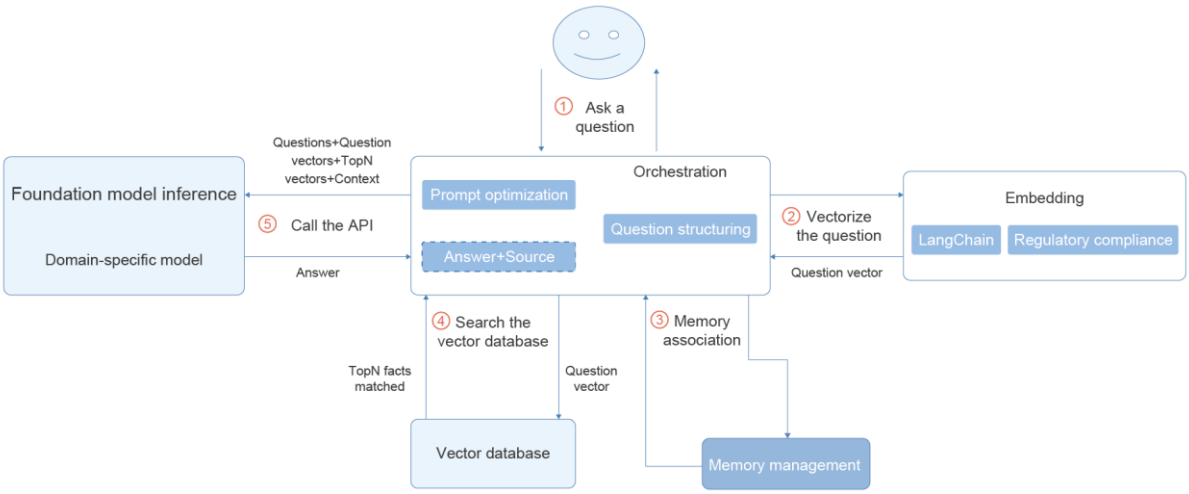
AI for OS

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.

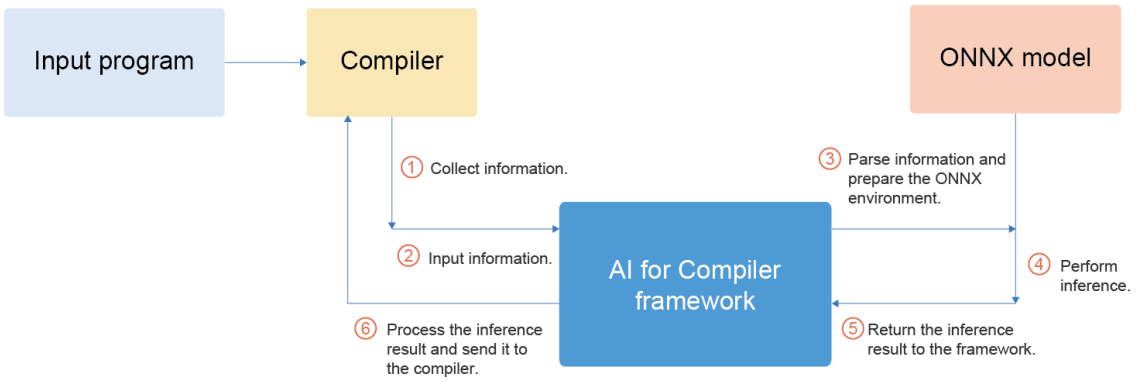
For details, see EulerCopilot Intelligent Q&A Service User Guide.

AI for Compiler

The AI for Compiler framework provides APIs that enable AI capabilities on an openEuler compiler to tune compilation options and optimize memory layout.

Feature Description

AI for Compiler receives information as the input, parses the information, prepares the Open Neural Network Exchange (ONNX) environment, and sends an inference request to the ONNX model. The ONNX model performs inference and returns the inference result to the AI for Compiler framework, which then parses and processes the inference result and sends the processing result to the compiler. This enables the compiler to implement optimization.



Application Scenarios

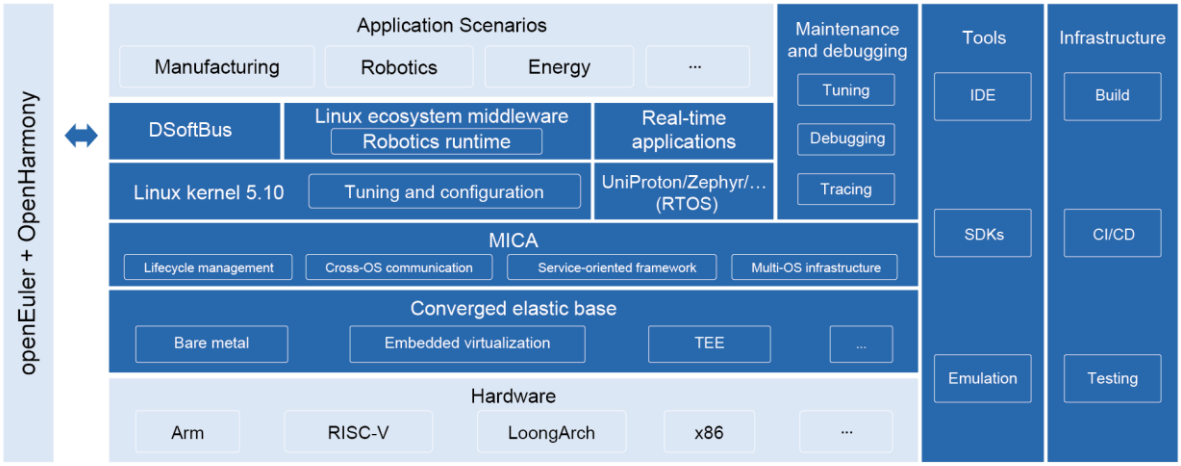
- **For openEuler users**: AI for Compiler helps better tune program performance.
- **For openEuler developers**: The compiler and model are decoupled, enabling developers to train models with ease and improve compiler optimization.

openEuler Embedded

openEuler 22.03 LTS SP4 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 SP4 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 SP4 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 22.03 LTS SP4 Embedded supports AArch64 and x86-64 chip architectures. In the future it 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 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 600 common embedded software packages can be built using openEuler.
- **ROS runtime**: openEuler 22.03 LTS SP4 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.

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 processor architectures like Cortex-M, AArch64, and x86_64, and supports M4, RK3568, RK3588, x86_64, Hi3093, and Raspberry Pi 4B.
- Connects with openEuler Embedded Linux on Raspberry Pi 4B, Hi3093, RK3588, 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

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

5 Kernel Innovations

openEuler 22.03 LTS SP4 runs on Linux kernel 5.10 and inherits the competitive advantages of community versions and innovative features released in the openEuler community.

Topology

- **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 NUMA-aware scheduling**: As the Moore's Law slows down, there is little benefit to scaling up hardware. This gives a platform for scale-out, which answers issues of resource locality. For the NUMA architecture, cross-NUMA memory access typically has much longer latency than that for local memory. Linux distributions focus on throughput and load balancing, which may break the optimal state of global cross-NUMA memory access. This feature enables affinity-based load balancing, overcoming typical constraints in the Linux scheduling model. If there is no resource bottleneck, this feature packs tasks with affinity relationships to reduce cross-NUMA memory access.
  - Affinity awareness: detects network and memory affinity between threads thanks to software and hardware collaboration.
  - Resource bottleneck detection: detects NUMA resource bottlenecks.
  - Optimized core selection and load balancing: performs scheduling within one or across NUMA nodes based on a programmable scheduling framework. If there is no resource bottleneck, the feature schedules tasks with affinity relationships to the same NUMA node.

Scheduling

- **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 light.
  - 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 (QoS).
  - The timer collects load and bandwidth information and detects the memory access bandwidth to avoid 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).
  - A task labeling mechanism attaches a QoS priority for each userspace process, where tasks with different priorities are bound to different CPU sets for scheduling. Power consumption policies can be configured for each QoS priority through the cpufreq module to save energy without impacting service QoS.

Memory

- **KSM at the cgroup level**: If the memory.ksm interface is used to enable kernel same-page merging (KSM) for processes, only processes in a memory control group (memcg) are configured, and this interface needs to be called again to enable KSM for processes newly added to the memcg. This feature simplifies KSM enablement with support for KSM status bits of memcgs. After **1** is written to the memory.ksm interface, KSM is automatically enabled for processes that are newly added to the memcg.
- **Non-KASLR memory range configuration**: The **CONFIG_NOKASLR_MEM_RANGE** option is added to the kernel command line. Up to four physical memory ranges can be configured to prevent kernel address space layout randomization (KASLR) on them during

system boot. For the AArch64 and x86_64 architectures, users can specify the non-KASLR memory ranges in boot parameters to avoid conflicts with the memory reserved for kdump and solve the problem of insufficient memory caused by excessive memory randomization.

- **Automatic memcg and blkcg binding for buffer I/O rate limiting**: A Meituan contribution to the openEuler community. Currently, subsystems in cgroup v1 are independent. For example, a blkio cgroup (blkcg) cannot detect the resource usage of a memcg when limiting resources. In this case, the blkcg and memcg needs to be manually bound. This feature automatically maps the blkcg and memcg of the same process in kernel space. Users can enable this feature using the **CONFIG_CGROUP_V1_BIND_BLKCG_MEMCG** option during compilation or setting **sysctl_bind_memcg_blkcg_enable** at run time.

Virtualization

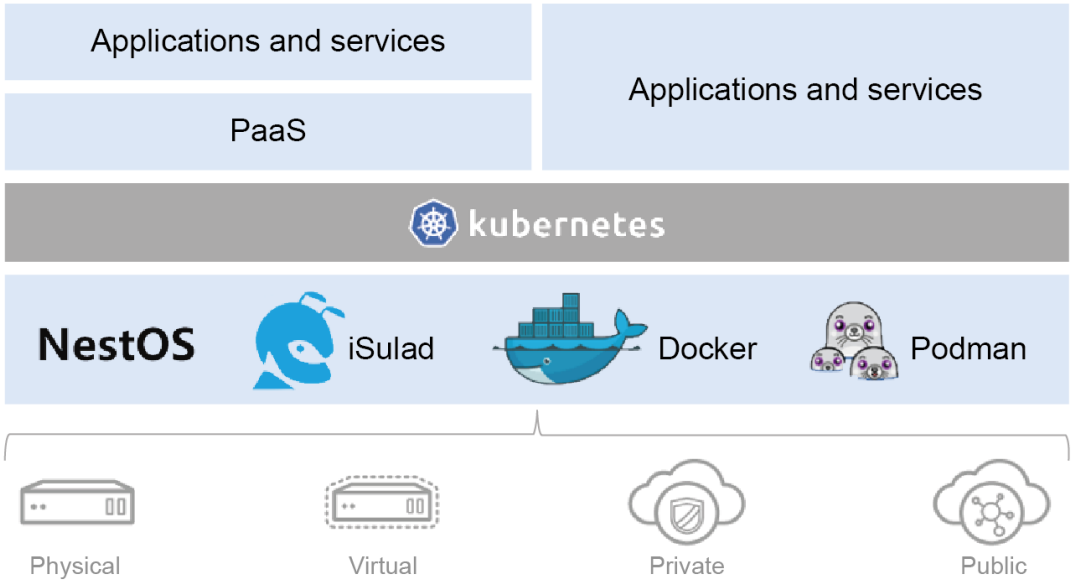
- **KVM TDP MMU**: An Intel contribution to the openEuler community. In Linux kernel 5.10 and later, KVM can scale to match demand for memory virtualization. Compared with the traditional KVM memory management unit (MMU), the two-dimensional paging MMU, or TDP MMU, offers more efficient handling of concurrent 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 supports KVM TDP MMU for Linux kernel 5.10 to 5.15. The SP4 version provides more KVM MMU optimizations and bug fixes for Linux kernel 5.15 to 6.2, including reduced and optimized KVM MMU unloading, better KVM MMU page fault handling, and optimized memslot updates.
- **Dirty ring**: A Kylinsoft contribution to the openEuler community. It optimizes the dirty bitmap of previous kernels that collects statistics on memory pages accessed by vCPUs at a KVM memory slot granularity. The dirty ring is per-vCPU or even per-memory page. It can be used in the user space and has good scalability for VMs with large memory specifications.

6 Cloud Base

NestOS

NestOS is a community cloud OS that uses nestos-assembler for quick integration and build. It runs rpm-ostree and Ignition tools over a dual rootfs and atomic update design, and enables easy cluster setup in large-scale containerized environments. Compatible with Kubernetes and OpenStack, NestOS also reduces container overheads.

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

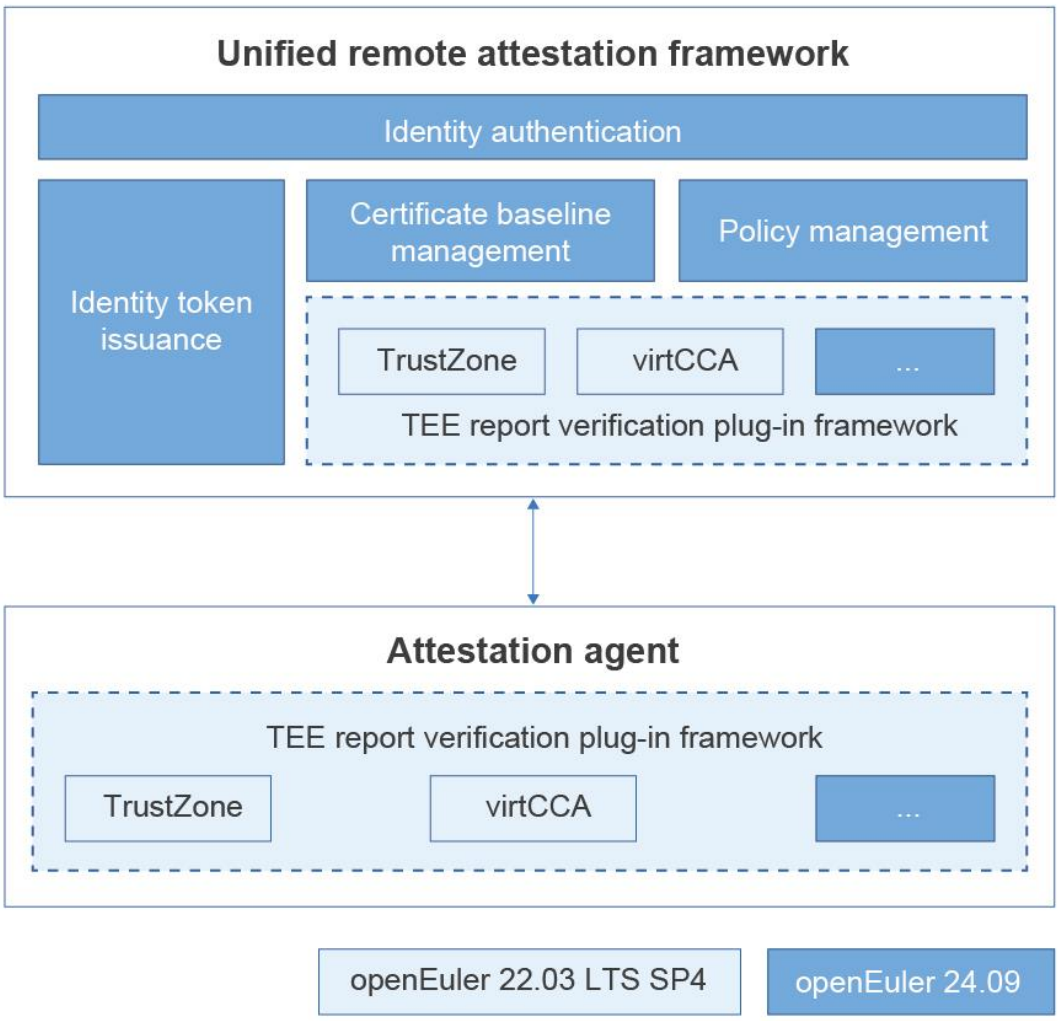
openEuler 24.03 LTS introduces the NestOS-Kubernetes-Deployer tool to resolve problems such as inconsistent and repeated O&M operations across stacks and platforms. These problems are typically caused by decoupling of containers from underlying environments when using container and container orchestration technologies for rollout and O&M. NestOS is developed for containerized cloud applications and ensures consistency between services and the base OS.

7 Enhanced Features

TEE Plug-in Framework: Remote Attestation for Confidential Computing

As confidential computing becomes a more viable option thanks to advances in hardware, new developments must address the differences in root of trust and remote attestation, which hinder trust between Trusted Execution Environments (TEEs). A remote attestation framework, comprising an attestation agent and TEE plug-in framework among other components, provide a unified verification process for diverse TEE attestation reports.

Feature Description



- The attestation agent can retrieve Kunpeng TrustZone and virtCCA attestation reports, which also enables runtime flexibility.
- When no attestation service is available, the framework can be integrated into the attestation agent for point-to-point verification.

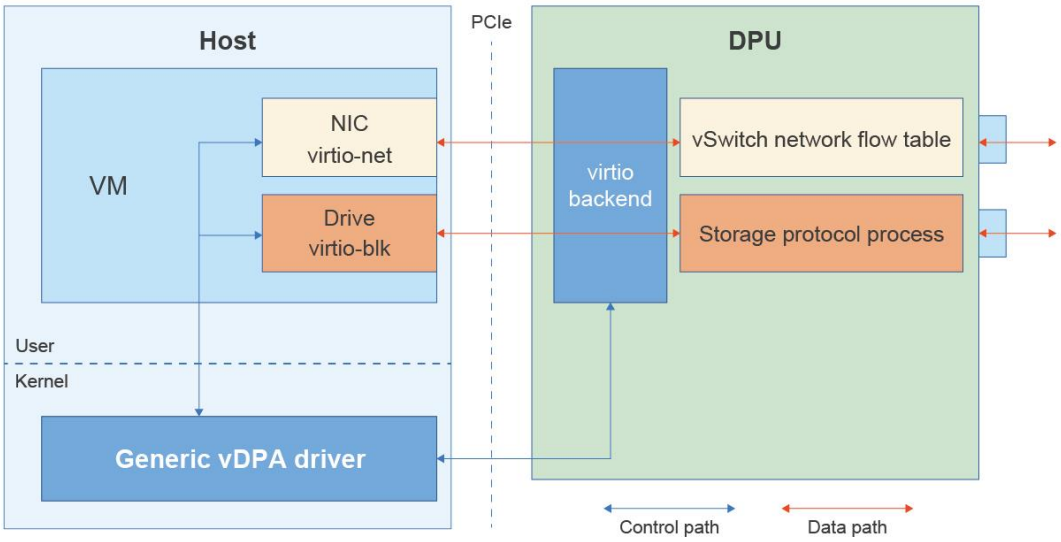
Application Scenarios

Remote attestation involves a challenger and a secure application, both of which equipped with a remote attestation agent. The challenger initiates a challenge to a secure application, which obtains a TEE remote attestation report and returns it to the challenger. The challenger then verifies the authenticity of the attestation report.

Enhanced vDPA: DPU Drive Support and Live Migration for VMs with vDPA NICs and Drives

The kernel-mode vHost Data Path Acceleration (vDPA) framework provides device virtualization as performant as passthrough and supports VM live migration across hardware vendors. openEuler 22.03 LTS SP4 supports a wide range of DPUs and exposes them to network and storage devices of frontend hosts. These DPUs can be integrated into the generic vDPA framework and support live migration.

Feature Description



openEuler 22.03 LTS SP4 inherits the support for vDPA NICs and vDPA NIC passthrough live migration from openEuler 22.03 LTS SP1/SP3. This support extends to DPU storage devices to enable vDPA drive passthrough live migration.

New features are as follows:

- vDPA drives configurable as VM data drives
- Hot-swappable vDPA drives on VMs
- Live migration for VMs equipped with vDPA NICs and drives

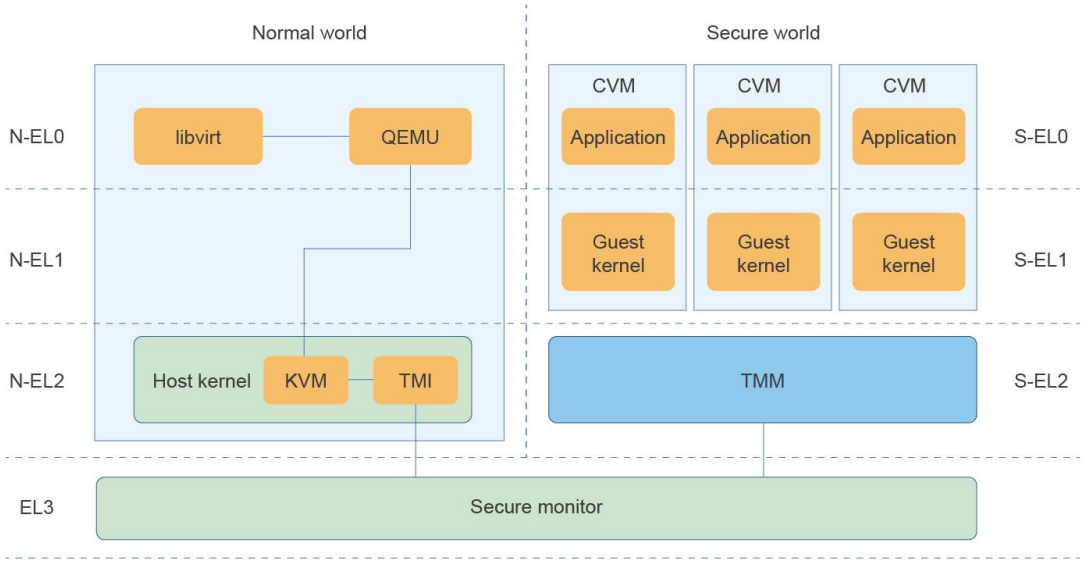
Application Scenarios

Generic vDPA frameworks are used for cloud computing virtualization where tasks are offloaded to DPUs. openEuler supports various virtio devices (including drives and NICs) through the generic vDPA framework and reduces the reliance on the Data Plane Development Kit (DPDK) for user-mode vDPA devices. This minimizes resource consumption on frontend hosts and simplifies the control plane architecture in full offload scenarios.

virtCCA-based VMs

virtCCA-based VMs, built on the Secure EL2 (S-EL2) of the Kunpeng 920 high-performance edition, allow regular VM software stacks to run securely within TEEs.

Feature Description



Based on the standard interface of the Arm Confidential Compute Architecture (CCA), openEuler builds a TEE virtualization management module upon the TrustZone firmware. This module supports memory isolation between confidential VMs, context and lifecycle management, and page table management, and enables seamless application migration to TEEs.

Technical Constraints

- Confidential VMs measure memory information such as the guest kernel each time they start. Therefore, they do not support the reboot capability.
- For security purposes, hyper-threading needs to be disabled in the host BIOS.

Application Scenarios

virtCCA-based VMs are suitable for general applications, clouds, data privacy protection, and other workloads.

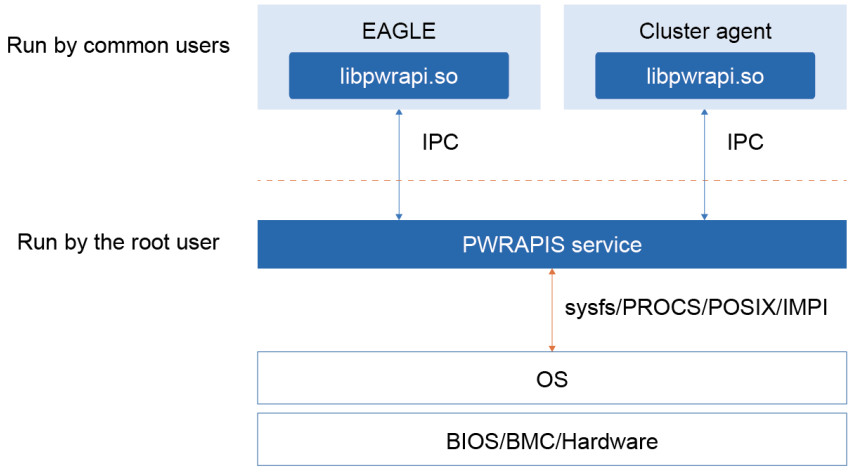
PowerAPI

PowerAPI is a collection of standardized APIs that monitor, adjust, and optimize system power consumption, helping boost system efficiency and reliability and slash power costs.

Feature Description

PowerAPI consists of the pwrapis program, the **libpwrapi.so** dynamic library, and OS-based APIs to improve upper-layer management on power consumption. It makes the following achievements:

- Masks differences between underlying APIs such as SCI, sysfs, PROCS, and IMPI, as well as between platforms.
- Eliminates the need for root privileges in power consumption management. For example, the cluster agent does not need to be run by the **root** user.



PowerAPI provides the following features:

- Data collection on system power consumption, CPU usage, and LLC misses.
- CPU frequency scaling and CPU sleep management.
- Watt and domain-based scheduling.

Application Scenarios

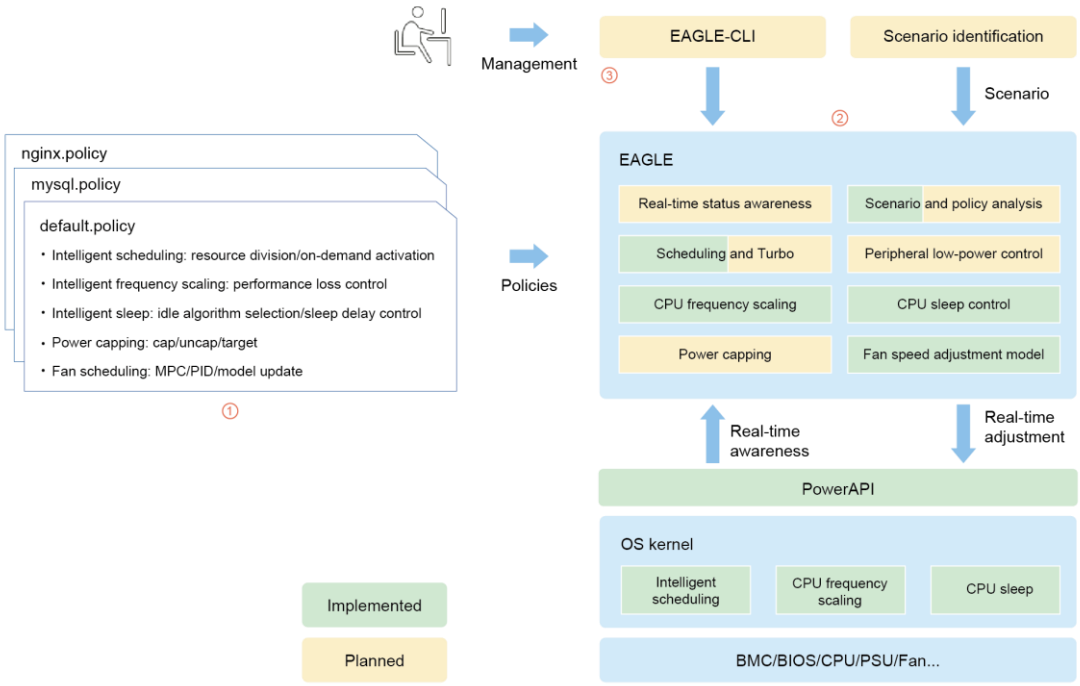
PowerAPI can help companies or users who want to reduce costs while improving the efficiency of their servers. With cluster scheduling, for example, the cluster agent can integrate PowerAPI to limit node power consumption based on the load.

EAGLE

Energy Aware intelliGent scheduler (EAGLE) is a service used to dynamically tune energy efficiency. It manages server power consumption by implementing custom policies, which improves energy efficiency and reduces costs.

Feature Description

EAGLE is developed on PowerAPI and implements energy tuning policies.



1. **Tuning policies**: Custom policies are supported to improve performance for specific services.
2. **Scenario-aware intelligent power consumption adjustment**: EAGLE uses PowerAPI to monitor power consumption and achieve optimal energy efficiency. To ensure service performance, it supports intelligent scheduling, frequency scaling, Turbo, and sleep control, and application priorities and other specifications (intensive CPU usage, intensive memory access, and latency sensitiveness).
   - Intelligent scheduling based on energy efficiency domains achieves optimal efficiency by domain division, on-demand resource activation, balanced scheduling, and intelligent Turbo.
   - Frequency scaling minimizes performance loss caused by CPU frequency scaling and detects service performance changes in real time.
   - Accurate intelligent sleep monitors service characteristics and formulates CPU and peripheral sleep policies that are suited to each service.
   - Multi-dimensional and multi-device power capping ensures optimal energy efficiency while ensuring the performance of high-priority applications.
   - Intelligent prediction of heat changes enables fans to respond in advance, improving heat reliability of the server and reducing power consumption.
3. **Human-machine interaction**: The Eagle-CLI monitors the system in real time and supports custom tuning policies.

Application Scenarios

Designed to help users who have requirements on server energy efficiency, EAGLE can improve environments or services that have low service loads (generally under 30%), with better efficiency and reduced costs.

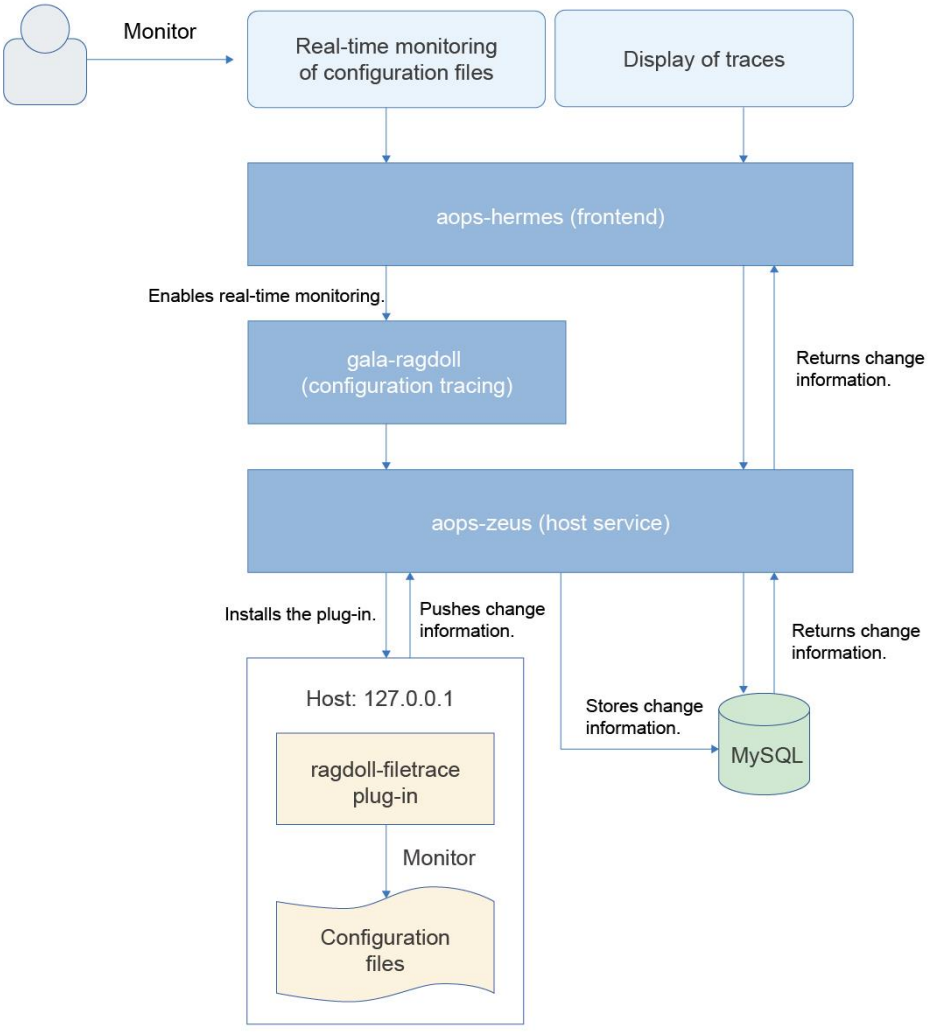
gala-ragdoll

gala-ragdoll monitors and records modifications to configuration files and help O&M personnel quickly trace these changes.

Feature Description

The ragdoll-filetrace plug-in of gala-ragdoll monitors configuration files so that, when a file is modified, information about the system call that operates the file is captured and saved to a MySQL database table. gala-ragdoll then provides database data to aops-hermes through APIs and displays the change information, such as the modification time, modification method, and process name, on the UI.

gala-ragdoll supports common text editing utilities, such as **vi**, **vim**, **sed**, **echo**, and **mv**.



Application Scenarios

gala-ragdoll enables you to trace any modifications to configuration files, which are typically difficult to monitor and trace. In this case, you can enable monitoring for the configuration domain,

so that any changes in the process and its child processes are traced and such data is stored in the database.

8 Copyright Statement

9 Trademark

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