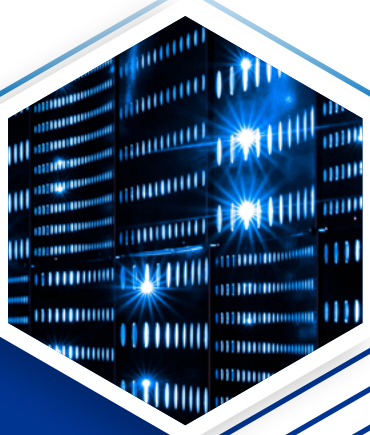
openEuler 23.03

Technical White Paper

CONTENTS



01 /
Introduction

openEuler has evolved from a simple server operating system (OS) into a digital infrastructure OS that fits into any server, cloud computing, edge computing, and embedded deployment. It provides a secure, stable, and easy-to-use open source OS that 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 a digital infrastructure OS for all scenarios.

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.

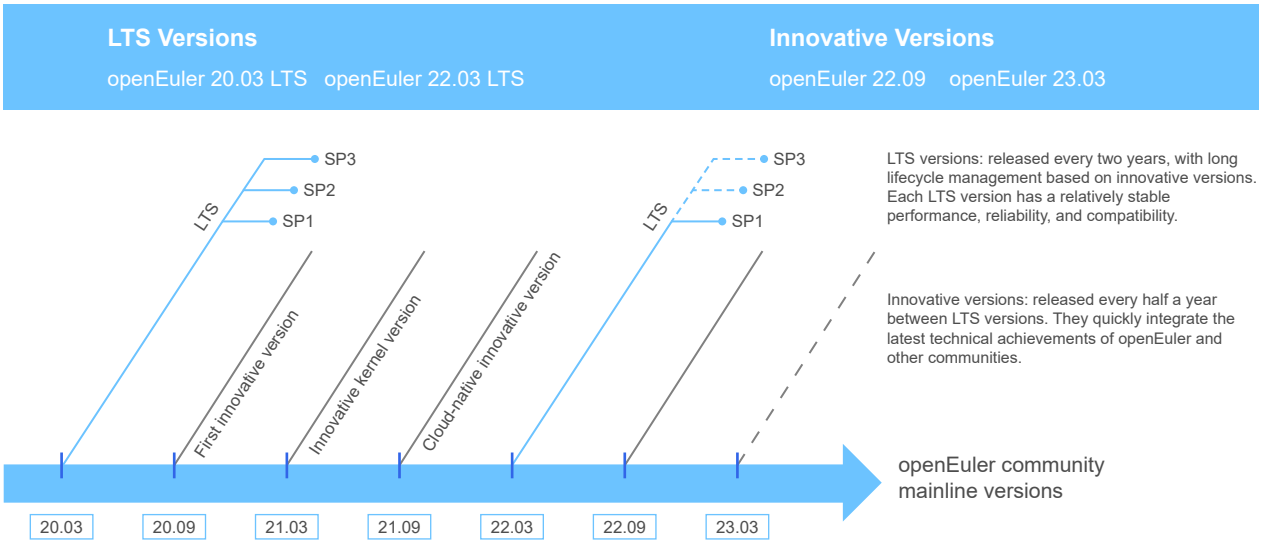
On March 31, 2021, the innovative kernel version openEuler 21.03 was released. This version is enhanced in line with Linux kernel 5.10 and also incorporates multiple new features, such as live kernel upgrade and tiered memory expansion. These features improve multi-core performance and deliver the computing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

openEuler Version Management

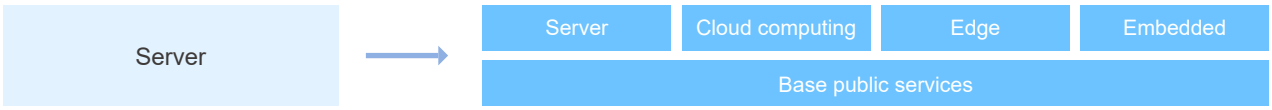


As an OS platform, openEuler releases an LTS version every two years. Each LTS version provides enhanced specifications and a secure, stable, and reliable OS for enterprise users.

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

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

Innovative Platform for All Scenarios



The openEuler 23.03 innovative version supports x86 and Arm processor architectures, and as part of the commitment for continuous improvement of diversified computing power, the future LTS version will support SW64, LoongArch, RISC-V, and PowerPC.

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 23.03 Edge and openEuler 23.03 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.

Continuous Contribution to the Linux Kernel

As a major contributor to the Linux kernel, the kernel development team is responsible for enhancing the processor architectures, Advanced Configuration and Power Interface (ACPI), memory management, file systems, media, kernel documents, bug fixes, and code rebuilds.

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 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.

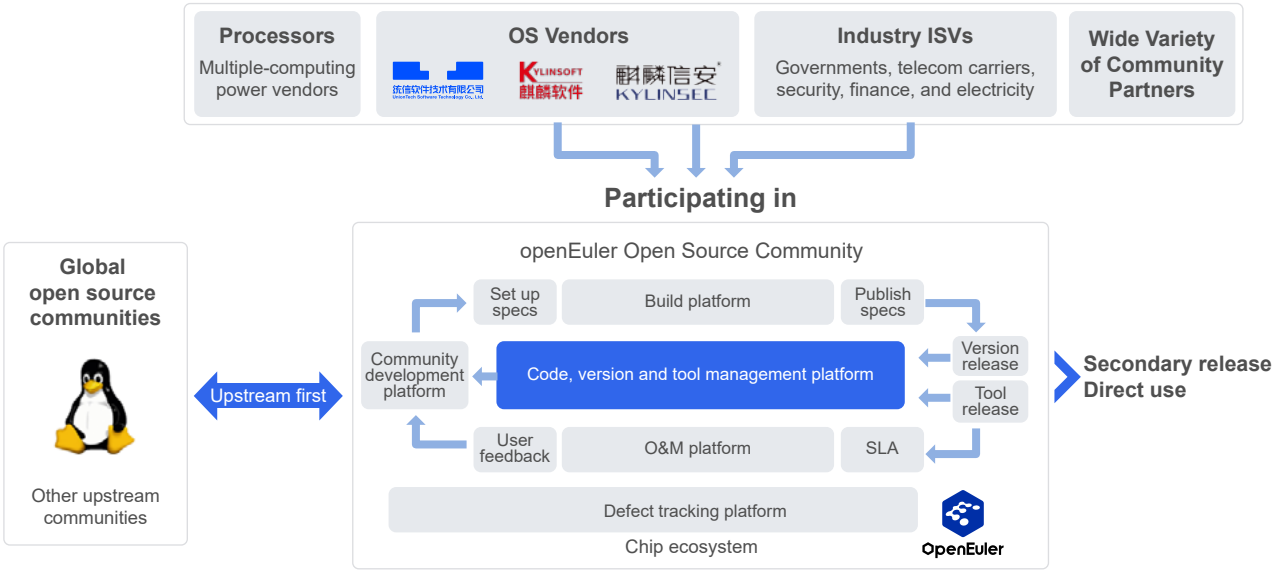
In the future, the openEuler open source community will continue to innovate, aiming to promote the ecosystem and consolidate the digital infrastructure.

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.



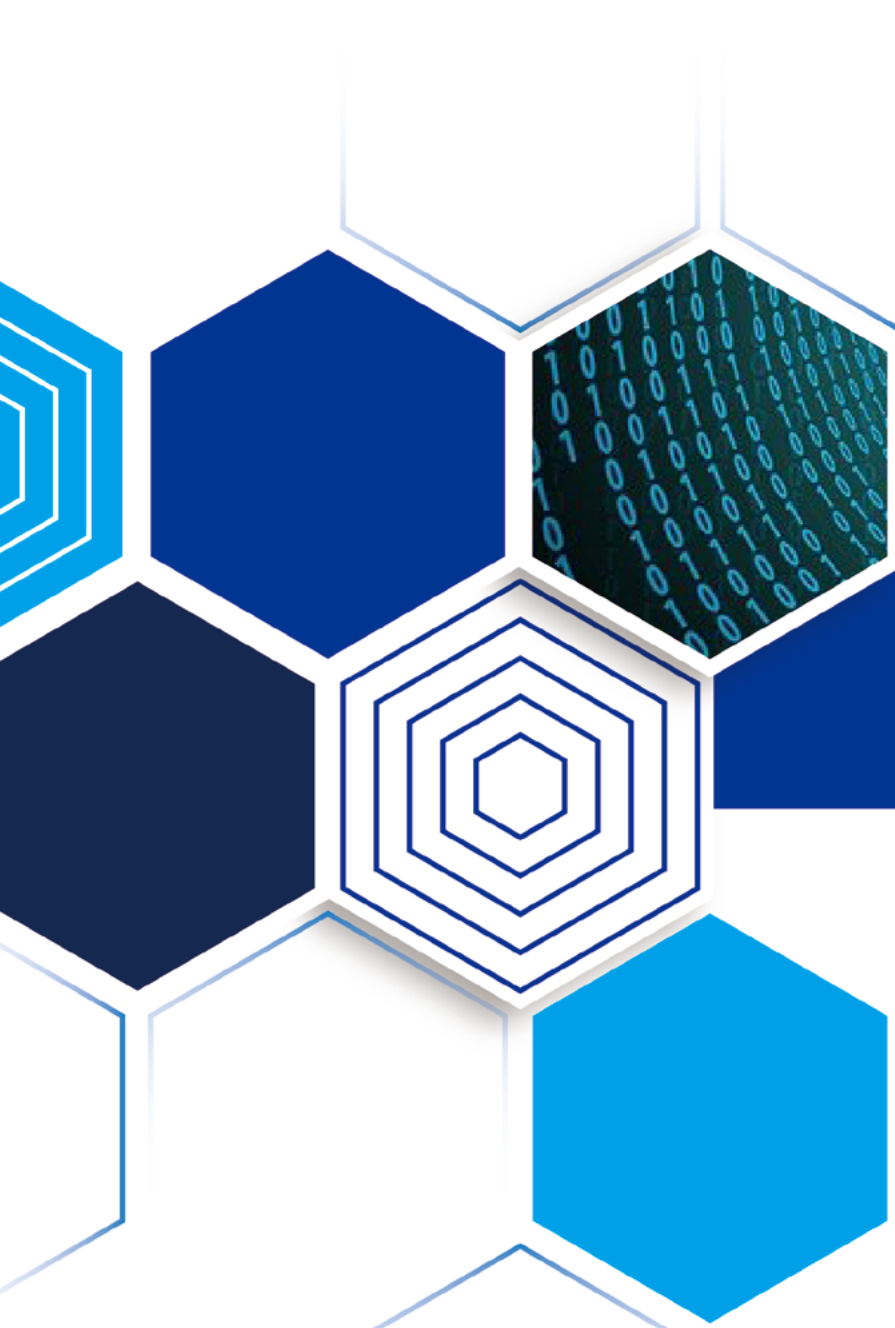
Hardware Support

The openEuler 23.03 innovative version has been verified on the x86 and Arm hardware products.

- **Servers:** TaiShan 200 2280 and 2288H V6
- **Chips:** Intel I350 and X710, Avago SAS3408, and Huawei Hi1822
- **Cards:** SP210, SR150-M, SP580, and SP330

For the most recent compatibility information, visit https://www.openeuler.org/en/compatibility/.

More products are undergoing validation tests. For details about the compatibility test process, visit https://www.openeuler.org/en/compatibility/hardware/.



03/ 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	At least 4 GB
Drive	At least 20 GB

VMs

To install openEuler on a VM, check that the VM meets the compatibility requirements.

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 embedded device, check that the embedded device meets the compatibility and minimum hardware requirements.

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

04/ Scenario-specific Innovations

Cloud-Native Service Mesh Data Plane (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

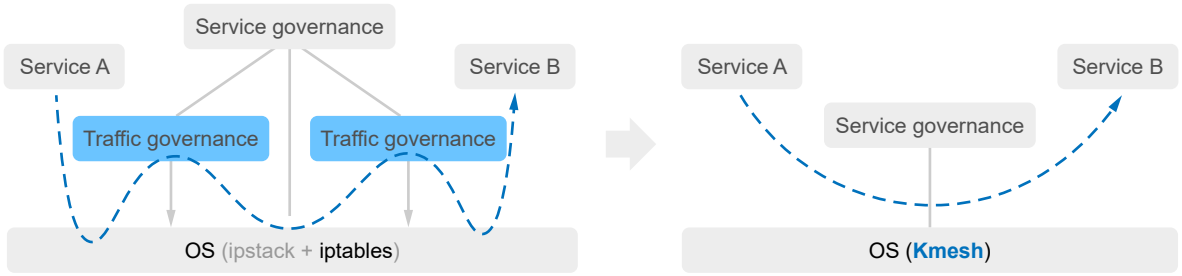
Take the typical service mesh Istio as an example. After meshing, its single-hop access latency increases by 2.65 ms, which is too high for core applications.

- Resource overhead

By default, each Istio sidecar occupies more than 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.

Offloads service governance to the OS; multiple hops → one hop



Other meshes: x times longer latency due to multiple hops along the traffic path

Kmesh: single-hop traffic path

Feature Description

Kmesh can connect to a mesh control plane (such as Istio) that complies with the Dynamic Resource Discovery (xDS) protocol. It orchestrates application traffic in the following ways:

- **Load balancing:** Various load balancing policies such as polling.
- **Routing:** L7 routing support.
- **Gray:** Backend service policies available in percentage mode.

Application Scenarios

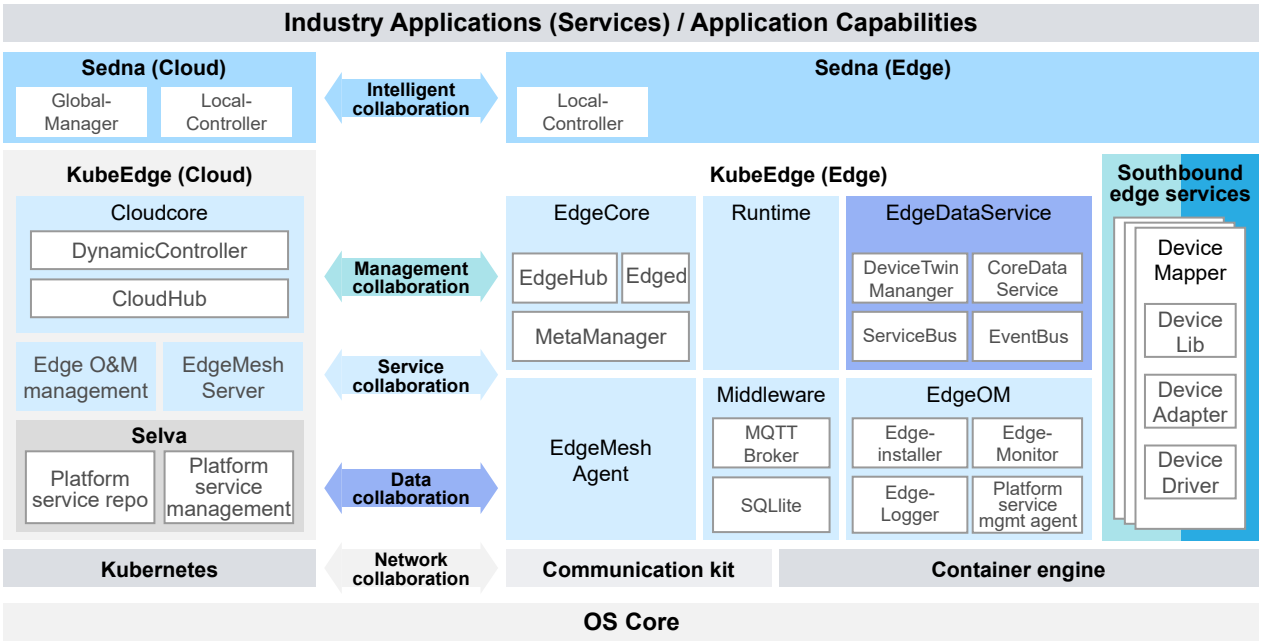
Optimized communication for cloud-native service meshes to meet latency-sensitive needs of e-commerce, billing, finance, logistics, short video, online conference, and cloud gaming deployments.

Edge Computing

As one of the 10 major technology trends, edge computing is dominating current and future business models. Smart city, autonomous driving, and industrial Internet applications are generating huge data volumes that cannot be processed by centralized cloud computing. IDC forecasts that in 2025, 48.6 ZB of data will be generated in China alone, and now high-speed, low-latency, and cost-efficient edge computing solutions are essential to many industry strategies.

openEuler 23.03 Edge integrates the KubeEdge+ edge-cloud framework into a unified management platform, on which edge and cloud applications can be provisioned. It delivers intelligent collaboration across edge and cloud to help streamline AI deployments, implement service discovery and traffic forwarding, and improve southbound capabilities.

Feature Description



openEuler 23.03 provides the edge computing framework KubeEdge+, which offers base capabilities such as communication, management, and deployment of applications across edge and cloud, as well as southbound peripheral management.

The following features will be available in future versions:

- **Enhanced edge-cloud collaboration:** The EdgeMesh Agent and EdgeMesh Server will be deployed on the edge and the cloud, respectively, to improve service discovery and routing.
- **Optimized southbound edge services:** The Device Mapper will be used for southbound access to provide the peripheral profile and parsing mechanisms, helping manage and control southbound peripherals and service streams. The southbound edge services will be compatible with the EdgeX Foundry open source ecosystem.
- **Edge data services:** On-demand persistence of messages, data, and media streams, in addition to data analysis and export operations.
- **Sedna:** This open source framework will enable collaborative inference, federated learning, and incremental learning on openEuler to unify edge and cloud environments. It will also support intelligent model and dataset management to simplify development, training, and deployment of new AI features.

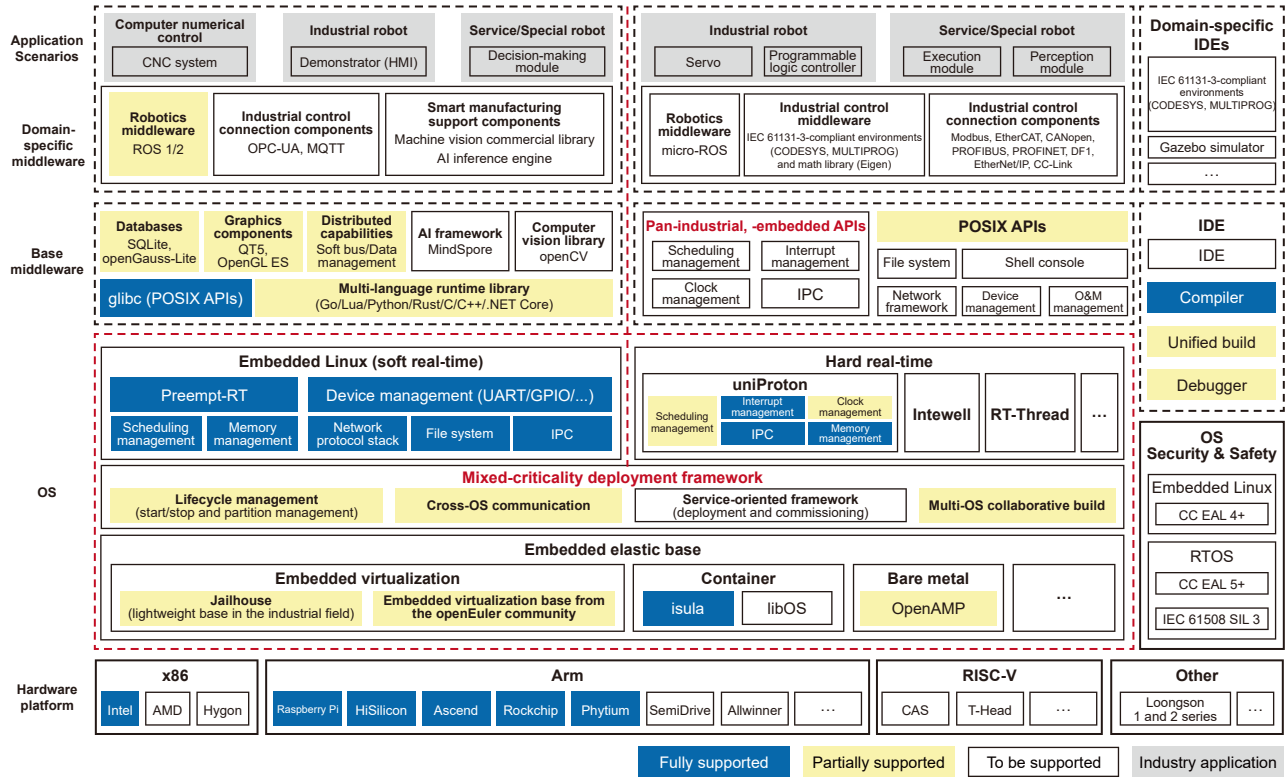
Application Scenarios

Edge-cloud collaboration in smart projects (smart manufacturing, smart gas stations, and smart campuses), transportation (urban transportation and tollway inspection), and healthcare (medical image recognition), among others.

Embedded

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 23.03 Embedded (running on Linux kernel 5.10) is equipped with an embedded virtualization base that is available in the Jailhouse or Zephyr-based Virtual Machine (ZVM) virtualization solution or the OpenAMP lightweight hybrid deployment solution. You can select the most appropriate solution to suite your services. openEuler 23.03 is also the first version to support Robot Operating System (ROS) 2 runtime environment, which integrates core software packages such as ros-core, ros-base, and SLAM. Future versions will rely on the contributions from ecosystem partners, users, and community developers, to increase the support for chip architectures such as RISC-V and LoongArch, and add capabilities such as industrial middleware, ROS middleware, and simulation systems.

Feature Description



openEuler 23.03 provides the following features for embedded scenarios:

- **Lightweight deployment:** Yocto is an open source lightweight framework on which you can customize or compress (< 5 MB) OS images, and slash OS startup time to under 5s.
- **Diverse hardware support:** Raspberry Pi, x86, Hi3093, and RK3568 can serve as the universal hardware for embedded deployments.
- **Soft real-time kernel:** Inherited from Linux kernel 5.10, this capability helps respond to soft real-time interrupts within microseconds.
- **Embedded virtualization base:** Multiple virtualization solutions are available to cater for different hardware and service scenarios.

  - 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.
- **Embedded software packages:** Over 140 common embedded software packages can be built using openEuler.
- **UniProton kernel:** The hard real-time kernel supports 75 POSIX APIs to control the context switch latency down to 3 μs and the interrupt latency to 2 μs.

The following features will be available in future versions:

- **Southbound ecosystem:** RISC-V and LoongArch.
- **Mixed-criticality deployment framework:** A mixed deployment framework will be available for critical services, delivering lifecycle management, cross-OS communication and collaboration, and service orientation, letting more soft and hard real-time OSs to join the openEuler ecosystem.
- **Embedded elastic base:** Virtualization capabilities, including Jailhouse and ZVM, will be optimized with a shorter latency to support the southbound ecosystem.
- **UniProton:** The hard real-time middleware will provide various POSIX APIs and common middleware to facilitate application development and porting.
- **Pan-industrial, -embedded APIs:** High-performance northbound APIs will be defined for performance-intensive scenarios in the Real-Time Operating Systems (RTOS) field.
- **Industry certifications:** openEuler 23.03 is currently under testing to be verified for different standards and certifications, including IEC61508 and CC EAL.

Application Scenarios

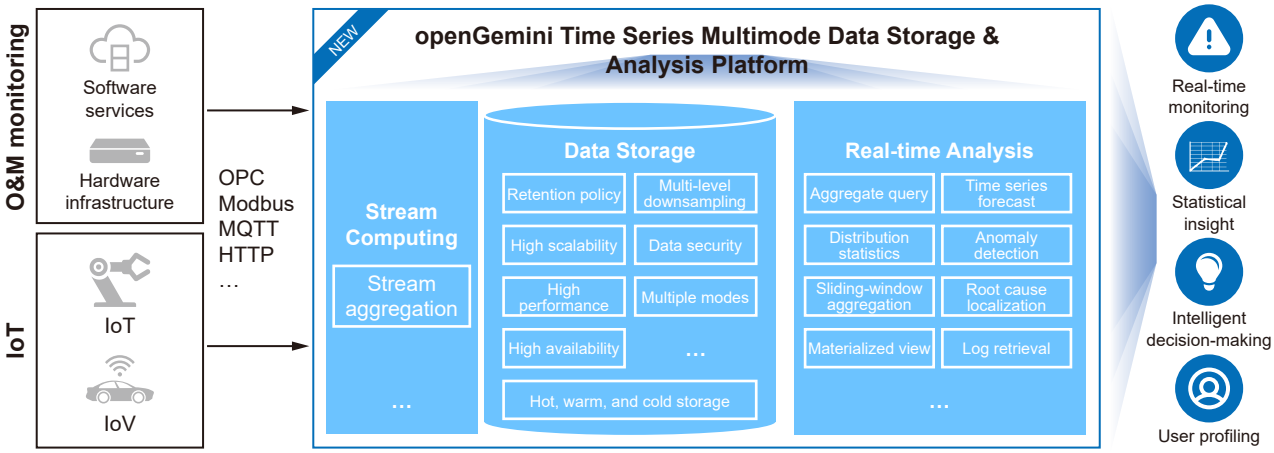
Embedded systems help supercharge computing performance in a wide range of industries and fields, including industrial and power control, robotics, aerospace, automobiles, and healthcare.

openGemini Time Series Database

openGemini is a cloud-native distributed time series database designed for IoT and O&M monitoring. It is a massively parallel processing (MPP) database that reduces database maintenance costs and improves production efficiency thanks to its advantages in performance, scalability, storage cost, and global analysis.

openGemini incorporates multiple innovative technologies, such as vectorization, stream computing, multi-level downsampling, LSM-tree optimization, AI4DB, and data compression, to build an integrated platform for IoT, O&M monitoring, and data storage and analysis. It resolves database performance and storage cost issues common with ultra-large time lines and massive time series data in fields such as observability, AIOps, and IoT.

Feature Description



openEuler 23.03 provides the following features for time series databases:

- **Distributed architecture:** The MPP architecture can be expanded horizontally.
- **Robust security:** Security measures include encrypted data transmission, user password authentication, weak password verification, and audit logs. In addition, HTTPS bidirectional authentication (mutual TLS) can be configured for communication between components in an openGemini cluster to ensure trusted links.

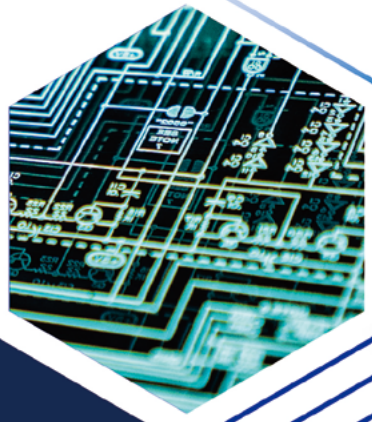
- **Wide platform compatibility:** openGemini is compatible with multiple Linux distributions such as openEuler and Ubuntu, and can work on AArch64 and x86 processor architectures.
- **Multiple development languages:** C/C++, C#, Java, Python, Go, Rust, JavaScript, and other.
- **Easy deployment options:** On-cloud, off-cloud, or edge deployment in containers, VMs, or physical servers.
- **Analytic operators:** openGemini provides more than 60 analytic operators, including aggregation operators (COUNT, SUM, MAX), statistical operators (PERCENTILE, DIFFERENCE), arithmetic operators (ABS, LN), Full Join, approximate statistical operators (PERCENTILE_OGSKETCH), and string operators (SUBSTR, STR).
- **Data compression:** In column-based storage, different compression algorithms are needed to fit the data type, while petabytes of metric data can be stored for a long time. The openGemini database reduces the storage cost to one-twentieth that of a traditional relational database and one-tenth that of NoSQL.
- **Data retention:** Each database can be assigned a data retention policy, and data is automatically deleted after they expire.
- **Write-ahead logging (WAL):** Cached data is not lost in case of a power failure.
- **Stream computing:** For large data volumes, the traditional downsampling method incurs an excess of drive I/Os and serious I/O amplification. The high-performance stream computing realizes data downsampling when data is written and with low network overheads.
- **Tiered storage:** Time series data is classified into hot, warm, and cold data to improve data query performance.
- **Kernel status observability:** openGemini uses the open source software TS Monitor to collect more than 260 key monitoring metrics of kernels and servers, helping better observe the database status and quickly isolate and rectify faults.
- **Multi-level downsampling:** When downsampling historical data of a time range, only the features of the downsampled data are retained and the data itself is deleted in place. Multi-level downsampling saves storage space by 50% and compute resources by 90%.
- **Vectorization:** For large data volumes, the MPP architecture improves query performance with batch data processing in each iteration.
- **Anomaly detection and forecast:** The openGemini database adopts an AI-based time series analysis and forecast framework, which supports batch-stream convergence and severity classification. This analysis framework contains 13 anomaly detectors and can analyze tens of thousands of real-time time lines per second. It works on common time series anomaly scenarios, such as outliers, value changes, thresholds, and continuous increase or decrease.

The following features will be available in future versions:

- **High reliability:** Multiple data copies.
- **Observability:** OpenTelemetry protocol (OTLP) support.
- **High time series cardinality:** No longer limited by the time series scale.
- **Standard SQL:** Lower learning cost.
- **Software-hardware synergy:** Maximized database performance with a data processing unit (DPU).

Application Scenarios

Electric power, transportation, telecommunications, healthcare, aerospace, Internet, smart home, smart campus, energy, and O&M monitoring.



05/

Kernel Innovations

What's New in the openEuler Kernel

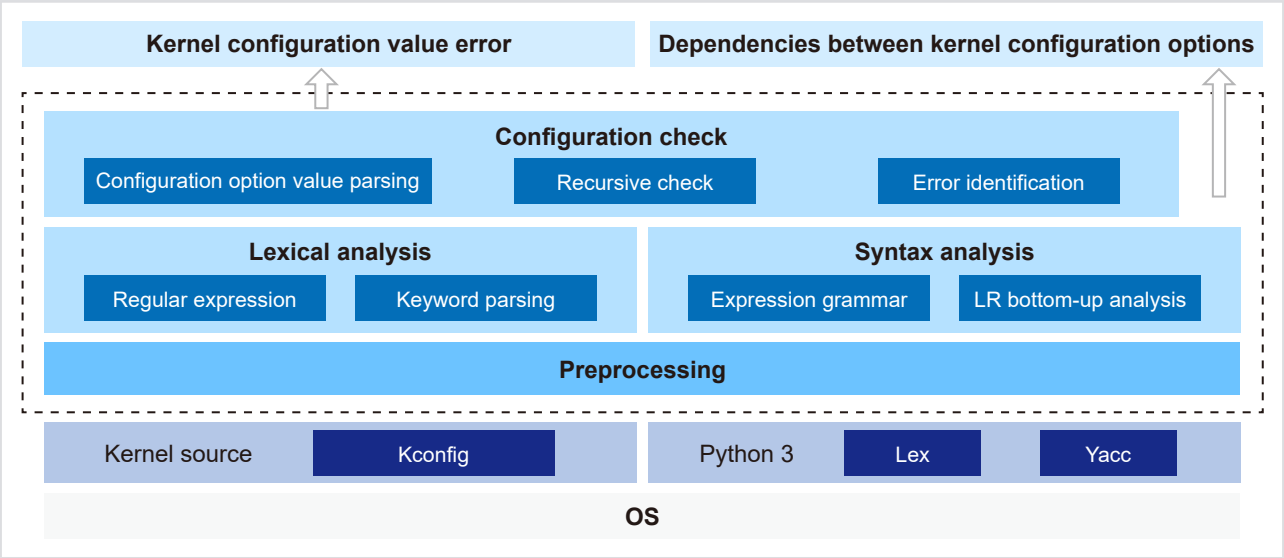
openEuler 23.03 runs on Linux kernel 6.1 and integrates the new features of Linux kernel 6.1.

- **Burstable CFS bandwidth controller:** Introduced via control groups (cgroups). It allows CPU-bound workloads to borrow their future quota resources, thereby reducing overall latency and improving batch processing.
- **SCHED_IDLE support for cgroups:** All tasks in a cgroup can be placed in the SCHED_IDLE class of the task scheduler so that these tasks are executed only when no other task is scheduled.
- **Optimized scheduling:** Reduced wakeup delay, alleviated NUMA node imbalance, and optimized select_idle_cpu() to accelerate query for idle CPUs on an overloaded system.
- **Optimized memory tiering:** New algorithms run on the enhanced memory management subsystem to identify and migrate hot and cold pages to NUMA nodes. In addition, the tiering information can be configured on the user space.
- **BPF CO-RE:** The Compile Once – Run Everywhere (CO-RE) technique allows BPF programs to be compiled once and run everywhere, resolving program porting considerations.
- **Improved XFS scale:** Deleted spin locks and global synchronization points to improve lock-free query for the buffer cache.
- **Optimized IO_URING:** The asynchronous buffer write feature improves the performance three-fold when XFS is enabled.
- **Optimized Pressure Stall Information (PSI):** The PSI feature provides a detailed view of the current system resource usage, with per-cgroup PSI and IRQ/SoftIRQ PSI and certain optimizations.

KconfigDetector to Check Value Errors in Kernel Configurations

Kconfig is a language in the kernel source that defines configuration options and interactions. It restricts the upstream and downstream dependencies between configuration options and limits mutual impact of kernel configuration options, which greatly complicates Linux kernel setup. If the kernel configuration file does not meet the value restrictions of configuration options defined by Kconfig, the kernel build may be incorrect, or an error or a security issue may arise during system running.

KconfigDetector is a tool that automatically detects value errors in kernel configuration options. It uses formal semantics and a Kconfig file parsing framework to detect value errors that do not meet dependencies and restrictions in the kernel configuration file. In addition, it can query parent and child entries to help configure the kernel. Common option errors include type, value, dependency, and matching, which can be verified and located to simplify verification and fault locating.



Feature Description

The KconfigDetector feature consists of two parts:

- Checking value errors in the kernel configuration file

The following error types in the **.config** file can be found:

- **Type error:** The value of a configuration option does not match its type.
- **Dependency error:** A configuration option is not started using **select** and dependencies are not met.
- **Dependency risk:** A configuration option is forcibly started using **select**, but dependencies are not met.
- **Configuration option not found:** The specified configuration option is not found in the kernel Kconfig file.
- **Invalid value range:** The value is not within the value range specified by **range**.
- **Value alarm:** The value of a configuration option does not meet the requirements of **default** or **imply**.

- Querying dependencies between kernel configuration options

The kernel configuration space is modeled to generate dependencies of kernel configuration options in a JSON file. All parent and child entries that are directly and indirectly dependent can be quickly queried, which is useful to secondary development.

Application Scenarios

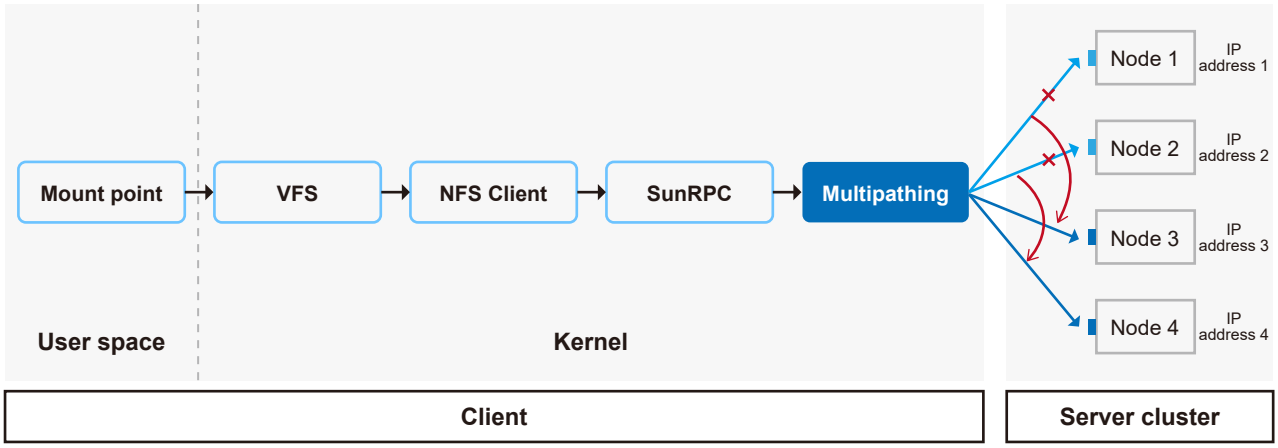
This feature is perfectly suited to streamline kernel testing, tailoring, and porting tasks, by directly modifying the **.config** file instead of using other kernel configuration tools. It can also check the updated **.config** file to ensure that the kernel configuration file meets dependencies.

NFS Multipathing

Network File System (NFS) is a distributed file system protocol first developed in 1984 by Sun Microsystems to let users on NFS clients access files on NFS servers over computer networks. As the NFS service is widely used in industries such as finance, EDA, AI, and container, higher requirements are imposed on NFS performance and reliability. Multiple innovations to NFS have been developed over the years, to resolve the following disadvantages:

- A mount point on a client is accessible only through a client and a server IP address. Even when multiple physical links exist between the client and the server, only one of them is usable.
- Conventional NFS was prone to single points of failure, such as a single link of a single mount point, meaning link failover cannot be performed and host services will be interrupted.

Feature Description



NFS multipathing solves the defects of conventional NFS. It ensures multiple links between the client and the server are connected for each single mount point to support I/O transmission over those links, improving the mount point performance. In addition, the link status is periodically checked to ensure fast I/O failover upon a link fault.

NFS multipathing provides the following features:

- NFSv3 supports the Round Robin link selection algorithm to balance the performance of multiple links.
- NFSv3 and NFSv4 support fast link failover to improve NFS reliability.

- An interface for registering link path selection algorithms is provided to let developers customize path selection algorithms.
- Checks link availability periodically.
- Displays the link status in real time.

Application Scenarios

UNIX/Linux environments, to improve processes in enterprise office automation (OA), Internet, VM, and high-performance computing (HPC) deployments.

06/

Enhanced Features

Enhanced DSoftBus

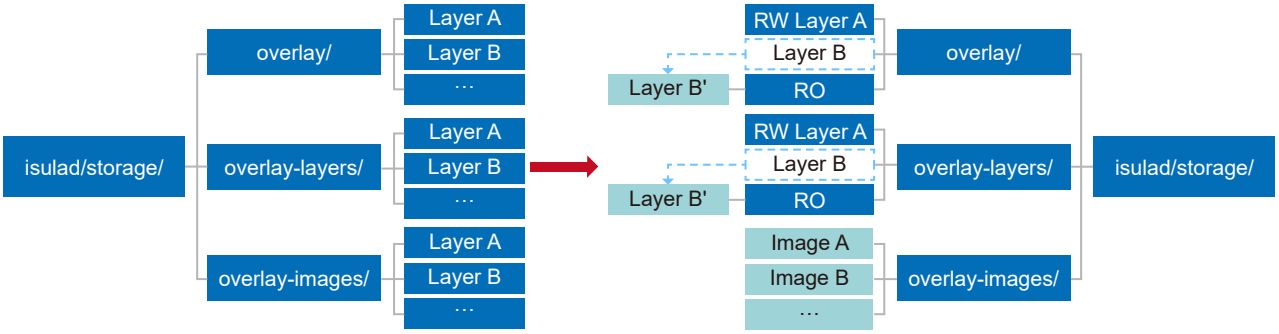
The DSoftBus system of openEuler 23.03 integrates the DSoftBus and point-to-point authentication module of OpenHarmony. It implements interconnection between openEuler-based embedded devices and OpenHarmony-based devices as well as between openEuler-based embedded devices. New features on openEuler 23.03 include Bluetooth Low Energy (BLE) discovery, file and stream transmission interfaces, and the nStack and FillP protocols.

iSulad: RO Overlay of Image Data Directories

Based on the read-only (RO) feature of image data, iSulad categorizes image data and overlays image data from container data to open new possibilities for storing image data in distributed storage systems.

Feature Description

The directories are overlaid as follows:



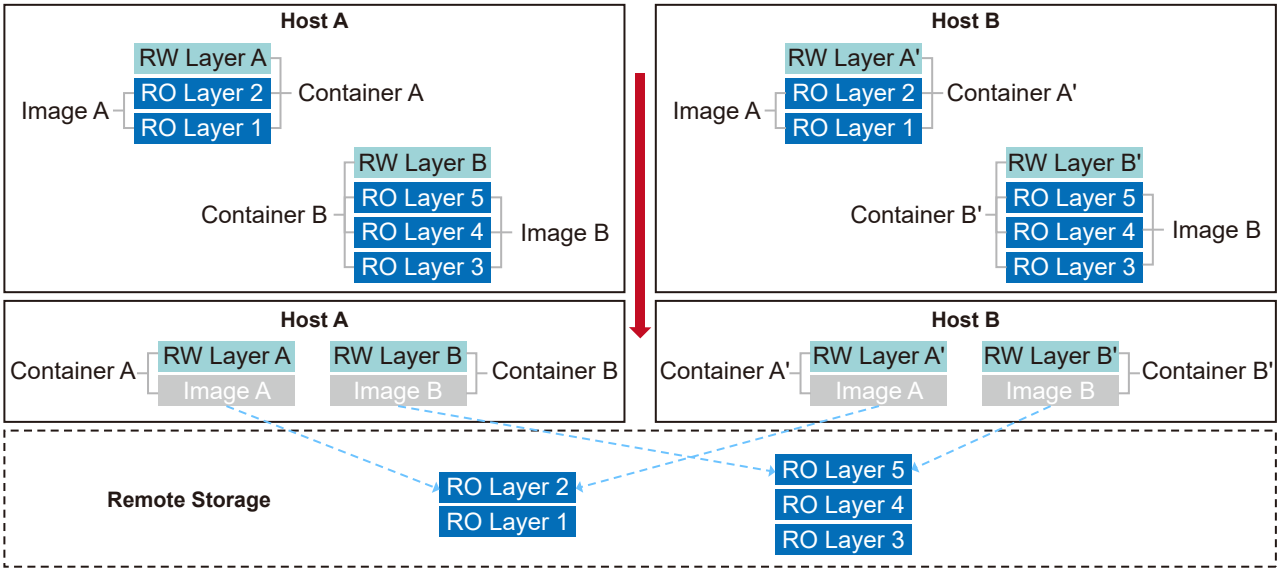
The following operations are available for the overlaid directories:

- It is configurable whether to enable RO overlay.
- **overlay-images** supports overlay of image metadata.
- **overlay-layers** supports overlay of layer metadata.
- **overlay** supports overlay of data after image decompression.

Application Scenarios

Scenario 1: Image sharing

Distributed storage is used for sharing image data, which greatly reduces the storage space for cluster images, accelerates the initial boot of containers, and reduces traffic caused by the container image repository.



Scenario 2: Local storage

Local independent partitions or drives are used as the storage space for image data, to facilitate image data management and transfer.

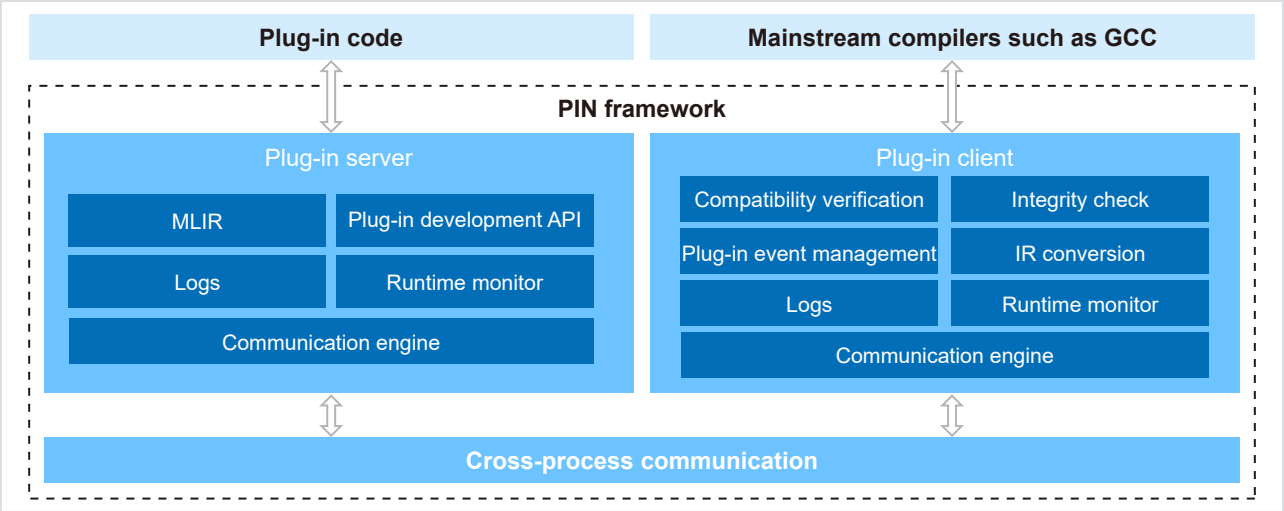
Constraints

- The image storage driver must be OverlayFS.
- In distributed storage setups, the Kubernetes layer synchronizes image deletion and download operations on multiple nodes.

Enhanced PIN Framework

The Plug-IN (PIN) framework is a plug-in development platform that provides MLIR-oriented interfaces to help develop plug-ins once but apply for multiple compilers and optimize features using the plug-ins, improving the development efficiency. The framework supports and maintains common capabilities such as plug-in compatibility and integrity checks.

Feature Description



- MLIR-based plug-in development and easy conversion of intermediate representations such as GIMPLE.
- The PIN framework supports 19 classes of GIMPLE statements.
- Base capabilities such as compatibility and binary integrity checks.
- Monitoring and verifying plug-in status, such as for compiler security and operations.
- Executing plug-in clients as GCC plug-ins, so functions can run without modifying the GCC compiler code.
- Link time optimization (LTO).

Application Scenarios

Scenario 1: Compilation tool build and integrity verification

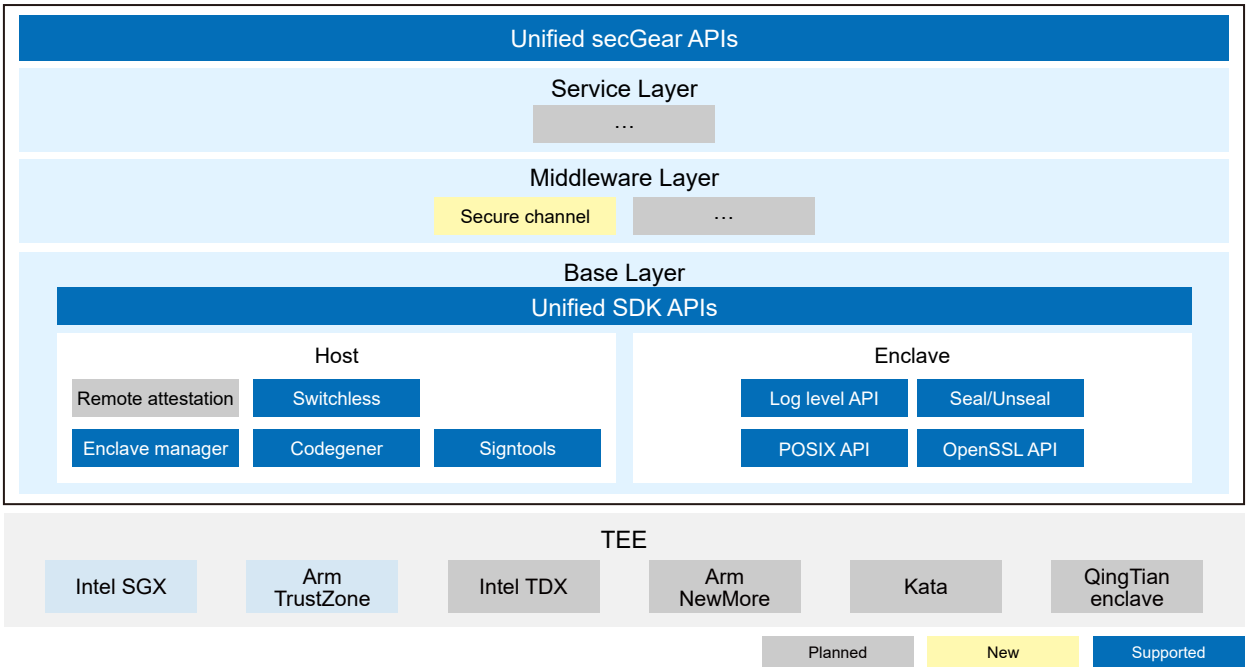
The PIN framework can work as the development platform to build compilation tools based on MLIR once but apply for multiple compilers such as GCC. The framework supports and maintains common capabilities such as compatibility and binary integrity checks.

Scenario 2: Quick enabling and verification of compilation tools

The PIN framework helps develop compilation tools as plug-ins to run on mainstream compilers such as GCC. There is no need to modify the source code of the compilers, streamlining the development efficiency.

secGear Confidential Computing Framework

secGear is a unified development framework that delivers confidential computing for the openEuler system. Compatible with popular trusted execution environments (TEEs), secGear masks the differences between TEEs and SDKs and simplifies APIs by sharing the same set of source code over multiple architectures. It streamlines the TEE applications while contributing to the confidential computing ecosystem.



secGear is logically divided into three layers that form the foundation of openEuler confidential computing software.

- Base layer: A unified layer that provides unified APIs for multiple TEEs, enabling different architectures to share the same set of source code.
- Middleware layer: A general component layer that runs the confidential computing software for users to quickly build solutions.
- Service layer: A confidential computing service layer that runs scenario-specific solutions.

In openEuler 23.03, the secure channel feature is added.

Feature Description

For cloud scenarios, data owners need to upload data to the cloud-based TEE for processing. Because the TEE is not connected to the Internet, the data needs to be transferred to the REE over the network in plaintext and then transferred to the TEE, but in doing so, the plaintext data is exposed in the REE memory, posing security risks.

A secure channel is a technique used to implement secure key negotiation between data owners and cloud-based TEEs. By combining remote attestation of confidential computing, it negotiates a session key, which belongs to the data owner for data encryption. The REE receives the ciphertext data, after which it transfers the data to the TEE for decryption and processing.

The secure channel SDK consists of the client, host, and enclave, which are executed by the client, client application (CA), and trusted application (TA), respectively.

- After the secure channel negotiation between the client in the REE and the server in the TEE is complete, both ends obtain the same key for encryption and decryption.
- The client and CA support encryption and decryption.

Application Scenarios

Scenario 1: Encrypted databases

An encrypted database provides SQL query and computing capabilities in the TEE. In a database client query, the ciphertext is transferred to the TEE. The client sends the ciphertext key to the TEE through a secure channel. Then the ciphertext is decrypted in the TEE to execute the query.

Scenario 2: TEE-based feature protection

In vertical federated learning (VFL), followers obtain intermediate results from the raw data and send the results to the leader. The leader applies labels to the intermediate results to obtain the gradient, which is sent to each follower. However, attackers may obtain the intermediate results uploaded by followers from the memory and deduce user information, which poses security risks. The intermediate result processing logic of the leader is deployed in the TEE, whereby the followers' intermediate results are transferred through a secure channel and displayed in plaintext.

radiaTest Community Test Platform

radiaTest is a management platform that performs testing in the openEuler community. Its key component is the web data mid-end, which streamlines and produces traceable community version tests, but also provides plug-ins for resource management and automated tests to connect to multiple test engines.

This feature is optimized in openEuler 23.03 to enhance authentication and supplement capabilities.

Feature Description

- Login authentication is redesigned to provide diverse authentication methods for organizations and communities. For organizations, any login to openEuler is authenticated based on the openEuler community instead of the enterprise code repository on Gitee.
- The testing progress of version baseline cases is displayed on the quality dashboard to monitor case execution of the whole version.
- IT-based testing design provides an easy entry to organization/community testing strategy management.
- Statistics on trouble tickets associated with test tasks can be collected.

Application Scenarios

Scenario 1: Early-stage testing strategies

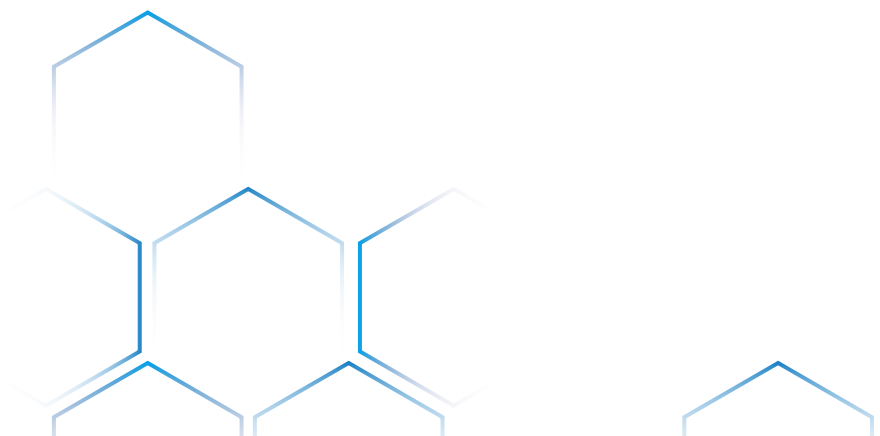
Community developers can find the test product version on the testing design page of the workbench and select the feature node to be compiled for testing. The platform lets users add, edit, and delete mind maps for testing strategies, and connect to the openEuler QA repository on Gitee to enable consistent archiving and format conversion, import, and export.

Scenario 2: Version quality monitoring and real-time test case execution tracing

Community developers, test managers, and QA personnel can view the execution status of version baseline cases on the quality dashboard to learn the real-time testing progress.

Scenario 3: Association between trouble tickets and test tasks

If a confirmed issue occurs in a test case, the tester can submit a trouble ticket on the test task page to ensure the task and ticket are synchronized to the Gitee repository. In this way, the test task is associated with the trouble ticket.



07/ Copyright

08/ Trademark

09/ Appendixes

Appendix 1: Setting Up the Development Environment

Environment Preparation	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