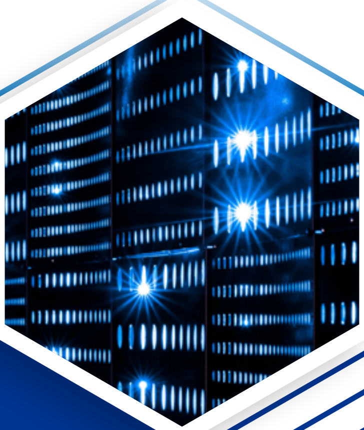
openEuler 22.03 LTS SP1

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 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.

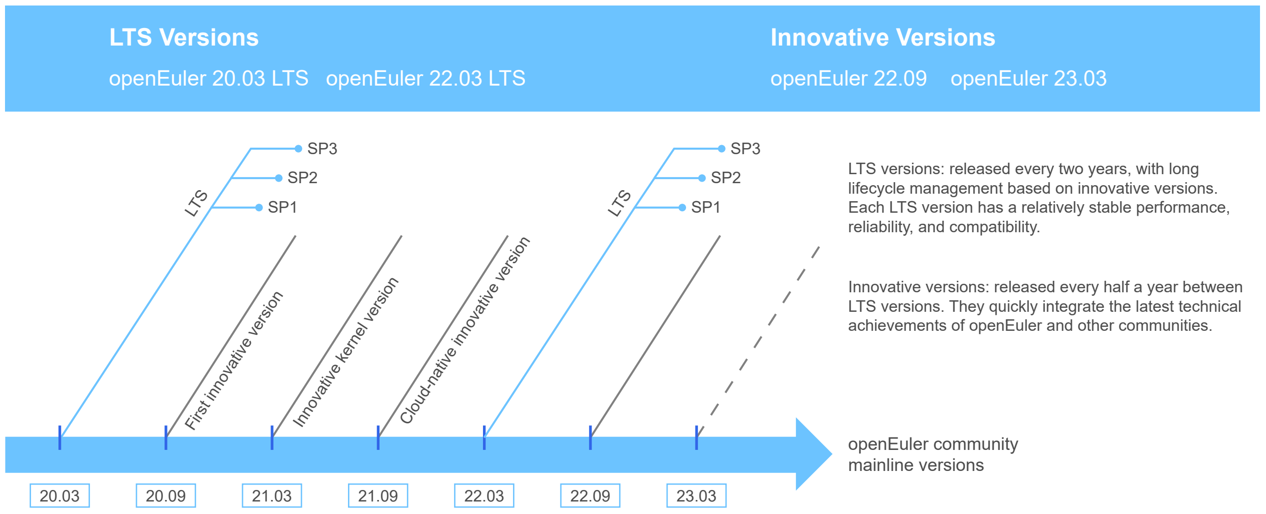
Fast forward to September 30, 2021, openEuler 21.09 was released. This premium version is designed to supercharge all scenarios, including edge and embedded devices. It enhances server and cloud computing features, and incorporates key technologies including cloud-native CPU scheduling algorithms for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

openEuler Version Management

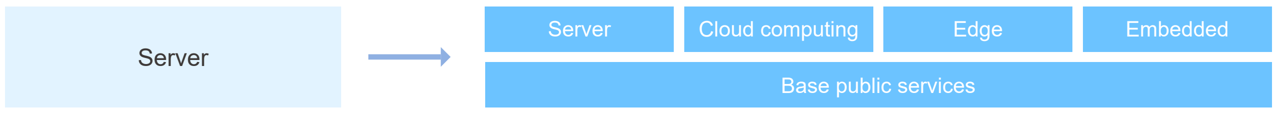

LTS Versions
openEuler 20.03 LTS
openEuler 22.03 LTS
Innovative Versions
openEuler 22.09
openEuler 23.03
LTS
SP1
SP2
SP3
First innovative version
Innovative kernel version
Cloud-native innovative version
LTS
SP1
SP2
SP3
LTS versions: released every two years, with long lifecycle management based on innovative versions. Each LTS version has a relatively stable performance, reliability, and compatibility.
Innovative 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
20.03
20.09
21.03
21.09
22.03
22.09
23.03

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

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

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

Innovative Platform for All Scenarios



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

The openEuler community is home to an increasing number of special interest groups (SIGs), which are dedicated teams that help extend the OS features from server to cloud computing, edge computing, and embedded scenarios. openEuler is built to be used in any scenario, and comprises openEuler Edge and openEuler Embedded that are designed for edge computing and embedded deployments, respectively.

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

Open and Transparent: The Open Source Software Supply Chain

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

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.

openEuler 22.03 LTS SP1 runs on Linux kernel 5.10 and provides POSIX-compliant APIs and OS releases for server, cloud native, edge, and embedded environments. It is a solid foundation for intelligent collaboration across hybrid and heterogeneous deployments. openEuler 22.03 LTS SP1 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.

Kernel innovations:

- **Enhanced cloud-native scheduling:** openEuler suits hybrid deployments of online and offline cloud services. Its innovative CPU scheduling algorithm ensures real-time CPU preemption and jitter suppression for online services. Additionally, its innovative memory reclamation algorithm against out of memory (OOM) allows online services to run reliably based on their higher service priorities.
- **EulerFS:** A new file system is designed for non-volatile dual in-line memory modules (NVDIMMs). It uses technologies such as soft updates and dual-view directories to accelerate file metadata synchronization and thus improve file read and write performance.
- **Tiered memory expansion etMem:** With the user-mode swap function, the discarded cold memory can be changed to the user-mode storage based on a preset policy. The user-mode swap delivers a higher performance than the kernel-mode swap and the whole swap process is transparent to users.
- **Enhanced memory reliability, availability, and serviceability (RAS):** The tiered-reliability memory technology preferentially allocates high-reliability memory for sensitive data, such as kernels and key processes. This technology reduces the system breakdown rate and enhances system reliability.

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:

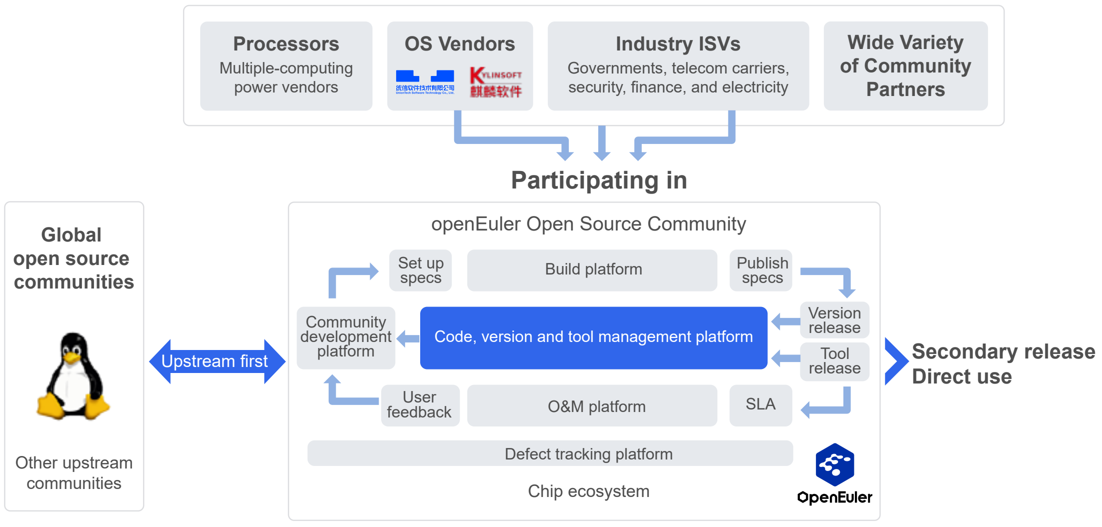
- **Edge computing:** openEuler 22.03 LTS SP1 Edge is released for edge computing scenarios. It integrates the KubeEdge+ edge-cloud collaboration framework to provide unified management, provisioning of edge and cloud applications, and other capabilities.
- **Embedded:** openEuler 22.03 LTS SP1 Embedded is released for embedded scenarios, helping compress images under 5 MB and image loading within 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.


Processors
Multiple-computing power vendors
OS Vendors
Industry ISVs
Governments, telecom carriers, security, finance, and electricity
Wide Variety of Community Partners
Participating in
Global open source communities
Other upstream communities
Upstream first
openEuler Open Source Community
Set up specs
Build platform
Publish specs
Community development platform
Code, version and tool management platform
Version release
Tool release
User feedback
O&M platform
SLA
Defect tracking platform
Chip ecosystem
OpenEuler
Secondary release
Direct use

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	ShenWei	Loongson	RISC-V
CPU	Intel, AMD, Zhaoxin, Hygon	Kunpeng, Phytium	ShenWei	Loongson	Nuclei, StarFive, UC Techip

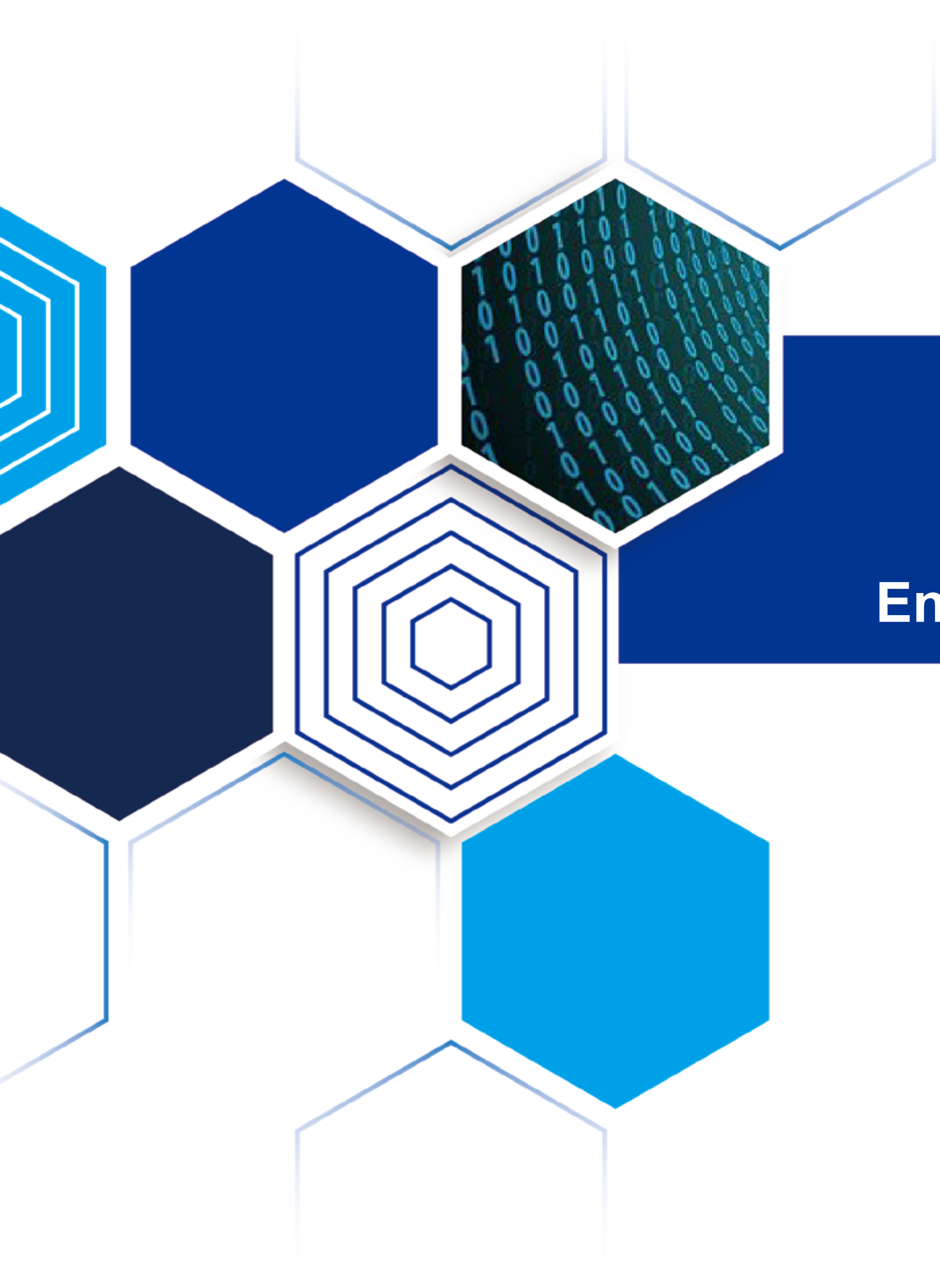
openEuler supports the following servers:

Hardware Type	x86	Arm	ShenWei	Loongson
Server	**Intel:** xFusion, H3C, Inspur, Lenovo, Nettrix, SUPERCLOUD, PowerLeader, Supermicro, ZTE, Dahua, Jaguar Microsystems **AMD:** H3C, Lenovo, Supermicro **Hygon:** H3C, Sugon/Suma **Zhaoxin:** Zhaoxin	**Kunpeng:** TaiShan, Tongfang, H3C, PowerLeader, Digital China, Yangtze Computing, Huanghe, Sichuan Hongxin, Xiangjiang Kunpeng, 100 Trust, Tiangong, SUPERCLOUD **Phytium:** QS, H3C, PowerLeader, Lenovo, Tongfang, Skysolidiss	ShenWei	Loongson

openEuler supports the following cards:

Hardware Type	x86	Arm
NIC	Huawei, Mellanox, Intel, Broadcom, Marvell, NetSwift	Huawei, Mellanox, Broadcom, Marvell, NetSwift, Intel
RAID	Huawei, Avago, PMC	Huawei, Avago, PMC
Fibre Channel	Huawei, Marvell, Qlogic, Emulex	Huawei, Marvell, Qlogic, Emulex
GPU & AI	Huawei, NVIDIA, AMD, Iluvatar CoreX, Intel	Huawei, NVIDIA, AMD, Iluvatar CoreX, Intel
DPU	Jaguar Microsystems	
SSD	Huawei, Samsung, Intel, Dera	Huawei, Samsung, Intel, Dera
Security	Sansec	Sansec, Xilinx

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



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

openEuler supports the following virtual machines (VMs):

1. centos-7.9 qemu 1.5.3-175.el7 libvirt 4.5.0-36.el7 virt-manager 1.5.0-7.el7
2. centos-8 qemu 2.12.0-65.module_el8.0.0+189+f9babebb.5
   libvirt 4.5.0-24.3.model_el8.0.0+189+f9babebb virt-manager 2.0.0-5.el8
3. fedora 32 qemu 4.2.0-7.fc32 libvirt 6.1.0-2.fc32 virt-manager 2.2.1-3.fc32
4. fedora 35 qemu 6.1.0-5.fc35 libvirt 7.6.0-3.fc35 virt-manager 3.2.0-4.fc35

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 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/

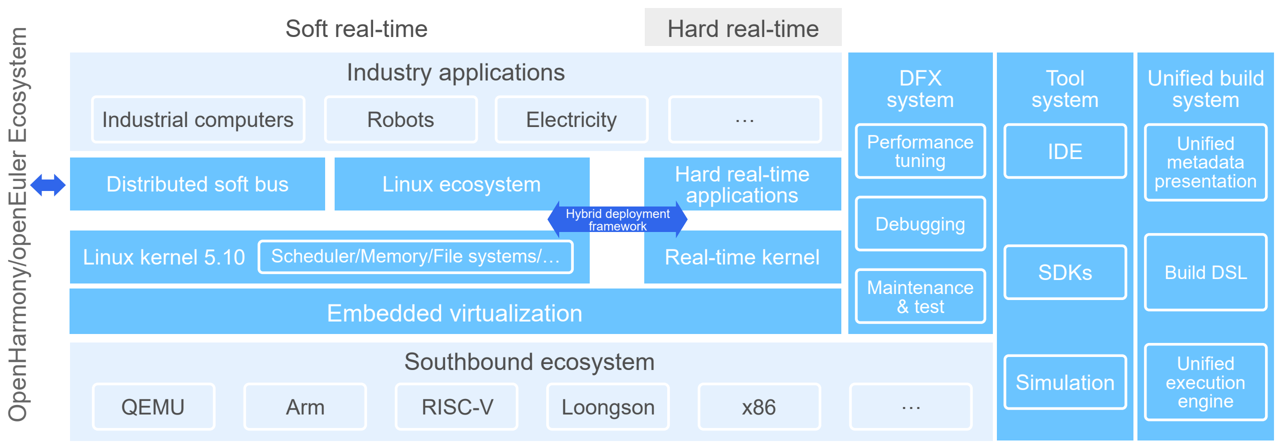
Scenario-specific Innovations

openEuler 22.03 LTS SP1 includes the Edge release for edge computing and the Embedded release for embedded systems. The OS empowers all-scenario collaboration over the new generation of digital infrastructure.

Embedded

openEuler 22.03 LTS SP1 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. Based on the participation of ecosystem partners, users, and developers of the openEuler open source community, openEuler 22.03 LTS SP1 Embedded helps to design efficient embedded OS solutions. It will attain greater support for chip architectures such as PowerPC and RISC-V, and add capabilities such as deterministic latency, industrial middleware, and simulation systems.

Feature Description



openEuler 22.03 LTS SP1 provides the following features for embedded scenarios:

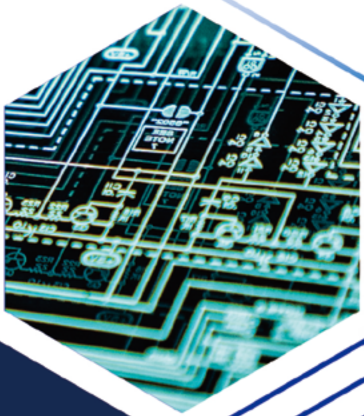
- **Lightweight deployment:** The open source Yocto is a small-scale and lightweight framework that allows you to customize OS images. It can compress OS images to under 5 MB and shorten OS startup time to under 5s.
- **Support for diverse hardware types:** Among others, Raspberry Pi is a new device that can serve as the universal hardware for embedded deployments.
- **Soft real-time kernel:** This capability is inherited from Linux kernel 5.10, and helps respond to soft real-time interrupts within microseconds.
- **Mixed criticality deployment:** Real-time (UniProton/Zephyr) and non-real-time planes can coexist on an SOC based on Raspberry Pi. The lifecycle management of real-time systems is also supported.
- **Distributed soft bus (DSoftBus):** The DSoftBus and HiChain point-to-point authentication module available in OpenHarmony are used to implement interconnection and interworking between embedded devices running on openEuler as well as between openEuler embedded devices and OpenHarmony devices.
- **Embedded software packages:** Over 140 common embedded software packages can be built using openEuler.
- **Hard real-time kernel:** The open source Real Time Operating System (RTOS) kernel, UniProton, supports 75 POSIX APIs. It controls the context switchover latency down to 2 μs and the interrupt latency to 1 μs.

The following features will be available soon:

- **Unified APIs:** The hard real-time kernel supports more POSIX APIs to simplify application development.
- **Industry security certifications:** openEuler 22.03 LTS SP1 is currently under review for different standards and certifications, including IEC61508 and EC62443.

Application Scenarios

Embedded systems help supercharge computing performance in a wide range of industries and fields, including aerospace, industrial control, telecommunications, automobiles, and healthcare. Mature 5G and AI technologies will enable embedded systems to meet the demands for intelligent management of IoT and edge computing devices.



05/ Kernel Innovations

What's New in the openEuler Kernel

openEuler 22.03 LTS SP1 runs on Linux kernel 5.10 and inherits the competitive advantages of community versions and innovative features released in the openEuler community.

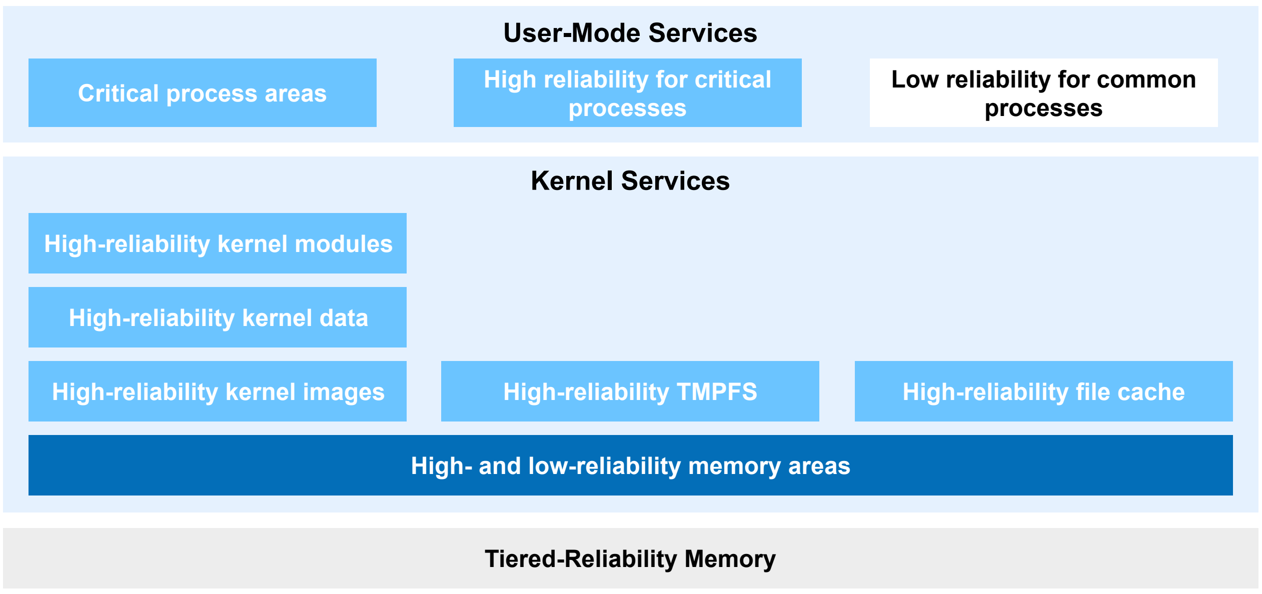
- **Memory RAS – reliable memory:** More extensive support for reliable memory, which is used by the kernel, key processes, memory file system, and file cache to prevent kernel resets caused by multi-bit errors (MBEs).
- **Memory RAS – enhanced UCE tolerance:** In the event the copy_from_user experiences an MBE, the affected processes can be killed to avoid a kernel reset.
- **Programmable kernel:** The kernel scheduler can dynamically extend scheduling policies to diverse requirements.
- **Resource isolation:** cgroup v1 supports iocost. Users can configure weights to allocate I/O resources.
- **CXL support:** cgroup v1 supports iocost. Users can configure weights to allocate I/O resources.
- **Commissioning:** The perf c2c tool now runs on AArch64 Statistical Profiling Extension (SPE) to detect cache pseudo-sharing and locate bottlenecks.
- **AF_UNIX socket optimization:** The connection delay and CPU usage are greatly reduced in concurrent tasks.

Tiered-Reliability Memory

Memory hardware faults may cause servers to break down. In the event of an uncorrectable MBE, the OS kernel and key service processes will be impacted, and the OS will be reset, causing a long downtime.

A server may be equipped with memory resources, such as host memory buffer (HMB), NVDIMMs, and Address Range Mirror, that vary in reliability specifications. That is, a server may run memory resources with high and low reliability. The OS can manage different memory reliability levels, and allocate more reliable memory resources to kernels and service processes that are susceptible to memory errors. The tiered-reliability memory feature minimizes system resets caused by memory errors and therefore improves system availability.

Feature Description


User-Mode Services
Critical process areas
High reliability for critical processes
Low reliability for common processes
Kernel Services
High-reliability kernel modules
High-reliability kernel data
High-reliability kernel images
High-reliability TMPFS
High-reliability file cache
High- and low-reliability memory areas
Tiered-Reliability Memory

- **Data protection:** Critical kernel data can be allocated to the highly reliable memory area, to prevent any possible memory errors caused by kernel data reads and writes.
- **High-reliability memory for processes:** By setting the attribute of the process, you can prioritize and allocate a process to a memory space from the highly reliable memory area.
- **High-reliability memory for TMPFS:** Read operations on the temporary file system, or TMPFS, are implemented in kernel mode. A memory error will cause a system reset. Using highly reliable memory reduces the probability of system resets.

- **High-reliability memory for file cache:** Read operations on the file cache (page cache) are implemented in kernel mode. A memory error will cause a system reset. Using highly reliable memory reduces the probability of system resets.
- **Optimization of high-reliability memory:** Thresholds can be set for the file cache, user-mode processes, and TMPFS.
- **UCE recovery:** Uncorrectable errors (UCEs) triggered by memory copy in specific scenarios, such as cow, copy_mc_to_kernel, copy_from_user, copy_to_user, get_user, and put_user, can be recovered.

Application Scenarios

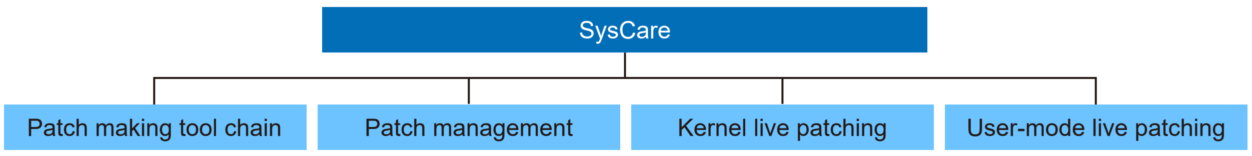
Semi-closed systems have high reliability demands, such as embedded systems.

SysCare for Live Patching

SysCare is an online live patching tool that automatically fixes bugs and vulnerabilities in OS components, such as kernels, user-mode services, and dynamic libraries. It has the following advantages:

- **Unified patches:** The tool masks differences in detail when creating patches, providing a unified management tool to improve O&M efficiency.
- **User-mode live patching:** It supports live patching of multi-process and -thread services in user mode, which takes effect when a process or thread is started or restarted.
- **Lazy mechanism:** SysCare fixes the ptrace defect (all kernel calls are ended) and improves the fix success rate.

Feature Description



SysCare supports live patching for kernels and user-mode services.

- One-click creation

  SysCare is a unified environment for both kernel and user-mode live patches that masks differences between patches, ensuring they can be created with just one click.

- Patch lifecycle operations

  SysCare provides a unified patch management interface for users to install, activate, uninstall, and query patches.

- Commercial use of kernel live patches

  OS: openEuler 22.03 LTS SP1

  Upper-layer applications: Redis and Nginx

- Limited support for user-mode live patches

  - SysCare supports hot fixes in Executable and Linkable Format (ELF), but not in interpreted languages.

  - It supports DWARF debug information, but not debug information at the G3 level.

  - It does not support cross compilation.

Application Scenarios

SysCare provides a live patching solution for bugs and common vulnerabilities and exposures (CVEs) in kernels, libraries, and user-mode services.

06/

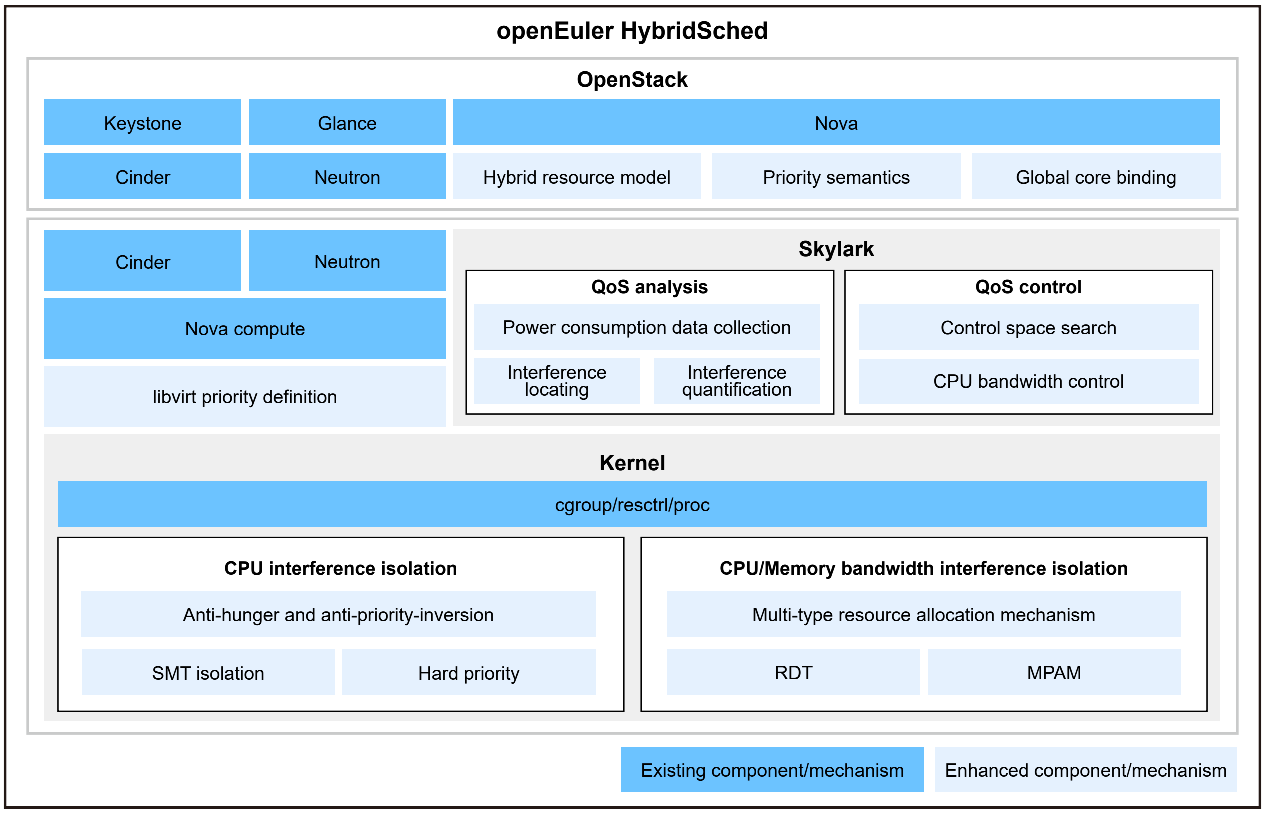
Cloud Base

HybridSched for Hybrid Virtualization Scheduling

Low resource utilization of cloud data centers is a common issue in the industry, and has fueled ways to improve this problem, such as deploying services based on priorities (hybrid deployment). The core technology of hybrid deployment is resource isolation and control.

HybridSched is a full-stack solution for hybrid deployment of VMs, covering enhanced OpenStack cluster scheduling, the single-node QoS management component Skylark, and kernel-mode base resource isolation. In particular, Skylark is a QoS-aware resource scheduler used when high- and low-priority VMs are deployed together, improving physical machine resource utilization while ensuring the QoS of high-priority VMs.

Feature Description



- **Enhanced cluster scheduling:** Enhances OpenStack Nova to support priority-based semantic scheduling.
- **Power consumption control:** Limits the CPU bandwidth of low-priority VMs to reduce the overall system power consumption and ensure the QoS of high-priority VMs.
- **Cache and memory bandwidth control:** Limits the LLC and memory bandwidth of low-priority VMs. Currently, only static allocation is supported.
- **CPU interference control:** Supports CPU time slice preemption in microseconds, SMT interference isolation, and anti-priority-inversion.

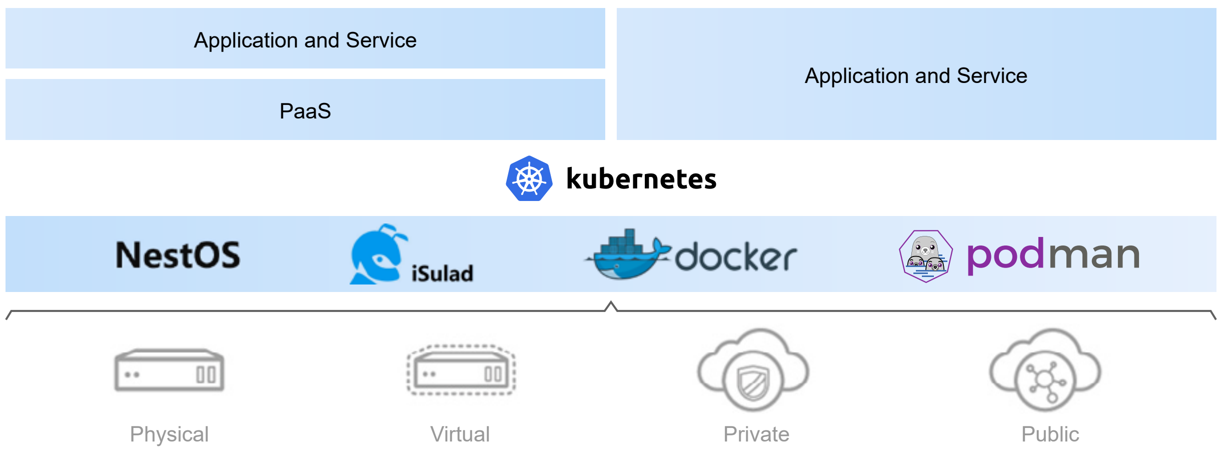
Application Scenarios

To improve resource utilization, services are classified into high- and low-priority services based on latency sensitivity, and deployed accordingly. Latency-sensitive services are recommended for high-priority VMs, such as web services, high-performance databases, real-time rendering, and machine learning inference; while services not limited by latency can be used for low-priority VMs, such as video encoding, big data processing, offline rendering, and machine learning training.

Container OS NestOS

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

Feature Description



- **Out-of-the-box design:** Integrates popular container engines such as iSulad, Docker, and Podman to provide lightweight and tailored OSs for the cloud.
- **Easy configuration:** Uses the utility Ignition 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, stable atomic updates.
- **Hitless node updating:** Uses Zincati to provide automatic node updates and reboot without interrupting services.
- **Dual rootfs:** Executes dual rootfs for active/standby switchovers, to ensure integrity and security during system running.

Application Scenarios

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

07/

Enhanced Features

Full-Stack Support for SM Cryptographic Algorithms

The openEuler OS now supports ShangMi (SM) cryptographic algorithms (SM2, SM3, and SM4) in key security features, and provides cryptographic services such as the SM cryptographic algorithm library, certificates, and secure transmission protocols for upper-layer applications.

Feature Description

SM cryptographic algorithms provide the following features:

- User-mode algorithm libraries, such as OpenSSL and Libgcrypt, support SM2, SM3, and SM4.
- OpenSSH supports SM2, SM3, and SM4.
- OpenSSL supports the Transport Layer Cryptography Protocol (TLCP) stack of the SM standards.
- SM3 and SM4 are supported for drive encryption (dm-crypt/cryptsetup).
- SM3 is supported for password encryption in user identity authentication (pam/libuser/shadow).
- SM3 is supported for data digest in intrusion detection (AIDE).
- SM2, SM3, and SM4 are supported in the kernel cryptographic framework (crypto), allowing algorithm performance optimization using instruction sets such as AVX, CE, and NEON.
- The SM3 message digest algorithm and SM2 certificate are supported in Integrity Measurement Architecture and Extended Verification Module (IMA/EVM) of the kernel.
- The SM2 certificate is supported in kernel module signing and module signature verification.
- SM4-CBC and SM4-GCM algorithms are supported in Kernel Transport Layer Security (KTLS).
- SM3 and SM4 are supported in the Kunpeng Accelerator Engine (KAE).
- The shim component supports secure OS boot based on SM2 and SM3

Application Scenarios

The SM capabilities provided by openEuler safeguard applications running on openEuler. For example, SM3 is used to encrypt service data based on the OpenSSL encryption API, and SM3 or SM4 is used to encrypt drives in dm-crypt.

GCC for openEuler

GCC is developed on open source GCC 10.3 to provide software and hardware collaboration, memory optimization, SVE vectorization, and math library and other functions.

- The compiler fully utilizes the hardware features of Arm processors to achieve higher operating efficiency. In the benchmark tests such as SPEC CPU 2017, it delivers much better performance than GCC 10.3 of the upstream community.
- It supports the mcmodel=medium, fp-model, quadruple-precision floating points, and vectorized math library.
- Automatic feedback optimization improves performance in scenarios such as databases.

Multiple GCC versions co-exist, including gcc-toolset-12 series whose installation packages run on GCC 12.2.0, to enhance Intel SPR features.

Feature Description

- **mcmodel=medium addressing:** Allows symbols (> 4 GB) to be properly accessed, which resolves the error caused by buffer overflow in the Arm architecture.
- **Quadruple precision:** Effectively improves the precision of 128-bit floating point arithmetic over the Arm architecture.
- **Vectorized math library:** Automatically searches for the available vectorized math library in the vectorization phase.
- **SVE vectorization:** Significantly improves program running performance for Arm-based machines that support SVE instructions.
- **SLP vectorization:** Analyzes and vectorizes reduction chains group and grouped_stores to improve stability of ongoing programs.

- **fp-model precision control:** Controls and refines the precision of floating-point calculations.
- **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 reduce the memory occupied by the structure and subsequent bandwidth pressure, while improving performance.
- **Pointer compression:** Compresses the 64-bit pointer in the structure field member to 8, 16, or 32 bits, which in turn reduces the memory occupied by the structure and 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 various application scenarios.
- **Automatic feedback:** Uses perf to collect and parse program information and optimizes this feedback in the compilation and binary phases, improving the performance of mainstream applications such as MySQL databases.

Application Scenarios

In the HPC test of Weather Research and Forecasting (WRF) and Microsystems Engineering and Materials Science (NEMO) applications, the GCC delivers 10% higher performance than GCC 10.3 of the upstream community.

BiSheng JDK

BiSheng JDK is an enhanced JDK developed based on Open Java Development Kit (OpenJDK). It features high performance and high availability and can supercharge production environments in any field or industry scenarios, and can optimize the performance especially for Arm-based scenarios. BiSheng JDK supports OpenJDK 8, 11, and 17 versions, and is compatible with Java Platform, Standard Edition (Java SE). BiSheng JDK offers the following advantages:

- **Stable and efficient:** In benchmark tests, such as SPECjbb 2015, BiSheng JDK delivers much better performance than OpenJDK.
- **Software and hardware collaboration:** BiSheng JDK fully utilizes the hardware features of Kunpeng servers for higher efficiency.
- **Premium security:** BiSheng JDK synchronizes release updates with OpenJDK community editions, performs strict analysis and control, and applies patches to CVEs as and when needed.
- **Open source:** BiSheng JDK provides releases and open source code, and continuously contributes to upstream communities.

Feature Description

- **Dynamic CDS (BiSheng JDK 8):** This technology extends application class-data sharing (AppCDS) for dynamic archiving of classes. It dumps classes loaded by Custom ClassLoader directly into a JSA file without creating a class list for every application, which creates a wider scope of shared classes and accelerates application startup.
- **Arm-based ZGC TBI optimization (BiSheng JDK 17):** The Top Byte Ignored (TBI) feature is introduced in ARMv8-A, to ensure hardware ignores the top byte (the most significant 8 bits) of a pointer when accessing memory. BiSheng JDK 17 now uses the TBI feature to implement Colored Pointer of ZGC on the AArch64 platform (replacing the original multi-mapping solution), effectively improving the Java ZGC performance and reducing dTLB load misses.

Application Scenarios

Application scenario 1: Big data

BiSheng JDK optimizes memory allocation and reclamation for GC in big data applications, and eliminates redundant memory barriers in JIT code. In related benchmark tests, BiSheng JDK delivers 5% to 20% higher performance than OpenJDK.

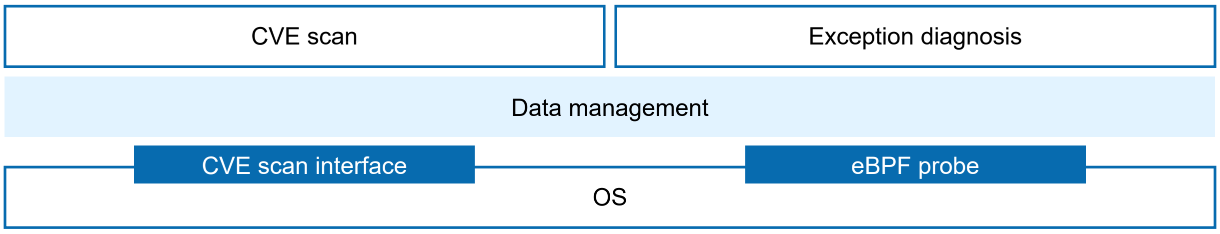
Application scenario 2: Java applications

BiSheng JDK optimizes the weak memory model of Kunpeng servers to avoid invalid memory barriers. NUMA-aware based on software and hardware collaboration improves the memory access efficiency of applications and fully unleashes application performance. BiSheng JDK enhances Java tools, such as JFR and jmap, to help developers quickly profile performance and locate faults.

A-Ops for Intelligent O&M

Big data and machine learning technologies are generating huge amounts of data, with estimates of a 2- to 3-fold increase every year. This trend is driving the need for new, intelligent O&M systems that improve efficiency at lower costs. openEuler A-Ops provides a smart O&M framework that runs premium capabilities such as CVE management and exception detection (for databases), to help enterprises quickly troubleshoot any issues and slash their O&M costs.

Feature Description



- Online CVE scans

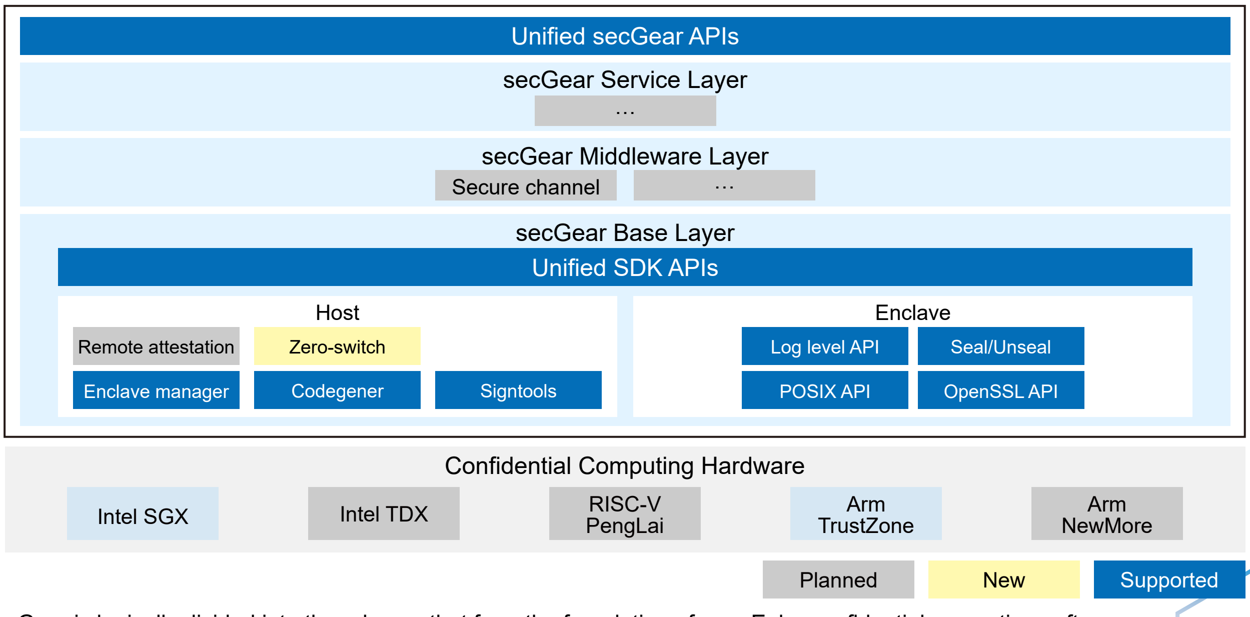
Online updates of the vulnerability database ensure systems are equipped to quickly scan for and fix live vulnerabilities with one click, ensuring the security of clusters with improved vulnerability remediation.

- Exception detection

Detects network I/O delays, packet loss, interruption, and high disk I/O loads in MySQL and openGauss service scenarios.

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) in the industry, secGear masks the differences between the TEE and SDK and simplifies development APIs by sharing the same set of source code over different architectures, reducing development and maintenance costs of confidential computing workloads. secGear streamlines the TEE ecosystem 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:** The unified layer of the confidential computing SDK provides unified APIs for different TEEs, enabling different architectures to share the same set of source code.

- **Middleware layer:** The general component layer provides confidential computing software for users to quickly build confidential computing solutions.
- **Service layer:** The confidential computing service layer runs dedicated solutions for typical situations.

The zero-switch feature uses shared memory to reduce the number of context switches and data copies, optimizing the interaction between the rich execution environment (REE) and TEE.

After confidential computing reconstruction, an application is typically divided into a non-secure client application (CA) and a secure trusted application (TA):

- Frequent calls from the CA to the TA cause prolonged calls, which severely affect the service performance.
- When large-block data is frequently exchanged between the CA and TA, multiple memory copies are generated during the enclave call (ECALL), deteriorating performance.

The zero-switch feature optimizes the interaction mechanisms and reduces performance loss caused by confidential computing reconstruction in the preceding two typical scenarios.

Feature Description

The zero-switch feature supports the following functions:

- ECALLs that improve call efficiency.
- Fast shared memory and zero-copy data exchange between the REE and TEE.
- Asynchronous zero-switch ECALLs.
- Configurable thread scheduling policies in the TEE, enabling flexible configuration of the resource mode for software and hardware resources.

Application Scenarios

Application scenario 1: Encrypted databases

An encrypted database provides SQL query and computing capabilities in the TEE. When a database client requests a query, the ciphertext is transferred to the TEE, sometimes in multiple times if the data volume is large. With a large number of query requests, the calls and data copies between the REE and TEE are frequently triggered, which significantly deteriorates performance. This feature implements zero-switch calls and zero-copy data exchange, greatly improving the end-to-end performance of data queries.

Application scenario 2: BJCA cryptographic module

This module supports SM2, SM3, and SM4 in the TEE and provides cryptographic services for external systems. The zero-switch feature reduces the context switches typically required for processing a large number of requests between the REE and TEE.

Collaboration Across the Edge and Cloud

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.

Feature Description

A complete edge computing platform supports collaboration across multiple components:

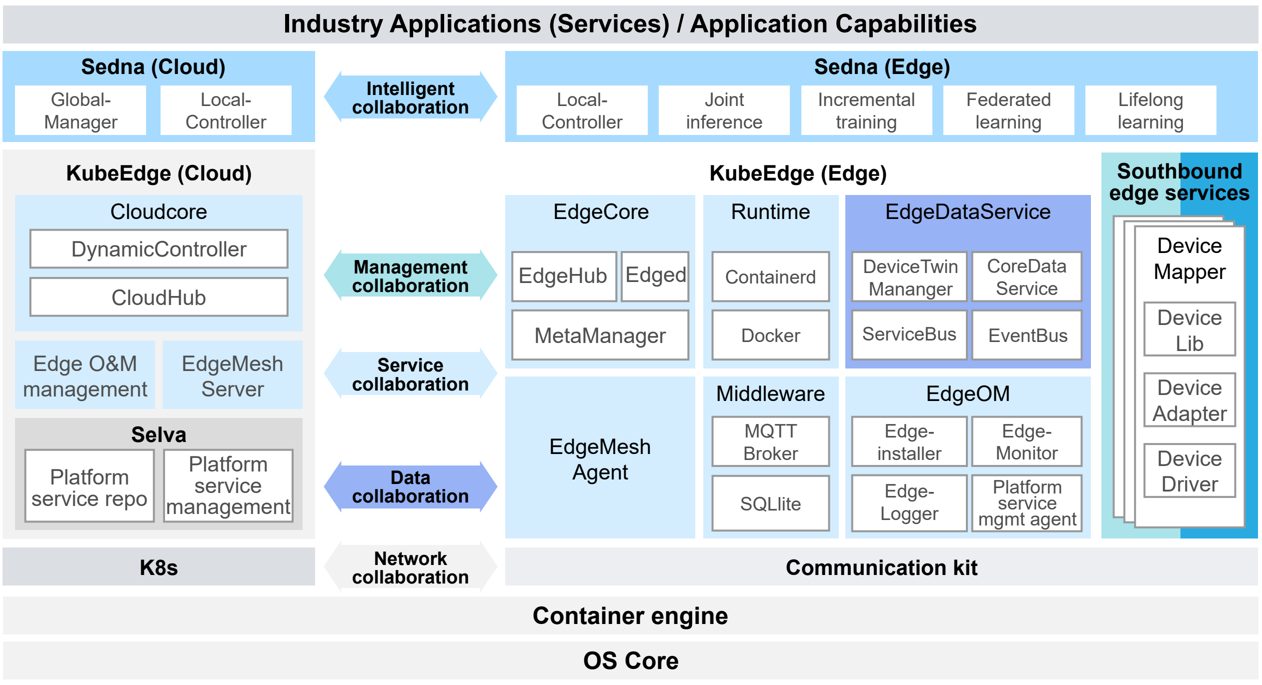
Network: The cross-subnet data communication mechanism allows edge nodes to process and share data with other nodes and with the computing center.

Service: The distributed service discovery mechanism combines with edge traffic governance policies to streamline service capabilities across the edge and cloud.

Data: Secure, efficient data transmission and synchronization between the cloud and edge, together with data collection, cleaning, and caching at the edge, help unify the management, processing, and operation of data in the computing center.

Management: The computing center gives a unified management platform to help operate resources, applications, and O&M.

Intelligence: Collaborative training, collaborative inference, and incremental learning are used to push AI applications seamlessly to the edge.



openEuler has supported KubeEdge for edge computing deployments since openEuler 22.03 LTS.

KubeEdge is an open source cloud-native edge computing platform project developed by Huawei Cloud in coordination with nearly 50 enterprises, organizations, and research centers. It is the first edge computing project that moves to the incubating level under the Cloud Native Computing Foundation (CNCF). KubeEdge is a leading cloud-native edge computing service, providing full-stack collaboration capabilities for resource management, data collaboration, and AI enablement. It is compatible with the Kubernetes ecosystem, and works with other applications in the ecosystem to build end-to-end edge computing solutions.

KubeEdge: As the base of the edge computing platform, KubeEdge is fully compatible with native capabilities of Kubernetes. It allows you to use native Kubernetes APIs to manage edge applications, devices, and data.

EdgeMesh: As the data plane tool of a KubeEdge cluster, EdgeMesh consolidates separated edge network environments, and discovers cloud-edge services and forwards traffic for applications. It masks complicated edge network topologies to simplify network configuration and uses the cloud-native service discovery mechanism and lightweight agents to collaborate across the edge and cloud.

Sedna: The first distributed edge-cloud AI framework enables collaborative inference, federated learning, incremental learning, and lifelong learning across the edge and cloud to better use AI applications in edge computing environments. Sedna simplifies the process of pushing AI algorithms to the edge and makes it easier to build AI services at the edge, thereby improving the end-to-end performance of AI systems.

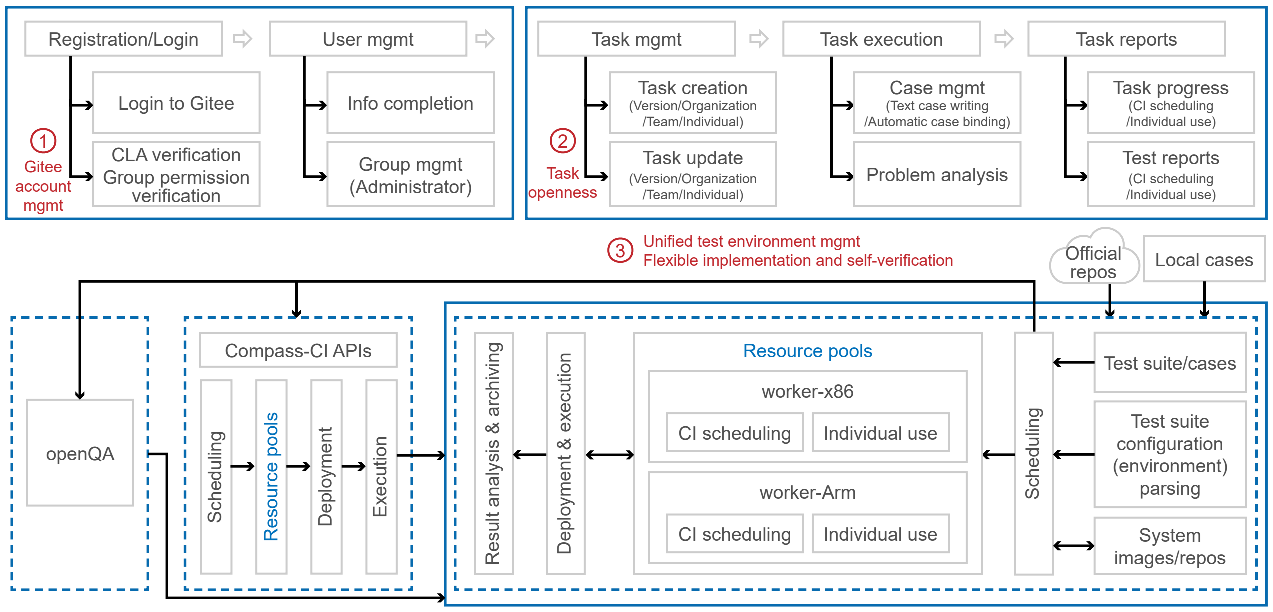
Application Scenarios

Edge-cloud collaboration scenarios, such as smart manufacturing, urban transportation, tollway inspection, smart gas stations, medical image recognition, and smart campuses.

radiaTest Community Test Platform

The radiaTest management platform carries end-to-end test activities in the openEuler community. Its key component is the web data mid-end, which streamlines and makes community version tests traceable, but also provides plugins for resource management and automated tests to connect to multiple test engines.

Feature Description


Registration/Login
User mgmt
Login to Gitee
Info completion
CLA verification
Group permission verification
Group mgmt (Administrator)
① Gitee account mgmt
Task mgmt
Task execution
Task reports
Task creation (Version/Organization/Team/Individual)
Task update (Version/Organization/Team/Individual)
Case mgmt (Text case writing/Automatic case binding)
Problem analysis
Task progress (CI scheduling/Individual use)
Test reports (CI scheduling/Individual use)
② Task openness
③ Unified test environment mgmt
Flexible implementation and self-verification
Official repos
Local cases
openQA
Compass-CI APIs
Scheduling
Resource pools
Deployment
Execution
Result analysis & archiving
Deployment & execution
Resource pools
worker-x86
CI scheduling
Individual use
worker-Arm
CI scheduling
Individual use
Scheduling
Test suite/cases
Test suite configuration (environment) parsing
System images/repos

- Management of static resources for physical machines, including password change, resource release, and system reinstallation; management of dynamic resources for VMs, with a web UI to hot-configure NICs and drives.
- Text case data management, version case baseline formulation, and case review.
- Milestone data management, data synchronization with the community's code repositories, version quality dashboards, and trusted tests.
- Test task data management, automated test triggering, and IT-based management of manual test cases; unified storage of test logs, with log splitting and marking by test step.

Application Scenarios

Application scenario 1: developer tests

The community test environment, including images, VMs, and containers, is managed on a unified platform, where test services are made available to connect partners' environments with the community environment. This allows for concurrent test scheduling and execution, as well as customized, self-service tasks.

Application scenario 2: version quality monitoring

For official community releases, the quality of multiple processes, such as software builds, AT execution, software change, test execution, problem closure, and requirement progress, is monitored using IT measures.

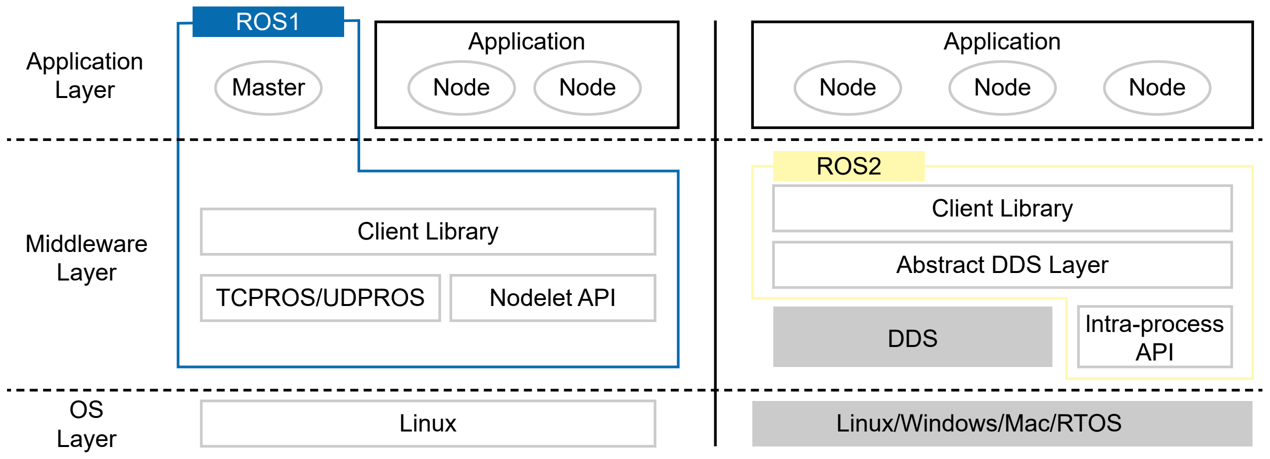
ROS

Robot Operating System (ROS) is a set of software libraries and tools designed to help you build robot applications. From drivers to algorithms and developer tools, ROS integrates standalone components to provide developers with a communication framework. ROS is not an actual OS, but a middleware that enables communication between the OS and developers' ROS applications. It can be seen as a runtime environment that runs on top of Linux, helping you efficiently organize and run robotic perception, decision making, and control algorithms.

openEuler supports two ROS versions, ROS 1 and ROS 2.

Feature Description

The following figure illustrates the ROS architecture:



The two key ROS components are available on openEuler:

- Communication framework
- Build tool

Application Scenarios

Robotics software development such as communication interaction and message processing. You can customize communication functions (synchronous or asynchronous) for topics and services to build a basic robotics framework.

openEuler WSL

Windows Subsystem for Linux (WSL) is an adaptation layer that allows you to run Linux user-mode software on Windows. You can download openEuler from Microsoft Store, to enjoy a native experience on Windows.

Feature Description

- **Out-of-the-box installation:** On Windows devices that support WSL, you can download the latest openEuler LTS version from Microsoft Store with just one click.
- **Full lifecycle support:** All openEuler LTS versions within the support period are available on Microsoft Store and are updated with official community versions.
- **User-friendly operations:** The openEuler WSL package is available on Microsoft Store, or build your own WSL applications using the open source code in the openEuler WSL repository.
- **Continuous feature updates:** Linux native UI applications, systemd, and other features will be supported in future openEuler WSL versions, to stay in line with Microsoft WSL iterations.

Application Scenarios

- Deploy and use an openEuler LTS version on Windows.
- Use Visual Studio Code and openEuler WSL to create a smooth cross-platform development experience.
- Build a Kubernetes cluster in openEuler WSL.
- Use openEuler command-line programs or scripts to process files and programs in Windows or WSL.

kiran-desktop-2.4

The Kiran desktop has two releases this year, kiran v2.3 that runs on openEuler 22.09, and kiran v2.4 that runs on openEuler 22.03 LTS SP1.

kiran v2.4 has the following added content over its predecessor:

- Front end of the control center
- Session management
- Qt themes

LoongArch Architecture

openEuler works with Loongson and other partners to port the openEuler 22.03 LTS version to the LoongArch architecture, during which 124 pull requests (PRs) were merged, 3,532 source packages and 14,378 binary packages were built, and over 100 build and test issues were resolved. To date, ISO files for CLFS, RootFS, Docker images, and the Loongson Beta4 version have been created but the official ISO version is still in development.

ShenWei Architecture

ShenWei provides the BIOS, Virtual Machine Manager (VMM), and OS functions to help schedule and manage hardware and software resources such as CPUs, memory, and processes. VMs can be used for secure virtual terminals, and new features are available in the openEuler community to support servers and workstations and provide virtualization capabilities.

PowerPC Architecture

Based on the open source RISC architecture, Hexin Tech has verified the openEuler PowerPC version on their HX-C1000 chip, meaning that openEuler PowerPC officially can be used for base development. The openEuler community has released the preview ISO version for early experience.

Intel Sapphire Rapids

openEuler 22.03 LTS SP1 is fully compatible with Intel Sapphire Rapids, ensuring the following key features are incorporated into the openEuler community:

- Intel Advanced Matrix Extensions (AMX) and virtualization support
- Intel Software Guard Extensions (Intel SGX) and virtualization support
- Intel Data Streaming Accelerator (Intel DSA)
- Sapphire Rapids new instructions and virtualization support
- Intel Platform Monitoring Technology (PMT) support
- Bus lock detection and rate limiting
- HBM EDAC support and enhanced MCA error recovery
- Intel PCIe non-transparent bridge driver

AMD EPYC Genoa

openEuler 22.03 LTS SP1 is fully compatible with AMD EPYC Genoa. Key features have been incorporated into the openEuler community, including:

Feature	Essential/Consider	Incorporated
crypto	Essential	Yes
sme	Essential	Yes
hwmon	Essential	Yes
RAS (edac/mce)	Essential	Yes
kvm 5-level pagetable	Essential	Yes
kvm support > 255 cpus	Essential	Yes
idle driver update	Essential	Yes
PerfMon V2-core events	Essential	Yes
PerfMon V2-uncore events	Essential	Yes
PerfMon V2-IBS	Essential	Yes
Perf BRS	Consider	Yes
Perf mem/c2c	Consider	Yes

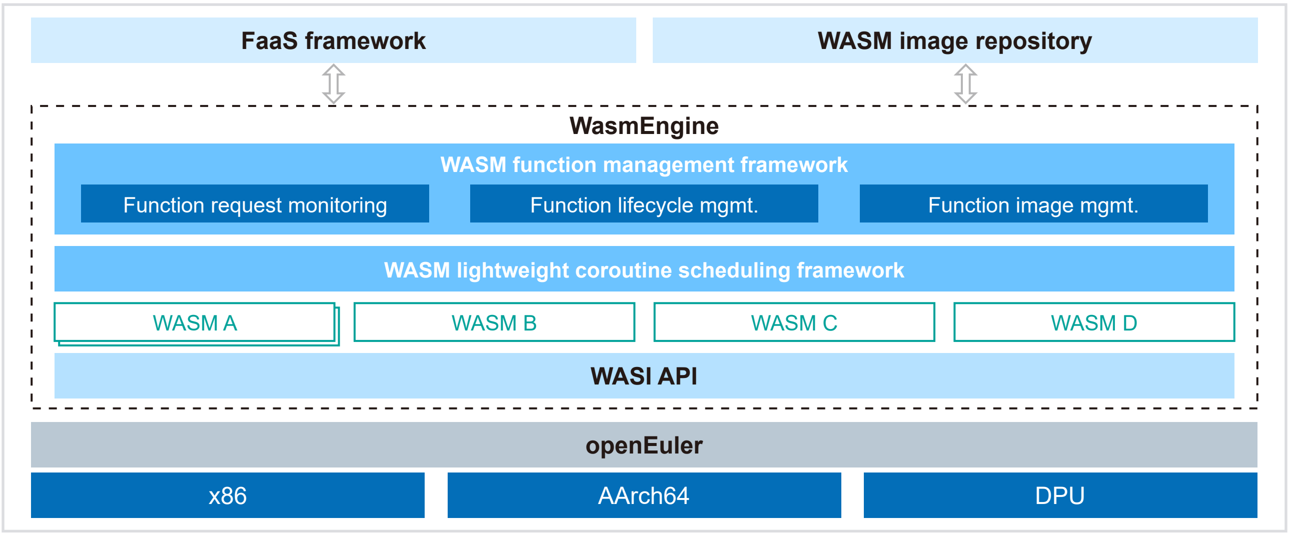
08/ Technical Preview

WasmEngine

Framework as a Service (FaaS) is a new computing paradigm of cloud computing. It features agile development, auto scaling, pay-per-use experience, and minimized O&M, helping users build any types of applications and services with ease. Conventional container-based FaaS decouples custom computing capabilities from content delivery network (CDN) services and implements fast iteration and updates. However, its cold start speed and memory overhead of containers make it insufficient for quick execution and processing of tens of thousands of instances on a single node, such as those in high-concurrency, heavy traffic scenarios.

To solve this, openEuler provides a WasmEngine sandbox solution based on the WebAssembly (WASM) technology to isolate functions in the WASM sandbox.

Feature Description



The functions of the lightweight WasmEngine are available thanks to the following two key components:

1. WASM function management framework

- Listens to and processes concurrent function requests.
- Manages functions throughout their lifecycle.
- Can work on Open Container Initiative (OCI) container images and manage local function image resources.

2. WASM lightweight coroutine scheduling framework

Abstracts the execution context of WASM instances, supports lightweight and high-performance user-mode coroutine scheduling models, and supports multiple WASM instance execution models such as JIT and AOT.

Application Scenarios

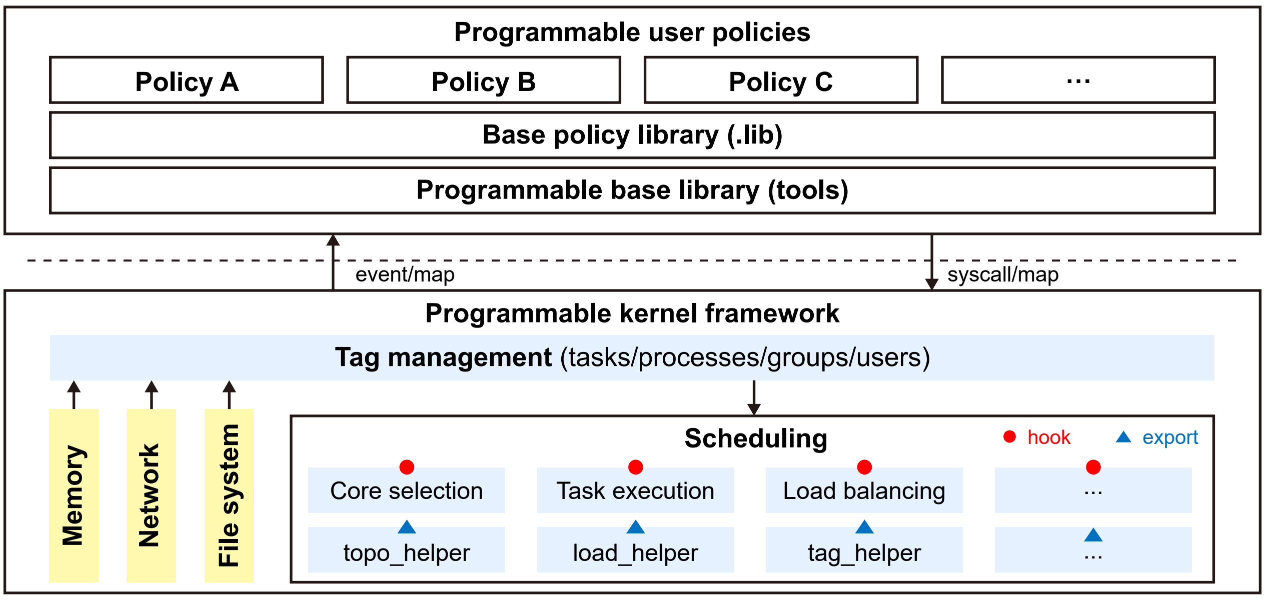
Stateless FaaS function tasks that run for a short period of time can be started on demand. For example, in the CDN edge computing scenario, custom request preprocessing functions allow for on-demand pulls and quick response.

eBPF-based Programmable Scheduling Framework

The eBPF-based programmable scheduling framework enables the kernel scheduler to extend scheduling policies and better meet varying loads. It has the following features:

- **Tag management mechanism:** The capability of tagging tasks and task groups is open. Users and kernel subsystems can tag specific workloads by calling interfaces. The scheduler can detect tasks of specific workloads by tag.
- **Policy extension:** The programmable scheduling framework supports policy extension for completely fair scheduling (CFS) preemption, core selection, and task execution, and adds new extension points and various auxiliary methods to extend policies.

Feature Description



- **Base library functions and policy library:** Provides basic library functions and custom scheduling policy templates for quick orchestration and extension of user-mode policies.
- **Tag management mechanism:** Supports user-defined extended tags for objects such as tasks, processes, groups, and users, and bears the semantics of collaborative scheduling between user-mode and kernel-mode components.
- **Scheduling component hook point and helper function:** Supports custom policy injections for CFS core selection, task execution, and preemption processes.

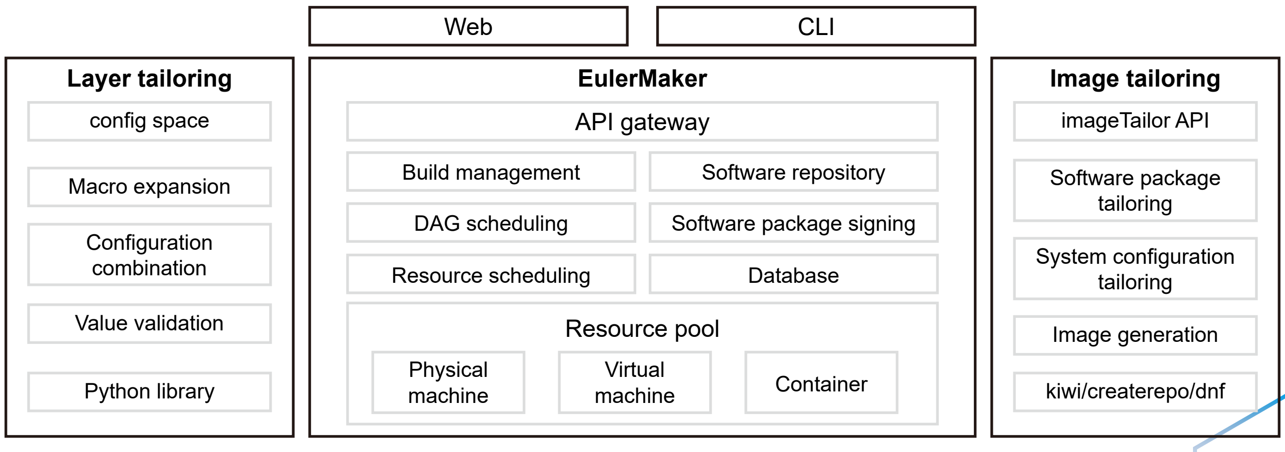
Application Scenarios

On the programmable kernel framework, developers and system administrators can create policies and dynamically load those policies to the kernel for execution.

EulerMaker Build System

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.
- **Gated build:** Listens to pull requests (PRs), uses dependency analysis to obtain the list of packages affected by changes, builds software packages, and verifies the installation of those packages.

- **Layer 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, embedded, and container OS images, and tailor the list of software packages and user login passwords for the images.

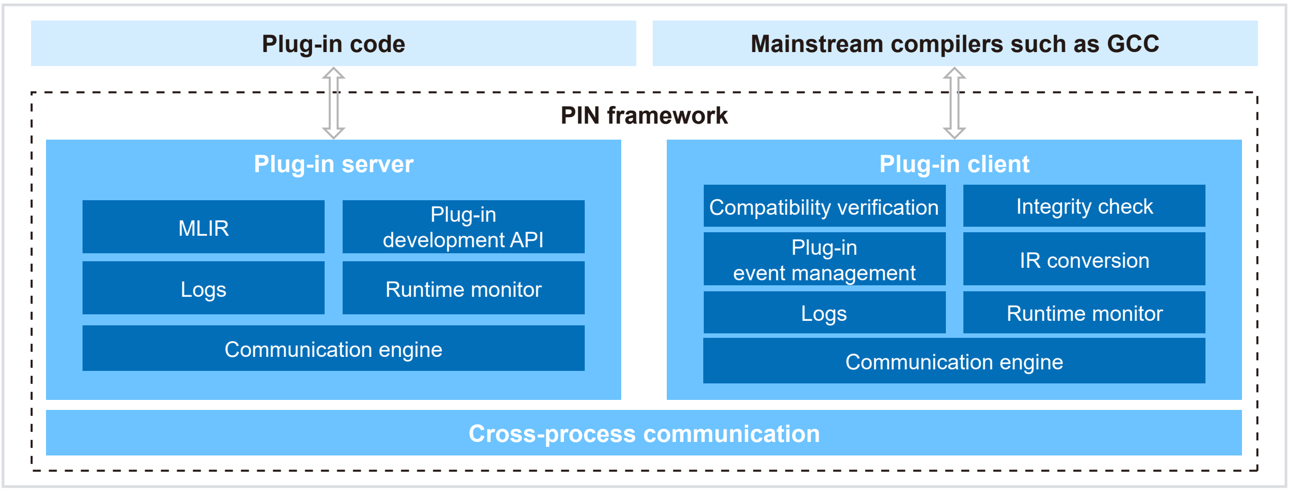
Application Scenarios

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

Plug-IN Framework

The Plug-IN (PIN) framework provides MLIR-oriented plug-in interfaces to help develop one plug-in for multiple compilers and optimize features using plug-ins, improving the development efficiency. The framework supports and maintains common capabilities such as tool compatibility and integrity check.

Feature Description



- MLIR-based plug-in development and easy conversion of intermediate representations such as GIMPLE.
- Common capabilities such as compatibility check and binary integrity check.
- Monitoring and verifying plug-in running, such as for compiler security and operation validity.
- Running plug-in clients in the form of GCC plug-ins, for plug-in functions to be executed without modifying the GCC compiler code.

Application Scenarios

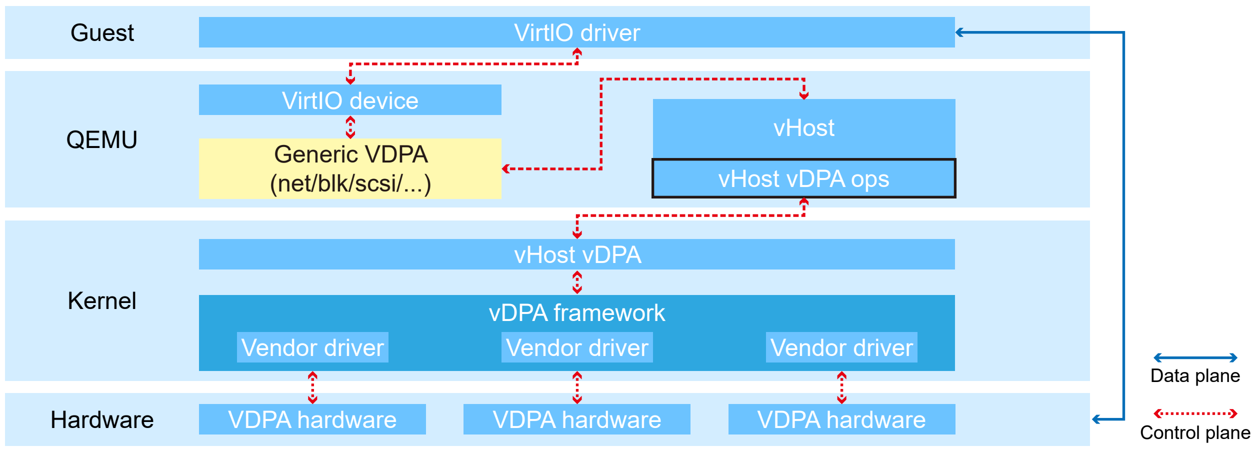
Those who frequently work on multiple compilers can use the PIN framework as the primary development platform, to develop tools on MLIR and run them on popular compilers such as GCC in the form of plug-ins, without modifying the compiler code.

Kernel-Mode vDPA Framework and Generic vDPA Device

The vHost Data Path Acceleration (vDPA) framework simplifies hardware vDPA implementation which in turn helps to develop and integrate hardware vDPA drivers. The framework can run on various hardware that supports data plane offload in kernel or user mode. Compared with user-mode vDPA, kernel-mode vDPA does not require an independent DPDK process that mounts the vDPA driver, making it suitable for offloading network, storage, and computing workloads, after which the host has less overhead and is easier to maintain.

The original kernel-mode vDPA framework mainly supports VirtIO-net devices, but Huawei successfully contributed the Generic vDPA Device to the Kernel and QEMU open source communities and the contribution has been incorporated. The Generic vDPA Device feature defines a set of general vDPA device models without distinguishing specific VirtIO device types, allowing a single framework to support all VirtIO devices, including VirtIO-net, VirtIO-blk, VirtIO-scsi, and VirtIO-fs.

Feature Description



- **Data and control plane isolation:** The data plane meets the VirtIO specifications, whereas the control plane uses a generic framework to interconnect with hardware from different vendors.
- **Generic vDPA devices:** openEuler supports simulated vDPA devices, but without the need to specify the device type, much like using Virtual Function I/O (VFIO) passthrough devices.
- **Kernel-mode vDPA framework:** The vDPA framework is added to the kernel to support interconnection with drivers from different vDPA hardware vendors.

Application Scenarios

Designed to meet VirtIO specifications of hardware devices, such as DPUs and SmartNICs from different vendors, and deliver a performance close to passthrough devices (theoretically). In the future versions, VMs can be live migrated between homogeneous and heterogeneous hardware devices.

QTFS

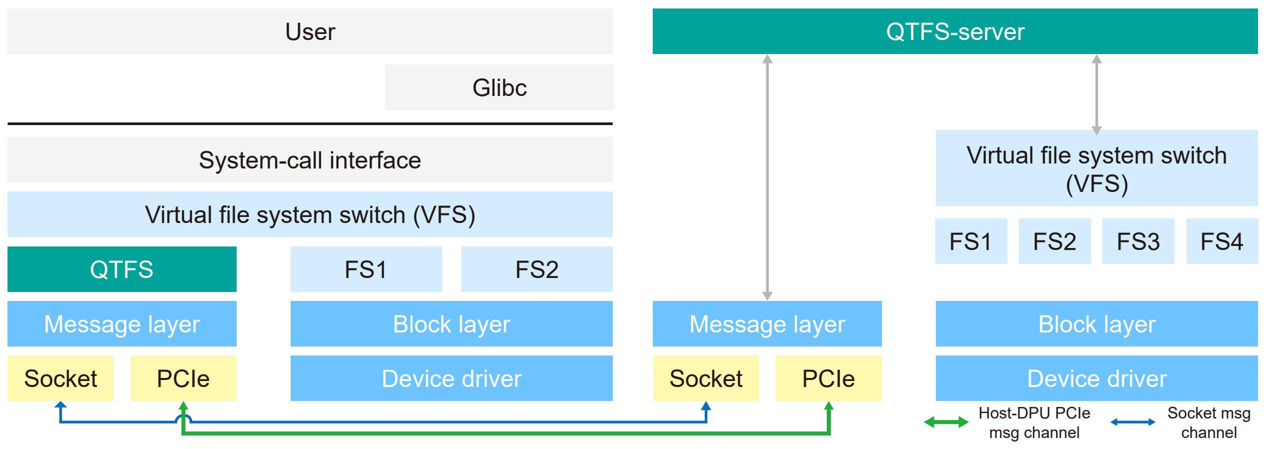
Quantum enTanglement File System (QTFS) is used for file system sharing between nodes. It is deployed on the host-DPU hardware architecture or between two servers, and works in client/server mode, to let clients access file systems on servers like accessing local file systems.

Feature Description

QTFS supports the following operations:

- Mount point propagation
- Sharing file systems such as proc, sys, and cgroup
- Transferring operations performed by clients on files in the qtfs directory to servers and sharing file reads and writes
- Remote mounting of a server's file systems on a client
- Customized processing of special files
- Remote FIFOs and UNIX sockets, and epoll, so that clients and servers can access these files in a way similar to through local communication
- PCIe underlying communication on the host-DPU architecture, at a higher performance level than network communication
- Modularized kernel development without intrusive modification to the kernel

Shared File System Framework



Application Scenarios

By offloading processes that run on the DPU management plane, QTFS masks environment differences between hosts and DPUs on the OS abstraction layer. It allows management processes, such as libvirt, docker, and kubelet, to seamlessly run on DPUs rather than hosts, to manage host service processes like VMs and containers. QTFS minimizes intrusive modification of management components and lowers upgrade and maintenance costs.

09/ Copyright

All materials or contents contained in this document are protected by the copyright law, and all copyrights are owned by openEuler, except for the content cited by other parties. Without a prior written permission of the openEuler community or other parties concerned, no person or organization shall reproduce, distribute, reprint, or publicize any content of this document in any form; link to or transmit the content through hyperlinks; upload the content to other servers using the "method of images"; store the content in information retrieval systems; or use the content for any other commercial purposes. For non-commercial and personal use, the content of the website may be downloaded or printed on condition that the content is not modified and all rights statements are reserved.

10/ Trademarks

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.

11/ Appendixes

Appendix 1: Setting Up the Development Environment

Environment Preparation	URL
Downloading and installing openEuler	https://www.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

Disclosure of Community Security Issues	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