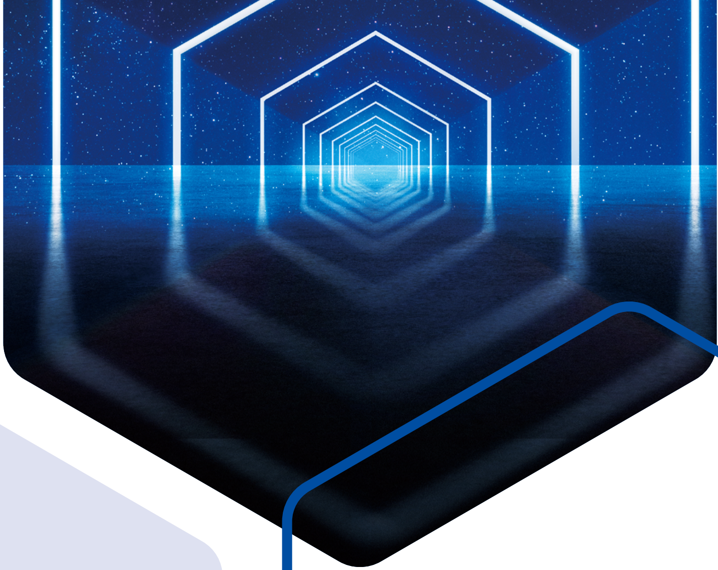

openEuler

openEuler 25.09

Technical White Paper



CONTENTS

Introduction 01

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

On September 30, 2024, openEuler 24.09 was released. It is an innovation version built based on Linux kernel 6.6 and brings more advanced features and functions.

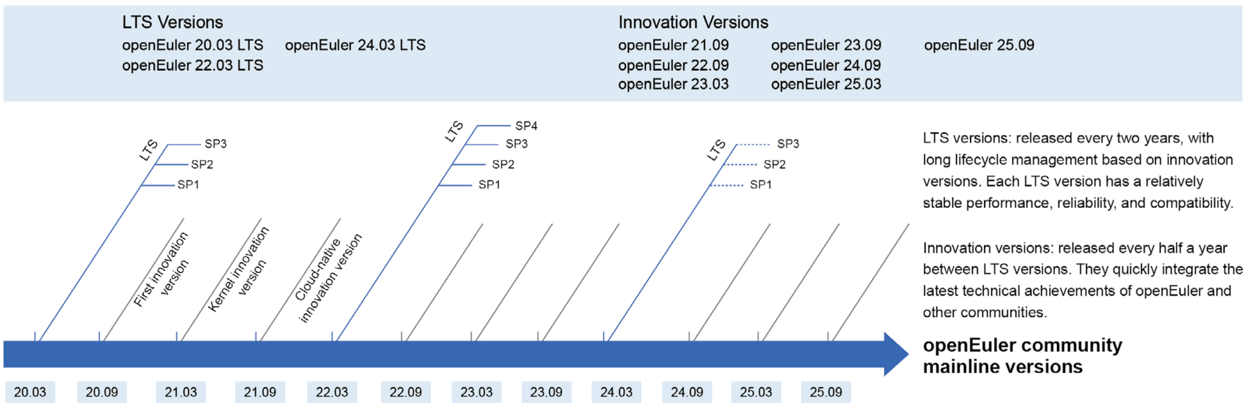
On December 30, 2024, openEuler 24.03 LTS SP1 was released. This enhanced and extended version of openEuler 24.03 LTS is developed on Linux kernel 6.6 and designed for server, cloud, edge computing, and embedded deployments. It offers new features and enhanced functions that streamline processes across a range of domains.

On March 30, 2025, openEuler 25.03 was released. It is an innovation version designed based on Linux kernel 6.6 and is suited for server, cloud, edge, and embedded scenarios. It provides a variety of new features and functions and brings brand-new experience to developers and users in diverse industries.

On June 30, 2025, openEuler 24.03 LTS SP2 was released. It is an enhanced version of 24.03 LTS based on the Linux 6.6 kernel. It is designed for server, cloud, AI, and embedded scenarios, introducing new functions and features covering kernel optimization, heterogeneous collaborative inference, high-density many-core computing, confidential containers, and multi-core and multi-instance hybrid deployment. It offers enhanced experience for developers and users across various industries.

More recently, on September 30, 2025, openEuler 25.09 was released. It is an innovation version based on the Linux 6.6 kernel. It offers new features and enhanced functionality for server, cloud, AI, and embedded scenarios. Key improvements include kernel optimization, heterogeneous collaborative inference, high-density many-core computing, confidential containers, confidential VMs, multi-core and multi-instance hybrid deployment, compilers, trusted computing, and cryptographic acceleration. This latest release enhances the experience for developers and users across a wide range of fields.

openEuler Version Management

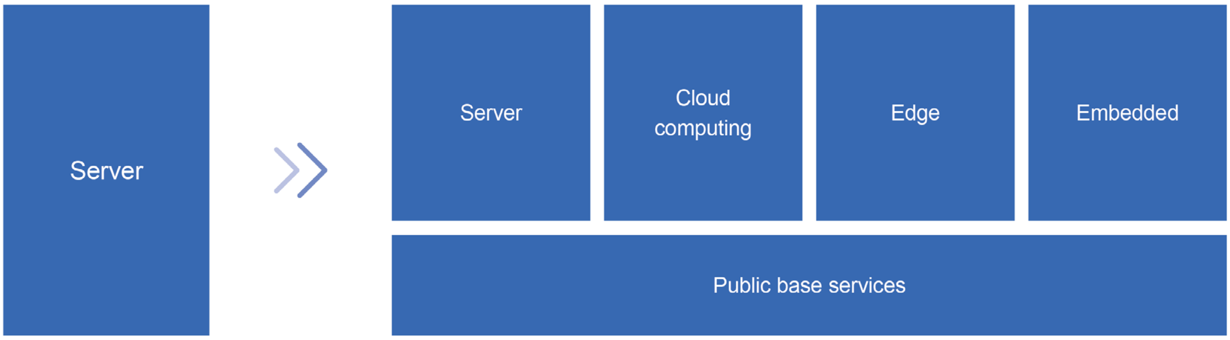


openEuler releases LTS versions to provide enhanced specifications and a secure, stable, and reliable OS for enterprise users.

openEuler is built on tried-and-tested technologies. Its innovation versions 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, LoongArch, and PowerPC), 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.

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 Software Supply Chain

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

Platform Architecture 02

System Framework

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

openEuler 25.09 runs on Linux kernel 6.6 and provides POSIX-compliant APIs and OS releases for server, cloud native, edge, and embedded environments. It is a solid foundation for intelligent collaboration across hybrid and heterogeneous deployments. openEuler 25.09 is equipped with a distributed soft bus and Kubernetes edge-cloud collaboration framework, among other premium features, making it a perfect choice for collaboration over digital infrastructure and everything connected models.

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

Cloud base

- **KubeOS for containers:** In cloud native scenarios, the OS is deployed and maintained in containers, allowing the OS to be managed based on Kubernetes, just as service containers.
- **Secure containers:** Compared with the traditional Docker+QEMU solution, the iSulad+shimv2+StratoVirt secure container solution reduces the memory overhead and boot time by 40%.
- **Confidential containers:** The iSulad+Kuasar+secGear confidential container solution offers privacy protection while remaining compatible with the cloud-native ecosystem.
- **Confidential computing:** Based on the Arm Confidential Compute Architecture (Arm CCA), this is the first community release to support CCA confidential VMs, providing confidentiality and integrity protection for data and code, even against privileged infrastructure software or cloud service providers.

New scenarios

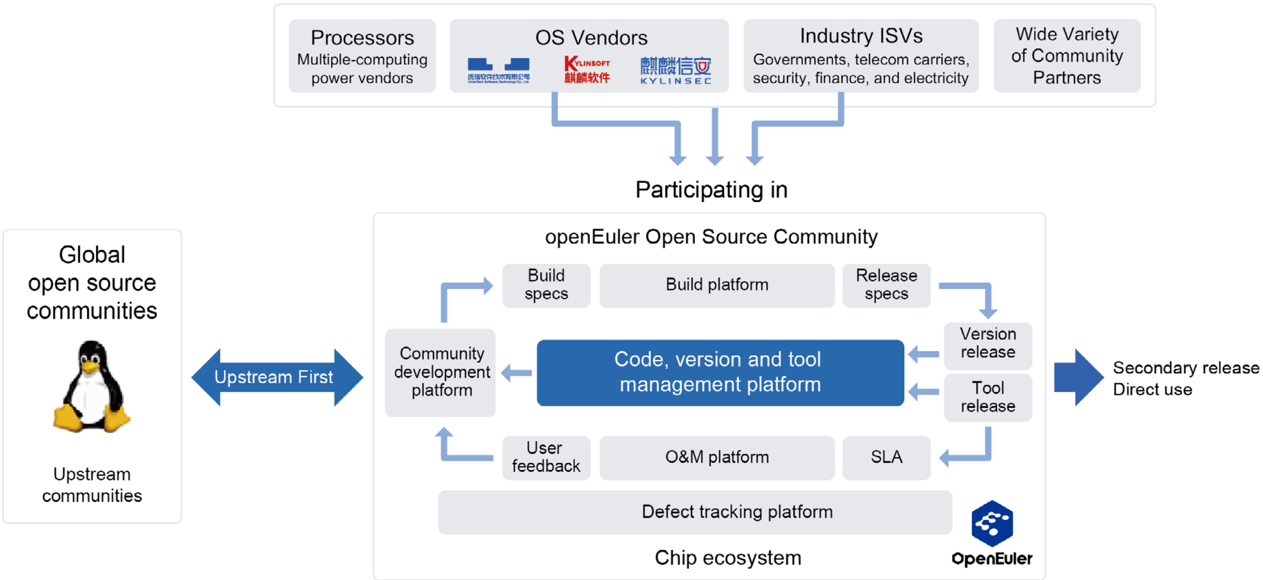
- **Edge computing:** openEuler 25.09 Edge is released for edge computing scenarios. It integrates the Kubernetes edge-cloud collaboration framework to provide unified management, provisioning of edge and cloud applications, and other capabilities.
- **Embedded:** openEuler 25.09 Embedded features an image smaller than 5 MB and a boot time of under 5 seconds. Its elastic virtualization base and mixed-criticality (MICA) deployment framework support concurrent deployment of multiple cores and instances. Cross-OS communication provides an efficient shared-memory mechanism for communication between different OSs.
- **AI-native OS:** The OS enables AI software stacks with out-of-the-box availability. Heterogeneous convergence of memory, scheduling, and training/inference resources reduces AI development costs and improves efficiency. The intelligent interaction platform of the OS streamlines development and administration.

Flourishing community ecosystem

- **Desktop environments:** UKUI, DDE, Kiran-desktop, and GNOME.
- **openEuler DevKit:** It supports OS migration, compatibility assessment, and various development tools such as secPaver which simplifies security configuration.
- **DevStation:** It offers an intelligent LiveCD desktop workstation that delivers an out-of-the-box, efficient, and secure development environment, streamlining the entire process from coding and deployment to building, packaging, and release. Additionally, it integrates a Model Context Protocol (MCP) AI engine to quickly invoke the community toolchain, improving efficiency from infrastructure setup to application development.

Platform Framework

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



Hardware Support

The openEuler open source community works with multiple vendors to build a vibrant southbound ecosystem. With participation of major chip vendors including Intel and AMD, all openEuler versions support x86, Arm, and RISC-V CPU architectures, and a wide range of CPU chips, such as Zhaoxin KaiXian and KaiSheng, Intel Sierra Forest and Granite Rapids, and AMD EPYC 4/5. 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_64	AArch64	RISC-V
CPU	Intel, AMD, Hygon, Zhaoxin	Kunpeng, Phytium	Sophgo, THead, etc.

Visit https://www.openeuler.org/en/compatibility/ to see the full hardware list.

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, RISC-V
Memory	≥ 4 GB
Drive	≥ 20 GB

VMs

Verify VM compatibility when installing openEuler.

Hosts running on openEuler 25.09 support the following software packages:

- libvirt-9.10.0-18.oe2509
- libvirt-client-9.10.0-18.oe2509
- libvirt-daemon-9.10.0-18.oe2509
- qemu-8.2.0-47.oe2509
- qemu-img-8.2.0-47.oe2509

openEuler 25.09 is compatible with the following guest OSs for VMs:

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

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

Edge Devices

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

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

Embedded Devices

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

Item	Configuration Requirement
Architecture	AArch64, AArch32, x86_64
Memory	≥ 512 MB
Drive	≥ 256 MB

Scenario-specific Innovations 04

AI

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

OS for AI

sysHAX

Feature Description

The sysHAX large language model (LLM) heterogeneous acceleration runtime enhances model inference performance in single-server, multi-xPU setups by optimizing Kunpeng + xPU (GPU/NPU) resource synergy.

- **Heterogeneous converged scheduling:** When GPUs are fully loaded, the prefill phase of an inference request can be executed on GPUs while the decode phase is handled by CPUs.

Application Scenarios

sysHAX is used to optimize Transformer models including DeepSeek, Qwen, Baichuan, and Llama. It fits into the following application scenario:

- **Data centers:** sysHAX assigns inference tasks to CPUs to fully utilize CPU resources and increase the concurrency and throughput of LLMs.

GMEM

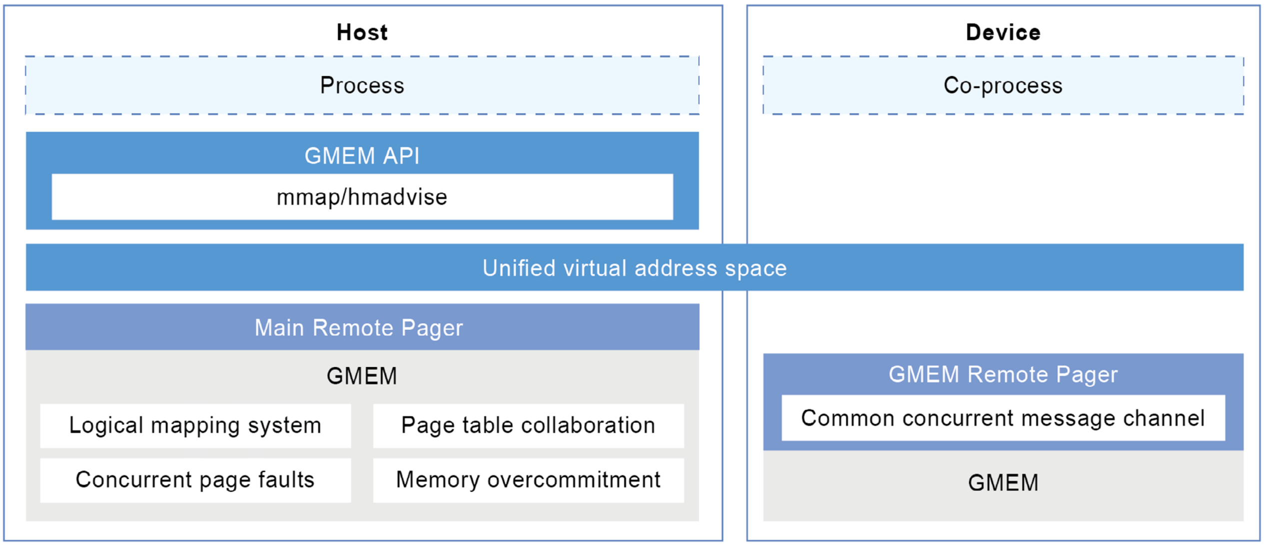
In the post-Moore era, there have been breakthroughs in GPUs, TPUs, FPGAs, and other dedicated heterogeneous accelerators. Similar to CPUs, these devices increase computing speeds by storing data in local memory (such as LPDDR SDRAM or HBM), but such design catalyzes more complicated memory systems.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

- Memory management is split between CPUs and accelerators. Explicit data migration makes it difficult to balance the usability and performance of accelerators' memory.
- The high bandwidth memory (HBM) available on accelerators is often insufficient for foundation models. Manual swapping is only feasible in limited scenarios and typically results in significant performance degradation.
- A large number of invalid data migrations occur in search & recommendation and big data scenarios, and no efficient memory pooling solution is available.

Heterogeneous Memory Management (HMM) is a Linux feature that is plagued by issues of poor programming, performance, and portability, while also relying heavily on manual tuning. As such, it is unfavored by most OS communities, and has fueled demand for an efficient solution for heterogeneous accelerators. Generalized Memory Management (GMEM) is one new option, which offers a centralized management mechanism for heterogeneous memory connections. GMEM APIs are compatible with native Linux APIs, and feature high usability, performance, and portability. After an accelerator calls

GMEM APIs to connect its memory to the unified address space, the accelerator automatically obtains the programming optimization capability for heterogeneous memory, and does not need to execute the memory management framework multiple times. This greatly reduces development and maintenance costs. Developers can apply for and release a unified set of APIs to achieve heterogeneous memory programming without memory migrations. If the HBM of an accelerator is insufficient, GMEM can use the CPU memory as the accelerator cache to transparently over-allocate the HBM without manual swapping. GMEM offers an efficient memory pooling solution thanks to a shared memory pool that eliminates the need for duplicate migrations.

Feature Description



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

- **GMEM**

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

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

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

- **Remote Pager**

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

- **User APIs**

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

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

- **Constraints**

  GMEM supports 2 MB huge pages only. Therefore, transparent OS huge pages must be enabled on hosts and NPUs to use GMEM.

  The heterogeneous memory obtained using **MAP_PEER_SHARED** cannot be inherited during fork.

  For details about how to use GMEM, visit:

  https://gitee.com/openeuler/docs-centralized/tree/master/docs/en/docs/GMEM

Application Scenarios

- Heterogeneous unified memory programming

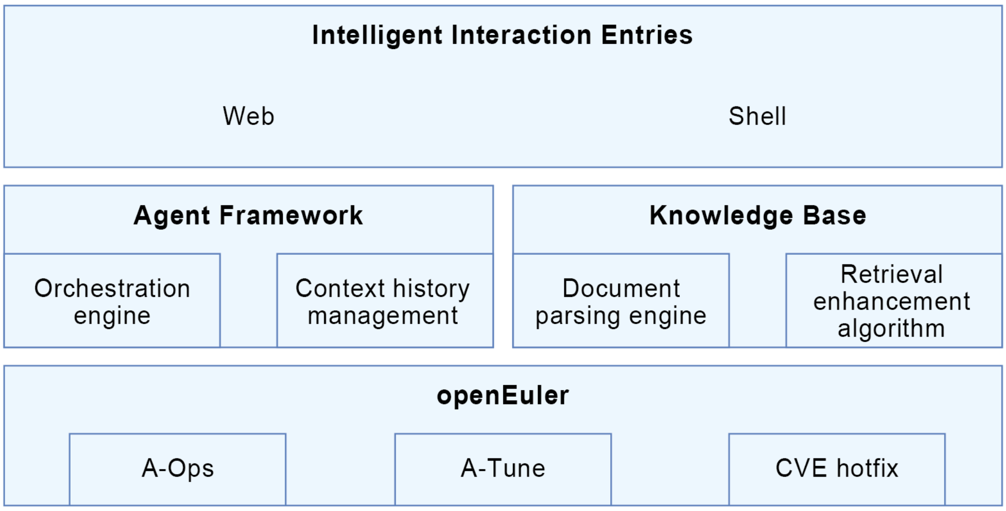
  To simplify heterogeneous memory programming, GMEM can allocate a unified virtual memory to CPUs and accelerators, which then share the same pointer. For example, an NPU memory management system can be connected to GMEM with just 100 lines of modified code in the NPU driver, a huge reduction on the original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

- Automatic memory overcommitment of an accelerator

  When a GMEM API is used to allocate memory to an application, the application is not limited by the physical memory capacity of the accelerator. Instead, the application can transparently over-allocate memory until the CPU's DRAM capacity is exhausted. GMEM swaps cold device memory pages out to the CPU memory to achieve high-performance and easy-to-start training and inference. This design makes GMEM deliver 1.6 times the performance of NVIDIA-UVM in ultra-large model training when the overcommitment ratio is 200%. (Test results compared the Ascend 910 NPU and NVIDIA A100 GPU under the same HBM conditions.)

AI for OS

AI makes openEuler more intelligent. openEuler Intelligence is an AI-powered Q&A platform built on openEuler data. It enables workflow orchestration through semantic interfaces and agent building through Model Context Protocol (MCP). Additionally, it integrates some system services to further improve the intelligence of openEuler.



Intelligent Q&A

Feature Description

The openEuler Intelligence system is accessible via web or shell.

- **Web:** Through the visual web entry, users can easily build and use the knowledge base, run automated tests on the knowledge base, and view test evaluation results. They can also register OpenAPI APIs, build and use workflow applications based on them, register and activate MCP, and build and use MCP-based agent applications. The web entry helps beginners access openEuler knowledge and leverage openEuler's AI capabilities effortlessly.
- **Shell:** Through the shell entry, users can trigger intelligent Q&A or pre-integrated agent applications within the agent framework. The shell entry allows O&M personnel to interact with openEuler in an OS-affinity and intelligent manner, reducing O&M costs.

Intelligent Planning, Scheduling, and Recommendation

- **Intelligent planning:** The agent applications of openEuler Intelligence can plan steps in real time based on user input and available tools, continuing until the user's objective is achieved or the maximum number of steps is reached.
- **Intelligent scheduling:** openEuler Intelligence allows users to define multiple workflows within a workflow application. When a query request is made, openEuler Intelligence automatically extracts the relevant parameters and selects the most suitable workflow to execute the query task.
- **Intelligent recommendation:** Based on users' query requests and workflow execution results, openEuler Intelligence recommends workflows that may be useful in future tasks, increasing the likelihood of task completion and making applications easier to use.

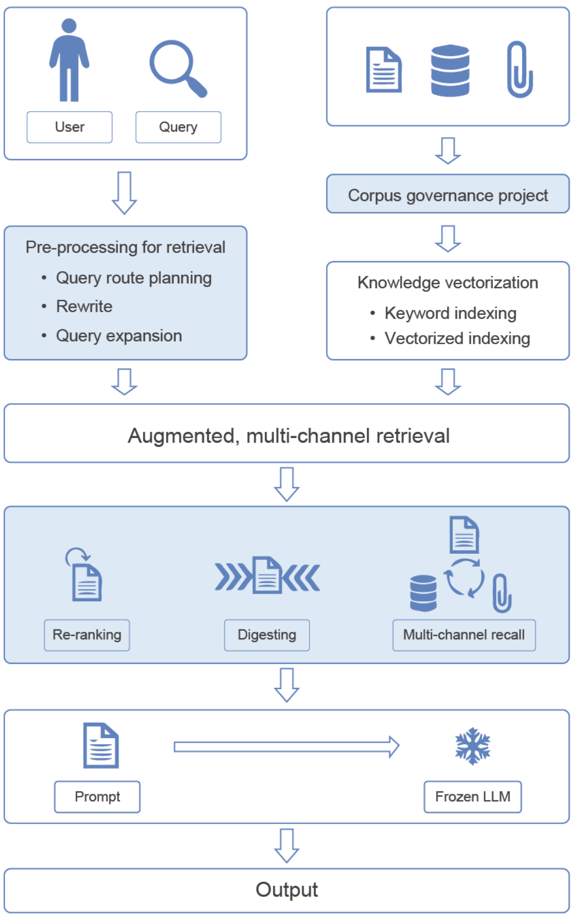
Workflow Applications

- **Semantic interfaces:** A semantic interface contains natural language comments. openEuler Intelligence supports two methods for registering semantic interfaces. The first method allows users to register APIs with openEuler Intelligence as OpenAPI (3.0 or later) YAML files. When writing an OpenAPI YAML file, users can add natural language comments to the APIs. When these APIs are called, the foundation model selects the right APIs and sets their parameters based on the comments. The second method allows users to register functions with openEuler Intelligence as Python code, and this method provides a more user-friendly option for using openEuler Intelligence. The semantic interfaces generated in both methods can be orchestrated into workflows in a visualized manner.
- **Workflow orchestration and invocation:** openEuler Intelligence enables users to visually connect built-in semantic interfaces and user-registered interfaces to create workflows. Users can debug these workflows, and then release and use them as applications. When workflows are debugged and executed, intermediate results are displayed to help lower debugging costs and enhance the overall user experience.

Agent Applications

- **MCP registration, installation, and activation:** MCP is a mainstream AI-related protocol. It uses SDKs to encapsulate complex and diverse services with natural semantic information, allowing AI to easily invoke tools and services reconstructed based on MCP.
- **Agent building and use:** openEuler Intelligence allows building agents based on MCP and various foundation models. These agents can decompose a user's objective into phased tasks using the configured model information and the user-provided objective. MCP tools are then used to complete each task until the user's objective is achieved.

RAG



Retrieval-augmented generation (RAG) is a technique for enhancing the long-term memory capability of large language models (LLMs). Used by the openEuler Intelligence system, RAG is essential to reducing model training costs and has the following highlights:

- **Pre-processing for retrieval:** When a user's query request lacks sufficient information, it can be rewritten based on historical context and inferred intent to enhance retrieval accuracy.
- **Knowledge indexing:** For documents of various formats and content types, openEuler Intelligence offers document parsing capabilities such as summarization, text feature extraction, tree-structured parsing, and optical character recognition (OCR). These capabilities help generate segment-level feature indexes to increase hit rates.
- **Retrieval enhancement:** openEuler Intelligence supports eight retrieval methods. Among them, the methods using dynamic keyword weighting and LLM-based filtering significantly boost out-of-the-box accuracy.
- **Post-processing for retrieval:** For fragments with missing context, a uniform randomized context completion method is provided to enhance the completeness of fragment information and reduce hallucination in model fitting. For fragments (sets) with the maximum number of tokens, various methods are provided to enable intelligent Q&A for scenarios with limited resources. These methods include the reranking method based on the Jaccard distance, token compression method that removes common words and applies random discarding, and token truncation method based on binary search.

These capabilities enable RAG for the openEuler Intelligence system to accommodate to more document formats and content types, and enhance Q&A services with minimal impact on system load.

Corpus Governance

Corpus governance is one of the basic RAG capabilities in the openEuler Intelligence system. It imports corpuses into the knowledge base in a supported format using context location extraction, text summarization, and OCR, increasing the retrieval hit rate.

- **Context location extraction:** The relative location relationship within a document is retained. Specifically, the global and local relative offsets of each segment are recorded to support context completion.
- **Text summarization:** Sliding windows and LLMs are used to generate text summaries for complex documents or segments, supplying foundational data for multi-level retrieval.
- **OCR augmentation:** For documents containing both images and text, summaries are generated based on image text and surrounding context to provide foundational data for answering image-related questions.
- **Document source tracing:** During answer generation, openEuler Intelligence displays a note in the lower right corner of the corresponding sentence based on the returned segment and related document information. Users can click the note to view the document name and abstract, enhancing the reliability of question answering.

Automated Testing

Automated testing is a core RAG capability in openEuler Intelligence. It detects shortcomings in the knowledge base and retrieval augmentation algorithms through automated dataset generation and evaluation.

- **Dataset generation:** After a user selects a parsed document, openEuler Intelligence randomly samples a portion of the parsing results and sends them to an LLM. The LLM then generates and filters Q&A pairs to create a high-quality test dataset that contains queries, reference answers, and original segments.
- **Test evaluation:** Automated tests are conducted using the user-generated dataset and the predefined parameters such as the retrieval augmentation algorithm, LLM, and top-k fragments. Tests score are calculated based on the queries, reference answers, original fragments, associated fragments, and model fitting results in the test dataset. After that, the test results are evaluated using metrics such as the precision rate, recall rate, fidelity value, explainability, longest common substring score, edit distance score, and Jaccard similarity coefficient.

Application Scenarios

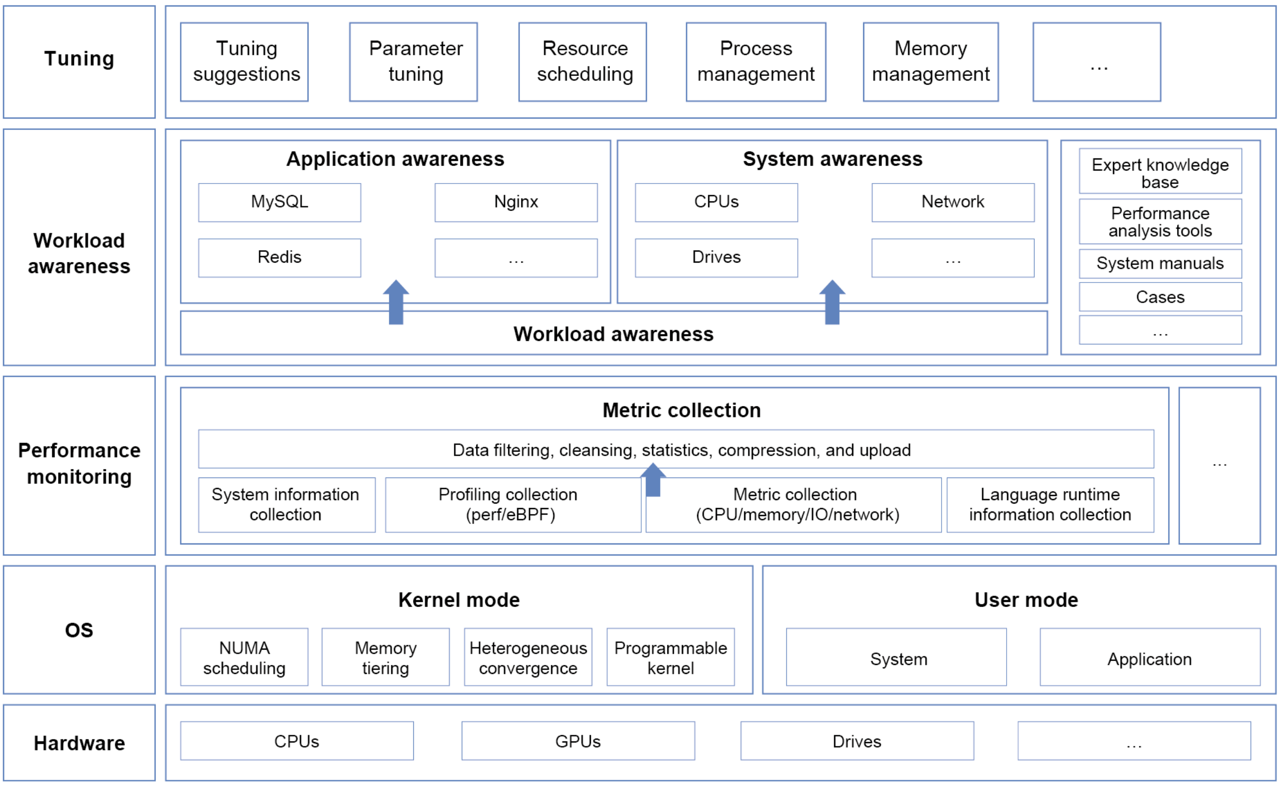
- Common openEuler users can build customized smart assistants using local files or documents from the openEuler community, such as white papers.
- openEuler developers can build custom workflows or agent applications using the web client, reducing repetitive work in certain scenarios, such as code audits.
- openEuler O&M personnel can interact with the OS using natural language through the shell, reducing the effort required to construct and modify complex commands.

For details, see openEuler Intelligence.

Intelligent Tuning

Feature Description

The openEuler Intelligence system supports the intelligent shell entry. Through this entry, you can interact with openEuler Intelligence using a natural language and perform heuristic tuning operations such as performance data collection, system performance analysis, and system performance tuning.

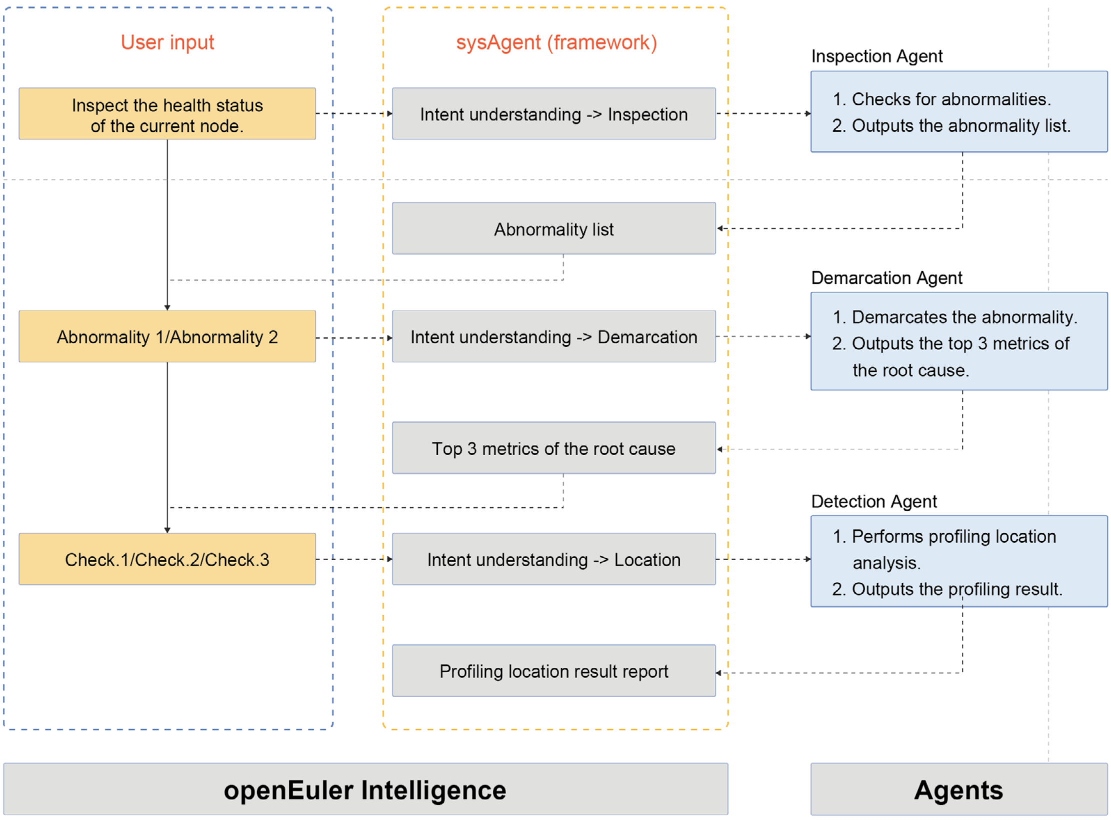


Application Scenarios

- **Gaining insights from key performance metrics:** You can learn about the system performance status based on collected performance metrics like CPU, I/O, drive, network, and application.
- **Analyzing system performance:** Performance analysis reports are generated, making it easier to locate performance bottlenecks across the entire system and in individual applications.
- **Receiving performance tuning suggestions:** The openEuler Intelligence system generates a performance tuning script, which can be executed with one click to tune the system and specific applications.

Intelligent Diagnosis

Feature Description



1. **Inspection:** The Inspection Agent checks for abnormalities of designated IP addresses and provides an abnormality list that contains associated container IDs and abnormal metrics (such as CPU and memory).
2. **Demarcation:** The Demarcation Agent analyzes and demarcates a specified abnormality contained in the inspection result and outputs the top 3 metrics of the root cause.
3. **Location:** The Detection Agent performs profiling location analysis on the root cause, and provides useful hotspot information such as the stack, system time, and performance metrics related to the root cause.

Application Scenarios

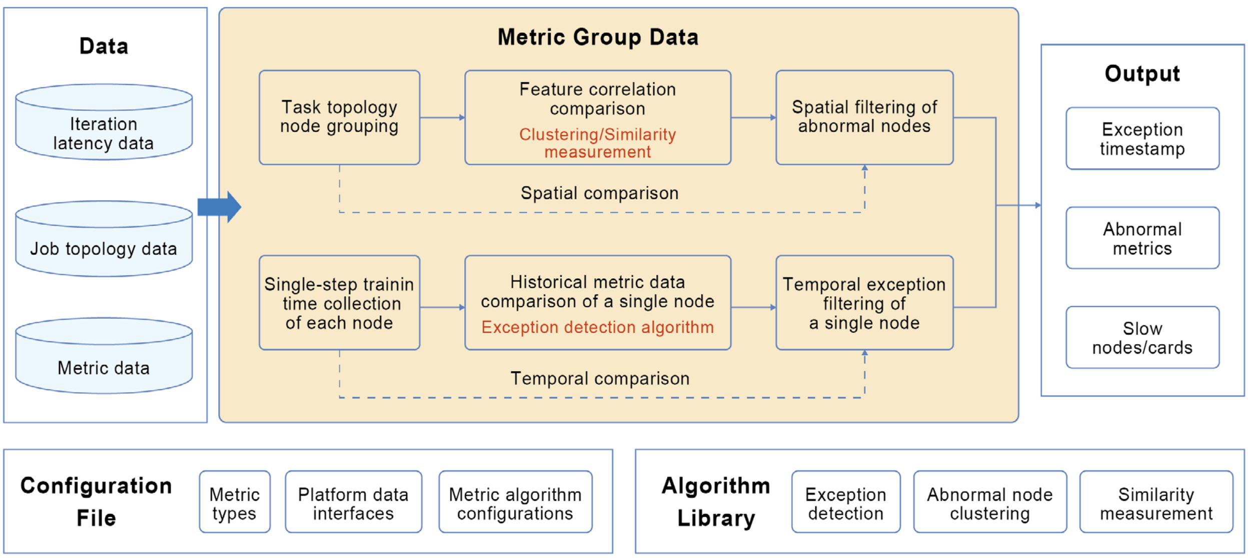
In openEuler 25.09, the intelligent shell entry enables capabilities like single-node abnormality inspection, demarcation, and profiling location.

- The inspection capabilities refer to single-node performance metric collection, performance analysis, and abnormality inspection.
- The demarcation capability is to locate the root cause based on the abnormality inspection result and output the top 3 metrics of the root cause.
- The profiling location capability refers to using a profiling tool to locate the faulty modules (code snippets) based on the root cause.

AI Cluster Slow-Node Demarcation

Performance degradation during AI cluster training is inevitable and often results from a wide range of complex factors. Existing solutions rely on log analysis after performance degradation occurs. However, it can take 3 to 4 days from log collection to root cause diagnosis and issue resolution on the live network. To address these pain points, an online slow node detection solution is offered. This solution allows for real-time monitoring of key system metrics and uses model- and data-driven algorithms to analyze the observed data and pinpoint slow or degraded nodes. This facilitates system self-healing and fault rectification by O&M personnel.

Feature Description



Grouped metric comparison helps detect slow nodes and cards in AI cluster training. This technology is built on Systrace and includes a configuration file, an algorithm library, and a slow node analysis mechanism based on both time and space dimensions. It outputs the exception timestamp, abnormal metrics, and IP addresses of slow nodes and cards. This technology enhances overall system stability and reliability.

- **Configuration file:** Contains the types of metrics to be observed, configuration parameters for the metric algorithms, and data interfaces, which are used to initialize the slow node detection algorithms.
- **Algorithm library:** Includes common time series exception detection algorithms, such as Streaming Peaks-over-Threshold (SPOT), k-sigma, abnormal node clustering, and similarity measurement.
- **Data:** Metric data collected from each node is represented by a time sequence.
- **Grouped metric comparison:** Supports spatial filtering of abnormal nodes and temporal exception filtering of a single node. Spatial filtering identifies abnormal nodes based on the exception clustering algorithm, while temporal exception filtering determines whether a node is abnormal based on the historical data of the node.

Application Scenarios

Systrace provides capabilities for detecting slow nodes, displaying alarms, and writing exception information to drives.

- **AI model training:** This feature quickly detects slow nodes for training jobs in large-scale AI clusters, facilitating system self-healing and fault rectification by O&M personnel.
- **AI model inference:** This feature identifies performance degradation across multiple instances of the same model. By comparing resource usage of multiple instances, it quickly locates underperforming instances to aid inference task scheduling and enhance resource utilization.

Intelligent Container Images

Feature Description

The openEuler Intelligence system can invoke environment resources through a natural language, assist in pulling container images for local physical resources, and establish a development environment suitable for debugging on existing compute devices.

This system supports three types of containers, and container images have been released on Docker Hub. You can manually pull and run these container images.

- **SDK layer:** encapsulates only the component libraries that enable AI hardware resources, such as CUDA and CANN.
- **SDKs + training/inference frameworks:** accommodates TensorFlow, PyTorch, and other frameworks (for example, tensorflow2.15.0-cuda12.2.0 and pytorch2.1.0.a1-cann7.0.RC1) in addition to the SDK layer.
- **SDKs + training/inference frameworks + LLMs:** encapsulates several models (for example, llama2-7b and chatglm2-13b) based on the second type of containers.

The following table lists the container images supported by the openEuler Intelligence system:

Registry	Repository	Image Name	Tag
docker.io	openeuler	cann	8.0.RC1-oe2203sp4
			cann7.0.RC1.alpha002-oe2203sp2
docker.io	openeuler	oneapi-runtime	2024.2.0-oe2403lts
docker.io	openeuler	oneapi-basekit	2024.2.0-oe2403lts
docker.io	openeuler	llm-server	1.0.0-oe2203sp3
docker.io	openeuler	mlflow	2.11.1-oe2203sp3
			2.13.1-oe2203sp3
docker.io	openeuler	llm	chatglm2_6b-pytorch2.1.0.a1-cann7.0.RC1.alpha002-oe2203sp2
			llama2-7b-q8_0-oe2203sp2
			chatglm2-6b-q8_0-oe2203sp2
			fastchat-pytorch2.1.0.a1-cann7.0.RC1.alpha002-oe2203sp2
docker.io	openeuler	tensorflow	tensorflow2.15.0-oe2203sp2
			tensorflow2.15.0-cuda12.2.0-devel-cudnn8.9.5.30-oe2203sp2
docker.io	openeuler	pytorch	pytorch2.1.0-oe2203sp2
			pytorch2.1.0-cuda12.2.0-devel-cudnn8.9.5.30-oe2203sp2
			pytorch2.1.0.a1-cann7.0.RC1.alpha002-oe2203sp2
docker.io	openeuler	cuda	cuda12.2.0-devel-cudnn8.9.5.30-oe2203sp2

Application Scenarios

- **Common openEuler operations:** Simplify the process of building a deep learning development environment while saving physical resources. For example, set up an Ascend development environment on openEuler.
- **openEuler development:** Developers familiarize themselves with the openEuler AI software stack to reduce the trial-and-error cost of installing components.

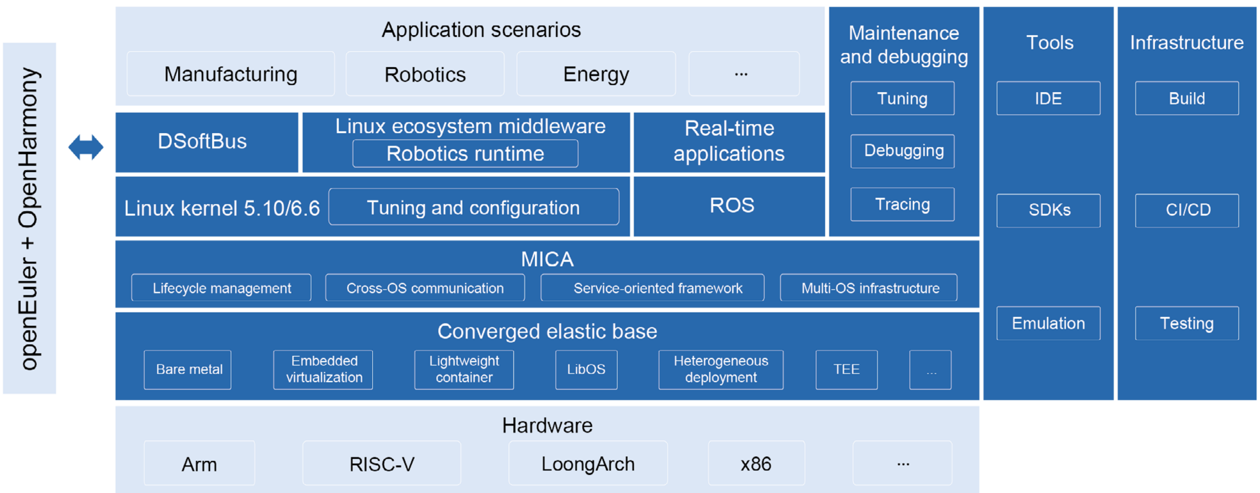
Embedded

openEuler 25.09 is suited for embedded applications, offering significant progress in southbound and northbound ecosystems, technical features, infrastructure, and implementation over previous generations.

openEuler Embedded provides a closed loop framework often found in operational technology (OT) applications such as manufacturing and robotics, whereby innovations help optimize its embedded system software stack and ecosystem. openEuler Embedded enhances its software package ecosystem by incorporating the oeBridge feature, which supports online software installation from an openEuler mirror site. When building Yocto images, oeBridge can be used to install openEuler RPM packages for easy image customization. openEuler Embedded also supports the oeDeploy feature for quick deployment of AI and cloud-native software stacks. Kernel support in openEuler is enhanced by optimizing the meta-openEuler kernel configuration and the oeAware real-time tuning feature. These updates help control interference and improve real-time system responsiveness.

Future versions of openEuler Embedded will integrate contributions from ecosystem partners, users, and community developers, increase support for chip architectures such as LoongArch and more southbound hardware, and optimize industrial middleware, embedded AI, embedded edge, and simulation system capabilities.

System Architecture



Southbound Ecosystem

openEuler Embedded Linux supports mainstream processor architectures like AArch64, x86_64, AArch32, and RISC-V, and will extend support to LoongArch in the future. openEuler 24.03 and later versions have a rich southbound ecosystem and support chips from Raspberry Pi, HiSilicon, Rockchip, Renesas, TI, Phytium, StarFive, and Allwinner.

Embedded Virtualization Base

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

- The bare metal hybrid deployment solution runs on OpenAMP to manage peripherals by partition at a high performance level; however, it delivers poor isolation and flexibility. This solution supports the hybrid deployment of UniProton/Zephyr/RT-Thread and openEuler Embedded Linux.
- Partitioning-based virtualization is an industrial-grade hardware partition virtualization solution that runs on Jailhouse. It offers superior performance and isolation but inferior flexibility. This solution supports the hybrid deployment of UniProton/Zephyr/FreeRTOS and openEuler Embedded Linux or of OpenHarmony and openEuler Embedded Linux.
- Real-time virtualization is available as two community hypervisors, ZVM (for real-time VM monitoring) and Rust-Shyper (for Type-I embedded VM monitoring).

MICA Deployment Framework

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

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

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

The MICA deployment framework provides the following functions:

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

Northbound Ecosystem

- **Northbound software packages:** Over 700 common embedded software packages can be built using openEuler.
- **Soft real-time kernel:** This capability helps respond to soft real-time interrupts within microseconds.
- **DSoftBus:** The distributed soft bus system (DSoftBus) of openEuler Embedded integrates the DSoftBus and point-to-point authentication module of OpenHarmony. It implements interconnection between openEuler-based embedded devices and OpenHarmony-based devices as well as between openEuler-based embedded devices.
- **Embedded containers and edges:** With iSula containers, openEuler and other OS containers can be deployed on embedded devices to simplify application porting and deployment. Embedded container images can be compressed to 5 MB, and can be easily deployed into the OS on another container.

UniProton

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

- Compatible with processor architectures like Cortex-M, AArch64, x86_64, and riscv64, and supports M4, RK3568, RK3588, x86_64, Hi3093, Raspberry Pi 4B, Kunpeng 920, Ascend 310, and Allwinner D1s.
- Connects with openEuler Embedded Linux on Raspberry Pi 4B, Hi3093, RK3588, and x86_64 devices in bare metal mode.
- Can be debugged using GDB on openEuler Embedded Linux.

Application Scenarios

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

05 Kernel Innovations

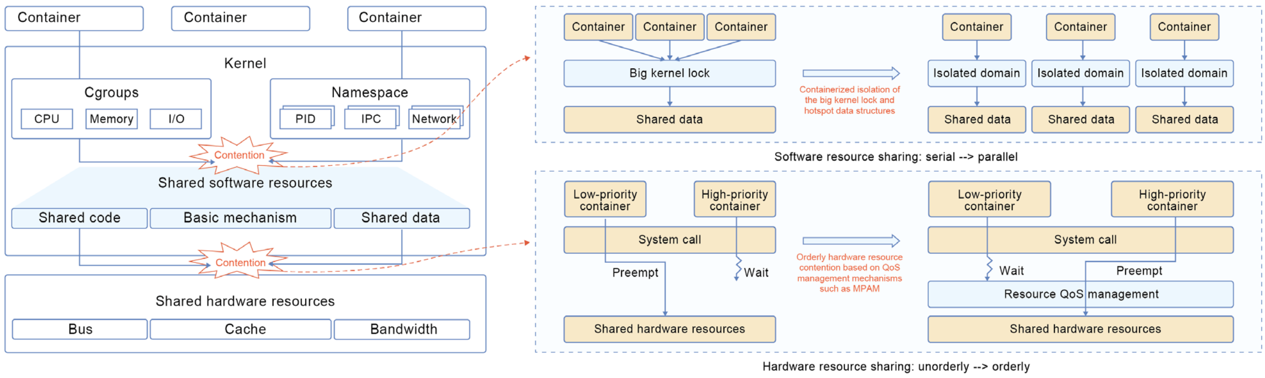
openEuler 25.09 runs on Linux kernel 6.6 and inherits the competitive advantages of community versions and innovative features released in the openEuler community.

- **Filesystem in USErspace (FUSE) pass-through:** FUSE is widely used in distributed storage and AI applications. In pass-through scenarios, FUSE skips additional processing for read and write I/Os. It only records metadata and forwards the I/O requests to the back-end file system. As a result, FUSE processing turns into the main bottleneck for I/O performance. The FUSE pass-through feature is designed to eliminate the overhead caused by context switches, wakeups, and data copying on the data plane when FUSE directly interfaces with the back-end file system. It allows applications to directly send read and write I/Os to the back-end file system within the kernel. In lab environments, FUSE pass-through has demonstrated notable performance gains. Specifically, fio tests show that read and write performance more than doubles for sizes between 4 KB and 1 MB. FUSE pass-through has also passed fault injection and stability tests, and is available for use as needed.
- **Enhanced Memory System Resource Partitioning and Monitoring (MPAM) features:** An improved quality of service (QoS) feature is introduced to optimize memory bandwidth and L3 cache control. In hybrid deployment scenarios, shared resources can be allocated based on the upper limit, lower limit, or priority-based policy. The new I/O QoS management feature collaborates with the system memory management unit (SMMU) to isolate I/O bandwidth traffic across hardware peripherals and heterogeneous accelerators. It supports monitoring by iommu_group, providing a new approach to I/O QoS management in heterogeneous environments. In addition, the L2 cache isolation feature enables monitoring of L2C usage and bandwidth traffic, offering core-level optimization and performance analysis in hybrid deployment scenarios. These MPAM features deliver significant performance improvements in test scenarios. In hybrid deployments, the interference rate of SPECjbb as an online service drops from 25.5% to below 5%.

06 Cloud Base

High-Density Many-Core Container Isolation

Server chips have evolved from multi-core to many-core architectures (typically exceeding 256 cores), posing new challenges to OSs. To boost rack-level computing density and reduce data center TCO, many-core servers have become the mainstream in the Internet industry. As cloud technologies and service scales advance, containerized deployment has also become the dominant model. Against this backdrop, serialization and synchronization overheads hinder system scalability, while interference and low resource utilization become increasingly prominent. These scalability issues in container deployments arise from contention for shared hardware and software resources.



Feature Description

Lightweight virtualization is used to partition resources by NUMA domain and enforce container-level resource isolation within each domain. This approach minimizes performance interference caused by hardware and software resource contention and enhances the container deployment scalability.

- **VM memory QoS control:** When multiple tenants' VMs are deployed on the same physical host, memory-intensive VMs may consume a large portion of the available memory bandwidth. This can lead to resource contention, preventing other VMs from obtaining sufficient bandwidth to meet their performance requirements. As a result, the overall system QoS may be degraded. Leveraging the MPAM feature of the Kunpeng processor and the OS-level resource control mechanisms, the system can perform fine-grained monitoring and dynamic control of memory bandwidth usage for up to 30 VMs. This capability enables configuration of upper and lower limits, as well as priority policies, to establish a memory bandwidth isolation and assurance framework in a multi-tenant environment.
    - Memory bandwidth upper limit: A maximum memory bandwidth threshold can be configured for each VM to prevent any single VM from consuming excessive bandwidth resources, thereby avoiding performance interference with other tenant VMs.
    - Memory bandwidth lower limit: A minimum bandwidth guarantee can be configured to maintain a baseline allocation even for lightly loaded VMs, enabling dynamic resource optimization and efficient bandwidth utilization.
    - Priority-based scheduling policy: The memory bandwidth priority of each VM can be configured based on service importance, ensuring stable bandwidth allocation for critical workloads and improving the availability and service quality of high-priority VMs.

- **NUMA affinity of virtual devices:** PCI devices possess NUMA affinity and can be directly accessed on the host. The OS scheduling mechanism optimizes task placement based on device affinity to minimize performance loss resulting from cross-NUMA access to PCI devices. In VM passthrough scenarios, PCI devices do not expose their NUMA node affinity to guest VMs. This feature expands the PCI device topology of VMs based on the PCI Expander Bridge (PXB). Within a VM, it displays the NUMA node where each virtual device is located. This helps the system optimize scheduling and allows users to deploy service applications according to the NUMA nodes of their virtual devices, thereby reducing performance loss caused by cross-NUMA resource access and improving the overall performance of VM service applications.
- **CPU scheduling by domain:** CPUs are divided into domains by cluster for container deployment. Each container operates within an independent scheduling domain. This design isolates interference between containers, reduces cross-cluster cache synchronizations, and eases contention for hardware resources like cache and NUMA memory. It improves performance by more than 10% in high-concurrency Redis scenarios.
- **Interference isolation in file system block allocation:** Optimizations to the group lock and s_md_lock in the EXT4 block allocation and release processes enhance the scalability of EXT4 block allocation. When the target block group is occupied, allocation can switch to an idle block group to reduce CPU overhead caused by multiple containers competing for the same group. Leveraging multiple EXT4 block groups helps ease group lock contention. Apart from that, the global target of the streaming allocation is split to multiple targets, so that the contention for the global lock s_md_lock is alleviated and the file data is more aggregated. In a 64-container concurrency scenario, the OPS increases by over 5 times in mixed block allocation and release workloads and by over 10 times in single-block allocation workloads.
- **Efficient slab reclamation:** In this openEuler release, the read and write locks used for slab memory reclamation are replaced with the read-copy-update (RCU) lock-free mechanism. Memory reclamation across different slabs operates independently, significantly improving reclamation efficiency. In multi-container concurrency scenarios, system calls are accelerated.
- **TCP hash interference isolation:** In high-concurrency scenarios, lock contention in tcp_hashinfo bash and ehash and frequent ehash calculations lead to reduced bandwidth and increased latency. The spin lock of tcp_hashinfo bash and ehash can be replaced with read-copy update (RCU), and the ehash calculation method can be changed to lport increment. These changes reduce the query time and calculations and reduce the lock contention in the TCP connection hash.
- **Enhanced control group (cgroup) isolation:** Original atomic operations are replaced with percpu counters to avoid parent node contention across namespaces and eliminate rlimit count interference between containers. This mechanism addresses the linearity issue in the **will-it-scale/signal1** test case and triples concurrent throughput performance in a 64-container deployment. Memory cgroups are released in batches to avoid contention for the same parent node's counter caused by frequent small memory releases, enhancing memory count scalability. In the **tlb-flush2** test case, this improves throughput by 1.5 times with 64 containers. Leveraging eBPF's programmable kernel capabilities, a host information isolation and filtering approach is provided to present container-specific resource views. Compared with the peer LXCFS solution, this openEuler solution avoids the overhead of switching between the kernel mode and user mode, and eliminates the performance and reliability bottlenecks associated with the LXCFS process. It doubles the resource view throughput in a single container and achieves a 10-fold increase in a 64-container deployment.
- **Interference monitoring:** Interference falls into three categories by result: instruction execution failure, instruction slowdown, and increased instruction execution. Interference is monitored from the kernel perspective, with statistics collected on each typical category during runtime. The system supports online monitoring of schedule latency, throttling, softirq, hardirq, spin lock, mutual exclusion (mutex), and simultaneous multi-threading (SMT) interference while incurring less than 5% performance overhead.
- **Kunpeng memory and cache QoS control:** The memory bandwidth traffic and cache usage at each level can be configured based on the upper limit, lower limit, or priority-based policy. Each thread is assigned a specific isolation policy to suit specific service requirements. The usage of shared resources is monitored in real time at both service and thread levels, and reported to the control policy to enable feedback-driven control. In addition, the MPAM feature and the SMMU combine to enhance peripherals' I/O QoS. They support bandwidth isolation for peripherals and heterogeneous accelerators and allow for resource monitoring at the device level.

- **Dynamic QoS policy configuration:** This openEuler release provides a cluster-level MPAM QoS management plugin. Using the QoS interfaces provided by MPAM, the plugin automatically assigns priorities to nodes based on user-defined policies and sets MPAM QoS priorities for offline tasks according to user configurations. This ensures efficient resource utilization in hybrid deployment scenarios. When online services are busy, the last-level cache (LLC) and memory bandwidth allocated to offline tasks are automatically preempted. When online services are idle, these resources are released to enhance offline service performance.

Application Scenarios

In a many-core server environment, service containers are deployed in a high-density configuration. The openEuler solution minimizes interference between containers, increases deployment density, and enhances resource utilization.

In memory-intensive scenarios, this feature mitigates bandwidth contention among multiple VMs. It monitors and controls the memory bandwidth usage of each VM, ensuring that performance-critical services run reliably.

07 Enhanced Features

LLVM for openEuler Compiler

LLVM for openEuler is a high-performance compiler based on open source LLVM software. It is crafted for compute-intensive scenarios such as server and Internet industries, cutting-edge data center applications, AI computing systems, and video encoding and decoding. In addition, it establishes an LLVM baseline within the openEuler community, providing a stable downstream LLVM distribution that is secure, reliable, and easily innovated upon. It currently supports mainstream languages (C and C++) and chip architectures (x86, AArch64, RISC-V, LoongArch, and Sunway).

Feature Description

LLVM for openEuler introduces the following compilation features in openEuler 25.09, optimizing the running efficiency of database and big data applications to unlock the ultimate software performance.

ICP optimization: Indirect call promotion (ICP) optimization converts indirect function calls into direct function calls based on feedback information. This increases potential inlining opportunities and reduces function call overhead.

Intelligent hash-based prefetch optimization: This feature identifies multi-layer indirect nested memory access scenarios in applications, automatically calculates actual memory access addresses, and inserts data prefetch instructions, thereby reducing the probability of data cache misses.

Adaptive memory copy optimization: By recognizing the characteristics of the source and destination pointers during memory copy, this feature adds runtime checks for memory copy method specialization (such as generating **memset** and **memmove** intrinsics).

Fast access to dynamic libraries: Conventional dynamic library function calls require jumping through the procedure linkage table (PLT), leading to extra memory access and jump instructions. This is optimized by directly calling functions with their addresses in the global offset table (GOT), eliminating PLT jump overhead.

Application Scenarios

The compilation optimizations of LLVM for openEuler improve the performance of mainstream database and big data applications such as MySQL, ClickHouse, and Doris.

Enhanced oeDeploy

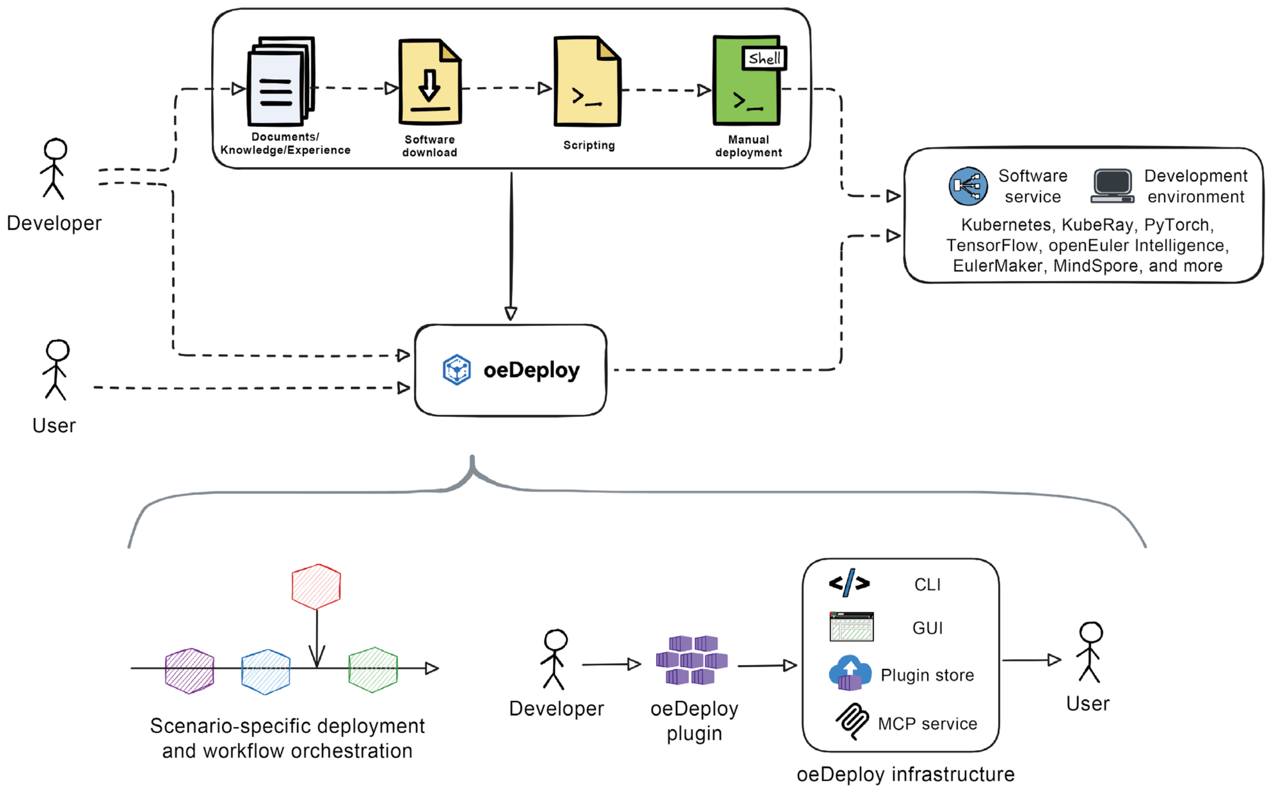
oeDeploy is a lightweight software deployment tool that accelerates environment setup across single-node and distributed systems with unmatched efficiency.

Feature Description

Multi-scenario support and quick software deployment: oeDeploy facilitates quick deployment for both single-node applications and clustered software environments. It now includes quick deployment capabilities for Kubernetes environments with multiple master nodes. It also extends support for community toolchains like openEuler Intelligence and DevKit Pipeline, as well as popular Retrieval Augmented Generation (RAG) software such as RAGFlow, AnythingLLM, and Dify.

Flexible plugin management and excellent deployment experience: oeDeploy provides an extensible plugin architecture for flexible management of diverse deployment capabilities, empowering developers to quickly release custom deployment plugins. It now supports plugin source management, enabling one-click plugin updates and one-click plugin initialization. While oeDeploy currently offers a streamlined CLI, a GUI and plugin store will soon launch, promising an even more efficient software deployment experience with less code.

Efficient deployment and intelligent development: oeDeploy introduces MCP servers, offering an out-of-the-box experience within DevStation. It leverages LLM inference to deploy software with natural language, boosting deployment efficiency by 2x. It can also convert user documents into executable oeDeploy plugins, increasing development efficiency by 5x.



Application Scenarios

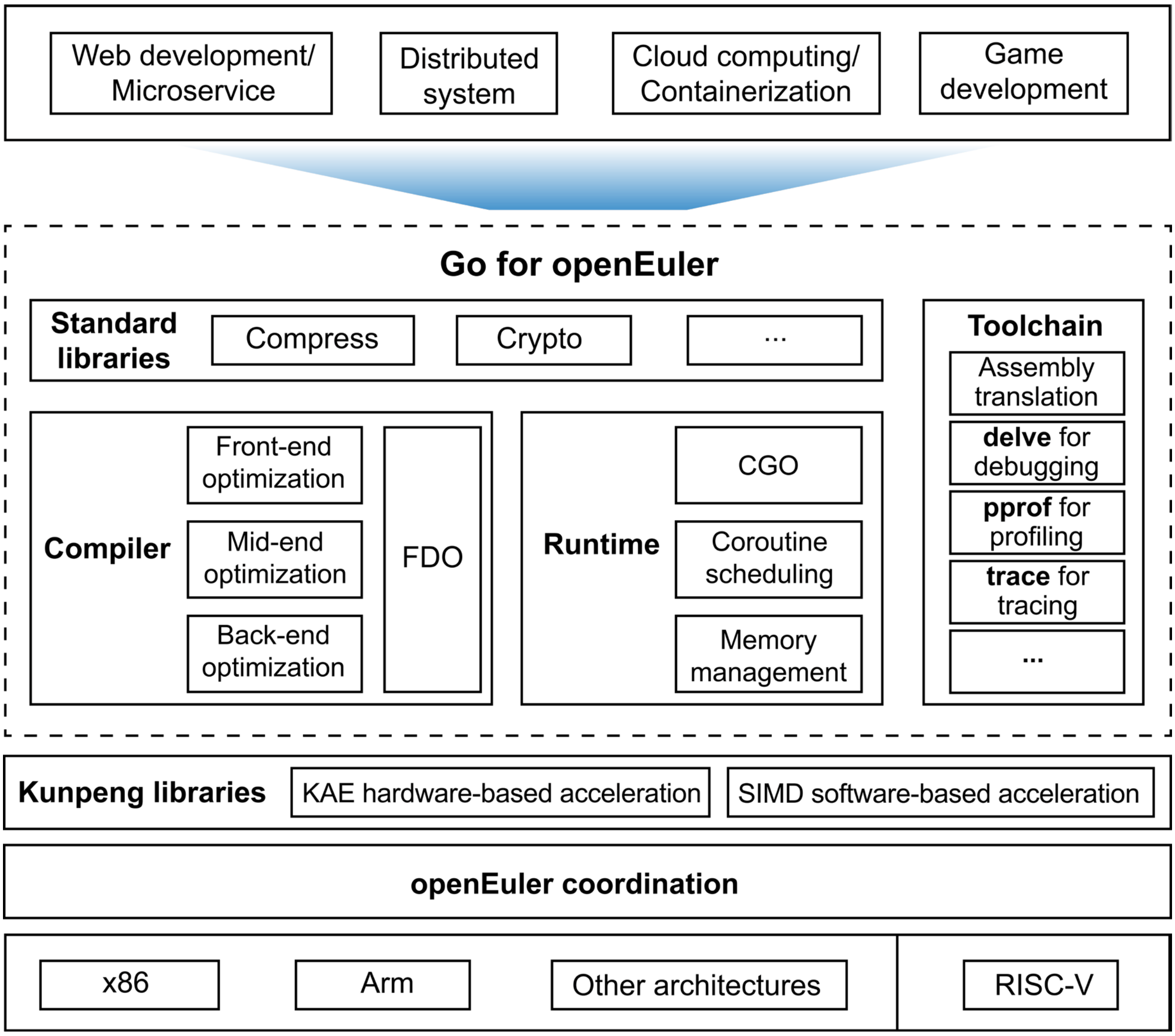
ISVs and development teams can adopt oeDeploy as a standardized solution for software product delivery. Its CLI tools and plugin framework minimize development overhead while ensuring smooth delivery, reducing user onboarding efforts, and enhancing satisfaction.

For developers and maintenance personnel, oeDeploy enables instant setup of complex environments, deploying mainstream AI training and inference frameworks in just minutes. This significantly streamlines software development and eliminates tedious configuration work. Developers can also extend oeDeploy by creating and sharing custom deployment templates, democratizing quick deployment for broader user communities. By leveraging foundation models and MCP capabilities, oeDeploy makes deployment more efficient and development more intelligent.

Go for openEuler Compiler

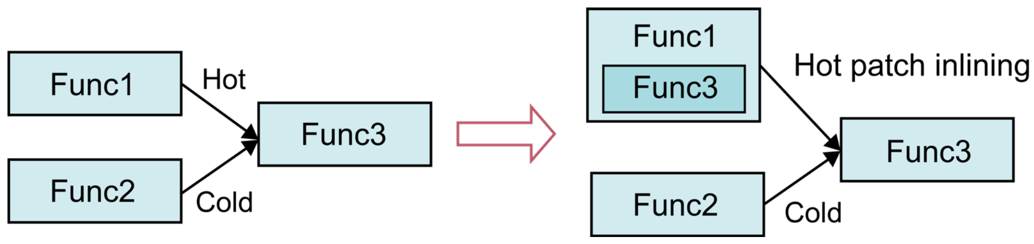
Go for openEuler is a reliable and easy-to-develop Go distribution. It is committed to building a high-performance compiler that offers wide ecosystem compatibility, openEuler affinity, and ultimate user experience. Primarily targeted at container cloud scenarios that require agile development and high performance, such as cloud native and microservice applications, Go for openEuler is optimized for mainstream Go workloads to address performance issues caused by insufficient native Go capabilities. The compiler adapts to hardware platforms such as LoongArch and Kunpeng to unleash their computing power.

Feature Description



Continuous Feature Guided Optimization (CFGO)

While ensuring the functional integrity of the program, this technique collects the program's runtime profile to make more accurate optimization decisions, resulting in a refined program with better performance. Based on the principle of program locality, it arranges hot instructions closely to improve cache/translation lookaside buffer (TLB) hits, effectively reducing front-end bottlenecks.



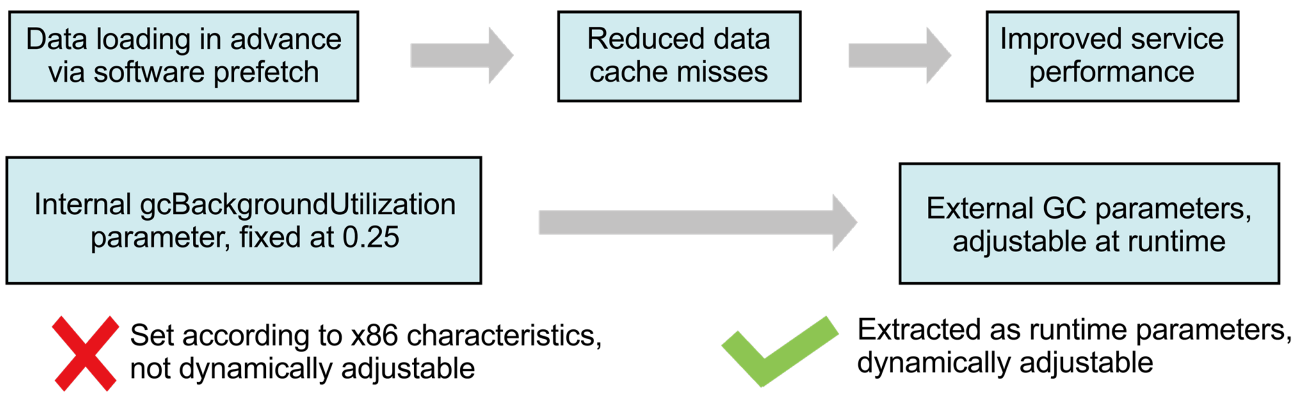
Function inlining

Arm atomic instruction optimization

In certain service scenarios, the Go runtime incurs significant overhead when invoking compare-and-swap (CAS) operations and load/store (LD/ST) instructions. Adopting an Arm-affinity instruction sequence delivers significant performance gains.

Runtime garbage collection (GC) optimization

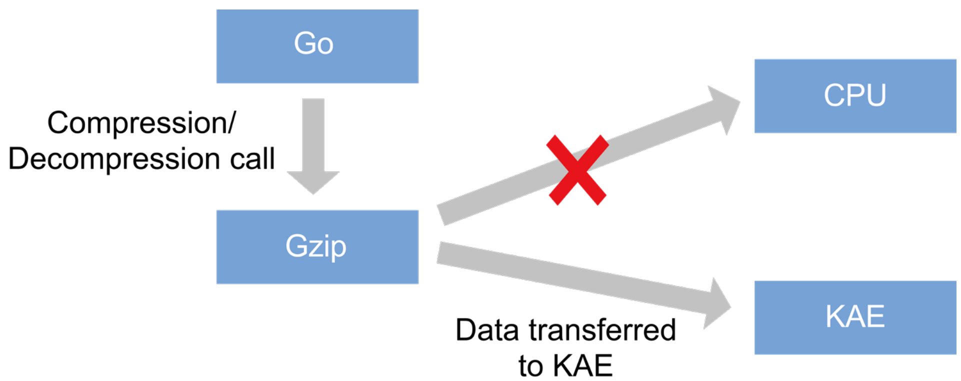
This optimization involves the insertion of software prefetch instructions based on identified program characteristics. Parameters governing the GC coroutine overhead are extracted as runtime parameters supporting dynamic adjustment according to varying service characteristics.



Runtime GC optimization

Kunpeng Accelerator Engine (KAE) for hardware-based acceleration

The compression/decompression logic of the Gzip compression library of Go is modified to enable underlying hardware-based acceleration.



KAE enablement

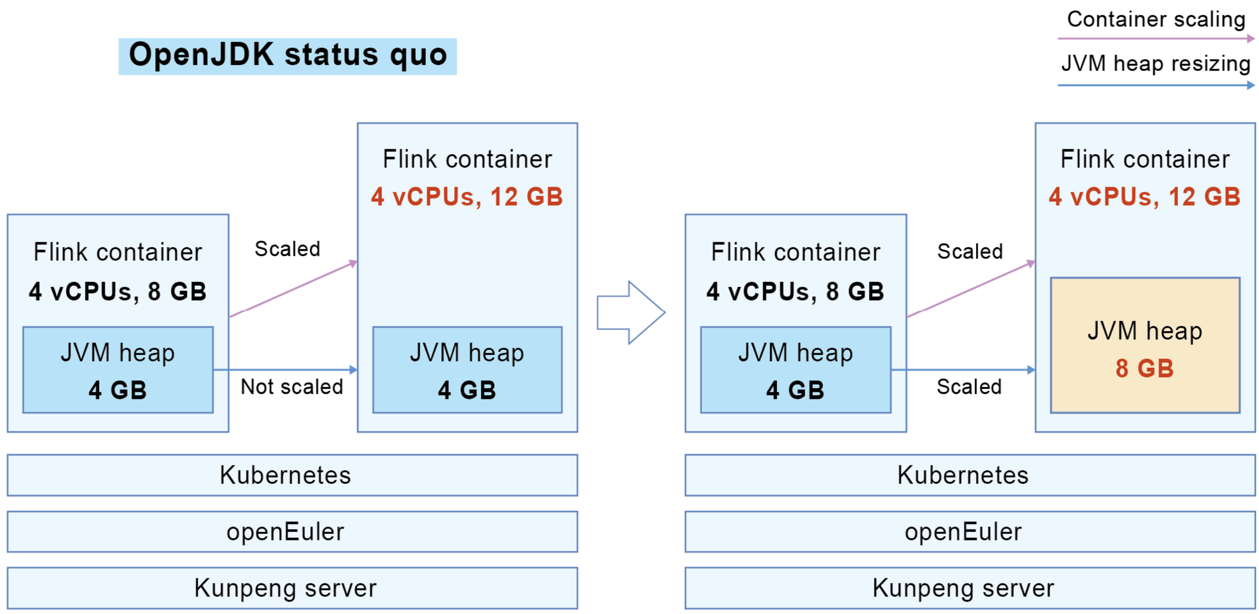
Application Scenarios

As a piece of foundational Go software, Go for openEuler is used in Linux environments such as openEuler, covering cloud native, distributed storage, and cloud gaming workloads.

Heap Resizing by the BiSheng JDK

Feature Description

In modern containerized deployments, most users' container environments support resource scale-up. However, OpenJDK has a significant limitation: its maximum heap size can only be configured at startup. This means it does not support online resizing, preventing Java applications from immediately using additional memory made available after a container scales up. Applications currently require a restart to reset their maximum heap. To address this limitation, BiSheng JDK introduces online heap memory resizing for the Garbage-First garbage collector (G1GC). This allows users to dynamically update the Java heap memory limit while an application is running, eliminating the need for a JVM restart.



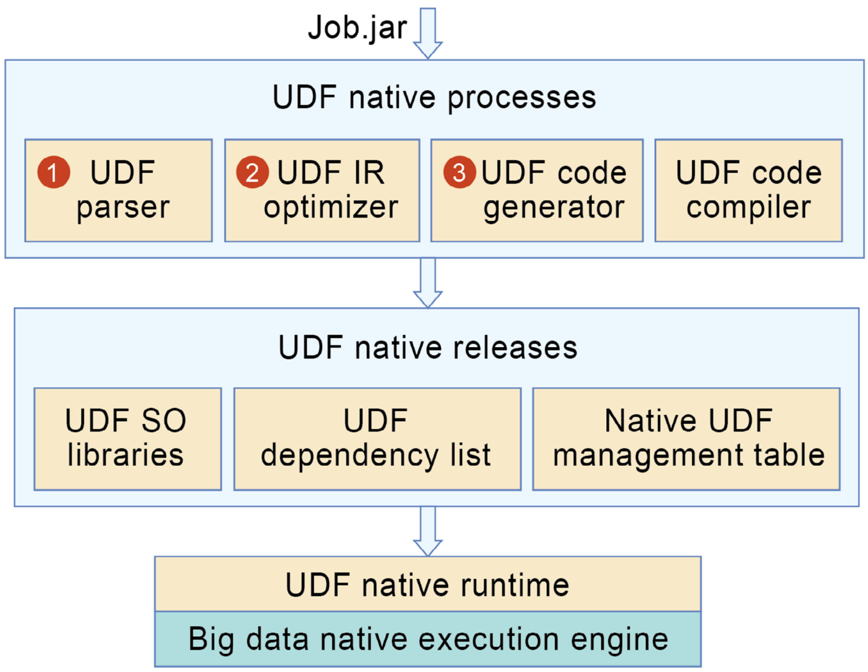
Application Scenarios

This capability is crucial for Internet and other container-based deployments that require Java application heap memory to resize online in response to container resource scale-up.

UDF Automatic Native Framework

Feature Description

The UDF automatic native framework addresses the inefficient JVM execution often seen in open source big data systems. It automatically converts Java user-defined functions (UDFs) into C/C++ native UDFs, significantly boosting big data processing performance through efficient memory management and hardware affinity. Essentially, the framework implements a seamless, automatic Java UDF native acceleration mechanism. It comprises the UDF parser, UDF IR optimizer, UDF code generator, and UDF code compiler modules.



The UDF parser automatically converts the bytecode of a service JAR package into Intermediate Representation (IR) code and extracts UDF code by identifying its specific features. The UDF IR optimizer optimizes the UDF IR from dimensions such as automatic memory object management and hardware-affinity acceleration. The UDF code generator translates the optimized UDF IR into native code. The UDF code compiler compiles the UDF native code into native binaries online. Finally, these UDF native binaries are deployed to big data execution nodes. The native execution engine of the big data system dynamically loads and executes the native binaries. This improves the big data processing performance.

Application Scenarios

The UDF automatic native framework is designed to seamlessly integrate with the Flink big data native execution engine. By leveraging a Flink DataStream native base library, it achieves automatic native acceleration for Flink DataStream UDFs, all without requiring any user intervention.

De-optimization Observability in BiSheng JDK 17

The JDK Flight Recorder (JFR) Streaming API of JDK 17 is a key feature that enables JFR to move from post-event static analysis to real-time monitoring.

In conventional JFR usage, a user must start recording, stop recording, dump contents into a .jfr file, and finally use Java Mission Control (JMC) tools for offline analysis. This is a post-event analysis mode, which is effective for troubleshooting problems that have occurred.

The Streaming API introduces a new mode, which allows a Java application to continuously subscribe to and consume JFR event streams from the JVM in real time without interrupting JFR recording or generating a complete .jfr file.

Feature Description

When using the Streaming API, a user can obtain JFR events such as de-optimization events before the current time from positions marked as ******.

// 1. Create a recording stream.

RecordingStream rs = new RecordingStream();

// 2. Enable events of interest and configure related settings.

rs.enable("jdk.GCPhasePause").withPeriod(Duration.ofSeconds(1));

rs.enable("jdk.Deoptimization").withPeriod(Duration.ofSeconds(1));

// 3. Subscribe to an event and set corresponding event handler (callback function).

rs.onEvent("jdk.GCPhasePause", event -> {

// Read the following fields from the event.

Duration duration = event.getDuration("duration");

String name = event.getString("name"); // For example, "GC Pause".

****

shell: jcmd JFR.start delay=-1 filename=xxx.jfr

****

});

// 4. Start the stream (non-blocking call).

rs.startAsync();

Application Scenarios

The combination of online monitoring and post-event analysis allows the JFR file used for post-event analysis to contain all JFR events within a period of time prior to the current moment.

Enhanced GCC for openEuler CFGO

The continuous growth in code volume has made front-end bound execution a common issue in processors, which impacts program performance. Feedback-directed optimization techniques in compilers can effectively solve this issue.

Continuous Feature Guided Optimization (CFGO) in GCC for openEuler refers to continuous feedback-directed optimization for multimodal files (source code and binaries) and the full lifecycle (compilation, linking, post-linking, runtime, OS, and libraries). The following techniques are included:

- **Code layout optimization:** Techniques such as basic block reordering, function rearrangement, and hot/cold separation are used to optimize the binary layout of the target program, improving I-cache and I-TLB hit rates.
- **Advanced compiler optimization:** Techniques such as inlining, loop unrolling, vectorization, and indirect calls enable the compiler to make more accurate optimization decisions.

Feature Description

CFGO comprises CFGO-PGO, CFGO-CSPGO, and CFGO-BOLT. Enabling these sub-features in sequence helps mitigate front-end bound execution and improve program runtime performance. To further enhance the optimization, you are advised to add the **-flto=auto** compilation option during CFGO-PGO and CFGO-CSPGO processes.

- **CFGO-PGO**

  Unlike conventional profile-guided optimization (PGO), CFGO-PGO uses AI for Compiler (AI4C) to enhance certain optimizations, including inlining, constant propagation, and devirtualization, to further improve performance.

- **CFGO-CSPGO**

  The profile in conventional PGO is context-insensitive, which may result in suboptimal optimization. By adding an additional CFGO-CSPGO instrumentation phase after PGO, runtime information from the inlined program is collected. This provides more accurate execution data for compiler optimizations such as code layout and register optimizations, leading to enhanced performance.

- **CFGO-BOLT**

  CFGO-BOLT adds optimizations such as software instrumentation for the AArch64 architecture and inlining optimization on top of the baseline version, driving further performance gains.

Application Scenarios

CFGO has good versatility and is suitable for C/C++ applications, such as databases and distributed storage, where the overall system performance bottleneck lies in CPU front-end bound execution. It can typically achieve a performance improvement of 5% to 10%.

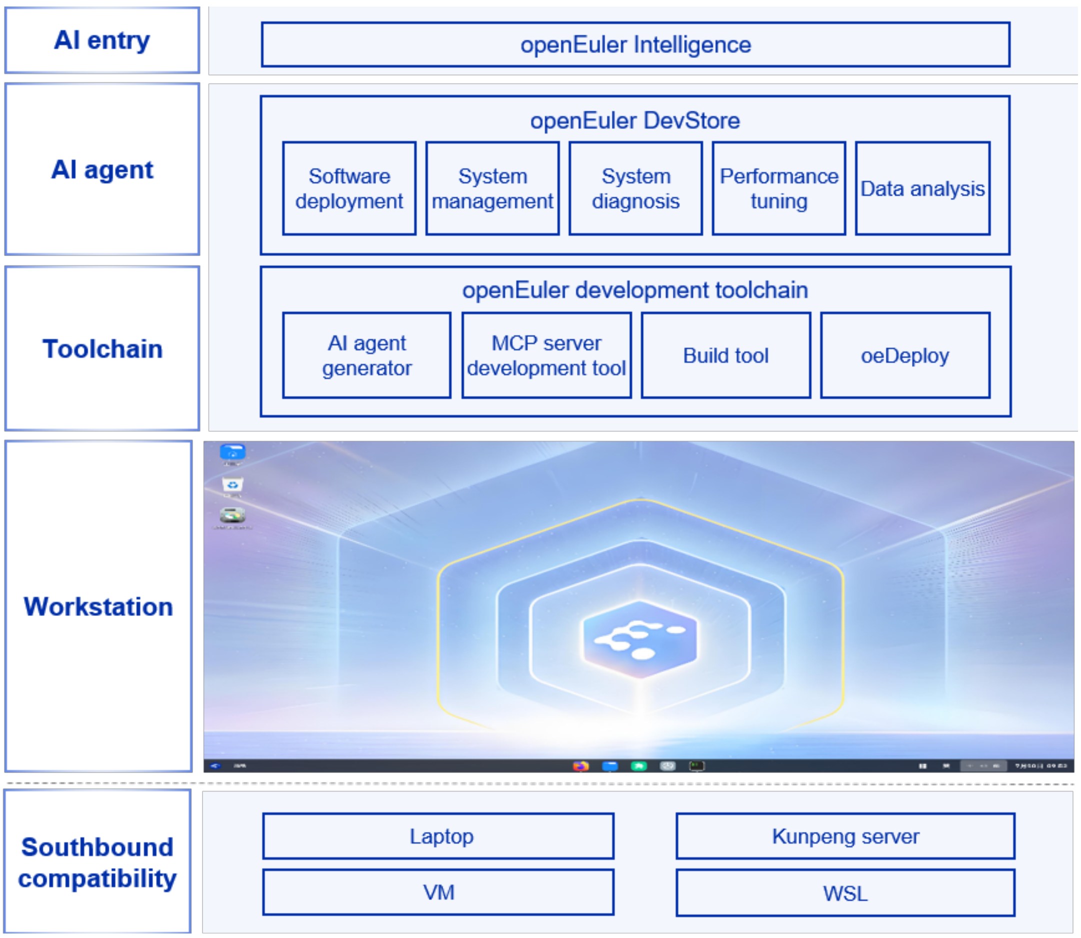
Enhanced DevStation

DevStation is an intelligent developer workstation built on openEuler, designed for geeks and innovators. It provides an out-of-the-box, efficient, and secure development environment that streamlines the entire workflow from deployment and coding to compilation, building, and releasing. By integrating a one-click runtime environment with a full-stack development toolchain, it enables a seamless transition from system boot to code execution. The new MCP AI engine allows for quick invocation of community toolchains, offering a significant leap in efficiency from infrastructure setup to application development, all without complex installation.

Feature Description

Developer-friendly integrated environment: Pre-installed with a wide range of development tools and IDEs like VSCodium, this distribution supports multiple programming languages to meet the needs of front-end, back-end, and full-stack developers.

Native community tool ecosystem: New tools like oeDeploy (a one-click deployment tool), epkg (an extended package manager), DevKit (a development toolchain), and openEuler Intelligence (an intelligent Q&A system) provide full-lifecycle support from environment configuration to code deployment. oeDevPlugin and oeGitExt are VSCodium plugins designed for the openEuler community. They provide visual management for issues and pull requests (PRs), facilitating code repository cloning, PR submission, and real-time task status synchronization. openEuler Intelligence supports natural language for generating code snippets, creating API references, and explaining Linux commands.



GUI-based programming environment: DevStation integrates graphical programming tools to streamline coding for beginners while offering powerful visual programming capabilities for veterans. It also comes pre-installed with productivity tools like Thunderbird.

MCP-based intelligent application ecosystem: DevStation deeply integrates the MCP framework to build a complete intelligent toolchain ecosystem. It includes pre-installed MCP servers like oeGitExt and rpm-builder, which provide capabilities for community task management and RPM packaging. It intelligently wraps conventional development tools like Git and RPM builders using the MCP protocol, offering a natural language interaction interface.

Enhanced system deployment and compatibility: DevStation offers extensive hardware support, especially seamless compatibility with mainstream laptop and PC hardware (such as touchpads, Wi-Fi, and Bluetooth), and a restructured kernel build script (**kernel-extra-modules**) to ensure bare metal deployment experience. It also supports flexible deployment options, including Live CD (instant run without installation), bare metal installation, and VM deployment.

New installation tool—heolleo: heolleo is a modern client tool designed specifically to simplify the DevStation installation process. Built with a modular design, it easily supports feature expansion across various hardware architectures (like x86 and Arm), file systems, and boot loaders (like GRUB). It offers flexible installation modes, supporting system file acquisition from both local ISO images and network sources (HTTP/FTP).

- Local ISO installation: heolleo provides a local ISO installation mode for users demanding extreme stability, high speed, or deployment in offline or restricted environments. By leveraging existing system image files, it delivers fast, reliable, and completely offline installation experience with automated partition setup.
- Network installation: heolleo's network installation mode aligns with modern system deployment trends. It eliminates the need for manual image downloads by allowing users to obtain the latest system files directly from Internet servers, which is the most convenient access to the newest DevStation version.

Note: DevStation released along with openEuler 25.09 does not integrate heolleo by default. Developers can download the DevStation software package from the openEuler community and run the **dnf install heolleo-frontend** command to install heolleo.

Application Scenarios

Multi-language development: DevStation is ideal for developers working on multi-language projects, such as Python, JavaScript, Java, and C++. With various pre-installed compilers, interpreters, and build tools, it eliminates the need for manual configuration.

Quick installation and deployment: DevStation integrates oeDeploy to deploy distributed software such as Kubeflow and Kubernetes within minutes. oeDeploy provides a unified plugin framework and atomic deployment capabilities, allowing developers to quickly release custom installation and deployment plugins.

Hardware compatibility and bare metal testing: For testers and developers focused on southbound compatibility, DevStation ensures robust hardware support for mainstream laptops and servers, and allows bare metal deployment for driver compatibility testing.

Improved developer efficiency: The MCP RPM-builder toolchain improves MCP usability by supporting automated packaging and releasing of RPM packages to the community, ensuring 100% availability for instant MCP installation. It helps build a complete MCP intelligent application ecosystem repository that covers scenarios like deployment, testing, and performance tuning, allowing developers to query assigned community issues, create PRs to submit code changes, and automate build and verification via CI/CD.

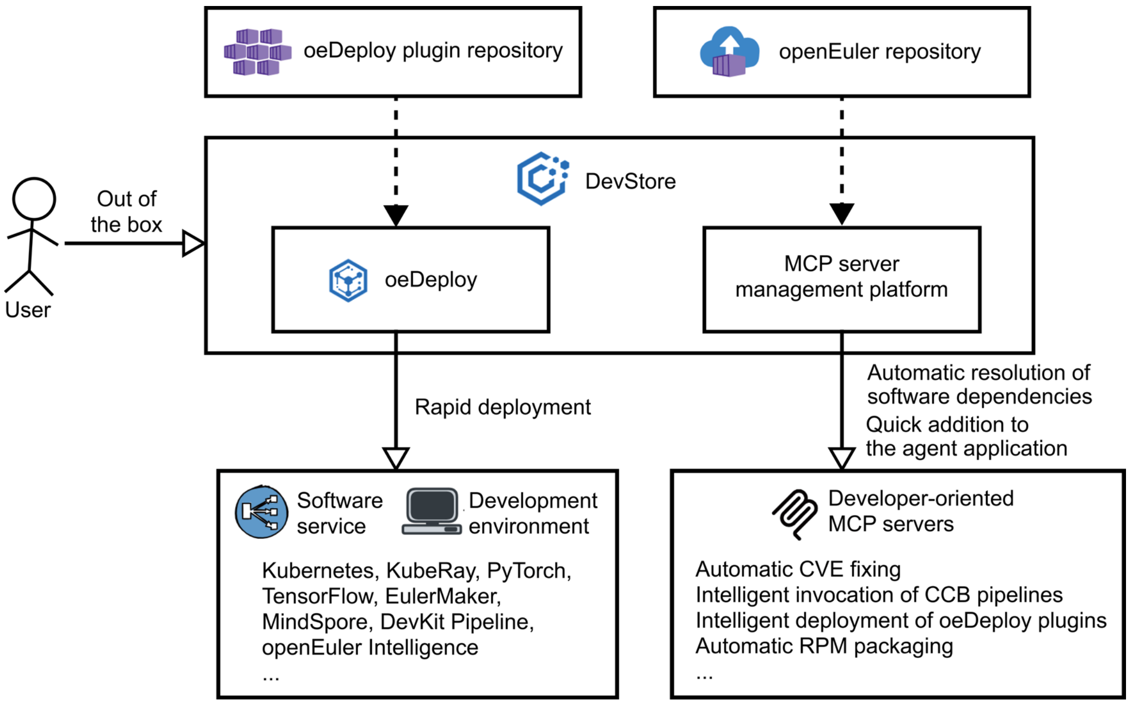
DevStore

DevStore is the application store for the openEuler desktop version, acting as a developer-centric software distribution platform. It supports the search and rapid deployment of MCP servers and oeDeploy plugins. DevStore is provided out-of-the-box on the DevStation platform.

Feature Description

Rapid installation of MCP servers: Leveraging openEuler community's extensive software ecosystem, DevStore packages the software dependencies required for MCP server operation as standard RPM files. Using built-in service management tools, DevStore quickly deploys MCP servers in agent applications. It automatically solves software dependency and MCP configuration issues for users, greatly improving user experience. Currently, DevStore supports the deployment of over 80 MCP servers.

oeDeploy for quick plugin deployment: DevStore utilizes the oeDeploy tool to enable the rapid deployment of mainstream software, substantially reducing the setup time. The supported software categories include AI software (like Kubernetes, KubeRay, PyTorch, TensorFlow, and DeepSeek), toolchains (like EulerMaker and openEuler Intelligence), and RAG tools (like RAGFlow, Dify, and AnythingLLM).



Application Scenarios

As the application store for the openEuler desktop, DevStore enables developers, including beginners, to swiftly obtain mainstream development tools, AI software, and MCP servers. Comprehensive user manuals and operation entries are available on the details page. Notably, all complex deployment operations are consolidated into a unified operation interface, significantly reducing the learning cost and improving user experience.

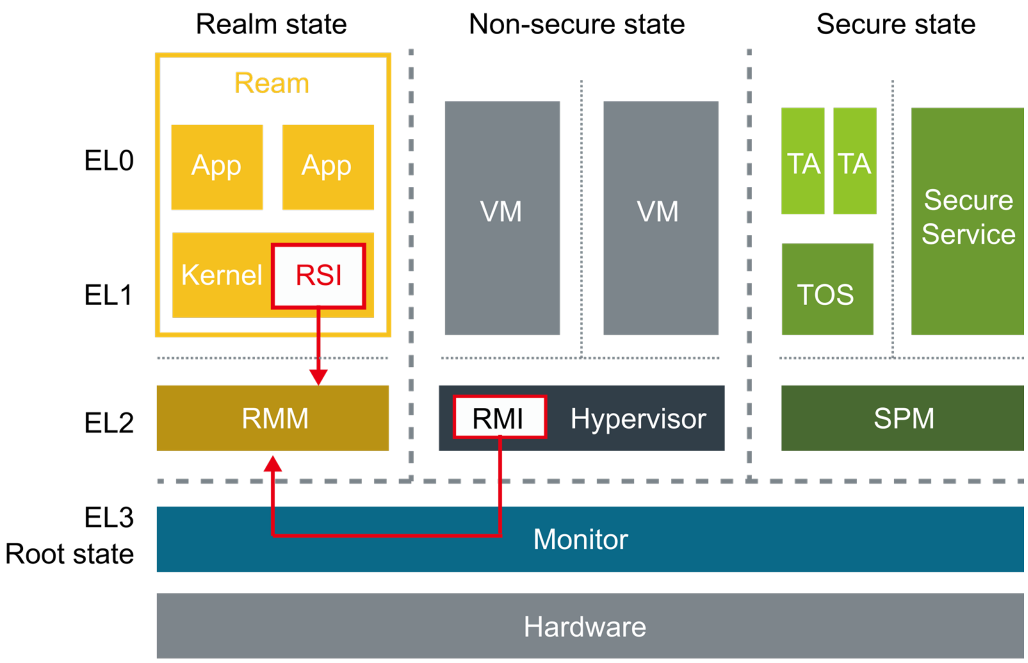
Confidential Computing Architecture (CCA)

Arm CCA is a confidential computing architectural specification newly introduced in Arm v9. It is designed to define a standard confidential computing solution for next-gen computing devices. CCA establishes realms as trusted execution environments (TEEs) to protect the confidentiality and integrity of data and code in use, ensuring effective protection even against privileged infrastructure software or cloud service providers.

Based on the Arm CCA specification, openEuler has implemented support for CCA across its relevant OS components, including KVM, QEMU, libvirt, and the guest kernel. The openEuler community edition provides native support for realm confidential VMs (cVMs), fulfilling the security requirement to protect data in use. Crucially, this implementation also offers the ease-of-use and compatibility necessary for seamless integration with traditional application ecosystems and existing VM management software.

Feature Description

Arm CCA utilizes the following core components working in synergy to construct a realm, an isolated and protected execution space. The realm is completely segregated from the normal world in terms of code execution and data access.



- **Realm:** A realm is the core abstraction of CCA. It is a new execution environment that runs parallel to the non-secure world and secure worlds (TrustZone). It is hardware-isolated and designed to host sensitive code and data. It is independent of the host OS and Hypervisor, which can manage a realm but cannot access the content within.
- **Dynamic management:** Hypervisor can dynamically create realms and allocate memory and CPU resources to them as required. However, once a realm is initialized, Hypervisor hands over control to the realm management monitor (RMM), a protected secure virtualization module, and can no longer access the data within the realm.
- **Memory management:** CCA extends the system memory management unit (MMU) to identify and isolate realm memory. Any access attempt from outside the realm (including Hypervisor) is blocked by the hardware, ensuring data confidentiality.

- **Remote attestation:** Each CCA-enabled processor has a unique, hardware-based identity. When a realm starts, it can generate an attestation token that is cryptographically signed by the hardware. Users can obtain this report, verify its signature, and check the component measurements to ensure that their workloads are running in a genuine, untampered Arm CCA environment.

Application Scenarios

cVMs are mainly used in cloud computing environments to provide high-level security isolation for core workloads. They enable the processing of sensitive data (such as financial records, healthcare information, and intellectual property) on untrusted public clouds while ensuring the confidentiality and integrity of the data. These cVMs can block any unauthorized access, even from cloud service providers. This effectively meets the key requirements of data sovereignty, regulatory compliance, and cross-agency privacy data collaboration (such as joint modeling and analysis).

Enhanced secGear

The unified remote attestation framework of secGear addresses the key components related to remote attestation in confidential computing, abstracting away the differences between different TEEs. It provides two components: attestation agent and attestation service. The agent is integrated by users to obtain attestation reports and connect to the attestation service. The service can be deployed independently and supports the verification of iTrustee and virtCCA remote attestation reports.

openEuler 25.09 introduces support for remote attestation featuring the platform token report for the new version of virtCCA cVMs. This enhancement completes the trust chain and supporting remote attestation services for virtCCA. While improving the remote attestation process, the system maintains backward compatibility with the legacy version of virtCCA cVMs, which perform attestation without the platform token report.

Feature Description

The new architecture for virtCCA cVMs introduces a platform token. When the attestation agent requests data from a cVM, the resulting virtCCA token now includes a platform token. The attestation service has been updated to accommodate this change:

- The root and intermediate certificate algorithms have been upgraded from RSA to ECCP521.
- The remote attestation process is enhanced with the addition of platform token signature verification and cVM token public key validation, completing the attestation chain.
- The system now allows using policies to verify the version and hash of the virtCCA platform software components.
- The attestation report now outputs the verification result of the platform software components based on policies.
- The report also explicitly outputs whether the current virtCCA supports the platform token.

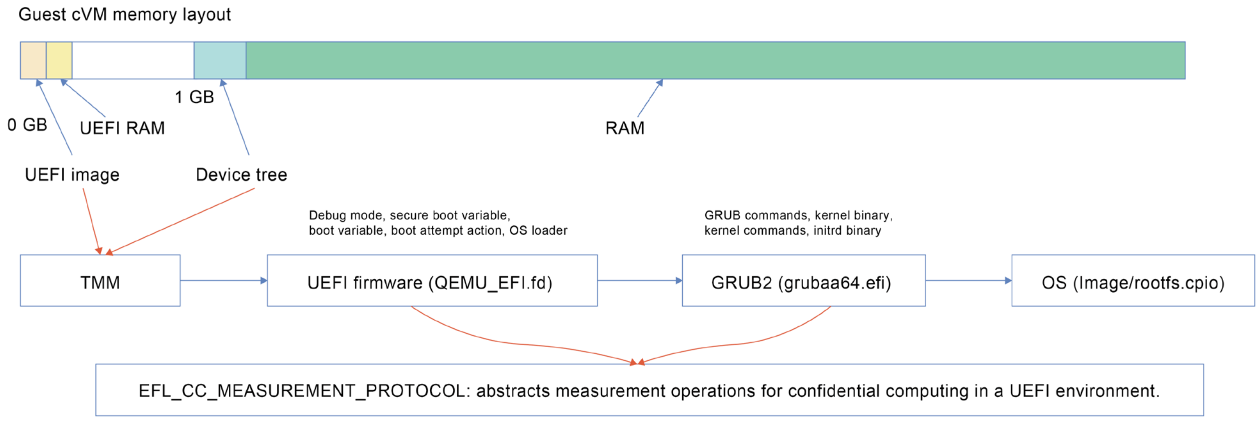
secGear retains the support for the legacy virtCCA remote attestation process. This is determined by whether the data sent from the attestation agent to the attestation service includes the platform token. If the token is absent, the system uses the legacy virtCCA remote attestation (certificate) process. In this case, the attestation policy cannot verify the related components, and the report will output **vcca.is_platform** as **False**, indicating that the current virtCCA implementation does not support the platform token.

Application Scenarios

In scenarios like finance and AI, where confidential computing is used to protect the security of privacy data during runtime, remote attestation is a technical means to verify the legitimacy of the confidential computing environment and applications. secGear provides components that are easy to integrate and deploy, helping users quickly enable confidential computing remote attestation capabilities.

Enhanced virtCCA

Feature Description



The current virtCCA architecture has this constraint: it only supports a boot mode where the kernel and rootfs are mounted separately. However, in mainstream cloud platforms, VMs typically rely on a GRUB bootloader. This requires integrating the Unified Extensible Firmware Interface (UEFI) firmware (like EDK2), kernel, and initial RAM file system (initramfs) into a single image, such as a QCOW2 file. Enhanced virtCCA addresses this by providing the following functions:

- Single image encapsulation
  - **Unified boot stack:** virtCCA integrates the EDK2 firmware, GRUB bootloader, kernel, and initramfs into a QCOW2 image, creating a complete boot stack.
  - **Streamlined boot process:** GRUB uses a configuration file (**grub.cfg**) to locate the kernel path, which requires the kernel and initramfs to reside on the same file system, for example, ext4 or XFS.
- Secure boot chain
  - **Secure boot:** EDK2 verifies the digital signatures of GRUB and the kernel, ensuring that the boot components have not been tampered with.
  - **Hardware resource collaboration:** virtCCA leverages UEFI runtime services to enumerate hardware devices, providing a virtualized resource pool for hypervisors like KVM.
- Cloud native optimization

  virtCCA supports snapshot cloning and dynamic rootfs expansion (depending on cloud-init in initramfs).

Application Scenarios

The adoption of UEFI as a boot mode in cloud environments is a core technology for modern virtualization. virtCCA's new UEFI support expands its use cases for cVMs:

- Fast instance boot and auto scaling

  UEFI uses parallel hardware initialization (such as simultaneous detection of CPUs, memory, and storage devices) to significantly shorten the boot time.

- Large-capacity drive support

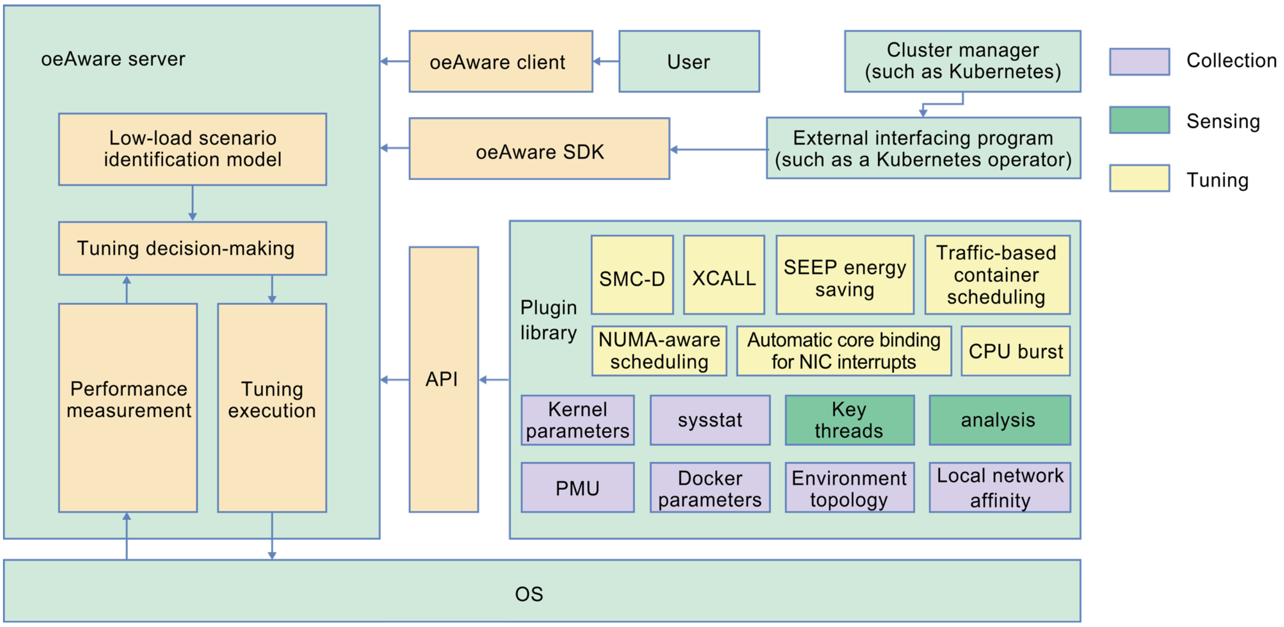
  Be leveraging the GUID Partition Table (GPT), UEFI breaks the traditional MBR limit of 2 TB. This allows virtCCA to support cloud drives of 100 TB or more (like Alibaba Cloud's ESSDs), which is essential for big data storage (such as HDFS) and AI training.

- Automated O&M and batch deployment

  UEFI-compatible images provided by cloud service providers streamline user deployment by enabling GPT partitioning by default.

oeAware Enhancements

oeAware is a framework that provides low-load collection, sensing, and tuning upon detecting defined system behaviors on openEuler. The framework divides the tuning process into three layers: collection, sensing, and tuning. Each layer is associated through subscription and developed as plugins, overcoming the limitations of traditional tuning techniques that run independently and are statically enabled or disabled.



Feature Description

Every oeAware plugin is a dynamic library that utilizes oeAware interfaces. The plugins comprise multiple instances that each contains several topics and deliver collection or sensing results to other plugins or external applications for tuning and analysis purposes.

- The SDK enables subscription to plugin topics, with a callback function handling data from oeAware. This allows external applications to create tailored functionalities, such as cross-node information collection or local node analysis.
- The performance monitoring unit (PMU) information collection plugin gathers performance records from the system PMU.
- The Docker information collection plugin retrieves specific parameter details about Docker containers in the environment.
- The system information collection plugin captures kernel parameters, thread details, and resource information (CPUs, memory, I/Os, network) from the current environment, as well as the TCP network affinity between local processes.
- The thread sensing plugin monitors key information about threads.
- The evaluation plugin examines NUMA and network information during service operations, suggesting optimal tuning methods.

- The system tuning plugins comprise stealtask for enhanced CPU tuning, smc_tune (SMC-D) which leverages shared memory communication in the kernel space to boost network throughput and reduce latency, xcall_tune which offers code paths that bypass non-critical processes to minimize system call processing overhead, realtime_tune which enables deep isolation and automatic real-time performance optimization, net_hard_irq_tune which dynamically modifies network interrupt affinity to improve network service performance, and NUMA-aware scheduling which enhances NUMA scheduling performance.
- The Docker tuning plugins contain cpu_burst which addresses CPU performance issues during sudden load spikes and traffic-based container scheduling which adjusts the CPU affinity of container services based on service loads to improve QoS.

Constraints

- **smc_tune:** SMC acceleration must be enabled before the server-client connection is established. This plugin is most effective in scenarios with numerous persistent connections.
- **Docker tuning:** This plugin is not compatible with Kubernetes containers.
- **xcall_tune:** The **FAST_SYSCALL** kernel configuration option must be activated.
- **realtime_tune:** This plugin must be used together with the Preempt-RT kernel.
- **net_hard_irq_tune:** This plugin applies only to network communication over TCP.

Application Scenarios

stealtask is ideal for scenarios aiming to boost CPU utilization, such as in Doris. This tuning instance effectively increases CPU utilization and prevents idle CPU cycles.

xcall_tune is designed for applications with substantial system call overhead. It offers code paths that bypass non-critical processes, optimizing system call handling and reducing overhead. However, this approach may compromise some maintenance and security capabilities.

smc_tune (SMC-D) excels in environments demanding high throughput and low latency, including high-performance computing (HPC), big data processing, and cloud platforms. By leveraging direct memory access (DMA), smc_tune significantly reduces CPU load and accelerates interactive workloads.

CPU burst is tailored for high-load container environments like Doris, addressing performance bottlenecks caused by CPU constraints.

realtime_tune is applicable to scenarios that demand low latency, minimal jitter, and strict latency restrictions, such as in the industrial manufacturing sector.

net_hard_irq_tune is best suited for services whose performance is heavily impacted by TCP networking, including applications like Redis and Nginx.

NUMA-aware scheduling applies to workloads, like Spark, where cross-NUMA memory access significantly degrades service performance and fixed core pinning is difficult to maintain.

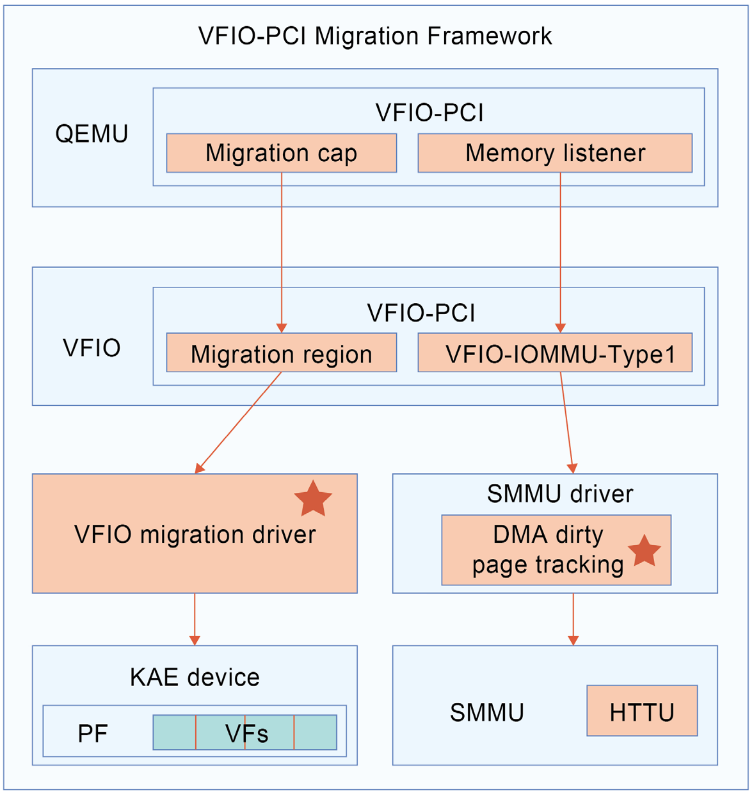
Traffic-based container scheduling is ideal for containerized services, such as containerized MySQL. When the service load is low, it satisfies QoS requirements while using the minimum necessary CPU resources. This reduces CPU migration, idle switching, cache misses, and cross-NUMA access, thereby enhancing resource locality. When the service load is high, it can bypass core pinning constraints to utilize more CPU resources, improving QoS and maximizing overall CPU resource utilization.

vKAE Passthrough Live Migration

Feature Description

The Kunpeng Accelerator Engine (KAE) is a hardware acceleration solution based on the new Kunpeng 920 processor model, featuring HPRE, SEC, and ZIP components for encryption, decryption, compression, and decompression. This allows KAE to significantly reduce processor overhead and boost efficiency. KAE passthrough live migration addresses the scenario where VMs are configured with KAE passthrough devices, offering enhanced operational flexibility and continuous service availability.

SMMU dirty page tracking is a key technology for efficient and reliable live migration of passthrough devices. In the Arm architecture, a purely software-based approach to dirty page tracking incurs significant performance overhead. Hardware Translation Table Update (HTTU) solves this by allowing the hardware to automatically update the SMMU page table status. During a write operation, the write permission bit of the corresponding page table entry is automatically set. During a live migration, the write permission bit of the page table is scanned to collect statistics on dirty pages.



Application Scenarios

This feature empowers live migration for VMs using KAE passthrough devices, making it ideal for fields with high requirements for data security and processing performance, such as finance, cloud computing, and big data processing. Ultimately, it enhances business continuity and operational stability.

Enhanced GTA Remote Attestation

Feature Description

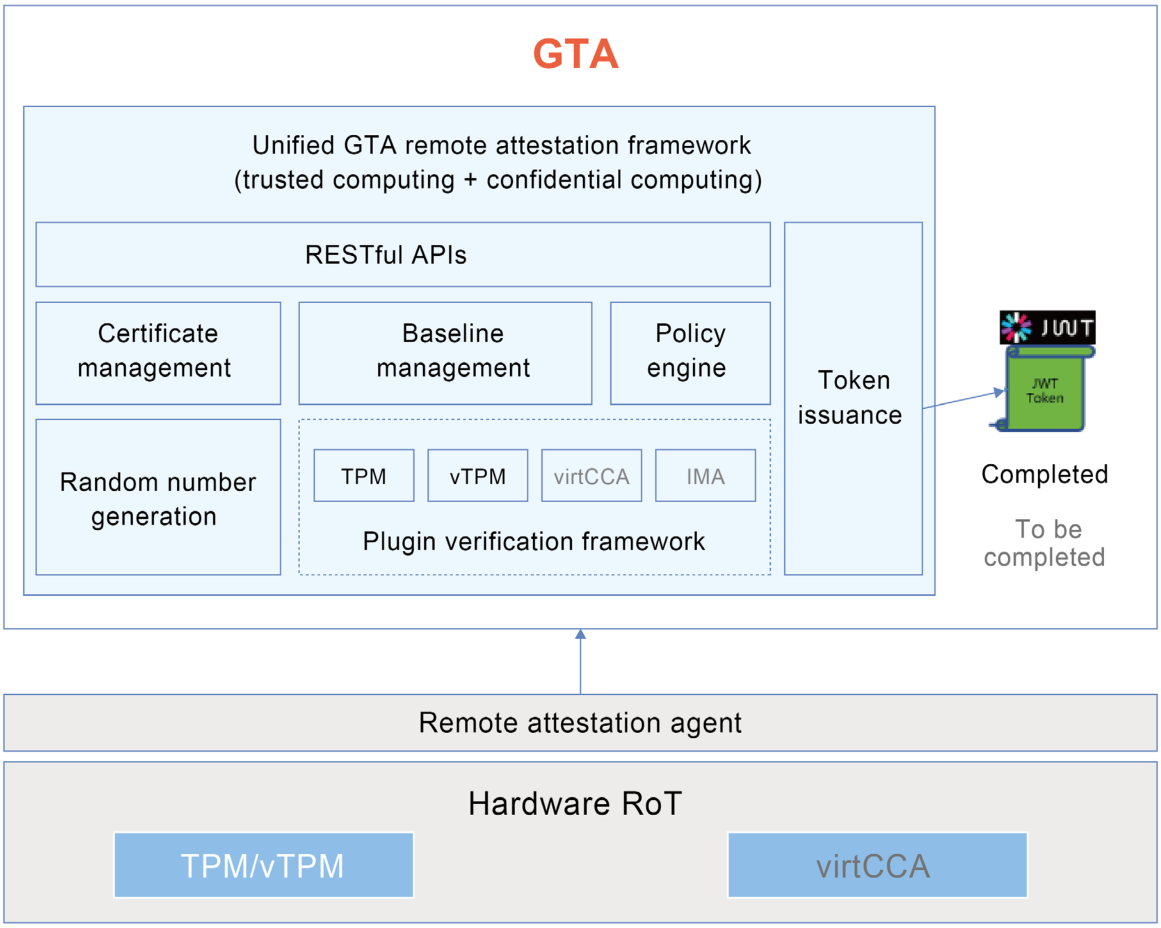
The Global Trust Authority (GTA) remote attestation component adopts a client-server architecture, supporting remote attestation of TPM/vTPM, virtCCA, and IMA.

- The server provides the remote attestation service framework, which is compatible with trusted computing and confidential computing. It supports the addition, deletion, modification, and query of certificates and policies, quote verification, random number generation, and JWT token generation.
- The client collects local TPM evidence and interacts with the server to verify quotes.

This component provides various capabilities in terms of security and usability.

GTA provides differentiated security competitiveness such as database integrity protection, data link encryption, anti-replay, SQL injection prevention, user isolation, and key rotation.

The passport and background-check models are available. The client supports multiple verification modes, such as scheduled reporting and challenge response. Both the client and server can be deployed using RPM packages and within Docker containers.



Application Scenarios

Remote attestation is a prerequisite for enabling confidential computing and trusted computing. Subsequent computations are allowed only when the operating environment is strictly proven to be secure and trustworthy in the cryptographic sense. Remote attestation should be included as a core component of all end-to-end solutions involving confidential computing. If the operating environment is untrusted, the subsequent tasks must be stopped. This service is widely used in AI model protection, user privacy protection, and key management.

Kuasar Confidential Container

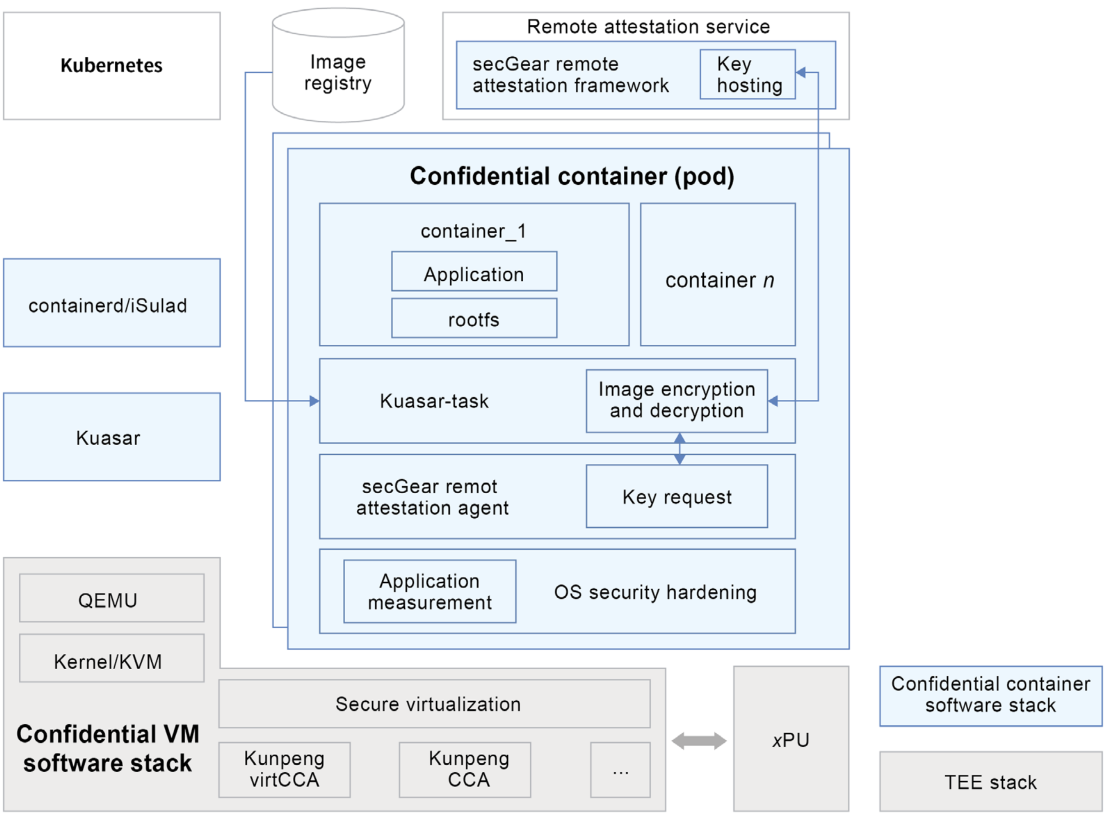
Feature Description

Kuasar has expanded its capabilities to include confidential container support while maintaining its existing secure container functionality. This support can be enabled through iSulad runtime configuration.

The current Kuasar confidential container implementation leverages the iSulad+Kuasar solution to significantly accelerate boot times and drastically reduce memory overhead. On the one hand, the Sandbox API eliminates the need to create a separate pause container during container creation, saving time spent on preparing the pause container image snapshot. On the other hand, the 1:N management model allows the Sandboxer process to be persistent. This avoids the cold-start time of the Shim process, greatly accelerating container boot and bringing memory benefits proportional to the number of pods. Finally, Kuasar is implemented in Rust. Compared to Go, Rust provides inherent advantages in memory safety and contributes to overall memory efficiency.

Key functions include:

- Native integration with the iSulad container engine preserves Kubernetes ecosystem compatibility.
- Hardware-level protection via Kunpeng virtCCA technology ensures confidential workloads are deployed in trusted environments.
- The secGear remote attestation framework, which complies with the remote attestation procedures (RATS) (RFC9334), allows containers running in a confidential computing environment to prove their trustworthiness to external trusted services.
- Container images can be pulled and decrypted in confidential containers to protect their confidentiality and integrity.



Application Scenarios

Kuasar addresses data security concerns while seamlessly integrating with cloud native ecosystems. It benefits confidential applications from cloud native advantages including high availability, auto scaling, and rapid delivery. The solution finds broad application in confidential computing scenarios spanning AI security, trusted data circulation, and privacy protection.

Native .NET Development Capabilities

Mono is a complete, cross-platform development tool set and runtime environment compatible with Microsoft's .NET Framework, allowing developers to use the C# language and the .NET Framework class libraries. .NET is an open source, cross-platform developer platform for building various applications. It provides a unified ecosystem encompassing programming languages, a runtime environment, a vast code library, and a rich set of development tools. openEuler offers native development capabilities for the .NET Framework, Mono, and .NET applications.

Feature Description

- **openEuler supports MonoDevelop and its dependencies:** MonoDevelop is a powerful, open source Integrated Development Environment (IDE) tailored for .NET developers. By configuring the Mono runtime, it supports the development, debugging, code management, compilation, building, and integrated testing of .NET Framework applications on openEuler.
- **MonoDevelop supports quick deployment:** Developers can easily deploy and uninstall MonoDevelop on the oeDeploy platform with a few clicks.
- **openEuler supports the .NET SDK and its dependencies:** The .NET SDK and its dependencies have been adapted and introduced to openEuler, currently supporting up to .NET 9. This enables the development of modern .NET applications directly on openEuler.
- **.NET SDK supports quick deployment:** Developers can deploy and uninstall the .NET SDK on the oeDeploy platform with a few clicks.

Application Scenarios

The openEuler .NET native development kit is designed for .NET application development scenarios. It allows developers who traditionally rely heavily on the Windows development environment to develop new .NET applications or iterate on existing ones within openEuler. This approach retains the familiar .NET development paradigm while granting access to the flexibility of the Linux ecosystem and its powerful toolchain, thereby reducing reliance on a single platform.

Raspberry Pi

Raspberry Pi is a series of single-board computers developed by the Raspberry Pi Foundation and Broadcom. They are widely used in industrial automation, robotics, Internet of Things (IoT), education, and enthusiast projects due to their low price, small size, low power consumption, high programmability, and abundant ecosystem. Raspberry Pi 4B and Raspberry Pi 5 are classic products, both using Arm processors. Raspberry Pi 4B is a cost-effective entry-level computer, and Raspberry Pi 5 is an innovative product with significant performance breakthroughs and extended capabilities, making it competitive in high-performance edge computing.

Feature Description

As typical open source hardware products, Raspberry Pi 4B and Raspberry Pi 5 support multiple Linux distributions such as Raspberry Pi OS, Ubuntu, and openEuler. They have extensive peripherals, powerful video encoding and decoding capabilities, LOM, and can be used as independent computer systems.

Application Scenarios

Raspberry Pi 4B and Raspberry Pi 5 are widely used in many fields:

- **Education:** programming language learning such as Python, and electronic experiments using the peripheral interfaces
- **Multimedia and entertainment:** media center or game console
- **IoT and smart home:** sensor nodes or smart home hubs for environment monitoring, automation control, and edge computing
- **Server and network applications:** home servers, lightweight web services, and containerized applications
- **DIY projects:** robot control, 3D printing management, and drone flight control
- **Research and development:** AI experiments and prototype validation in embedded development
- **Industrial automation:** device monitoring, man-machine interface, and machine vision

08 Copyright Statement

09 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.

10 Appendixes

Appendix 1: Setting Up the Development Environment

Environment Setup	URL
Downloading and installing openEuler	https://www.openeuler.org/en/
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/docs/en/vulnerability-management-process/security-process-en.md
Security disclosure	https://gitee.com/openeuler/security-committee/blob/master/docs/en/vulnerability-management-process/security-disclosure-en.md
Security strategy overview	https://gitee.com/openeuler/security-committee/blob/master/docs/en/vulnerability-management-process/security-strategy-overview-en.md