openEuler OS Technical Whitepaper

Innovation Projects

(June, 2023)

OpenAtom openEuler Community

CONTENTS

CONTENTS

01
Introduction

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

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

The openEuler open source community collaborates with global developers to create an inclusive and diverse software ecosystem that caters to all digitalization scenarios. This ecosystem empowers enterprises to develop their software, hardware, and application ecosystems.

Development Roadmap

2023.06
Signed a letter of intent for donations to Vtopia agent, secScanner, KsPack, and secShield

2023.04
Received 5 project donations: GoStone, CTinspector, ZVM, Rust-Shyper, and QuickPool

2022.12
Set up the openEuler Project Group and openEuler Committee
Started independent project donations

2022.11
Won the World Leading Internet Scientific and Technological Achievements Award

2022.09
Debuted at Open Source Summit Europe 2022

2022.03
The first versatile-scenario digital infrastructure release, openEuler 22.03 LTS
Nine partners released commercial distributions based on openEuler 22.03 LTS, with a total of 1.7+ million installations

2021.12
1+ million openEuler-based installations

2021.11
Huawei and community partners donated openEuler to the OpenAtom Foundation
300+ enterprise members

2021.09
Upgraded to an open source OS for digital infrastructure

2021.08
100+ enterprise members

2020.12
Community Council officially established
30,000 commercial installations

2020.03
The first LTS release, openEuler 20.03 LTS
Commercial releases by Kylinsoft, iSOFT, UnionTech, and Software Research Institute, Chinese Academy of Sciences

2019.12
Code officially open-sourced
Community established

2019.09
Huawei unveiled openEuler

02

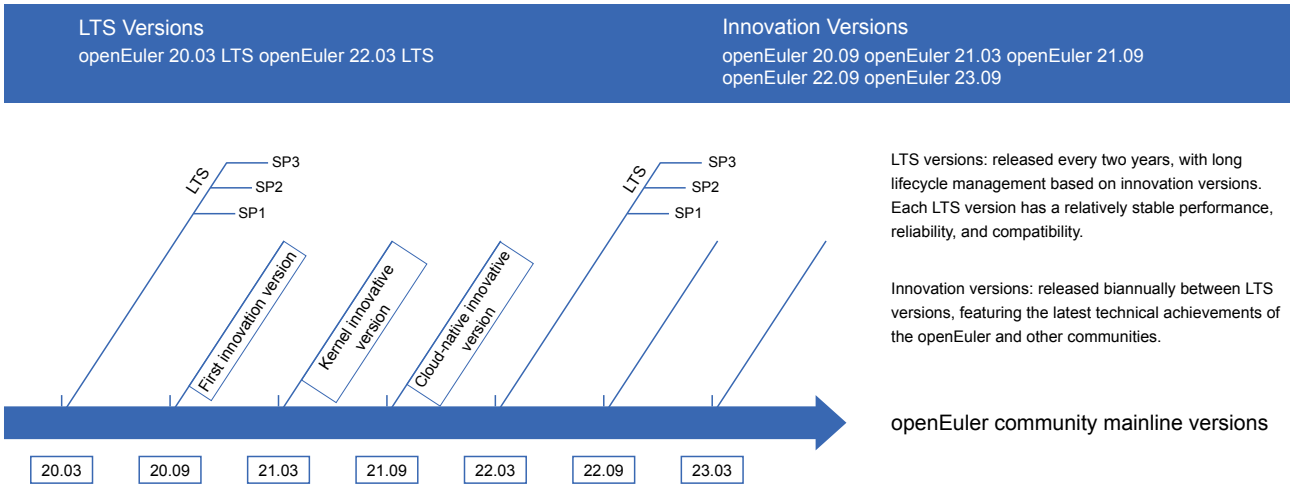
Technology Ecosystem

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

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

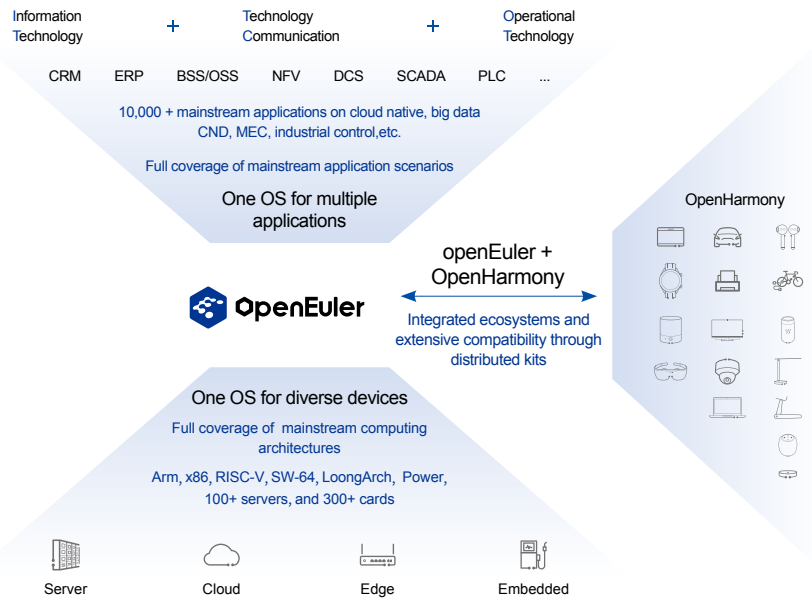
Technical capabilities are first tested in the open source community, and continuously incorporated into each openEuler release. In addition, each release is built on feedback given by community users to bridge the gap between innovation and the community, as well as improve existing technologies. openEuler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

openEuler Version Roadmap



Innovative Platform for Versatile Scenarios

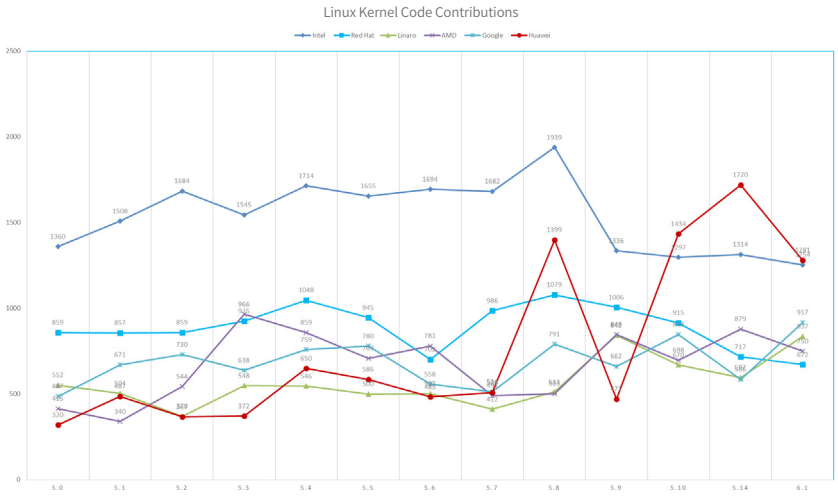
openEuler supports a diverse range of devices, and covers various application scenarios, and interfaces with other OSs such as OpenHarmony, achieving ecosystem interoperability through shared capabilities. With a unified OS architecture supporting all mainstream computing architectures, openEuler is one of the best open source OSs for diverse computing powers. It introduces the concept of the versatile-scenario OS, which achieves flexible version build and service composition through a full-stack atomization decoupling and Lego-style architecture, making it ideal for servers, cloud computing, edge computing, and embedded systems. This white paper aims to provide a comprehensive overview of openEuler's architecture and its capabilities in supporting various digital infrastructure scenarios.



Everlasting Contribution to the Linux Kernel

Huawei, Loongson, Kylinsoft, UnionTech Software, Phytium, Chengdu JRLC, China Telecom, China Mobile, and other members of the openEuler community continuously contribute to the Linux kernel, covering chip architecture, Advanced Configuration and Power Interface (ACPI), memory management, file systems, media, kernel documents, bugfixs for kernel quality hardening, and code rebuilds.

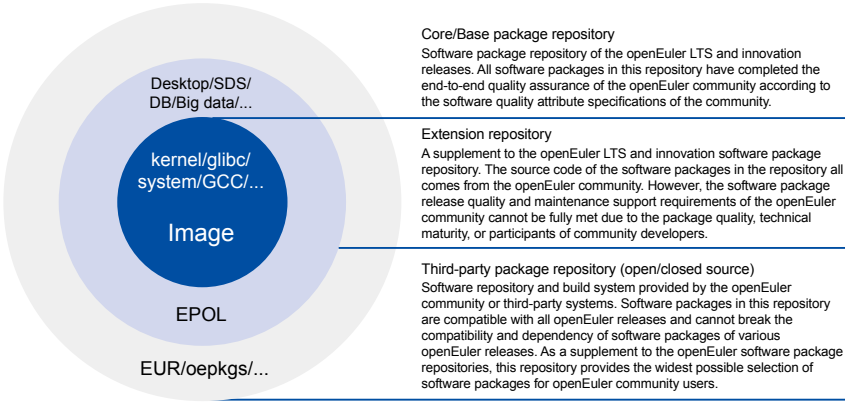
The openEuler community boasts 905 community partners and over 14,000 contributors, which continues to grow. The total number of merged PRs has surpassed 100,000, while the community supports 34,000 software packages, which has achieved over 1.38 million global downloads.



Huawei, a strategic member of openEuler, ranks No.1 in Linux Kernel contributions (5.10, 5.14, and 6.1).

Software Package Repositories

The openEuler community works with third-party developers to provide a vast array of user-friendly software packages. The community has categorized its software repositories into three types based on the source, quality attribute, and package maintenance mode. You can configure software repositories according to their specific needs. The redistribution of software packages across different repositories is subject to community rules, which are based on usage, stability, and maintenance status.



Open and Transparent Management of the Open Source Software Supply Chain

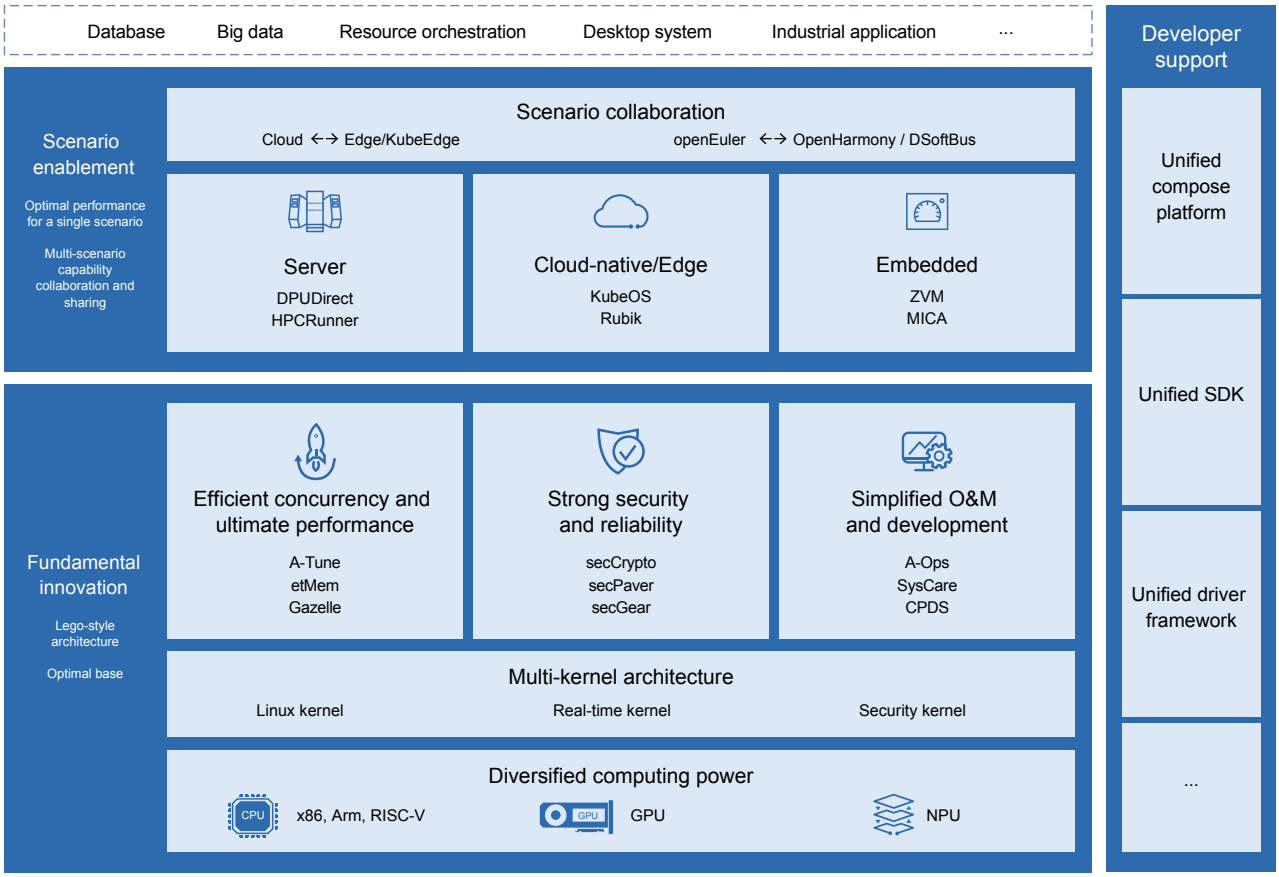
The process of building an open source OS relies on supply chain aggregation and optimization. A reliable open source software supply chain is fundamental to a large-scale commercial OS. openEuler combs through its software dependencies based on real user scenarios, sorts out the upstream community addresses of all the software packages, and verifies its source code in comparison to the upstream communities. This is a complete lifecycle management throughout build, verification, and distribution. The build, runtime dependencies, and upstream communities of the open source software form a closed loop, realizing a complete, transparent software supply chain management.

Community-certified openEuler Distributions

(Sorted by certification time)

Partner	System
xFusion Digital Technologies Co., Ltd.	FusionOS 22
UnionTech Software Technology Co., Ltd.	UOS V20 (1050e)
Hunan Kylinsec Technology Co., Ltd.	Kylinsec OS V3 (openEuler distribution)
SUSE	SUSE Euler Linux 2.0
H3C Technology Co., Ltd.	H3Linux 2.0.2
Jiangsu HopeRun Software Co., Ltd.	HopeEdge V1.0 HopeStage V1.0
Nanjing Fiberhome Starrysky Co., Ltd.	fitstarryskyos 22.06
Beijing Linx Software Co., Ltd.	linxos 6.0.99
China Telecom e-Cloud Technology Co., Ltd.	CTyunOS
TurboLinux Inc.	TurboLinux Enterprise Server 16
Kylinsoft Co., Ltd.	Galaxy Kylin Advanced Server Operating System V10
Shenzhen Archforce Technology Co., Ltd.	ArchforceEuler 22.09
iSoftStone Information Technology (Group) Co., Ltd.	ISSEL 22 LTS
CSG Digital Power Grid Group Co., Ltd.	pegaspegasus server v1.0
DBAPPSecurity Co., Ltd.	dasos e2.1.0
China Mobile (Suzhou) Software Technology Co., Ltd	BC-Linux for Euler 21.10
iSOFT Infrastructure Software Co. Ltd.	iSoft Server OS V5.1
Eversec (Beijing) Technology Co., Ltd.	EversecOS 20.03
PowerLeader Computer System Co., Ltd.	RedderStar V1.0
China Unicom Digital Technology Co., Ltd.	CULinux
Guangdong ZTE NewStart Technology Co.,Ltd	NewStartOS Server V6.02

openEuler Open Source OS Architecture



03

Scenario-specific Innovations

DPUDirect

Server Cloud

DPU SIG

DPUDirect creates a collaborative operating environment for services, enabling them to be easily offloaded and ported between hosts and DPUs. The feature includes a process-level seamless offload function and a cross-host and -DPU collaboration framework. This allows management-plane processes to be split and offloaded to the DPU without requiring reconstruction. Once offloaded, these processes can continue managing processes on the host side. The DPUDirect feature significantly reduces service offloading costs in DPU scenarios, simplifies O&M, and significantly reduces subsequent maintenance costs.

Challenges

iNICs are gradually evolving into DPUs/IPUs, becoming an increasingly important part of cloud and data center infrastructure. In addition to accelerating I/O on the data plane, DPUs/IPUs are also supporting the offloading of management- and control-plane components. This means that all management and control components of the data center infrastructure can be offloaded to the DPU, resulting in better architecture and more flexible deployments. Most mainstream offload solutions involve splitting components. However, this approach has several drawbacks.

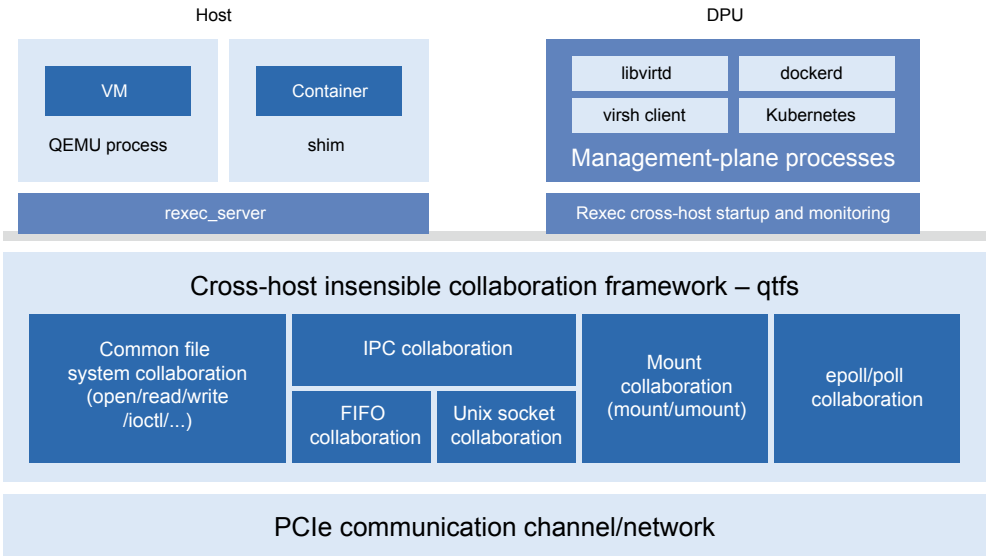
- Developers must understand the code of the components being offloaded.
- Cloud vendors maintain a large number of components, making offloading a burden.
- The code of offloaded components is difficult to inherit between upgrades and must adapt to the newest version, which is complicated and expensive.

Project Introduction

DPUDirect builds a cross-host collaboration framework at the OS layer of the host and DPU, providing a consistent runtime view for the management-plane processes offloaded to the DPU and the service processes on the host. In this way, applications are unaware of offloading. Only a small amount of service code on the management plane is needed to ensure software compatibility and evolving services, as well as reducing component maintenance costs.

Features

The figure illustrates the architecture of DPU management plane seamless offload framework. qtfs, a cross-host collaboration framework is established at the host and DPU OS layers to provide a consistent runtime view for the management plane processes offloaded to the DPU and the service processes on the host. This approach ensures that applications remain unaware of the offloading process.



- DPUDirect allows you to combine the following collaboration mechanisms to achieve seamless offloading in various scenarios. File system collaboration supports cross-host file system access and provides a consistent file system view for host and DPU processes. It also supports special file systems such as proc, sys, and dev.
- IPC collaboration enables unaware communication between hosts and DPU processes. It supports FIFO and Unix domain sockets for cross-host communication.
- Mounting collaboration performs the mounting operation in a specific directory on the host, which can adapt to the container image overlay scenario. The offloaded management-plane process can construct a working directory for the service process on the host, providing a unified cross-node file system view.
- epoll collaboration supports epoll operations for cross-host access of remote common files and FIFO files, and supports read and write blocking operations.
- Process collaboration uses the rexec tool to remotely start executable files. The rexec tool takes over the input and output streams of the remotely started processes and monitor the status to ensure the lifecycle consistency of the processes at both ends.

By combining these mechanisms, policies can be tailored for different scenarios to fulfill the service requirements of the management-plane processes, eliminating the need to split and reconstruct too many services.

Application Scenarios

DPUDirect facilitates the complete offloading of the container management plane such as kubelet and dockerd, as well as the virtualization management plane libvirtd. It eliminates the need of splitting over 10,000 lines of code, thereby reducing the workload of adapting and maintaining by almost 20-fold. Furthermore, the service logic of the management plane remains unaltered, ensuring service software compatibility and evolution.

Repositories

https://gitee.com/openeuler/dpu-utilities

eNFS

Server | Cloud

Kernel SIG

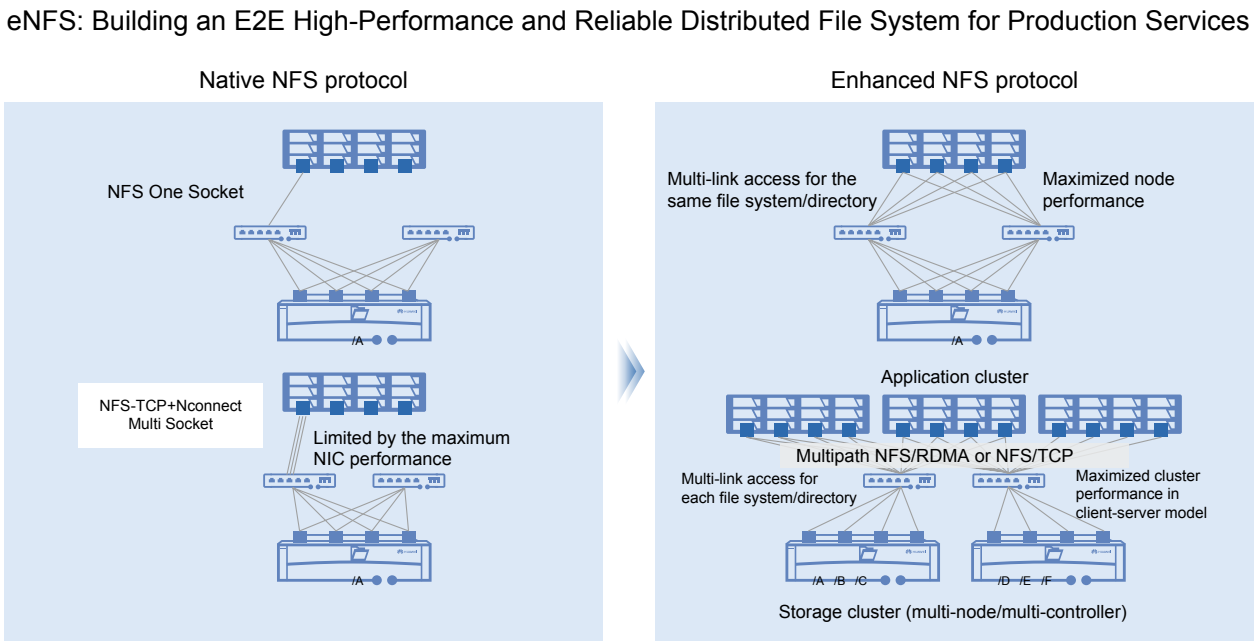
The exponential growth of unstructured data has seen Network Attached Storage (NAS) become the most viable option for managing massive unstructured production services. Enhanced Network File System (eNFS) helps improve the performance and reliability of NAS. It establishes multiple links for I/O load balancing and automatically switches I/Os to other available paths in case of a link failure, ensuring uninterrupted high-performance and high-reliability services.

Challenges

With the emergence of diverse application scenarios, data importance is increasing, and various industries are putting forward higher requirements for the reliability and performance of NAS. However, traditional NFS only specifies one server IP address for a single mount point, which poses several challenges.

- When an NIC or link is faulty, the mount point becomes inaccessible, suspending service I/Os and compromising reliability.
- The performance of a mount point is limited by the performance of a physical link, which poses challenges for the performance of important services.
- NAS is often deployed in the public zone, and accessing it from the host requires crossing three layers of networking. If one end is faulty, the IP address cannot detect the fault. File systems are manually mounted on the application layer, and active-active links cannot be automatically switched over.

Project Introduction

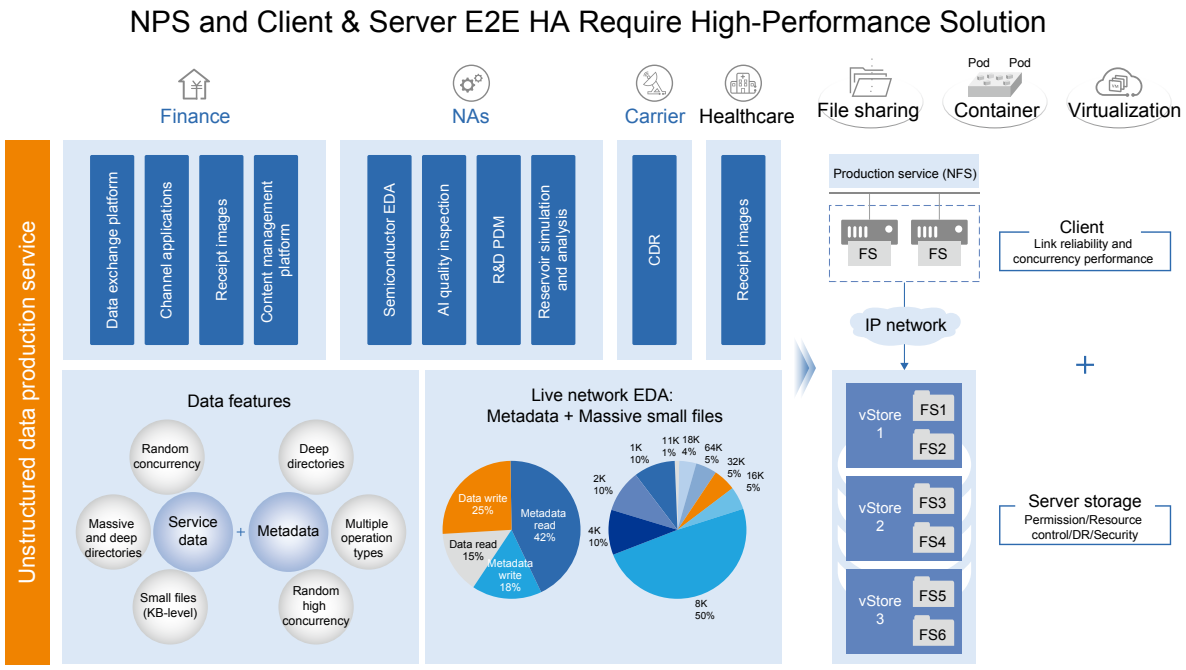


The eNFS protocol is a driver module running in the openEuler OS kernel, consisting of the mounting parameter management module at the NFS protocol layer and the multipathing management module at the transport layer. eNFS specifies multiple local and server IP addresses to establish multiple TCP/RDMA links for different IP addresses, achieving functions such as multi-path link setup, fault recovery and failover, and load balancing.

Compared to native NFS, eNFS boasts three key innovations.

- It enables second-level failover in the event of software or hardware faults on I/O paths. It also ensures multiple links between the client and server for each single NFS mount point to support I/O transmission over those links, achieving cross-controller and cross-site reliability. All configurations are recorded in one file, making it easy to deploy HPC applications on different hosts by simply modifying the configuration file.
- Multi-link aggregation, covering NIC ports, NICs, and nodes, significantly improves host access performance.
- Notably, eNFS is the industry's first protocol to offer active-active path failover on a three-layer network. It enables cross-site active-active failover in the event of storage faults or host-side I/O timeouts, effectively resolving cross-engine failures and host unawareness issues.

Application Scenarios



eNFS provides high-performance data sharing capabilities beyond local file systems and also solutions to faults between the client and server, ensuring service continuity and replacing native NFS.

Repositories

https://gitee.com/openeuler/kernel

HPCRunner

Server | Cloud

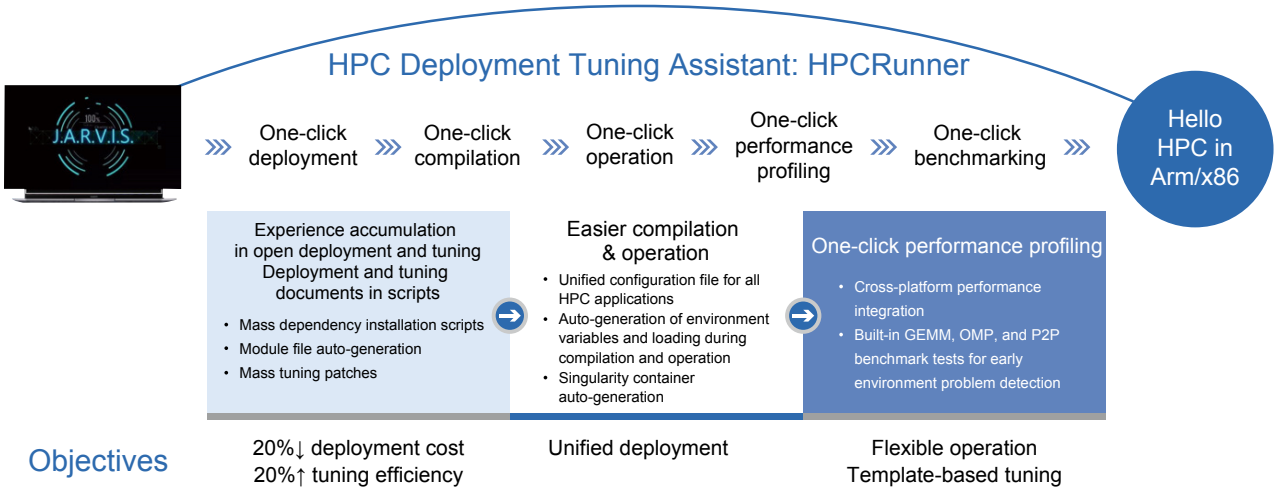
HPC SIG

With a mission to make HPC accessible to everyone, HPCRunner is committed to offering comprehensive solutions for application deployment and tuning, with the aim of making HPC accessible to more people. Our closed-loop ecosystem includes dependency management, performance analysis, application compilation, and automatic containers, catering to the needs of developers, supercomputing centers, and scientific research institutes. Our goal is to simplify the HPC process and provide a one-stop application for all your needs.

Challenges

The deployment process for HPC applications is not particularly straightforward. Each dependency must be installed manually, and various compilers and Message Passing Interface (MPI) libraries are needed for compilation. Additionally, the application of multiple hardware-related parallel and communication technologies lead to the increasing complexity of porting between different architectures, and deploying and optimizing HPC applications can be quite demanding. To tackle this challenge, openEuler has designed and implemented HPCRunner, an HPC deployment tuning assistant that significantly reduces deployment costs and improves optimization efficiency.

Project Introduction



HPCRunner is composed of two parts: HPC dependency management and HPC application management. All third-party HPC dependencies are uniformly managed by modules. With an installation script provided, HPCRunner automatically downloads, decompresses, compiles, installs, and configures dependencies, and generates packages. This solves the problem of multiple versions coexisting, which cannot be implemented by Yum. Additionally, HPCRunner supports deployment of closed-source components with just a few clicks. As for HPC application management, HPCRunner abstracts all HPC applications as a monolithic configuration file. You only need to compile commands for compilation, environment, running, and batch running in the configuration file to automatically compile and run applications.

Features

- HPCRunner supports the Arm/x86 architecture, 100+ dependencies, and the deployment and installation of 60+ applications within a few clicks. It uses the authoritative dependency directory structure in the industry to manage massive dependencies and automatically generates module files.
- HPCRunner implements one-click compilation and operation, CPU/GPU performance profiling, and benchmarking based on HPC configurations.
- All configurations are recorded in one file. To deploy HPC applications on different hosts, you only need to modify the configuration file.
- The log management system automatically records all information during HPC application deployment.
- The software itself does not need to be compiled and can be used out of the box on the Python environment.
- HPC application containerization is also supported, with automatic connection to Singularity containers.

Application Scenarios

Scenario 1: Supercomputing center application management

HPCRunner is a powerful tool that simplifies the management of dependencies among different software packages. It ensures that these packages are compatible with the required library and tool chain versions, making it easy to upgrade compilers and parallel libraries. Furthermore, HPCRunner automates and simplifies the deployment of software packages of the same version on multiple hosts, reducing manual input and avoiding errors.

Scenario 2: Scientific computing experiment

HPCRunner offers flexible configuration options for compiling applications, allowing you to optimize performance and resource utilization as needed. You can also record software package versions and compilation options to ensure that identical binary files are generated on different platforms with the same chip architecture, ensuring the test-retest reliability. Furthermore, to adapt to the Kunpeng platform, HPCRunner integrates with the Kunpeng software full stack, which includes a compiler, mathematical library, communication library, and porting and tuning tool chain, significantly improving the application running efficiency of the Kunpeng platform.

Repositories

https://gitee.com/openeuler/hpcrunner

WayCa Scheduler

Server

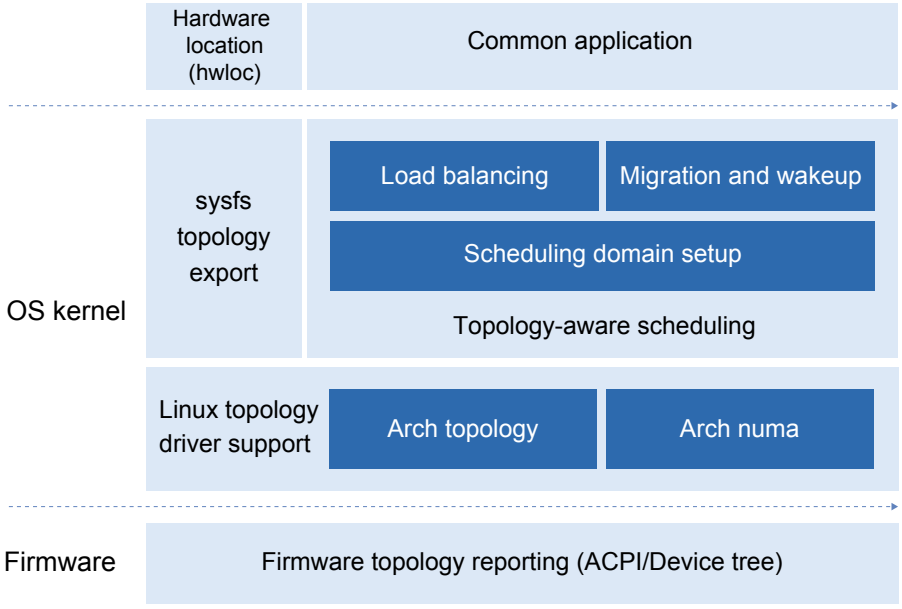
WayCa SIG

WayCa Scheduler provides a set of software libraries and tools to optimize and complete the hardware topology export and task scheduling of the Kunpeng platform based on Linux.

Challenges

The number of server cores is on the rise, and the cache and interconnection structures are becoming increasingly complex. Server designs differ greatly from vendor to vendor. Although Linux is a general-purpose OS and supports hardware from various vendors, it does not fully support the newly introduced hardware topology of Kunpeng. This presents a challenge in fully utilizing the Kunpeng hardware performance on the existing software platform.

Project Introduction



By optimizing the firmware topology description, hardware topology establishment, and topology information export in the Linux kernel, as well as the kernel scheduling algorithm, WayCa Scheduler enables applications to fully utilize components such as the CPUs, cache, memory, and I/O peripherals. This feature improves system hardware utilization and memory bandwidth, while reducing memory, cache, and peripheral access latency, thereby significantly improving application performance on Kunpeng servers.

WayCa Scheduler offers several features, including topology discovery and export, scheduling support and optimization, and user-mode topology support.

- The Linux ACPI and topology driver can enumerate and create hardware topologies for CPUs, cache, NUMA, and devices, and search for hardware topology information through kernel interfaces such as sysfs.

- Cluster and NUMA scheduling domains are established based on the hardware topology, and the scheduler supports load balancing and migration based on hardware clusters and NUMA nodes. This fully utilizes L3 cache and memory resources, reducing system latency and improving throughput.
- Furthermore, WayCa Scheduler provides user-mode topology support, which allows for optimization based on specific service characteristics and requirements. For example, specific CPU or device binding policies can be implemented. To meet the requirements of these applications, hwloc is adapted to provide hardware topology information for applications.

Application Scenarios

WayCa Scheduler has been integrated into the openEuler kernel and its related user-mode tools. This integration enables hardware topology awareness and optimizes scheduling for applications on the Kunpeng server and openEuler OS, for example, general-purpose databases, thereby improving system performance. In certain application scenarios, such as HPC applications, tools like hwloc can be utilized to obtain topology information that meets the optimization policy requirements, further improving the application performance.

Repositories

https://gitee.com/openeuler/wayca-scheduler

HybridSched

Cloud

Virt SIG

Low resource utilization of cloud data centers is a common issue in the industry, and has fueled ways to improve this problem, such as deploying priority-based services (hybrid deployment).

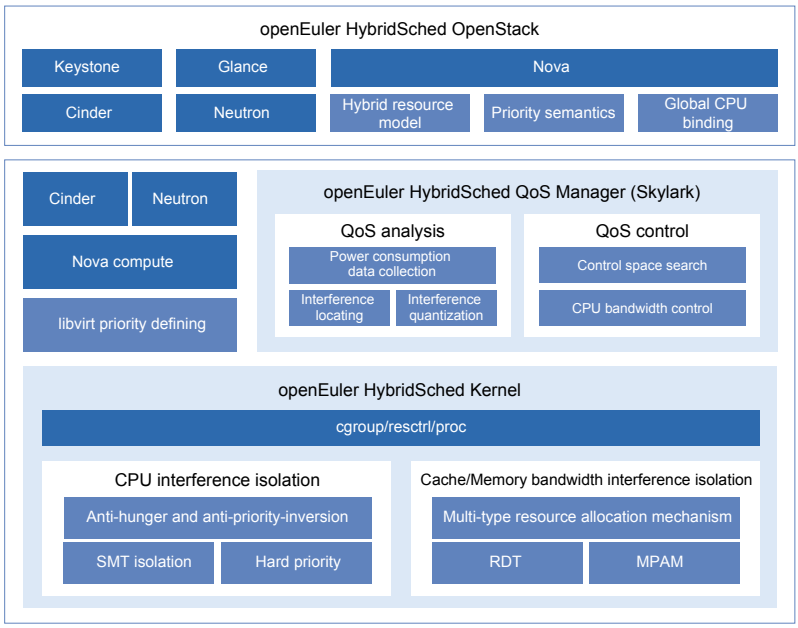
HybridSched is a full-stack solution for hybrid deployment of VMs, covering enhanced OpenStack cluster scheduling, new single-node QoS management component Skylark, and kernel-mode basic resource isolation. In particular, Skylark is a QoS-aware resource scheduler used when high- and low-priority VMs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

Challenges

The core technology behind hybrid deployment is resource isolation and control. Server resources are multi-dimensional, and the requirements of the upper-layer services for hardware resources are changing dynamically. Not only this, internal services of VMs are invisible, and the underlying scheduling logic cannot detect the degree of service loss. These two aspects hinder the hybrid scheduling technology of the virtualization platform.

Project Introduction

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



Application Scenarios

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

Repositories

https://gitee.com/openeuler/skylark

KubeOS

Server | Cloud | Edge

Cloud Native SIG

KubeOS is an OS designed specifically for cloud-native scenarios, with a focus on containerization. KubeOS is a lightweight and secure operating system that can be centrally managed through Kubernetes. It enables unified management of both containers and node OS through Kubernetes, including atomic upgrades and API-based operations.

Challenges

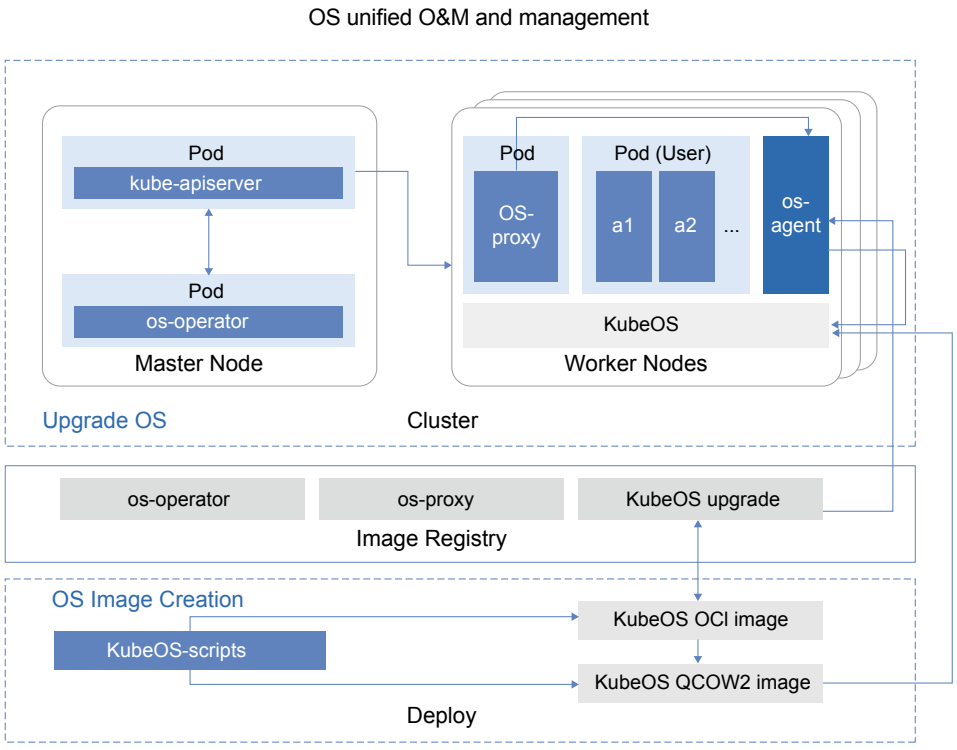
In cloud-native scenarios, containers and Kubernetes are widely used. However, the management of OSs is affected.

- With applications being containerized, new challenges arise for OSs. Traditional OSs are too heavy and no longer fully applicable.
- Containers and OSs are maintained and managed separately, which results in redundant management functions and difficult scheduling.
- Siloed package management causes problems such as scattered and inconsistent container OS versions in the cluster. This requires a unified container OS management mechanism.

To address these issues, we launched KubeOS, an OS O&M solution that uses Kubernetes to manage containers and OSs in a unified manner.

Project Introduction

KubeOS leverages Kubernetes' CRD+Operator extension mechanism to create custom resources for OSs. This enables the OS to be treated as a Kubernetes resource, monitored by the OS-Operator and OS-Proxy within the cluster. You can use Kubernetes to set the desired state and information of the OS, and trigger processes such as upgrades, which are completed with the help of the OS-Agent for operational tasks.



Features

- Unified management: KubeOS connects the OS to the cluster as a component, uses Kubernetes to manage the OS and service containers, and manages the OSs of all nodes.
- Collaborative scheduling: It can detect the cluster status before the OS changes to implement collaborative scheduling of service containers and OSs.
- API O&M: Kubernetes-native declarative APIs are used to manage and maintain OSs in a standard manner.
- Atomic management: Based on the Kubernetes ecosystem, the atomic upgrade and rollback of the OS are implemented to ensure consistency between cluster nodes.
- Lightweight security: Only components required for container running are included, reducing the attack surface and vulnerabilities, overheads, and reboot time of the OS. The rootfs is read-only to protect the system from attacks and malicious tampering.

Application Scenarios

KubeOS is mainly used as a cloud-native infrastructure, providing a basic operating environment for cloud services, helping cloud vendors and customers in the telecom industry solve OS O&M problems in cloud-native scenarios.

Repositories

https://gitee.com/openeuler/KubeOS

NestOS

Server | Cloud | Edge | Embedded

K8s Distro SIG

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

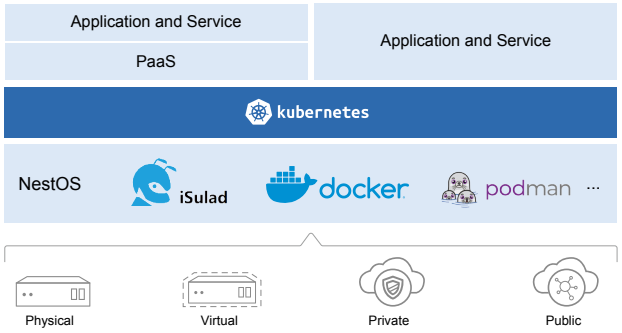
Challenges

Various runtimes and management software have been emerging as containers and Kubernetes are widely adopted in cloud native scenarios. Technologies such as container and orchestration further decouple service rollout and O&M from the underlying environment. Without a unified O&M stack, O&M platforms need to be built repeatedly.

Project Introduction

Features

- Out-of-the-box design: Integrates popular container engines such as iSulad, Docker, and Podman to provide lightweight and tailored OSs for the cloud.
- Easy configuration: Uses the Ignition feature to install and configure a large number of cluster nodes with a single configuration.
- Secure management: Runs rpm-ostree to manage software packages and works with the openEuler software package source to ensure secure, stable atomic updates.
- Hitless node updating: Uses Zincati to provide automatic node updates and reboot without interrupting services.
- Dual rootfs: Executes dual rootfs for active/standby failovers, to ensure integrity and security during system running.



Application Scenarios

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

Repositories

https://gitee.com/openeuler/NestOS

Rubik

Server | Cloud | Edge

sig-CloudNative

Rubik is a hybrid container deployment engine with adaptive single-node computing power optimization and quality of service (QoS) assurance. It schedules and isolates resources to improve node resource utilization while ensuring the QoS for mission-critical services.

Challenges

The expenditure on global cloud infrastructure services is enormous. However, the average CPU usage of user clusters in data centers is low, ranging from only 10% to 20%. This results in significant resource waste and additional O&M costs, which have become a critical issue for enterprises seeking to improve computing efficiency. Therefore, improving data center resource utilization has become an urgent matter that needs to be addressed. One solution is to deploy online and offline jobs together, utilizing idle online cluster resources to meet the computing requirements of offline jobs. This approach has become a research hotspot in academia and industry.

However, the hybrid deployment of multiple service types can also lead to a peak-sharing problem, which can result in a loss of QoS for mission-critical services.

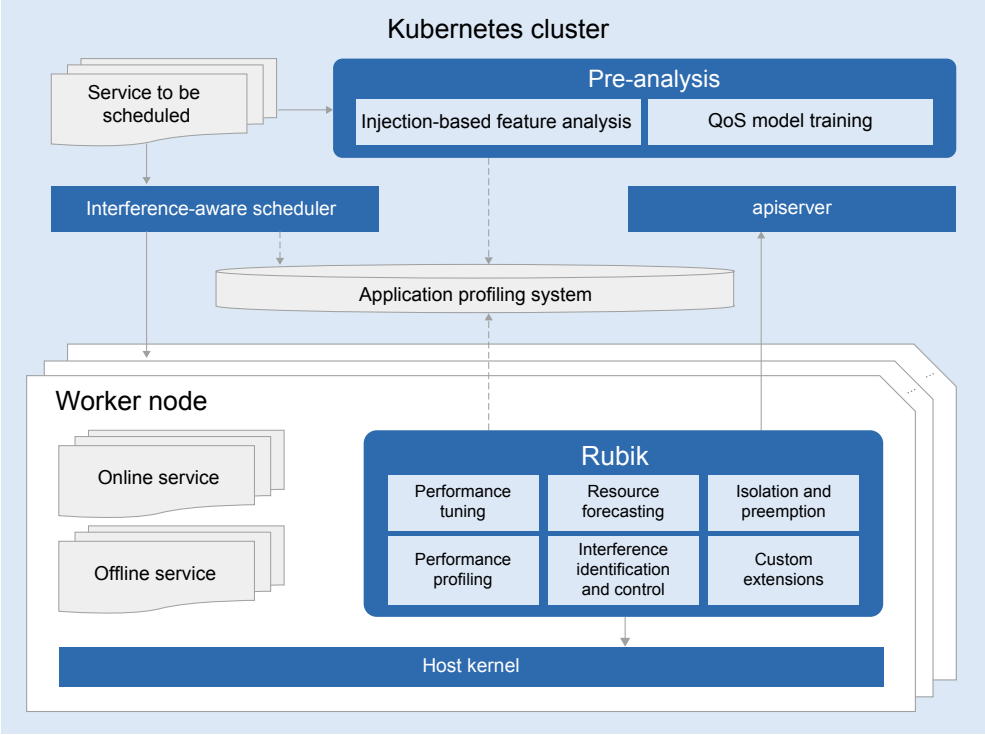
Therefore, ensuring that QoS is not affected after improving resource utilization is a key technical challenge that needs to be addressed.

Project Introduction

Features

- Compatible with the native Kubernetes system: Capabilities are extended based on extended APIs of native Kubernetes.
- Compatible with the openEuler system: Enhanced features (such as hierarchical kernel resource isolation) provided by openEuler are automatically enabled. For other Linux distributions, only restricted management capabilities are provided because some kernel features are missing.
- Interference identification and control during system running: Real-time interference identification and control capabilities are provided for mission-critical services.
- Adaptive dynamic optimization: The performance of mission-critical services is optimized to ensure efficient and stable operation, and the ratio of resources for online and offline services is dynamically adjusted to reduce QoS violations of mission-critical services.
- Custom extensions: Advanced users can develop custom extensions for specific service scenarios.

Rubik solution



Application Scenarios

Rubik is widely used in hybrid deployment of cloud service containers, covering hybrid deployment of web services, databases, big data, and AI. It helps customers in industries such as Internet and communications achieve a data center resource utilization rate of over 50%.

Repositories

https://gitee.com/openeuler/rubik

GearOS

Edge | Embedded

Industrial Control SIG

GearOS is a real-time operating system (RTOS) developed by the Industrial Control SIG of the openEuler open source community. It runs on openEuler Embedded and Yocto and enhances the real-time performance and reliability. GearOS can be used in automotive control, robotic control, programmable logic controller (PLC), and machine tool control.

Challenges

Modern industries must be equipped with intelligent information technologies that are deployed in industrial systems, which helps realize an industry and application ecosystem. Consider the example of smart manufacturing. Such integration of big data, AI, robotic control, and device-cloud collaboration systems can deliver automated production, network-based collaboration, customization, and other service transformations.

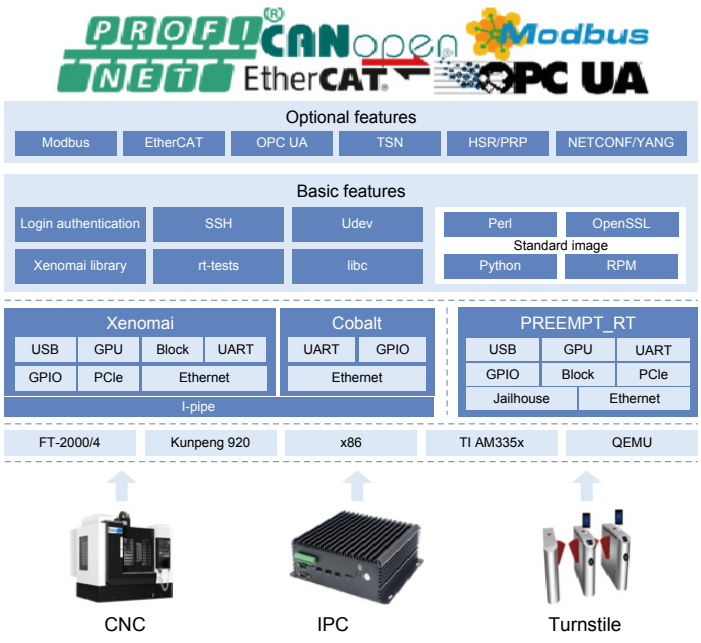
These technologies are not suited to Linux, which was initially designed to solve issues of throughput and scheduling but not real-time response. Due to their unique application requirements, industrial systems have an ever-increasing demand for real-time response, deterministic reaction, and functional security, making RTOSs a natural choice in industrial fields.

In practice, there are many challenges involved in integrating information technologies into industrial systems, but GearOS is equipped to resolve these problems.

Project Introduction

GearOS is built on AArch64 and contains two kernels and two file system images.

- Two kernels (8 MB each) are rebuilt on openEuler kernel 4.19, in which one supports PREEMPT_RT and Jailhouse virtualization, and the other supports Xenomai.
- Two file system images, including a compact file system image and a standard one. The compact one was created using BusyBox, with a size of 5.4 MB.



Key Features

- Compatible with the FT-2000/4, Kunpeng 920, TI AM335x, QEMU ARM64, and x86 platforms
- Minimum kernel size: 3.3 MB
- Serial port, network, block device, USB, and PCIe drivers
- Minimum file system size: 5.4 MB
- Boot within 5s
- PREEMPT_RT and Xenomai
- Jailhouse virtualization
- Compact file system image: login authentication, Udev, SSH, Xenomai library, and rt-tests
- Standard file system image: Python, Perl, OpenSSL, SQLite, and RPM

Additional Features Oriented to Industrial Fields

- Modbus protocol
- Ethernet for Control Automation Technology (EtherCAT) protocol
- Open Platform Communications United Architecture (OPC UA) protocol
- Time-Sensitive Networking (TSN) protocol
- High-availability Seamless Redundancy (HSR)/Parallel Redundancy Protocol (PRP)
- Network Configuration Protocol (NETCONF)/Yet Another Next Generation (YANG)

Real-Time Features

The following table shows the cyclictest results of GearOS on the FT-2000/4 and Kunpeng 920 hardware platforms.

Platform	Test Environment (No Load)			No CPU Isolation	CPU Isolation
	Test options: **cyclictest -m -h 100 -q -l 10000000 -i100 -t 1 -n -p 99**				
Kunpeng 920	openEuler 20.03 LTS SP1	General-Purpose Operating System (GPOS)	Linux only	76	3
			Linux + Xenomai	74	58
		RTOS	Xenomai	3	51
	GearOS	GPOS	Linux + Xenomai	7	6
			Linux + PREEMPT_RT	4	3
		RTOS	Xenomai	1	1
FT 2000/4	openEuler 20.03 LTS SP1	GPOS	Linux only	138	4
			Linux + Xenomai	633	13
		RTOS	Xenomai	7	4
	GearOS	GPOS	Linux + Xenomai	36	18
			Linux + PREEMPT_RT	10	7
		RTOS	Xenomai	2	1

Virtualization Features

- Impact on host machines

  A comparison test compared the impacts of virtualization on a bare metal server and a server with Jailhouse using UnixBench, lmbench, IOzone, and Netperf. The results show that enabling virtualization on the server causes little performance loss on the CPU, memory, storage, network, and I/O.

  The impact of Jailhouse on passthrough performance of the guest Linux bus on the FT-2000/4 hardware platform:
  - » The multi-dimensional tests on Ethernet NIC passthrough using Iperf and Netperf show almost no performance loss.
  - » In the PCI USB device passthrough tests using the dd, hdparm, and IOzone utilities, there is minimal performance loss.
  - » In the Iperf network tests on the Inter-VM shared memory device (ivshmem), the performance is better than that of gigabit NICs.

- Impact on guest systems
  - » According to the interrupt tests (5 million external interrupts), the average and maximum interrupt response time increases from 290 ns to 1,200 ns, and from 2,400 ns to 2,500 ns, respectively.
  - » The interrupt jitter is about 1,000 ns.
  - » According to the Iperf tests on Ethernet NIC passthrough (10 Mbit/s), there is almost no performance loss. (LwIP can only run on 10 Mbit/s.)
  - » The passthrough tests of the Inter-Integrated Circuit (I2C bus) show almost no performance loss (average value taken from 1,000 tests).
  - » The passthrough tests of the Serial Peripheral Interface (SPI bus) show near-zero performance loss (average value taken from 1,000 tests).
  - » The passthrough tests of the Controller Area Network (CAN bus) show little performance loss (average value taken from 1,000 tests).

Application Scenarios

This project aims to simulate and reflect typical computing workloads of the power grid project, smart turnstiles, and computer numerical control (CNC) machine tools.

Repositories

https://gitee.com/openeuler/GearOS

MICA

Edge | Embedded

Embedded SIG

Mixed-criticality (MICA) is a multi-core SoC framework that supports mixed-criticality deployment of real-time and non-real-time OSs or secure and non-secure OSs based on hardware-assisted virtualization, trusted execution environment (TEE), and heterogeneous architectures. It utilizes the characteristics of each OS to meet the multi-objective constraints of embedded systems, such as security, real-time response, and extensive functions.

Challenges

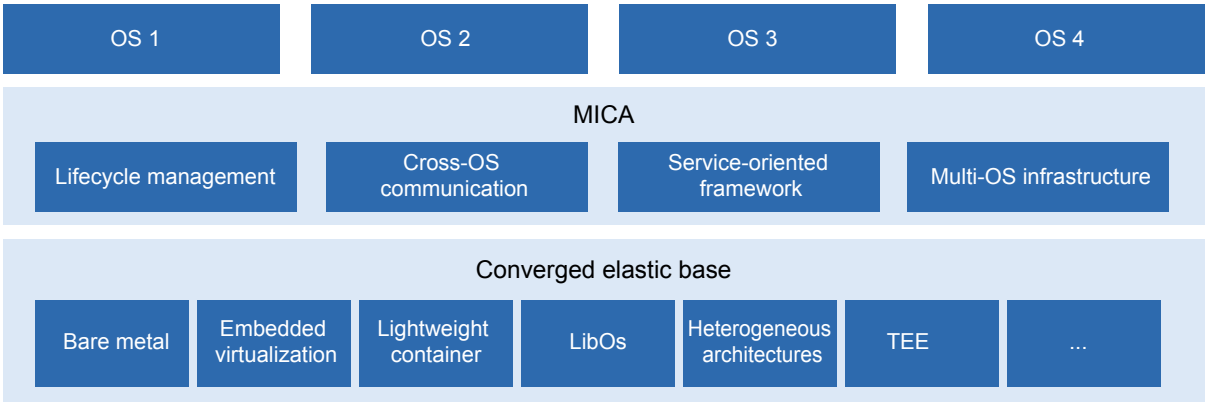
Improvements in hardware have allowed embedded systems to run on complex OSs like Linux. Embedded applications have higher requirements on interconnection, AI, and iterative upgrade than general-purpose workloads, but they typically have constraints in terms of resources, power consumption, real-time performance, reliability, and security. Because the Linux architecture cannot meet such constraints, simplified dedicated OSs such as RTOSs or even bare metal runtime systems are more suitable.

For embedded systems, hybrid deployment of multiple OSs faces the following challenges:

- Deployment of multiple OSs requires collaboration on the same multi-core (homogeneous or heterogeneous) SoC to implement system functions.
- System isolation to prevent impact of failures, such as a crash or fault, to OSs that have high security, reliability, and real-time requirements.
- Improved resource scheduling and utilization to ensure full use of hardware resources.

Project Introduction

The figure shows the overall architecture of MICA.



MICA needs to be in coordination with a converged elastic base, a set of technologies that enable multiple OSs or runtimes to run concurrently on a multi-core SoC. The base includes bare metal servers, embedded virtualization, lightweight containers, library operating system (LibOS), TEE, and heterogeneous architectures. This setup allows MICA to use the advantages of various technologies, such as high performance from the bare metal servers, better isolation and protection from embedded virtualization, and the usability and flexibility of lightweight containers.

MICA supports hybrid deployment of Linux and other OSs or runtimes, while the multi-core capabilities ensure the general-purpose Linux and dedicated RTOS complement each other. In this way, different systems can be developed and deployed independently or as a whole.

MICA consists of 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.

Application Scenarios

The MICA project is positioned for mid-range and high-end embedded systems in manufacturing, energy, and robotics fields. It currently supports OpenAMP and Jailhouse, and is in development to work with ZVM and Rust-Shyper.

Repositories

https://gitee.com/openeuler/mcs

Rust-Shyper

Embedded

Virt SIG

Rust-Shyper is an embedded type-1 hypervisor built with Rust, and is oriented to embedded scenarios such as autonomous vehicles and robotics. It is designed to improve resource utilization without compromising real-time performance, isolation, or security of VM memory. Further, it supports VM migration and live hypervisor update, and can dynamically fix software vulnerabilities without affecting VM running.

Challenges

Embedded systems have developed towards universal and mixed-criticality systems, which vary in terms of reliability, real-time performance, and authentication. Resource isolation and real-time performance of critical tasks are major issues, and while the virtualization solution can solve issues of isolation, current embedded virtualization options are still insufficient. The following features need to be improved:

- Enhanced isolation and security between VMs to prevent malicious attacks
- Improved communication and real-time performance between VMs to avoid delay or jitter
- Superior stability and reliability of the hypervisor to avoid downtime from failures

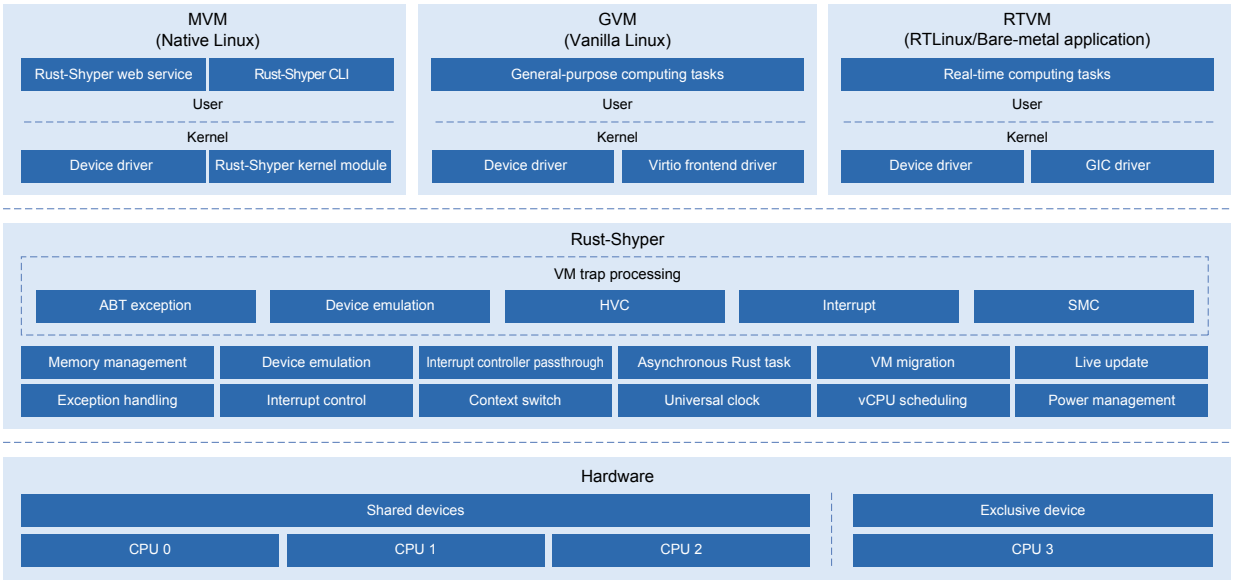
Project Introduction

Rust-Shyper was initially developed by Professor Wang Lei's team from Beihang University, before it was donated to the openEuler open source community in April 2023, where it has been continuously developed by the Virt SIG.

Rust-Shyper is a type-1 hypervisor based on AArch64 and consists of three layers:

- The bottom layer is the hardware layer, corresponding to the ARMv8 EL3 firmware layer.
- The middle layer is the hypervisor layer, corresponding to the ARMv8 EL2 virtualization layer. This layer is also the privilege layer of the Rust-Shyper code.
- The top layer is the VM layer, corresponding to the ARMv8 EL1 and EL0 layers.

Rust-Shyper delivers differentiated virtualization services by using three types of VMs: management VM (MVM), guest VM (GVM), and real-time VM (RTVM).



Rust-Shyper's design concepts and features:

- Memory security: Uses the Rust's type system and security model to ensure memory security of the hypervisor.
- Strong isolation: Uses hardware-assisted virtualization to implement security and fault isolation between VMs.
- Extensive device models: Provides improved resource utilization and wide support for passthrough, mediation, full emulation, and other devices.
- Real-time virtualization: Helps manage resource passthrough and real-time performance status.
- Live update: Implements VM migration and live hypervisor update.

Application Scenarios

The Rust-Shyper project is in development, and aims to meet demands for mid-range and high-end complex embedded systems in manufacturing, energy, robotics, and automotive electronics fields.

Repositories

https://gitee.com/openeuler/rust_shyper

UniProton

Cloud | Embedded

Embedded SIG

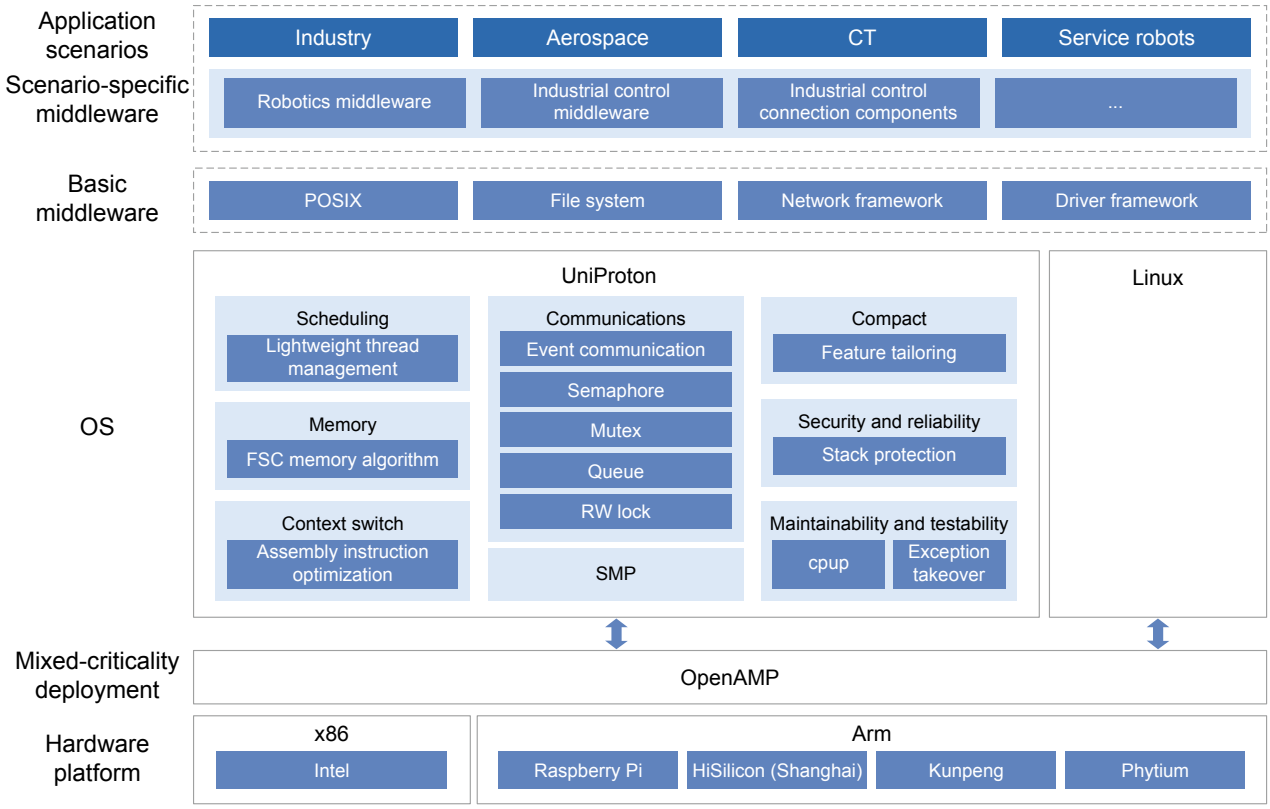
UniProton is a hard RTOS that delivers ultra-low latency in microseconds and mixed-criticality deployment. It can be deployed together with general-purpose OSs such as openEuler Embedded and supports microcontrollers (MCUs) and multi-core CPUs in industrial control systems.

Challenges

Industrial control systems require low deterministic latency from Oss, which is not possible with Linux due to the large size and complex functions. Small embedded systems like UniProton feature a lightweight kernel, optimized performance, and custom functions that combine to achieve ultra-low latency.

Project Introduction

Huawei is an experienced vendor specializing in hard real-time scenarios in the CT field who, in June 2022, released UniProton as an open source software to the openEuler community, where the software has been developed by the Embedded SIG.



UniProton has the following features:

- Low Latency
  - » Deterministic latency: maximum scheduling latency meeting service requirements
  - » Ultimate performance: task scheduling and interrupt latency in microseconds
  - » Lightweight: running in an environment with only tens of KB of memory
- Universality
  - » POSIX-compliant APIs
  - » Compatible with multiple instruction set architectures (ISAs) such as x86, AArch64, and Cortex-M, and chips such as Intel and Raspberry Pi
  - » Multi-core processors delivering efficient low-latency, multi-core computing
- Usability
  - » Customized functions
  - » Maintenance and testing, including cpup and exception takeover
  - » Mixed-criticality deployments using Linux and RTOS capabilities
- Extensive Middleware
  - » Unified and standard driver development framework that improves driver development efficiency
  - » Various network protocol stacks and standard NICs
  - » Industrial middleware that works with mainstream protocols and standards, and bus communication capabilities such as EtherCAT

Application Scenarios

Thanks to its low deterministic latency, UniProton is a hard real-time solution suited to manufacturing, healthcare, energy, electric power, aerospace, and other industries that perform industrial production and robotic control.

Repositories

https://gitee.com/openeuler/UniProton

ZVM

Edge | Embedded

Zephyr SIG

Zephyr-based Virtual Machine (ZVM) is an embedded real-time VM. Developed on the Zephyr RTOS and hardware-assisted virtualization, it supports hybrid deployment and mixed-criticality scheduling of multiple runtimes, including Linux, RTOSs, and bare metal programs.

Challenges

Embedded real-time virtualization allows multiple OSs to run on a single hardware platform at the same time while maintaining deterministic and time-critical performance. It equips typical embedded systems with hardware integration, system isolation, and brilliant flexibility, reliability, security, and scalability. In practice, embedded real-time virtualization has been used to support advanced applications such as smart cars, CNC machine tools, and 5G devices.

There are several challenges when developing embedded real-time virtualization software. Questions include: how to ensure isolation and security between different guest OSs, especially those with different levels of criticality and credibility; how to share or allocate I/O devices among different guest OSs that require device emulation or passthrough; and how to deliver consistent latency and throughput when the RTOS is running as a guest OS.

Future developments in this area will provide mandatory isolation, secure and efficient interrupt handling, flexible I/O device management, and hardware support.

Project Introduction

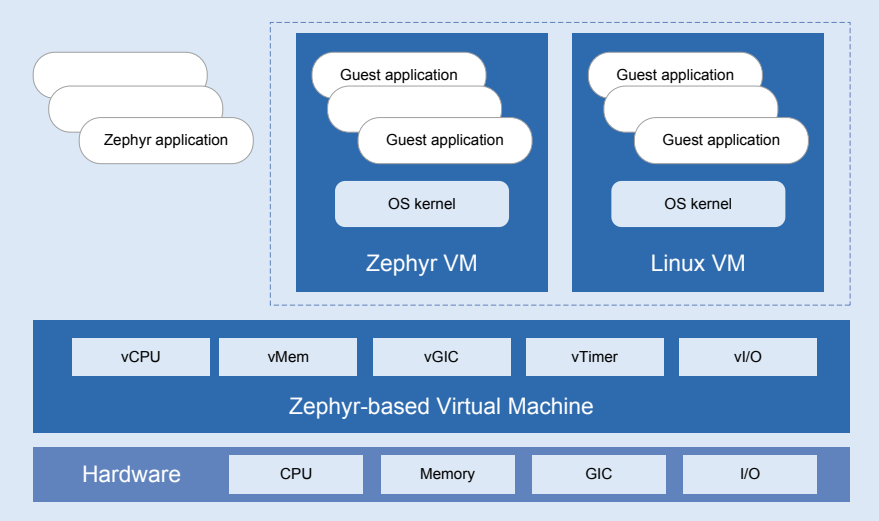
ZVM was developed by Professor Xie Guoqi's team from the Key Laboratory for Embedded and Network Computing of Hunan University, Hunan Province. In April 2023, the software was donated to the openEuler community, where it has been managed and improved upon by the Zephyr SIG.

ZVM uses architecture-level hardware-assisted virtualization and virtualization host extension (VHE). It implements guest OS isolation, device allocation, and interrupt handling, ensuring system security and real-time performance.

ZVM provides benefits in security isolation, device management, and system performance.

- Security isolation: The virtualization technology runs applications of different privileges, isolating guest OSs, especially those with different criticality levels, for security purposes. Further, virtual address spaces (VASs) and virtual devices are allocated to guest OSs to isolate VMs and ensure system security.
- Device management: Management programs that support device emulation and passthrough are used to share or allocate I/O devices between guest OSs. Devices that need to be exclusively occupied by the interrupt controller are allocated in full virtualization mode. Non-exclusive devices such as the Universal Asynchronous Receiver/Transmitter (UART) are allocated in device passthrough mode.
- System performance: AArch64 virtualization extensions reduce the context overhead for processors, hardware-based two-stage address translation reduces performance overhead for memory, and hardware-based interrupt injection reduces the context overhead and interrupt latency.

The following figure shows the overall architecture. ZVM supports two types of guest OSs: Linux and Zephyr RTOS. The Zephyr RTOS has a virtualization module to implement CPU, memory, interrupt, timer, and I/O virtualization.



- CPU virtualization: This level virtualizes an independently-isolated context for each vCPU of the guest OS, in which each vCPU is a thread and scheduled by ZVM on a unified platform. To improve vCPU performance, the AArch64 architecture provides VHE that enables the host OS to migrate to the EL2 privilege level without modifying the OS code. In this process, VHE redirects Arm registers so that the Zephyr RTOS kernel can be migrated to the EL2 level to develop ZVM, reducing system redundancy and improving system performance.
- Memory virtualization: To protect the physical memory, the system needs to isolate the memory space of different guest OSs and monitor the access of guest OSs to the physical memory. In this case, ZVM isolates memory addresses between guest OSs. To implement this function, AArch64 provides a two-stage address translation mechanism, in which the virtual address of the guest OS is translated into a physical address, and then the physical address of the guest OS is translated into that of the host OS. Arm provides independent hardware for stage 2 to improve address translation performance.
- Interrupt virtualization: The Generic Interrupt Controller (GIC) of Arm is used to configure virtual interrupts. It provides a unified platform to route the interrupts of the guest OS to ZVM, which then allocates them to different vCPUs. Virtual interrupts are injected through the virtual CPU interface or list register in the GIC.
- Timer virtualization: This defines a group of virtual timer registers for each CPU. They separately count and throw interrupts after a preset period of time, and then the host OS forwards the interrupts to the guest OS. In addition, during guest OS switchover, the virtual timer calculates the actual running time of the guest OS, compensates for the time when the guest OS exits, and provides the timer service for the guest OS.
- Device virtualization: ZVM uses the Arm memory-mapped I/O (MMIO) method to map device addresses to virtual memory addresses to enable guest OSs to access device addresses. Specifically, ZVM builds a virtual MMIO device and allocates the device to a specified guest OS during the creation of the guest OS, to implement I/O virtualization. For certain non-exclusive devices, ZVM allows access through device passthrough.

Application Scenarios

The ZVM project is being developed for mid-range and high-end complex embedded systems in manufacturing, energy, robotics, and automotive electronics fields.

Repositories

https://gitee.com/openeuler/zvm

DSoftBus

Edge | Embedded

Distributed Middleware SIG

To develop a digital infrastructure OS and improve collaboration between devices and edges, openEuler marks an industry feat by applying DSoftBus technology to the server, edge, and embedded. DSoftBus provides a unified platform to enable collaboration and communication for distributed devices, achieving fast device discovery and efficient data transmission.

Challenges

Collaboration between edge devices includes the discovery, connection, networking, and transmission phases, though interconnection between edge devices has the following difficulties:

- Different types of devices: There is no unified solution to cover various hardware capabilities and supported connection modes, such as Wi-Fi, Bluetooth, and near field communication (NFC).
- Unstable and slow networking: It is difficult to automatically create and allocate network management roles between edge devices and maintain network stability after devices exit, power off, or experience a fault.
- Poor transmission: Data transmission performance cannot be guaranteed between edges and devices, especially for devices that have certain restrictions on power consumption.
- Difficult API adaptation: There are no unified APIs that mask differences in underlying hardware and networking, to allow one service process can be reused by different edge devices.

Project Introduction

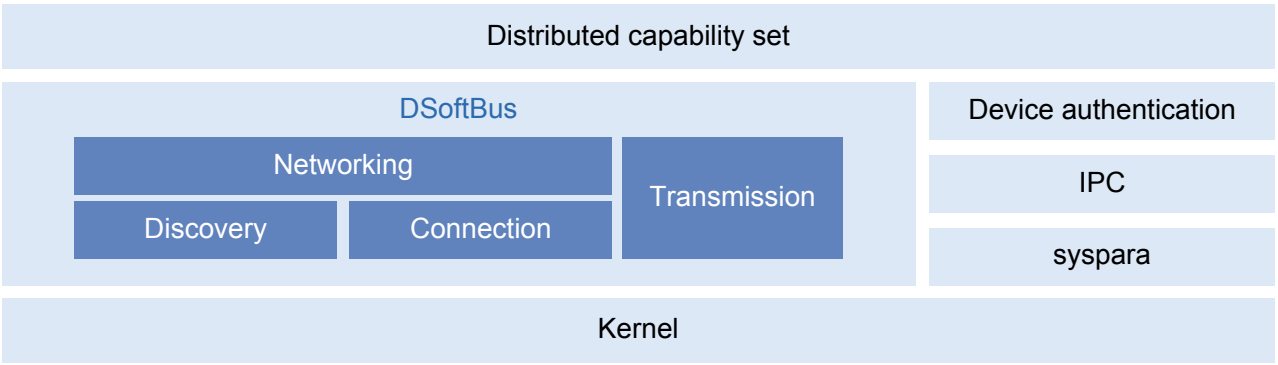
Key Features

- Device discovery and connection based on Wi-Fi, wired network, and Bluetooth
- Unified networking and topology of device and transmission information
- Transmission channel for data bytes, streams, and files

Architecture

DSoftBus consists of four basic modules: discovery, networking, connection, and transmission.

Relationship between DSoftBus and external modules

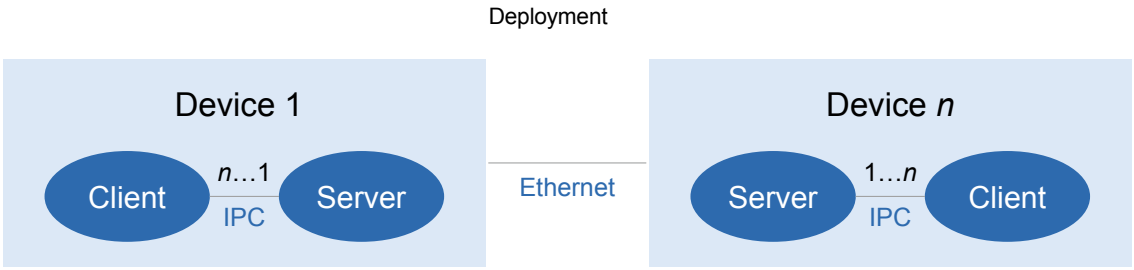


For southbound devices, DSoftBus supports Wi-Fi, wired network, and Bluetooth connection, whereas for northbound distributed applications, it provides unified APIs that mask the underlying communication mechanism.

DSoftBus depends on peripheral modules such as device authentication, inter-process communication (IPC), log, and system parameters (SNs). In embedded environments, such modules can be tailored or replaced to provide basic functions of DSoftBus, meet actual service scenarios, and extend DSoftBus capabilities.

Deployment

- DSoftBus can be deployed on multiple devices in the LAN, but the devices communicate with each other through the Ethernet.
- A device consists of a server and a client that communicate through the IPC module.
- Multiple clients can concurrently access a server on a device.



DSoftBus provides services for external systems through an independent process by executing the main program of the server.

Application Scenarios

DSoftBus is best suited for device discovery and interconnection between openEuler edge servers and common and OpenHarmony embedded devices.

It provides unified APIs and protocol standards to enable self-connection, self-networking, and plug-and-play of multi-vendor, multi-type devices, implementing the peripheral access needed in industrial production lines and campus management.

Repositories

https://gitee.com/openeuler/dsoftbus_standard

openEuler Edge

Cloud | Edge

Edge SIG

openEuler Edge positioned as an edge computing edition for edge-cloud collaboration. It uses KubeEdge, an open source project incubated by the Cloud Native Computing Foundation (CNCF) that delivers basic capabilities such as unified management and provisioning for edge and cloud applications, to streamline AI deployments and service collaboration for edge-cloud service discovery and traffic forwarding, and offer enhanced data collaboration to improve southbound capabilities.

Challenges

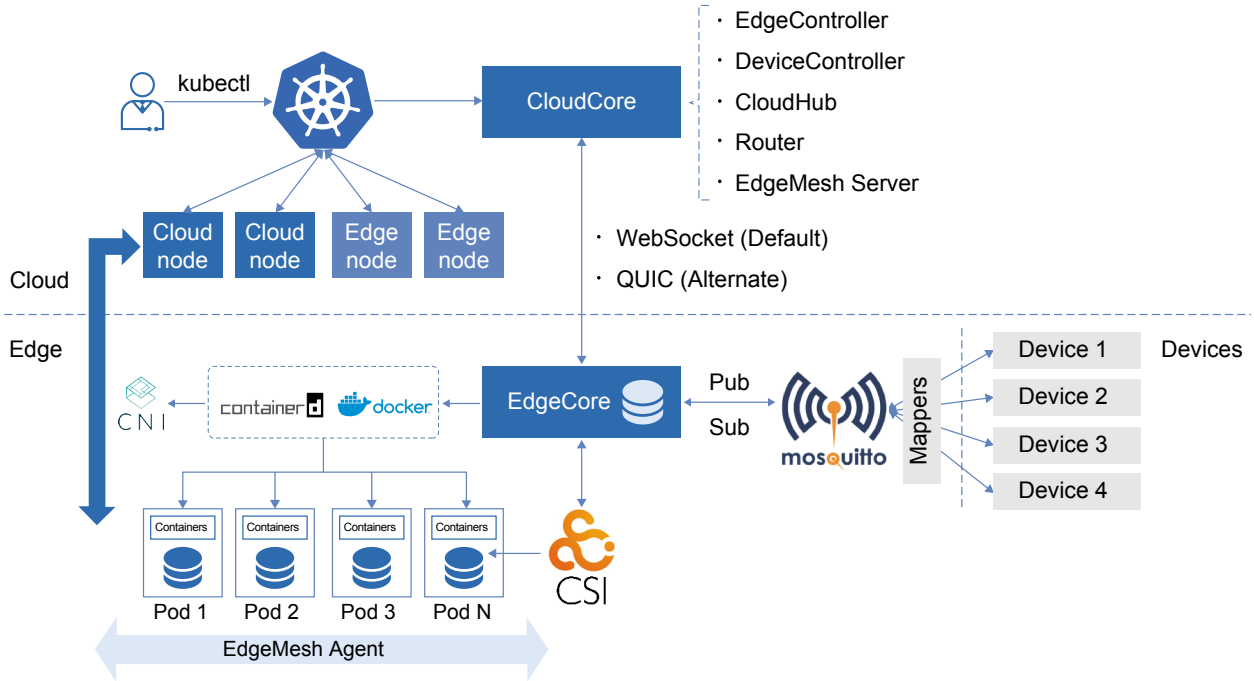
Edge computing is one of the top 10 major technology trends, and as such, is dominating current and future business models. Consider how emerging fields like smart city, autonomous driving, and industrial Internet applications are generating huge data volumes that cannot be processed by centralized cloud computing. For example, IDC forecasts that in 2025, 48.6 ZB of data will be generated in China alone. There is a demand for high-speed, low-latency, and cost-efficient edge computing solutions.

Project Introduction

openEuler Edge integrates KubeEdge into a unified management platform, on which users can easily provision edge and cloud applications. It delivers collaboration across edge and cloud to help streamline AI deployments, implement service discovery and traffic forwarding, and improve southbound capabilities.

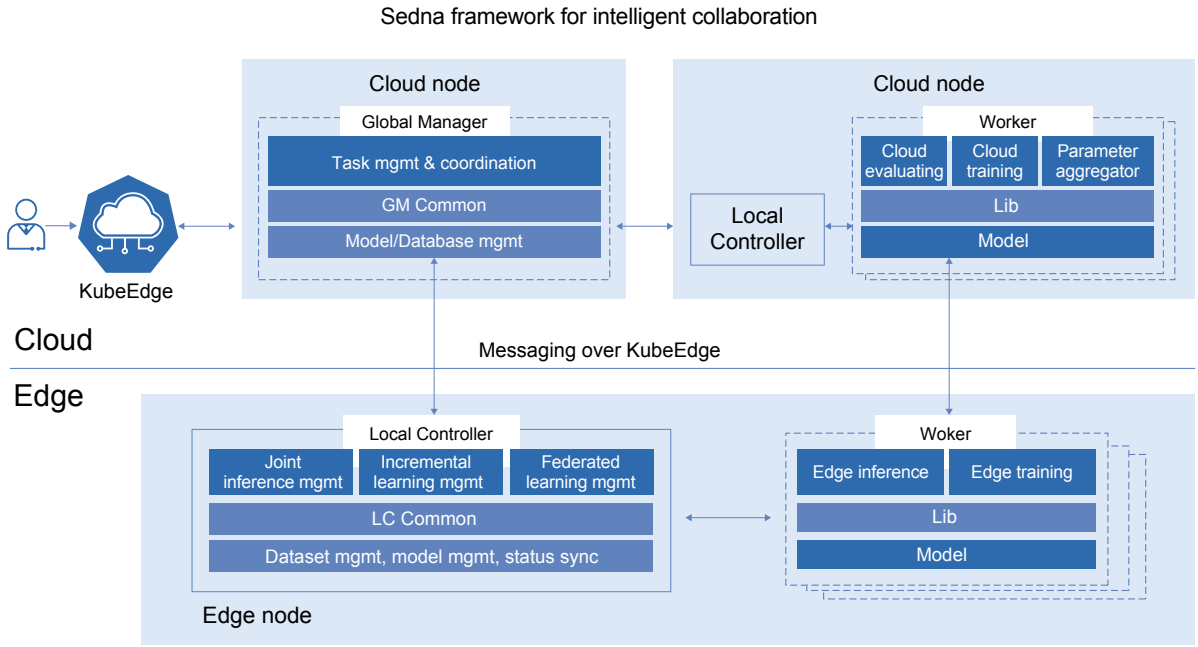
Features

KubeEdge framework for edge-cloud collaboration



KubeEdge is equipped with the following features:

- Edge-cloud management collaboration allows users to provision applications and manage southbound peripherals across edge and cloud.
- Improve service discovery and routing by deploying EdgeMesh Agent and EdgeMesh Server at the edge and on the cloud, respectively.
- Optimized southbound edge services using Device Mapper, which provides the peripheral profile and parsing mechanisms to help manage and control southbound peripherals and service streams. The southbound edge services are compatible with the EdgeX Foundry open source ecosystem.
- Edge data services, including on-demand persistence of messages, data, and media streams, and data analysis and export operations.



The Sedna framework provides the following features:

- Sedna GM and LC: General Manager (GM) is used for global task coordination, and Local Controller (LC) provides local dataset and model management, status synchronization, and edge autonomy. These capabilities enable edge-cloud collaborative inference and federated and incremental learning, improving training and deployment of edge-cloud AI.
- Sedna Lib: enables developers to efficiently develop edge-cloud AI collaboration features.

Application Scenarios

openEuler Edge is a popular option for edge-cloud collaboration in fields like public safety, energy, transportation, manufacturing, finance, healthcare, campus, and unattended operation.

Repositories

https://mirror.sjtu.edu.cn/openeuler/openEuler-23.03/edge_img/x86_64/openEuler-23.03-edge-x86_64-dvd.iso

https://github.com/kubeedge/kubeedge

04

Basic Capability Innovations

A-Tune

Server

A-Tune SIG

A-Tune is an AI-powered OS performance tuning engine. It uses AI technologies to enable the OS to learn services statuses, simplify IT system tuning, and deliver excellent performance.

Challenges

The development of hardware and software applications over the past few decades has coincided with a larger, more comprehensive Linux kernel. In openEuler, the sysctl command is used to configure kernel parameters and has over 1,000 parameters (**sysctl -a | wc -l**). A typical IT system covers the CPU, accelerator, NIC, compiler, OS, middleware framework, and upper-layer applications, and uses 7,000 parameters, most of which are default settings, which cannot tap into the full system performance. Parameter tuning faces the following difficulties:

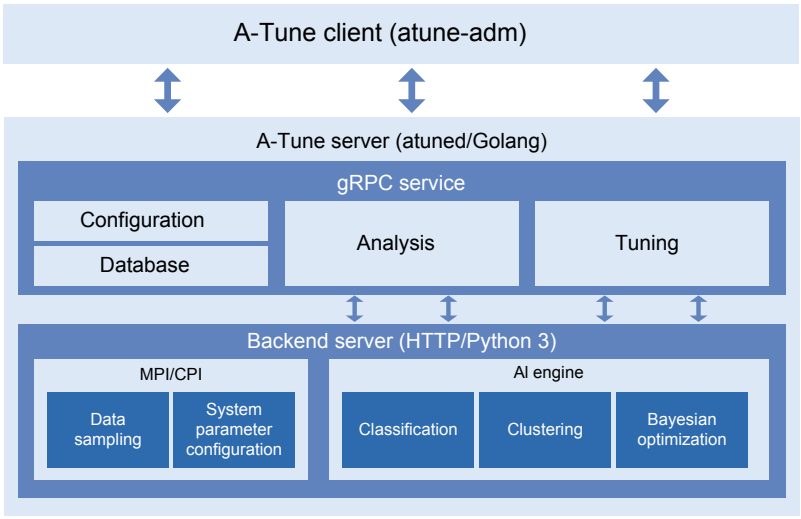
- A large number of parameters that depend on each other.
- Various types of upper-layer application systems require varying tuning of parameters.
- Complex and diversified loads require the parameters to vary accordingly.

Project Introduction

Features

A-Tune is in a client/server (C/S) architecture. The atune-adm on the client is a command-line tool that communicates with the atuned process on the server through the gRPC protocol. The atuned process contains a frontend gRPC service layer (implemented by Golang) and a backend service layer, the former of which manages optimization configurations and data and provides tuning services for external systems, including intelligent decision-making (Analysis) and automated tuning (Tuning). By contrast, the backend service layer is an HTTP service layer executed by Python that consists of the Model Plugin Interface (MPI)/Configurator Plugin Interface (CPI) and AI engine. The MPI/CPI discovers system configurations, and the AI engine provides machine learning capabilities for the upper layer, including classification and clustering for model identification and Bayesian optimization for parameter search.

A-Tune software architecture



Efficient Concurrency and Ultimate Performance

A-Tune delivers intelligent decision-making and automated tuning.

Intelligent decision-making is to sample system data and identify the corresponding loads using the clustering and classification algorithms of the AI engine, obtain the types of service loads, extract optimization configurations from the database, and select the optimal parameter values to fit each service load.

- Automatically selects key features and removes redundant ones for precise user profiling.
- Two-layer classification model accurately identifies service loads using the classification algorithm.
- Load awareness proactively identifies application load changes and implements adaptive tuning.

Automated tuning uses configuration parameters and performance metrics of the system or application as reference, to then repeatedly perform iteration using the parameter search algorithm of the AI engine. This can obtain parameter configurations that deliver optimal performance.

- Automatically tunes core parameters to reduce search space and improve training efficiency.
- Allows users to select the optimal tuning algorithm based on the application scenario, parameter type, and performance requirements.
- Adds load features and optimal parameters to the knowledge base to improve future tuning operations.

Application Scenarios

A-Tune is widely used in Linux environments that handle big data, databases, middleware, and HPC workloads. Commonly used in industries such as finance and telecom, the software improves performance of applications such as MySQL, Redis, and BES middleware by 12% to 140%, respectively.

Repositories

https://gitee.com/openeuler/A-Tune

BiSheng JDK

Server | Cloud | Edge

Compiler SIG

BiSheng JDK is an open source edition of the Huawei JDK developed on OpenJDK. It is a high-performance OpenJDK distribution that can run in the production environment thanks to its extensive development by the dedicated team, who solved many issues caused by the native OpenJDK defects in actual deployments. On May 26, 2022, BiSheng JDK was released in Eclipse Adoptium Marketplace, offering a stable, reliable, high-performance, and easy-to-debug JDK, and can even be supercharged when run on the Kunpeng AArch64 architecture.

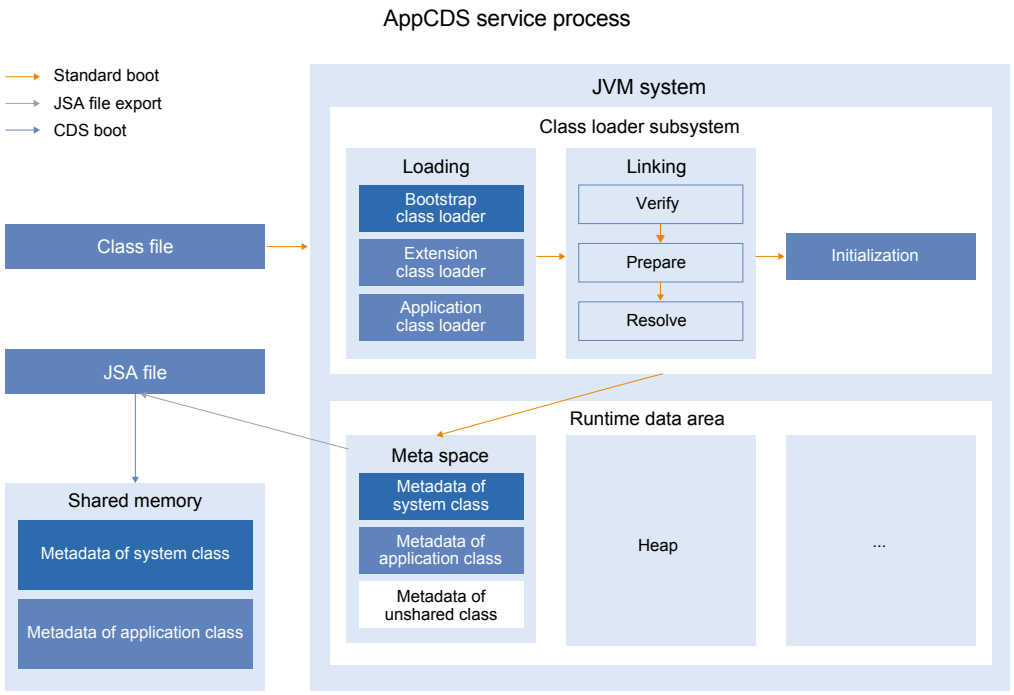
Challenges

JDK is the core component for Java environments that has been plagued by issues of performance and stability. Despite difficulties in introducing new major features to OpenJDK 8 during the maintenance period, optimization features have been added to resolve Java startup, garbage collection (GC), latency, encryption and decryption, and communication.

Project Introduction

Feature 1: AppCDS

In the initial phase of running a Java program, class loading is time-consuming and needs to be performed each time the program is running. Class data sharing (CDS) saves the loaded class data to a file, so that the next time a user runs a Java program, the loaded class data is directly restored from the file to the memory. Based on CDS provided by OpenJDK, BiSheng JDK 8 extends application class data sharing, namely, AppCDS to support application classes, delivering performance 7% higher on average than in the Hive SQL scenario.



Feature 2: Promptly Return Unused Committed Memory from G1

The Garbage-First garbage collector (G1 GC) may not return committed Java heap memory to the OS in a timely manner, and may perform it only after a full GC operation. Since G1 tries hard to completely avoid full GCs, it will not return Java heap memory in many cases unless forced to do so externally.

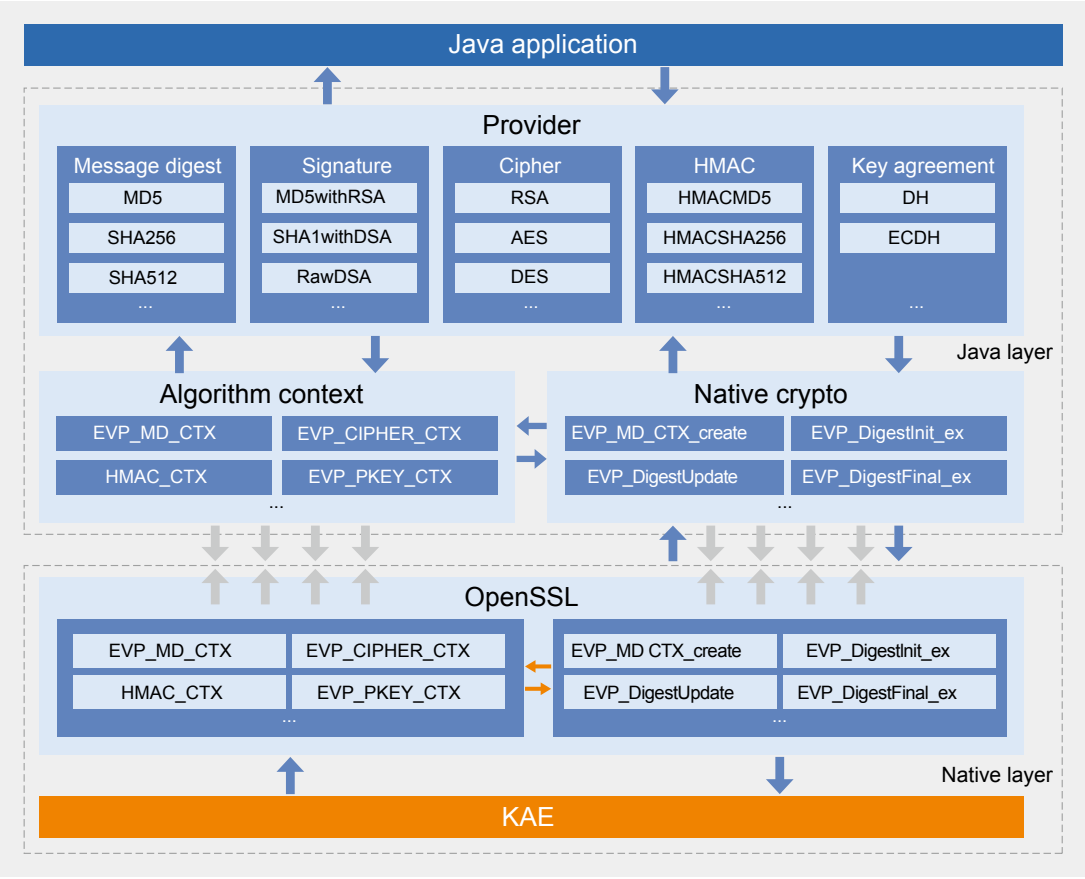
This behavior is particularly disadvantageous in container environments where resources are paid by use. Even when the JVM only uses a fraction of its assigned memory resources due to inactivity, G1 will retain all of the Java heap, causing users to pay full price the resources or cloud providers unable to fully utilize their hardware. One solution is to ensure the JVM can detect phases of Java heap under-utilization, and automatically reduce its heap usage. Assume 49 microservices are running and this feature is enabled. G1 can return unused committed memory when the CPU is idle, reducing the physical memory committed by G1 by 40% compared with that of the default settings.

Feature 3: KAE Provider

The Kunpeng Accelerator Engine (KAE) is an encryption and decryption module that supports the RSA, SM3, SM4, DH, MD5, and AES algorithms, and provides high-performance symmetric and asymmetric encryption/decryption algorithms. The KAE is compatible with OpenSSL 1.1.1a and later versions, and can work in synchronous or asynchronous mode.

BiSheng JDK adopts the Provider mechanism to support KAE encryption and decryption of Kunpeng servers, helping to improve the security and related services on Kunpeng AArch64 servers. In HTTPS scenarios, the performance is doubled.

KAE Provider architecture



The KAE Provider supports the following algorithms.

Algorithm	Description
Digest algorithms	MD5, SHA256, SHA384, SM3
AES	ECB, CBC, CTR, GCM
SM4	ECB, CBC, CTR, OFB
HMAC	HMACMD5, HMACSHA1, HMACSHA224, HMACSHA256, HMACSHA384, HMACSHA512
RSA	512-bit, 1024-bit, 2048-bit, 3072-bit, and 4096-bit keys
DH	DHKeyPairGenerator and DHKeyAgreement; 512-bit, 1024-bit, 2048-bit, 3072-bit, and 4096-bit keys
ECDH	ECKeyPairGenerator and ECDHKeyAgreement; secp224r1, prime256v1, secp384r1, and secp521r1
RSA signatures	RSASignature and RSAPSSSignature

Application Scenarios

BiSheng JDK is a common Java software developed and distributed on OpenJDK. It is widely used in Linux environments to process big data, middleware, and encryption and decryption tasks, in industries like finance, middleware, carrier, and Internet. On average, it improves the Spark performance by 10%, while for encryption and decryption improves by over 100%.

Repositories

BiSheng JDK 8, BiSheng JDK 11, and BiSheng JDK 17 are open sourced, with version updates and new features downloaded every three months. Java developers can obtain the latest information and communicate with each other in the BiSheng JDK open source community.

Software	Delivery Type	URL
BiSheng JDK 8	Open source code repository	https://gitee.com/openeuler/bishengjdk-8
BiSheng JDK 11	Open source code repository	https://gitee.com/openeuler/bishengjdk-11
BiSheng JDK 17	Open source code repository	https://gitee.com/openeuler/bishengjdk-17

etMem

Server | Cloud

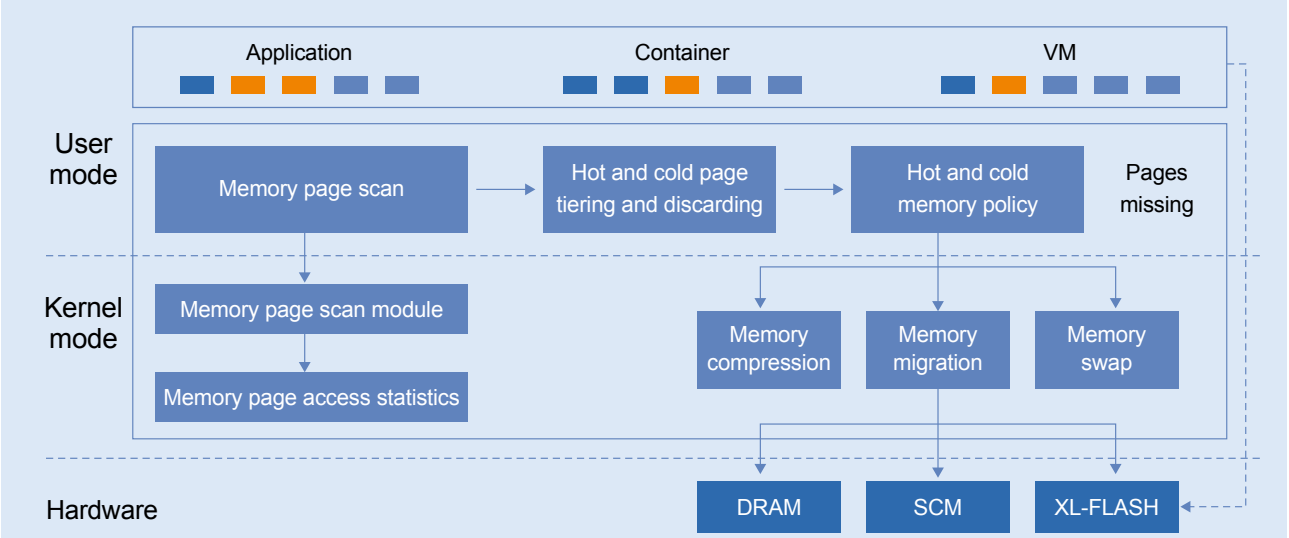
Storage SIG

etMem uses DRAM, low-speed memory (such as storage class memory), or hard drives to form tiered memory for application processes. Its automatic memory scheduling redirects hot data to the high-speed DRAM and cold data to the low-speed media, expanding available memory and improving service performance.

Challenges

Despite the relative cost effectiveness of X per CPU, the memory manufacturing process has plateaued, making it difficult to innovate in the short term. As a result, the memory cost accounts for an increasing proportion of the total server cost. Further, database, VM, big data, AI, and deep learning workloads are in demand of diversified computing power and large memory, making it essential to slash memory costs and expand memory capacity.

Project Introduction



etMem is divided into the kernel-mode and user-mode modules to provide the following features:

- Process control: etMem configuration file can expand the memory in a more flexible and refined way when compared with the native LRU-based pageout kswap mechanism.
- Cold and hot tiering: In user mode, a memory access scan can be performed for a designated process. Cold and hot tiering policies are used to classify memory access results into hot and cold memory.
- Discarding policy: The cold memory is discarded when it meets the conditions specified in the etMem and system environment policies. This process is based on the native kernel, a secure and reliable mechanism that does not affect user experience.

Kernel-Mode Module

A memory page scan module and a memory discarding (compression, migration, and swap) module are available in the kernel mode.

- Memory page scan module identifies page table features. It periodically collects memory page access statistics, and then reports the statistics to the user mode.
- Memory discarding module receives the unwanted memory page addresses from the user mode, and uses open source native capabilities to compress, migrate, or swap out those pages.

User-Mode Module

- Memory scanning triggers memory page scanning and collects results.
- Hot and cold tiering classifies memory access results into hot and cold memory according to the policy.
- Hot and cold memory policy performs actions on the hot and cold memory based on the configuration.

Application Scenarios

etMem is designed for service software that requires a large amount of memory but infrequently accesses the memory, such as MySQL, Redis, and Nginx. In MySQL TPC-C scenarios, the performance is improved by 40% under the same cost conditions.

It supports tiered memory expansion for node service processes and does not involve cross-node remote operations. In a user-mode storage framework, the userswap function can enable user-mode storage to be a swap device.

Repositories

https://gitee.com/openeuler/etmem/

EulerFS

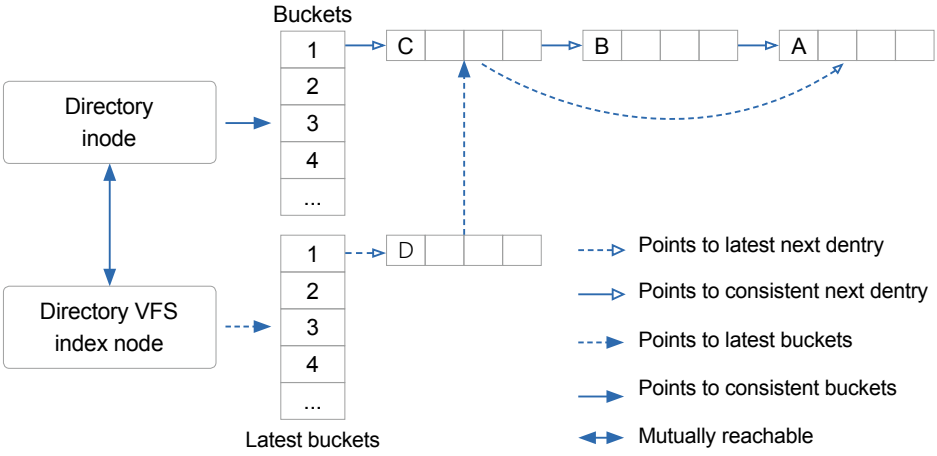
Server | Cloud

Kernel SIG

Non-volatile dual in-line memory modules (NVDIMMs), for example Intel Optane, are a high-speed storage medium that offers byte-level access. The kernel file system EXT4 can work with the direct access (DAX) feature to improve the data read and write performance of NVDIMMs. However, NVDIMMs are hampered with heavy metadata management overhead and write amplification issues due to their journal synchronization mechanism.

The EulerFS file system has the innovative soft metadata update technology, which employs pointer-based dual directory views to reduce the metadata synchronization overhead. EulerFS improves the performance of system calls including create, unlink, mkdir, and rmdir. Compared with EXT4 DAX, EulerFS reduces the metadata operation latency by 25% to 75% and increases the bandwidth by 0.2 to 4 times.

Features



- Hash table directory: Hash tables are used to manage directory entries, accelerating linear search and reducing pseudo-sharing.
- Unified allocator: Data structures are arranged by a unified allocator, which breaks the boundaries between different data structures and facilitates memory management.
- Soft update: This lightweight technology ensures file system consistency. It simplifies the implementation of file system consistency.
- Pointer-based dual directory views: This mechanism reduces metadata synchronization overhead and improves the read and write performance of file systems.
- Dependency tracing: Operations such as directory entry creation and deletion are not made persistent immediately. After the operations are complete, dependencies are traced only in the index node, while subsequent persistence is performed in asynchronous mode to improve the performance.

Application Scenarios

EulerFS works on all NVDIMM-based media and is an alternative to file systems such as EXT4 and XFS. It enables high-performance data storage in standalone and cloud-native distributed applications.

Repositories

https://gitee.com/openeuler/eulerfs

Gazelle

Server | Cloud | Edge | Embedded

High-Performance Network SIG

Gazelle is a lightweight user-mode protocol stack developed based on Data Plane Development Kit (DPDK) and lightweight IP (lwIP). It boosts application performance and is also versatile due to its broad compatibility and usability.

Challenges

As a key path that modern applications must pass, protocol stacks have always been a hot topic in performance research. As software and hardware technologies have evolved over recent years, protocol stacks face the following new issues:

Hardware

- CPUs lag behind NICs in the increase of computing power. The single-core CPU architecture cannot fully utilize sufficient NIC bandwidth.
- In a many-core architecture, the protocol stack design must avoid the NUMA swap out problem.

Software

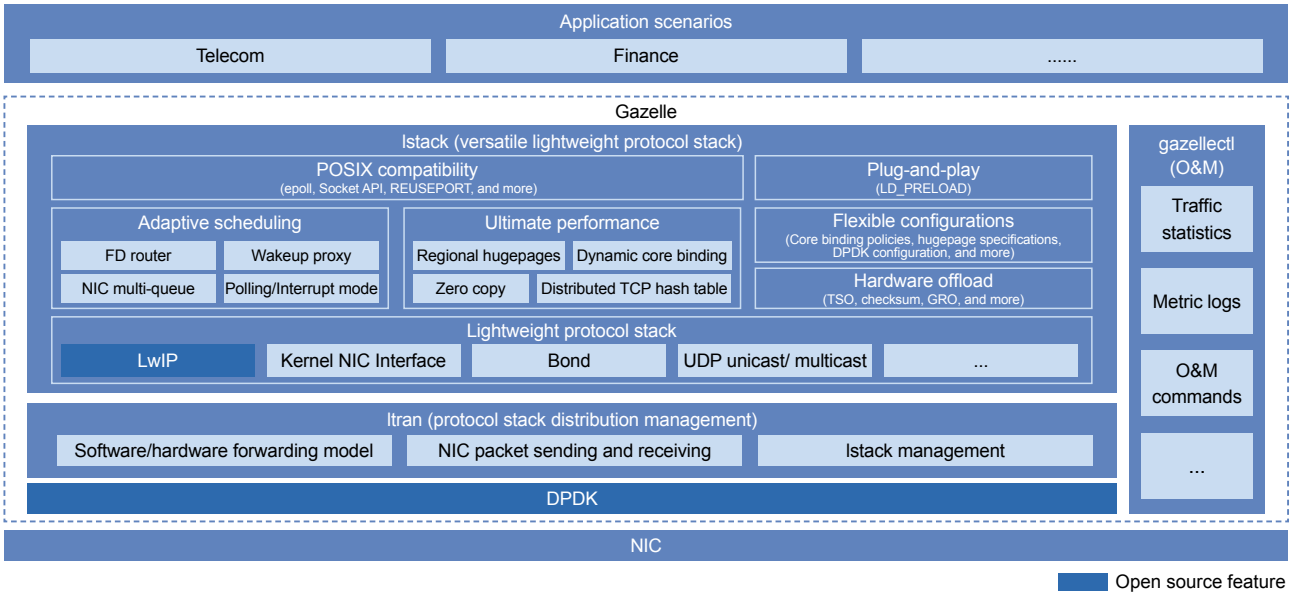
- Modern large-scale software generally relies on a multi-thread architecture to make full use of hardware resources such as CPUs and NICs. The software performance is expected to increase linearly as the number of threads increases.
- Diverse application network models call for a common protocol stack.
- The kernel protocol stack features high versatility and layered decoupling, but its performance is far from what is desired. In comparison, the user-mode protocol stack is oriented to specific scenarios, trying to deliver maximum performance while failing to fulfill versatility needs.

Design

In database scenarios, there are various network models and high requirements on network performance. Implementing protocol stack software that balances performance and versatility is challenging.

Project Introduction

The Gazelle software architecture consists of ltran, lstack, and gazellectl. ltran distributes traffic to protocol stacks. lstack provides lightweight protocol stack capabilities. gazellectl functions as an O&M tool to collect traffic statistics and generate diagnosis logs.



Efficient Concurrency and Ultimate Performance

Gazelle has the following features:

High performance

- Ultra-lightweight implementation: The high-performance protocol stack works based on DPDK and LwIP.
- Ultimate performance: The highly linearizable concurrent protocol stack leverages technologies such as regional hugepage splitting, dynamic core binding, and full-path zero-copy.
- Hardware acceleration: Hardware offload approaches, including TCP segmentation offload (TSO), checksum offload, and generic receive offload (GRO), streamline the vertical acceleration of software and hardware.

Versatility

- POSIX compatibility: Full compatibility with POSIX APIs eliminates the need to modify applications when running them on Gazelle.
- General-purpose network model: The adaptive network model scheduling, powered by socket routers, proxying wakeup, and other mechanisms, fits into any network application scenario.

User-friendliness

Plug-and-play: With the LD_PRELOAD environment variable, protocol stack acceleration is felt immediately when Gazelle is installed.

Easy O&M

O&M tool: Traffic statistics, logs, and command lines are available to streamline O&M.

Application Scenarios

Gazelle is best suited for database acceleration. It has been used in telecom and finance industries to improve MySQL transaction processing test performance by more than 20% and Redis throughput by more than 50%.

Repositories

https://gitee.com/openeuler/gazelle

GCC for openEuler

Server | Cloud | Edge | Embedded

Compiler SIG

The GCC for openEuler compiler is developed based on the open source GNU Compiler Collection (GCC). The open source GCC is the de facto standard of cross-platform compilers, and it complies with the GPLv3 license, becoming the most widely used C/C++ compiler on Linux. GCC for openEuler inherits capabilities of the open source GCC. It also has optimizations on C, C++, and Fortran languages and delivers enhanced features such as automatic feedback-directed optimization (FDO), software and hardware collaboration, memory optimization, and automatic vectorization. GCC for openEuler is compatible with a wide range of hardware platforms such as Kunpeng, Phytium, and Loongson, fully unleashing the computing power of these hardware platforms.

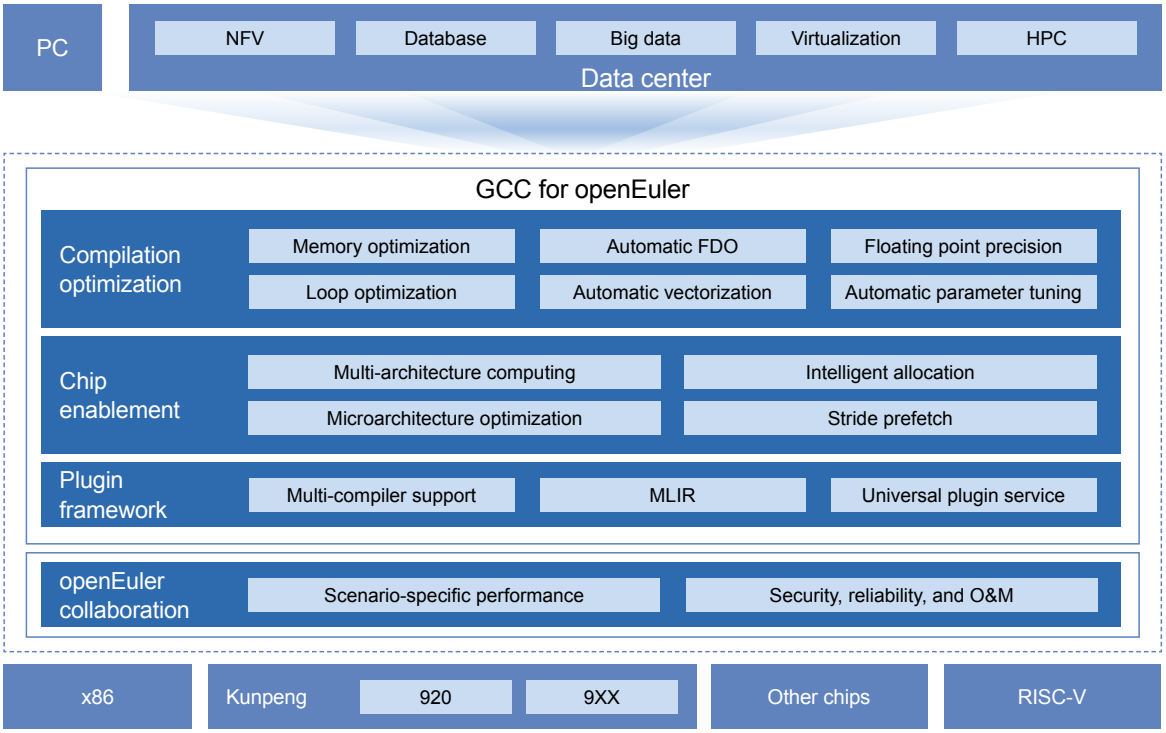
Challenges

GCC is fundamental to the OS as it is the default compiler of the Linux kernel and the de facto standard of cross-platform compilers. Any change made to GCC may heavily impact upper-layer applications. Therefore, GCC developers must familiarize themselves with the knowledge needed, such as compilation principles while improving feature security and robustness. GCC for openEuler aims to boost the performance of upper-layer applications by providing more advanced features than the open source GCC. In the Compiler SIG biweekly meetings the GCC is a fixed topic. All are welcome to attend.

Project Introduction

GCC for openEuler is compatible with mainstream hardware platforms including Kunpeng and x86, for use in openEuler performance, security, reliability, and O&M projects, for example, working in the compiler plugin framework to provide universal plugins. GCC for openEuler supports multi-architecture computing and microarchitecture optimization to implement intelligent memory allocation, memory optimization, and automatic vectorization. Besides that, GCC for openEuler incorporates industry-leading FDO technologies to implement automatic FDO, improving application performance for databases. GCC for openEuler has made major breakthroughs in the following aspects:

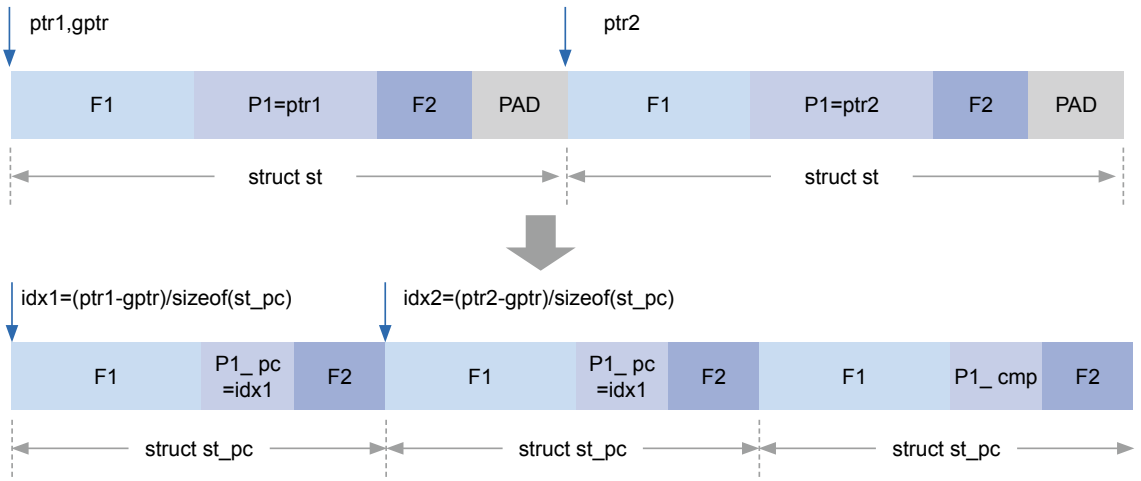
- Basic performance: Improves computing performance in general scenarios and also applies to multi-architecture computing.
- FDO: Integrates industry-leading FDO technologies to implement multi-modal FDO throughout the process, improving key applications such as databases in cloud-native applications.
- Chip enablement: Supports multi-architecture computing instruction sets and leverages computing advantages based on hardware systems such as memory to improve scenario-specific performance such as HPC.
- Plugin framework: Offers one set of plugins that is compatible with different compilation frameworks, streamlining the GCC and LLVM ecosystems.



Feature 1: Pointer compression (basic performance)

When structure field members contain structure pointers, an 8-byte pointer may cause alignment gaps between structure members. Not only that, when the pointer is used together with a basic data type that is less than 8 bytes, memory padding may occur, wasting memory space. As a result, the page table is refreshed frequently, memory access is delayed, and program performance is compromised.

Structure pointer compression works for this application scenario. This feature compresses the structure pointers in structure field members from 64 bits to 8-bit, 16-bit, or 32-bit integers. This reduces the memory occupied by the structure, reduces the bandwidth pressure when data is read from or written to the memory, and streamlines data access.



Feature 2: End-to-end FDO

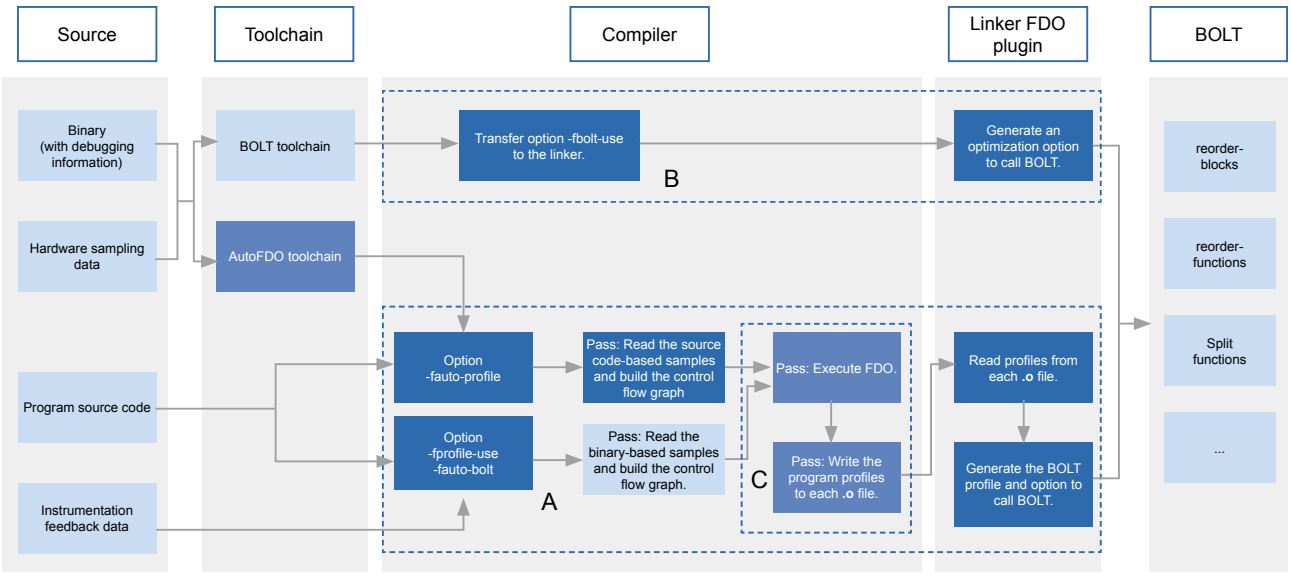
Major FDO technologies include profile-guided optimization (PGO) and AutoFDO, which work during compilation, along with BOLT, which works during binary generation.

PGO is a compiler optimization technology which collects runtime profiles to improve optimization decision-making. The compiler then makes use of runtime profiles and leverages various compilation optimization technologies to make more accurate optimization decisions and generate target programs.

AutoFDO samples program running information to indirectly obtain the program execution status. It uses perf to collect profiles while minimizing impact on program performance. AutoFDO decouples program source code from profile data and is not sensitive to program code changes. The profiles collected in the development and test phases can be used to optimize the target program, which can be reused multiple times.

BOLT presets relocation information when the compiler generates binary files, and performs binary-level optimization operations, such as reordering basic blocks, reordering functions, and splitting cold and hot areas, to implement global binary optimization.

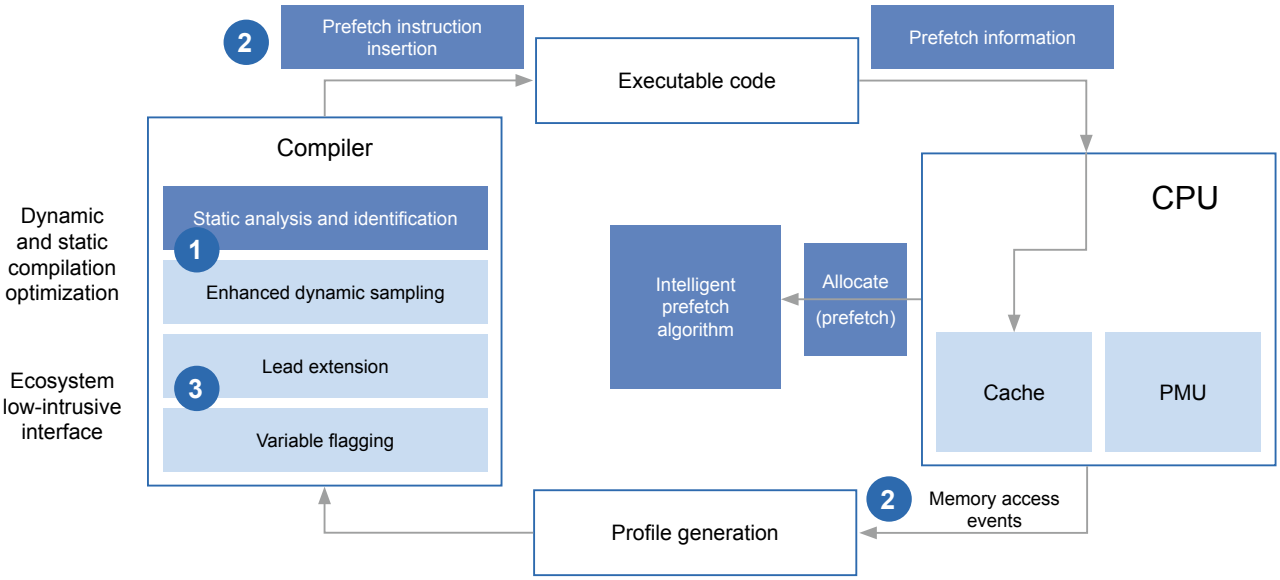
GCC for openEuler enables FDO throughout the process by integrating PGO, AutoFDO, and BOLT. It also leverages minimum cost flow algorithm correction and discriminator support to enhance optimization.



Feature 3: Intelligent allocation (chip enablement)

GCC for openEuler supports intelligent allocation for static optimization analysis of HPC applications. It analyzes memory access data reuse and inserts prefetch instructions to improve the performance of the openFOAM, SPMV, and WRF kernel functions by 30% on average.

- Dynamic and static compilation optimization: Static analysis and data reuse scoring models modified based on dynamic execution feedback are added to GCC for openEuler to analyze and identify high-concurrency hotspot data.
- Intelligent memory allocation: GCC for openEuler generates and inserts prefetch instructions to explicitly notify hardware of prefetch information. Together with the cache replacement policy, GCC for openEuler increases the cache utilization rate and hit ratio.
- Ecosystem low-intrusive programming interface: GCC for openEuler will provide CUDA-like variable attributes and OpenMP lead extension. It will automatically generate code to streamline development and improve ecosystem compatibility.



Application Scenarios

GCC for openEuler is developed based on the open source GCC. It is widely used in Linux environments such as openEuler and is perfect for databases, virtualization, and HPC. On the Arm platform, GCC for openEuler delivers 1.2 times higher basic performance of SPEC CPU 2017 than the open source GCC, boosting the performance of the MySQL database by more than 15%.

Repositories

GCC for openEuler is upgraded together with official openEuler releases. GCC developers can find the latest updates in the openEuler community.

Software Product	Delivery Type	Link
GCC for openEuler	Code repository	https://gitee.com/openeuler/gcc
	Package repository	https://gitee.com/src-openEuler/gcc

HSAK

Server | Cloud

Storage SIG

The Hybrid Storage Acceleration Kit (HSAK) improves the I/O performance of NVMe devices. The software library implements an I/O software stack of high-performance NVMe devices. It is advanced for its user mode, asynchronization, lock-free, and polling features. Compared with the NVMe device I/O software stack in the native Linux kernel, the HSAK greatly reduces the latency of NVMe commands and improves the I/O processing capability (IOPS) of a single CPU.

Challenges

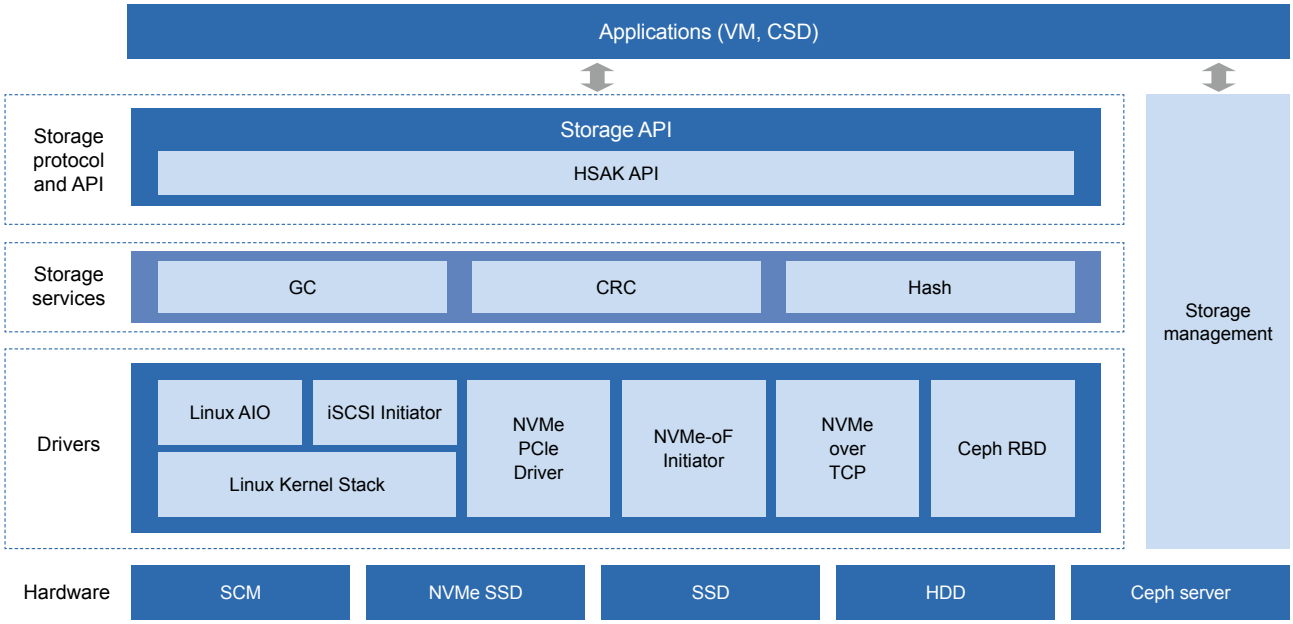
Evolving storage media such as NVMe solid state drives (SSDs) and storage class memory (SCM) have resulted in decreasing access latencies caused by the media layer in the traditional I/O software stack. On the contrary, the software latency gradually becomes a performance bottleneck, so when high-performance storage media are used, the overhead of the traditional kernel I/O software stack accounts for more than 60% of the total I/O overhead. The reasons are as follows:

- When a user-mode service process delivers a request to a drive, the memory data needs to be copied multiple times.
- The drive requires two interrupts to respond to the I/O request, causing overheads such as process scheduling and context switching.

There are many solutions to the previous problems, but most of them have the following drawbacks:

- The software is updated frequently and the external interfaces are inconsistent.
- The I/O data plane provides simple functions and cannot leverage hardware capabilities provided by NVMe drives.
- The capabilities of the management plane are insufficient, and device management or I/O monitoring methods are absent.

Project Introduction



Efficient Concurrency and Ultimate Performance

The HSAK consists of the three-layer data plane and the storage management module.

- Storage protocol and API layer: provides stable and unified northbound storage APIs to mask storage protocol differences.
- Storage service layer: provides various storage media services, such as garbage collection, cyclic redundancy check, and hash.
- Driver layer: connects to different devices or logical volumes in the southbound direction and provides unified device driver registration APIs for using various storage media.
- Storage management: provides functions such as device management, I/O monitoring, and maintenance and test tools to manage devices.

Application Scenarios

The kit is ideal for distributed storage services carried by NVMe drives or traditional storage services. It uses the user-mode NVMe driver to take over drives, while also increasing the read and write performance of raw drives by 10-fold, and decreasing the I/O overhead by more than 50%, compared to the kernel I/O stack.

Repositories

https://gitee.com/openeuler/hsak

iSulad

Server | Cloud | Edge | Embedded

iSulad SIG

iSulad is a lightweight container engine developed using C/C++. It is not restricted by hardware architecture or specifications, has low memory overhead, and can be used in a wider range of fields.

Challenges

A container is an isolated environment that streamlines application packaging and distribution. Compared with virtualization technologies, containers accelerate distribution and reduce overhead, effectively improving development and deployment efficiencies. As the Docker container engine, Kubernetes container orchestration and scheduling, and cloud-native deployments have become more widespread, the container ecosystem is developing rapidly.

There are an increasing number of user requirements on containers, including the following:

- Containers need to be deployed and started quickly.
- Resources consumed by containers must be limited to a reasonable range.
- Containers should adapt to Internet of Things (IoT) and edge computing scenarios.

Based on these user requirements, we propose the iSulad container solution, a lightweight and fast container engine.

Project Introduction

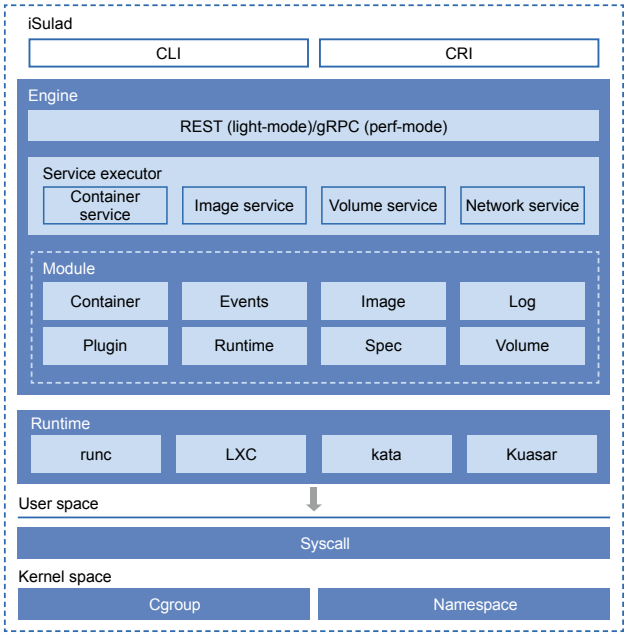
Features

The openEuler container engine, iSulad, features a unified architecture design tailored for the requirements of ICT fields. Compared with the Docker container engine developed using Go, iSulad occupies fewer resources, starts containers faster, and can be used in a wider range.

iSulad is named after the small isula ant, which is a small but very powerful insect. The same can be said about iSulad. It is a flexible, stable, and secure container base for various application scenarios.

iSulad provides commands similar to those of Docker, for greater usability. It supports the Container Runtime Interface (CRI) in the northbound direction and can connect to Kubernetes. You can use iSulad as the container base to orchestrate and schedule containers through Kubernetes. It also supports the Open Container Initiative (OCI) Runtime Specification in the southbound direction and is compatible with multiple container runtime environments, such as runc, LXC, kata, and Kuasar.

iSulad software architecture



Efficient Concurrency and Ultimate Performance

Core capabilities of iSulad include the container service, image service, volume service, and network service. The container service manages the lifecycle of containers, while the image service enables operations on container images. iSulad complies with the OCI Image Specification and supports mainstream image formats in the industry. In addition, iSulad supports the external rootfs image format in system container scenarios and the embedded image format in embedded scenarios. The volume service manages data volumes of a container, and the network service works together with Container Network Interface (CNI)-compliant network plugins to provide network capabilities for containers.

As a general-purpose container engine, iSulad supports system containers and secure containers as well as common containers.

- Common containers: They are traditional application containers.
- System containers: They have extended functions based on common containers, possessing the systemd management service capability, as well as being able to dynamically add or release drives, NICs, routes, and volumes when the container is running. System containers are mainly used in computing-intensive, high-performance, and heavy-concurrency scenarios to accommodate computing-intensive applications and cloudified services.
- Secure containers: They are a combination of virtualization and container technologies. Unlike common containers that share the same host kernel, secure containers clearly isolate containers from each other through the virtualization layer. Each secure container has its own kernel and a lightweight VM environment, ensuring that different secure containers on the same host do not affect each other.

Compared with Docker, iSulad features faster container startup and lower resource overhead. That is because iSulad is developed using C/C++ and has lower running overhead than in other languages. iSulad optimizes the call chain at the code layer. iSulad calls functions directly through the link library, whereas Docker call fork and exec functions on binary files for multiple times. The shorter call length enables iSulad to start containers faster, and what's more, the C language is a system programming language that allows iSulad to fully play its role on embedded and edge devices. In contrast, Docker is developed using Go and has a narrower application scope.

According to tests, iSulad brings only 30% of the memory overhead incurred by Docker, and in Arm and x86 environments, iSulad can start 100 containers concurrently in less than half of the time Docker takes. These advantages enable iSulad users to start up containers faster and reduce resource consumption, minimizing the impact on containerized applications.

Application Scenarios

iSulad has been widely used in cloud computing, ICT, embedded, and edge scenarios, empowering banking, finance, communications, and cloud services. The openEuler community has been developing the iSulad+Kuasar+StratoVirt solution for a more effective full-stack secure container solution.

Repositories

https://gitee.com/openeuler/iSulad

Kmesh

Cloud | Edge

eBPF SIG

Kmesh is a high-performance service mesh data plane software. Based on the programmable kernel, Kmesh offloads traffic governance from proxies to the OS and shortens the traffic path from multiple hops to one hop. It significantly improves application access performance in a service mesh.

Challenges

The boom of AI and live streaming applications has seen data centers expand to connect with more cluster services and manage soaring volumes of data. It is a big challenge to efficiently implement traffic governance between microservices in a data center.

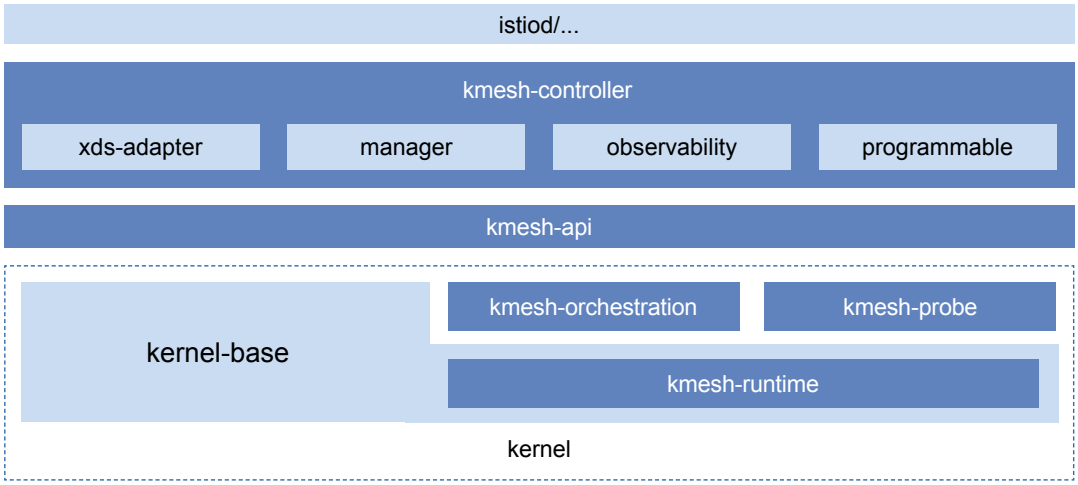
Service meshes are one of next-generation microservice technologies that separate traffic governance from services and offload it to the mesh infrastructure, implementing application-unaware traffic governance. However, their proxy architecture introduces extra latency and overhead. For example, the service mesh software Istio increases the single-hop service access latency by 2 ms to 3 ms, making Istio unable to meet the Service Level Agreement (SLA) requirements of latency-sensitive applications.

Application-unaware, high-performance traffic governance is a challenge that must be tackled.

Project Introduction

Kmesh works based on the programmable kernel to offload traffic governance to the OS. Kmesh supports the following features:

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



As shown in the figure, the Kmesh software architecture consists of the following components:

- kmesh-controller: Kmesh management program, which is responsible for Kmesh lifecycle management, xDS protocol interconnection, and O&M monitoring.
- kmesh-api: API layer provided by Kmesh for external systems, including orchestration APIs converted by xDS and O&M monitoring channels.
- kmesh-runtime: Runtime that supports L3 to L7 traffic orchestration implemented in the kernel.
- kmesh-orchestration: L3 to L7 traffic orchestration implemented based on eBPF, such as routing, gray, and load balancing.
- kmesh-probe: O&M monitoring probe, which provides end-to-end monitoring capabilities.

Application Scenarios

Latency-sensitive applications such as e-commerce, cloud gaming, online conferencing, and short videos. Kmesh brings a 5-fold forwarding performance increase in HTTP tests, compared to Istio.

Repositories

https://gitee.com/openeuler/Kmesh

LLVM for openEuler

Server | Cloud | Edge | Embedded

Compiler SIG

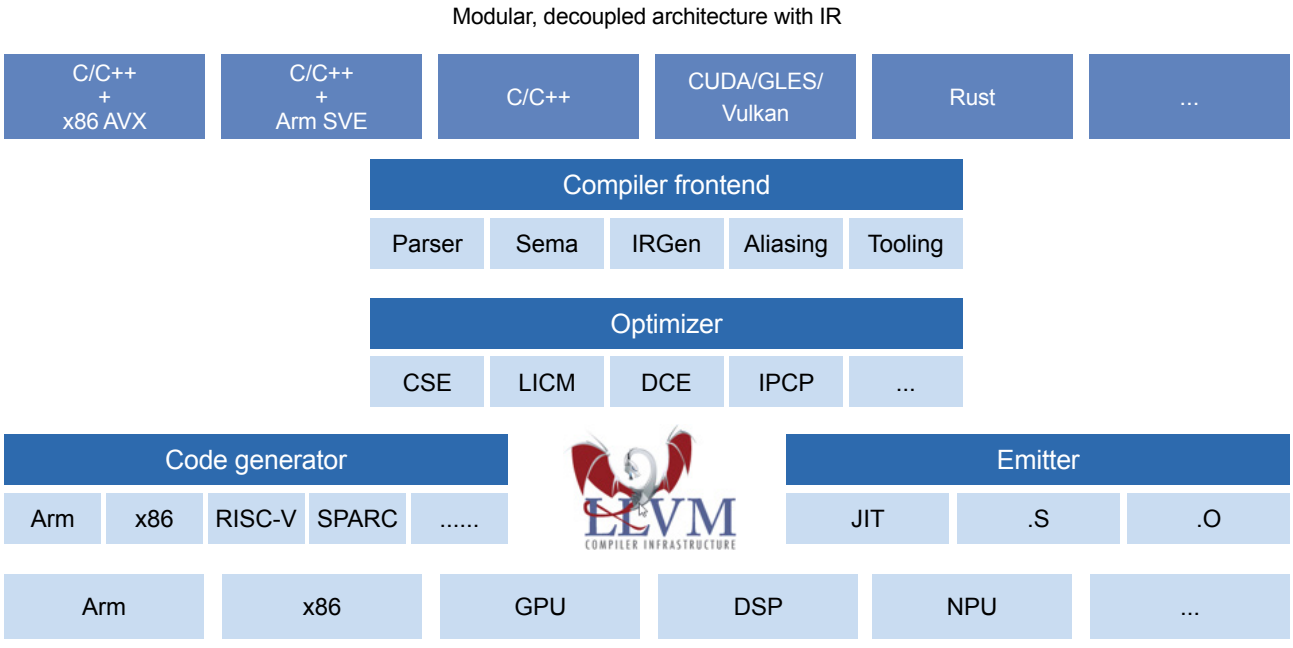
The open source LLVM project is a collection of modular and reusable compiler and toolchain technologies. This project has attracted widespread attention from developers, and commercial companies have launched their commercial compilers based on the LLVM project. LLVM for openEuler is innovative in terms of compatibility, performance, and development-state secure coding. It adapts to multiple hardware platforms, such as Kunpeng, Phytium, and Loongson, to fully unleash diversified computing power.

Challenges

LLVM is an alternative compiler on openEuler and needs to bring more competitive capabilities than GCC. Specifically, it needs to provide powerful and scalable optimization capabilities to bring more performance benefits in major computing scenarios, such as databases, software-defined storage, and virtualization. In addition, LLVM for openEuler needs to have a complete ecosystem to expand the user base. Specifically, it must be compatible with existing software packages, and also support the efficient compilation of newly developed software packages.

Project Introduction

LLVM has a modular architecture design and divides the compilation process into multiple independent phases, such as the frontend, optimization, and backend. This design makes LLVM more flexible and scalable, facilitates the independent evolution and innovation of modules in each phase, and combines different modules through the unified intermediate representation (IR). The LLVM project contains multiple subprojects, such as Clang, Flang, LLVM, MLIR, and LLD.



Feature 1: Sanitizers

The LLVM Sanitizers are a collection of tools used for dynamic code analysis and bug detection. They help developers detect and debug common memory errors and security problems. These tools are closely integrated with the LLVM compiler and runtime library to provide a convenient way for detecting and diagnosing code bugs.

Function	Usage	Issues to Detect
Fast memory error detection	-fsanitize=address	• Out-of-bounds accesses to heap/stack/globals • Use-after-free • Use-after-return • Use-after-scope • Double-free • Invalid free • Memory leaks
Data race detection	-fsanitize=thread	• Data races
Memory detector	-fsanitize=memory	• Uninitialized reads • Use-after-destruction
Undefined behavior detection	-fsanitize=undefined	• integer-divide-by-zero • Bitwise shifts that are out of bounds for their data type • Dereferencing misaligned or null pointers • Signed integer overflow
Hardware-assisted memory error detection	-fsanitize=hwaddress	• Same as AddressSanitizer
Stack protection	-fsanitize=safe-stack	• Protects programs from stack buffer overflow attacks.

Feature 2: Clang Extra Tools

Clang Extra Tools are a group of additional tools provided by the LLVM project. They are used together with the Clang C/C++ compiler to support static code analysis, code refactoring, and code style check.

- Clang-Tidy: A powerful static code analyzer used to check for common errors, potential problems, and code style violations in C, C++, and Objective-C code. It automatically detects and fixes problems in code, helping developers write higher-quality code.
- Clang-Format: A code formatter that automatically formats C, C++, and Objective-C code. It can automatically adjust the indentations, line feeds, and spaces of the code according to the configured rules. It helps members of a team be consistent on the code style, improving code readability and maintainability.
- Clang-Check: A tool for writing a custom static analyzer. It lets developers customize static analysis rules for detecting specific problems or potential errors in code. It provides powerful APIs and frameworks, enabling developers to create tailored code check tools based on their requirements.

Feature 3: Clang+LLVM – the Parallel Universe Program

As the LLVM project develops quickly and developers demand more, is it possible to build openEuler releases based on the LLVM technology stack? Compiler SIG and RISC-V SIG jointly initiate the LLVM Parallel Universe Program to explore the possibility.

Benefit Analysis

- Basic performance: Compared with GCC, LLVM enhances compilation optimization and has more promising performance potential. On top of that, LLVM has more powerful link-time optimization (LTO) capabilities.
- Software package performance: Software package maintainers can choose GCC or LLVM as the build toolchain, so that they can channel more time on software function implementation.
- Code security: The Clang+LLVM combination has stricter compliance requirements for C/C++. It can detect potential defects of software packages through static check and Sanitizer dynamic detection.

Application Scenarios

As a C/C++/Rust compiler, LLVM for openEuler can be used to build server, cloud computing, edge computing, and embedded applications. It shortens the time of building the openEuler 23.03 Embedded image by 16%, compresses the code size by 1.5%, and increases the CoreMark performance by 6%. As a general-purpose compiler, LLVM for openEuler is perfectly suitable for general-purpose computing scenarios, such as databases, software-defined storage, and virtualization.

Repositories

https://gitee.com/openeuler/llvm-project (source code)

https://gitee.com/src-openEuler/clang

https://gitee.com/src-openEuler/llvm

https://gitee.com/src-openEuler/lld

https://gitee.com/src-openEuler/llvm-bolt

OneAll

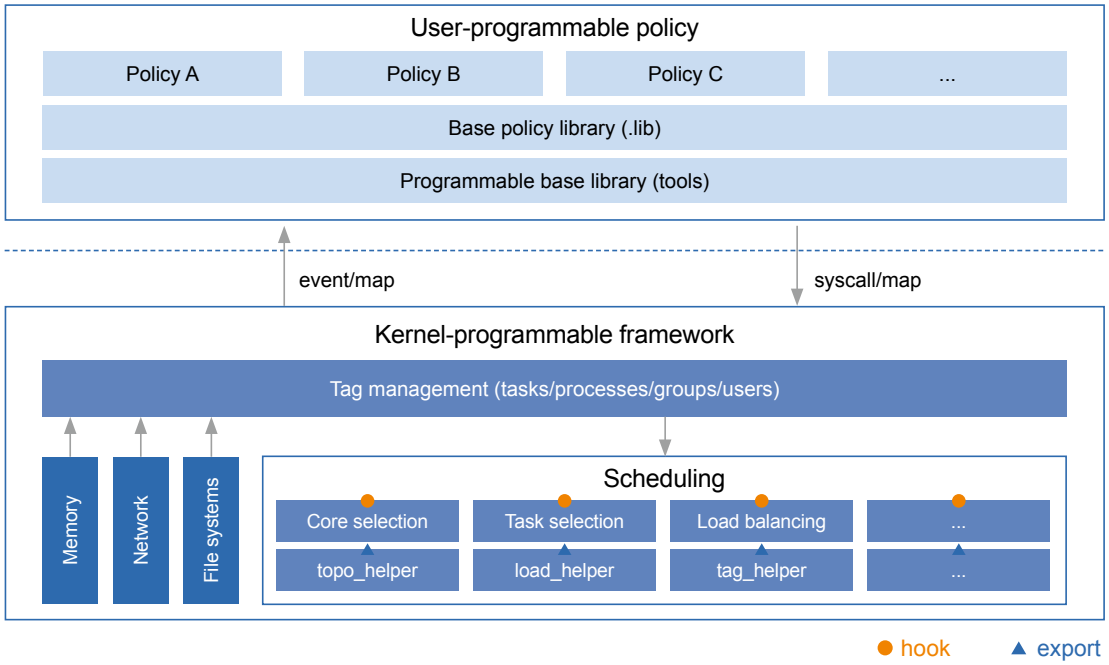
Server | Cloud

eBPF SIG

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

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

Features



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

Application Scenarios

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

StratoVirt

Cloud

Virt SIG

StratoVirt is an enterprise-class virtualization platform oriented to cloud data centers. It offers a unified architecture that fits into the three scenarios: VMs, containers, and serverless computing. StratoVirt is lightweight and causes low memory overhead, supports software and hardware collaboration, and is safe at the Rust language level.

Challenges

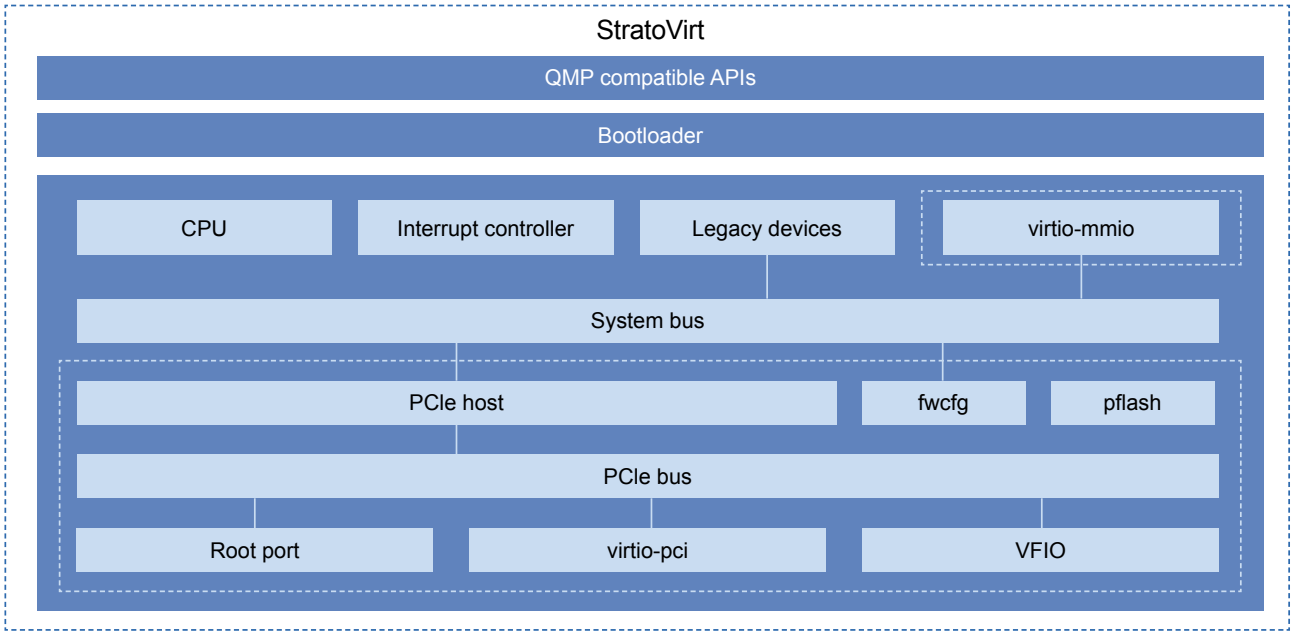
As the QEMU virtualization software has been gradually evolving, the code scale of its core open source components is becoming increasing large, among which there is a large amount of outdated code. In recent years, CVE security vulnerabilities frequently occur, and problems such as poor security, code redundancy, and low efficiency are awaiting handling. A practicable solution is the rust-vmm architecture, which is developed using the memory-safe programming language Rust. General-purpose virtualization technologies for all scenarios (data centers, terminals, and edge devices) are the future trend, due to their security, light weight, and performance advantages. StratoVirt emerges as a next-generation virtualization technology designed for openEuler.

Project Introduction

Features

StratoVirt is an open-source lightweight virtualization technology based on Linux Kernel-based Virtual Machine (KVM). It reduces memory consumption and accelerates VM startup while maintaining the isolation and security capabilities of traditional virtualization technologies. StratoVirt can be applied in serverless scenarios such as microservices or function computing, and retains virtualization interfaces and designs for quickly importing more features to supplement general virtualization capabilities.

StratoVirt software architecture



Efficient Concurrency and Ultimate Performance

The core architecture of StratoVirt is divided into three layers from top to bottom:

- External APIs: StratoVirt uses the QEMU Machine Protocol (QMP) to communicate with external systems, is compatible with OCI, and supports interconnection with libvirt.
- Bootloader: In lightweight scenarios, StratoVirt uses a simple bootloader to load kernel images, much faster than the traditional BIOS+Grub boot method. In general-purpose virtualization scenarios, StratoVirt supports UEFI boot.
- Emulated mainboard – microVM: To improve performance and reduce the attack surface, StratoVirt minimizes the emulation of user-mode devices. With the emulation capability, KVM-based devices and paravirtualization devices are available, such as generic interrupt controller (GIC), serial, real-time clock (RTC), and virtio-mmio devices.
- General-purpose VMs: StratoVirt provides an advanced configuration and power interface (ACPI) table to implement UEFI boot. Virtio-pci and Virtual Function I/O (VFIO) passthrough devices can be added to further improve the VM I/O performance.

Application Scenarios

StratoVirt, iSula, and Kubernetes combine to form a complete container solution, which processes serverless loads efficiently.

Repositories

https://gitee.com/openeuler/stratovirt

Compiler Plugin Framework

Server | Cloud | Edge | Embedded

Compiler SIG

The compiler plugin framework is a plugin development platform that provides MLIR-oriented interfaces, helping develop a plugin while applying it on multiple compilers. The framework supports and maintains common capabilities such as plugin compatibility and integrity checks.

Challenges

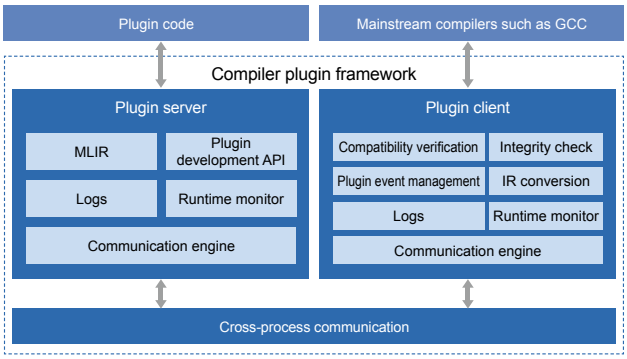
There are two major compiler frameworks: GCC and LLVM. A large number of compilation tools and extended compilation capabilities are developed based on the two compiler frameworks. The code for one compiler framework cannot be reused on another. Therefore, compilation tool development faces the following difficulties:

- The compiler needs in-depth modification, which also complicates compiler maintenance.
- Repetitive coding occurs when you develop the compilation tool based on the two compilation frameworks.
- Lack of base capabilities such as compatibility increases tool development and maintenance costs.

Project Introduction

Features

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



Application Scenarios

Scenario 1: Compilation tool build and integrity verification

The compiler plugin framework can work as the development platform to build compilation tools while applying them on multiple compilers such as GCC. The framework supports and maintains common capabilities such as compatibility and binary integrity checks.

Scenario 2: Quick enabling and verification of compilation tools

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

Repositories

https://gitee.com/openeuler/pin-gcc-client

https://gitee.com/openeuler/pin-server

IMA

Server | Cloud | Edge | Embedded

Security Facility SIG

The Integrity Measurement Architecture (IMA) is a mandatory access control (MAC) subsystem that provides file integrity protection in Linux kernel 2.6 and later. With IMA digest lists, the IMA generates and protects file measurement digest base values in the build phase, verifies imported measurement base values in the boot phase, and protects the integrity of key system files during the running phase.

Challenges

As the operating environment on the network is complex, it may be exposed to various types of attacks when they are running. Attackers can replace the executable files carried by the system or implant unknown malicious programs, causing unpredictable damage to the system. The IMA is an extension to the trusted boot mechanism and provides enhanced user-mode file integrity protection based on kernel-mode trustworthiness.

The IMA can measure files accessed through system calls such as execve() and mmap() based on user-defined policies. The measurement result can be used for measurement and appraisal:

- Measurement: Detects accidental or malicious modifications to files, with local or remote attestation.
- Appraisal: Measures a file and compares it with a pre-stored reference value to protect the integrity of the local file.

The native IMA feature of the Linux kernel is confronted with the following shortcomings:

- Complicated deployment: The native IMA stores file integrity information through file extended attributes. To enable IMA verification, you need to set the system to fix mode, generate and mark the extended attributes, and then reboot the system to enter the enforce mode.
- Performance deterioration: In the native IMA, each time file measurement is triggered, the Platform Configuration Registers (PCRs) of the trusted platform module (TPM) are extended. The TPM is a low-speed chip, and the extension process is very time consuming. Besides, each time file verification is triggered, the signature or hash-based message authentication code (HMAC) stored in the file extended attributes is verified. The verification process is also long and deteriorates performance.

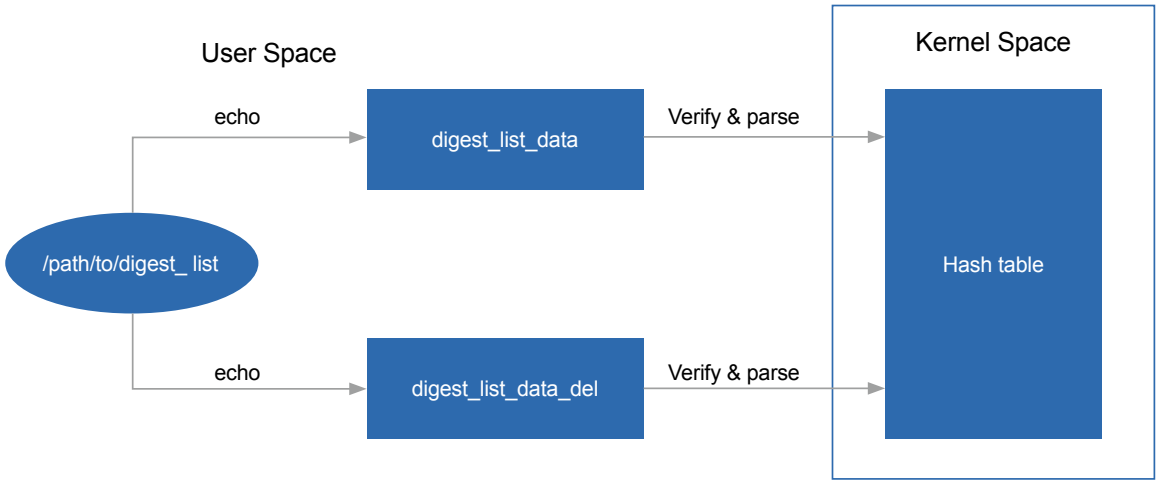
Project Introduction

Features

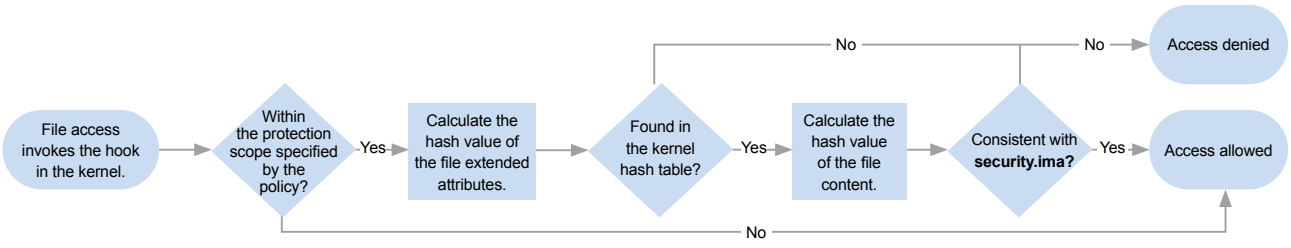
The IMA is a trusted computing implementation method in openEuler, connecting trusted applications to the trusted OS along the trust chain. IMA digest lists are provided by openEuler to enhance the native kernel integrity protection mechanism, replacing the native IMA mechanism for greater file integrity protection.

Digest lists are binary data files in a special format. Each digest list corresponds to an RPM package and records the hash values of protected files (executable files and dynamic library files) in the RPM package.

After the boot parameters are correctly configured, the kernel maintains a hash table (invisible to users) and provides interfaces (digest_list_data and digest_list_data_del) that update the hash table through securityfs. The digest lists are signed using a private key when they are built. When uploaded to the kernel through interfaces, the digest lists are verified by a public key in the kernel.



When IMA appraisal is enabled, each time an executable file or dynamic library file is accessed, the hook in the kernel is invoked to calculate the hash values of the file content and extended attributes, and search the kernel hash table for the hash values. If the calculated hash values match the hash table, the file is executed. Otherwise, the file access is denied.



Application Scenarios

IMA digest lists are used in data center, cloud computing, edge, and embedded scenarios to protect system file integrity, representing a key technology in trusted computing. IMA digest lists help set up a trusted local environment so that the trust chain of trusted computing can be extended to the application layer. In addition, measurement logs can be used for remote attestation to verify whether the files loaded to the tested platform and the system running status are trusted.

Repositories

https://gitee.com/src-openEuler/digest-list-tools

https://gitee.com/openeuler/digest-list-tools

https://gitee.com/openeuler/kernel

KunpengSecL

Server | Cloud | Edge | Embedded

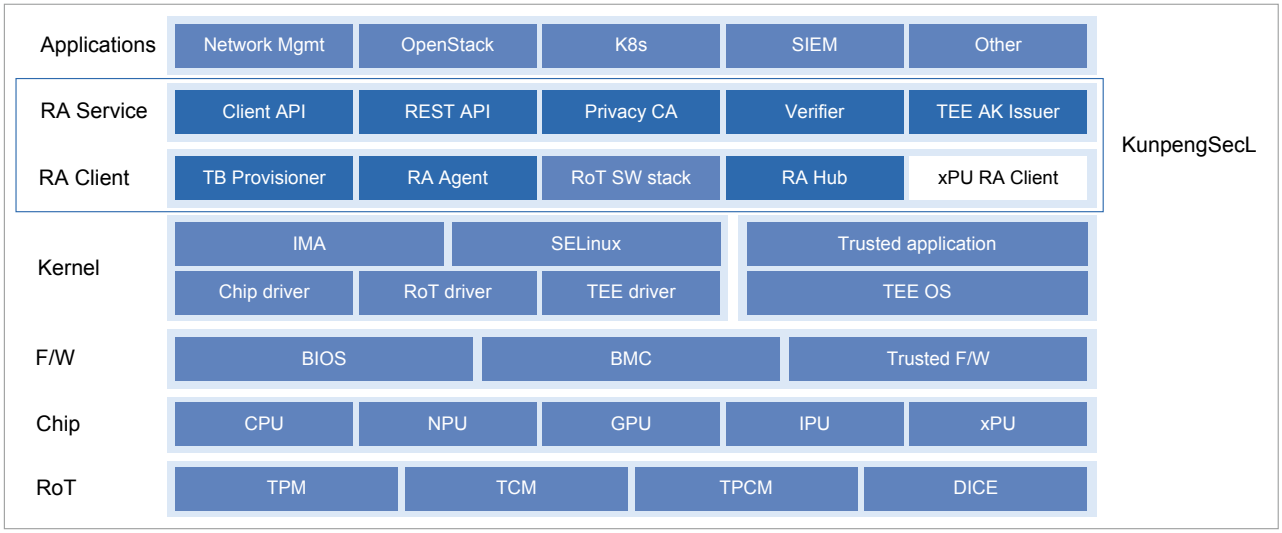
Security Facility SIG

Kunpeng Security Library (KunpengSecL) is a fundamental security software component running on Kunpeng processors. In the early stage, KunpengSecL focuses on trusted computing fields such as remote attestation.

Challenges

Server security technologies for trusted computing, confidential computing, and trusted AI are developing rapidly. The security features provided by server hardware are far from being easy to use for common software developers and users. Security software middleware is a practicable solution to shorten the gap between software developers and hardware security features.

Project Introduction



Each KunpengSecL feature comprises components and services.

- The components are deployed on worker nodes that provide resources (compute, storage, and network) for user workloads. They encapsulate platform security and trustworthiness capabilities into software interfaces, which are available to the services.
- The services are deployed on independent management nodes to aggregate security and trustworthiness capabilities from all worker nodes and provide them for users and specified management tools. The services are tailored for system security and trustworthiness design.

Remote attestation is the first KunpengSecL security feature, and is an end-to-end trusted computing solution that obtains the trustworthiness status of software and hardware on worker nodes. Various resource management tools can formulate policies based on trustworthiness reports to schedule and use server resources in a differentiated manner.

The remote attestation feature of KunpengSecL supports:

- TPM-based remote attestation for universal platforms.
- Remote attestation for the Kunpeng trusted execution environment (TEE).

Application Scenarios

Scenario 1: trusted cloud hosts

By combining trusted boot of cloud physical servers and remote attestation of the platform, trusted verification can be performed on the host environment where VMs are running to provide underlying security support for cloud host users. In addition, the virtual Trusted Platform Module (vTPM) is used to support trusted boot and remote attestation of VMs. In this way, users can perceive the security and trustworthiness status of trusted cloud hosts, thereby enhancing user confidence regarding cloud hosts.

Scenario 2: key cache management

Remote attestation for the platform and TEE and local attestation for the TEE are used to harden security in scenarios where TAs obtain and cache keys from enterprises' or cloud Key Management Service (KMS), ensuring the security of keys in transmission, storage, and use.

Repositories

https://gitee.com/openeuler/kunpengsecl/

secCrypto

Server | Cloud

Security Facility SIG

openEuler provides support for cryptographic services such as Chinese cryptographic algorithm libraries, certificates, and secure transmission protocols. Key security features that use cryptographic algorithms, including OS user authentication, drive encryption, and integrity protection, are enhanced by the Chinese cryptographic algorithms (also known as ShangMi algorithms, or SM for short).

Challenges

SM algorithms are being increasingly adopted in the industry. However, most open source software in openEuler's repository that uses cryptographic algorithms does not support SM algorithms. As a result, openEuler and its upper-layer applications cannot utilize SM algorithms natively to ensure service security.

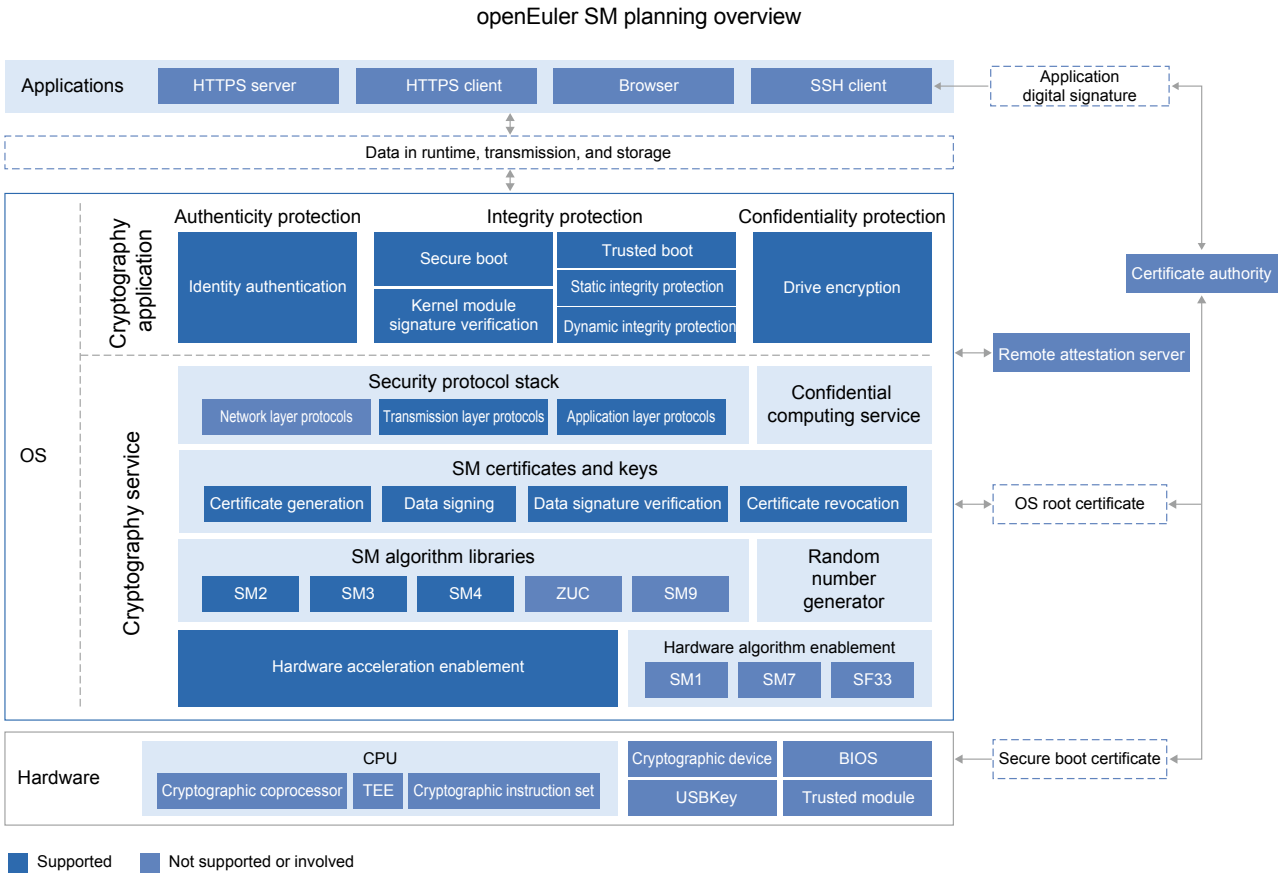
In addition, the current SM algorithm implementations still bring performance loss, which need to be optimized to fully realize the potential of software and hardware collaboration.

Project Introduction

Features

Currently, the following SM features are supported by openEuler:

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



Application Scenarios

SM algorithm stack of openEuler is mainly used in server and cloud scenarios, supporting SM2, SM3, and SM4 in both kernel and user modes. openEuler, now has been reconstructed for SM algorithms, serving as the information system foundation to enable SM algorithms for various industries and help them meet the cryptography assessment.

Repositories

https://gitee.com/src-openEuler/openssl
https://gitee.com/src-openEuler/nss
https://gitee.com/src-openEuler/openssh
https://gitee.com/src-openEuler/pesign
https://gitee.com/src-openEuler/shim
https://gitee.com/src-openEuler/aide
https://gitee.com/src-openEuler/pam
https://gitee.com/src-openEuler/libxcrypt
https://gitee.com/openeuler/kernel

secGear

Server | Cloud

Confidential Computing SIG

secGear is a security application development kit (SADK) that delivers confidential computing for the computing industry. It is a unified development framework that masks the differences between trusted execution environments (TEEs) and software development kits (SDKs). It provides development tools and common security components to help security application developers focus on services and improve development efficiency.

Challenges

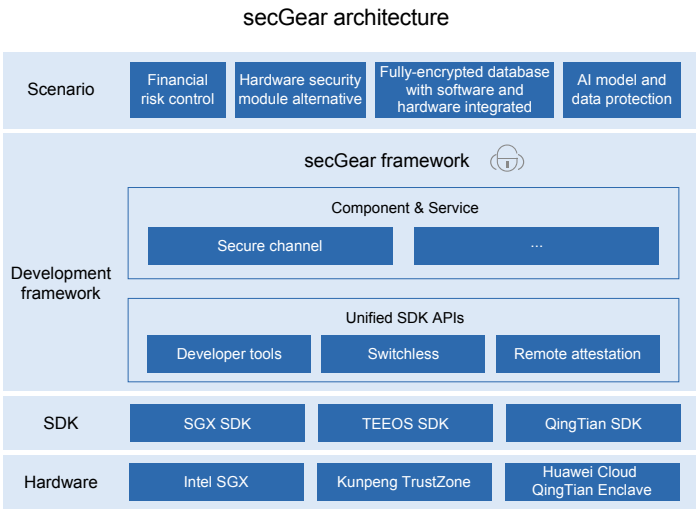
The rapid development of confidential computing technologies allows chip vendors to launch their own confidential computing solutions, but it poses the following problems for security application developers:

- Ecosystem isolation: To deploy a confidential computing application on a different platform, secondary development is necessary based on the SDK of the target platform.
- Difficult development: Some platforms provide only awkward bottom-layer interfaces, which come at a high cost in terms of learning and development.
- Low performance: A confidential computing application is designed to run in both the rich execution environment (REE) and TEE. However, frequent context switches between these two environments can significantly degrade performance.

Project Introduction

secGear features the following benefits:

- Architecture compatibility: It masks differences between different SDK APIs to share the same set of source code across multiple architectures.
- Easy development: The development tools and common security components allow users to focus on services, significantly improving development efficiency.
- High performance: The switchless feature improves the interaction performance between the REE and TEE by more than 10-fold in typical scenarios such as frequent interactions between the REE and TEE and big data interaction.



Application Scenarios

secGear is widely used in scenarios such as databases, hardware security module alternatives, AI model and data protection, and big data. It helps customers in industries such as finance and telecom quickly port services to confidential computing environments and protect data runtime security.

Repositories

https://gitee.com/openeuler/secGear

secPaver

Server | Cloud

Security Facility SIG

secPaver is a SELinux policy development tool. Its core concept is to abstract and encapsulate a group of common policy description methods and policy operation interfaces. During policy development, developers do not need to understand security mechanism details but rather use secPaver to configure policy descriptions. The specific security policies are automatically generated by secPaver.

Challenges

Although Linux can be hardened by mandatory access control (MAC) mechanisms such as SELinux and AppArmor, they have yet to be widely used in actual scenarios due to the complicated security policy configurations:

- SELinux contains hundreds of thousands of policy rules. At least hundreds of rules are required to configure a security policy for a single application.
- SELinux policy functions as an allowlist. If a policy is not correctly configured, applications will not run as expected.
- Application developers are not skilled enough in defining security policies, which results in the security policies of most applications not being defined during the development process. Hence, system administrators are unable to understand the running and resource access of each application and cannot define detailed security policies for each application.

To address these problems, a simplified security policy configuration tool is required to help developers and system administrators quickly define security policies for applications to run.

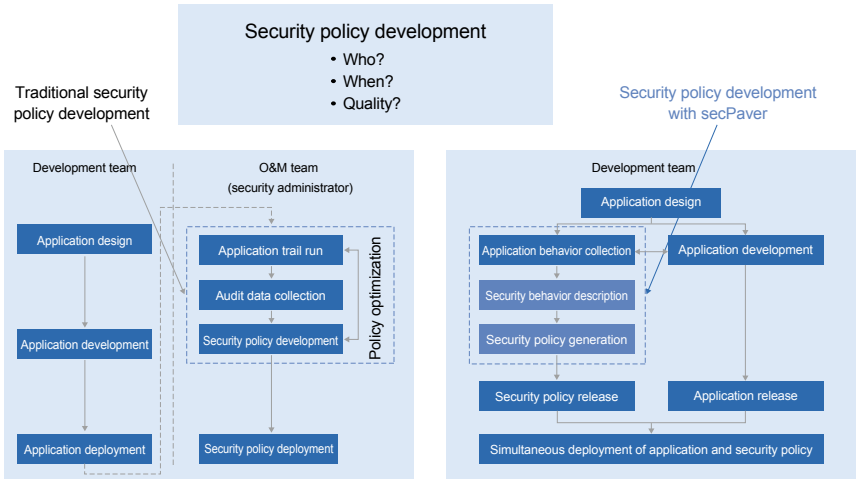
Project Introduction

Features

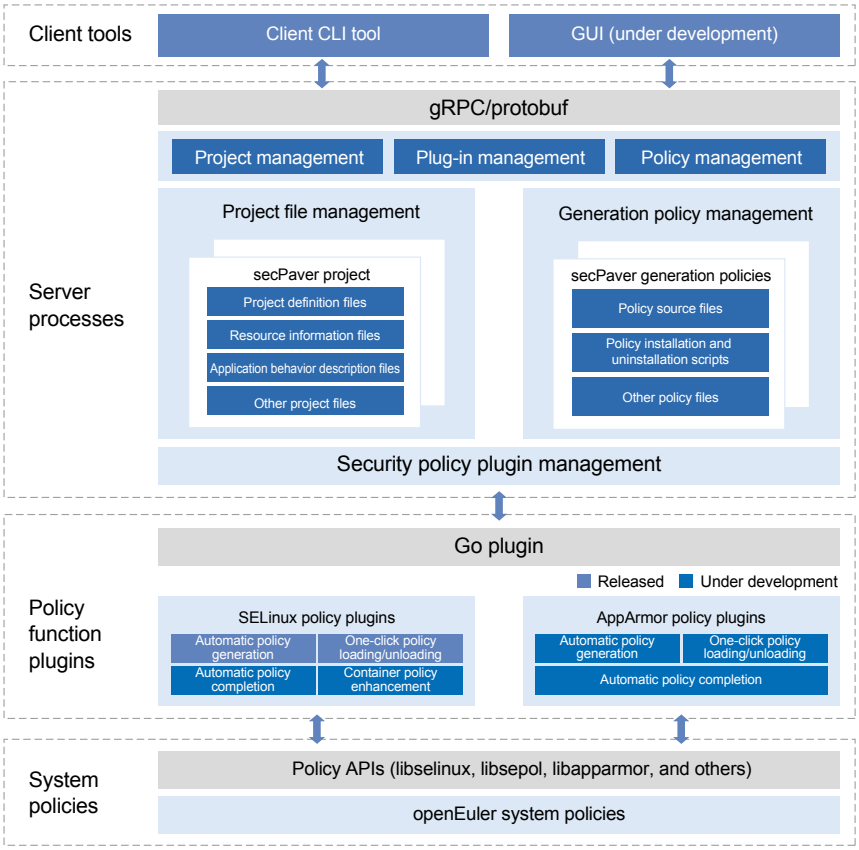
An application security policy sets security rules for the running of an application, where only behavior that complies with the rules is allowed. Proper security policies can effectively improve system security and resilience, so that even if an application is attacked, actions beyond the rules cannot be performed.

The core of security policies is access control. That is, the access of applications to resources, such as file read/write and socket usage, is checked by the OS to determine whether the access needs to be blocked. openEuler provides secPaver to help generate security policies for applications, simplifying the management of complex security policies by system administrators.

Compared with traditional security policy development methods, secPaver reduces the knowledge requirements on security mechanisms for developers, simplifies the policy development process, improves policy development efficiency, and enables the simultaneous release of software and security policies.

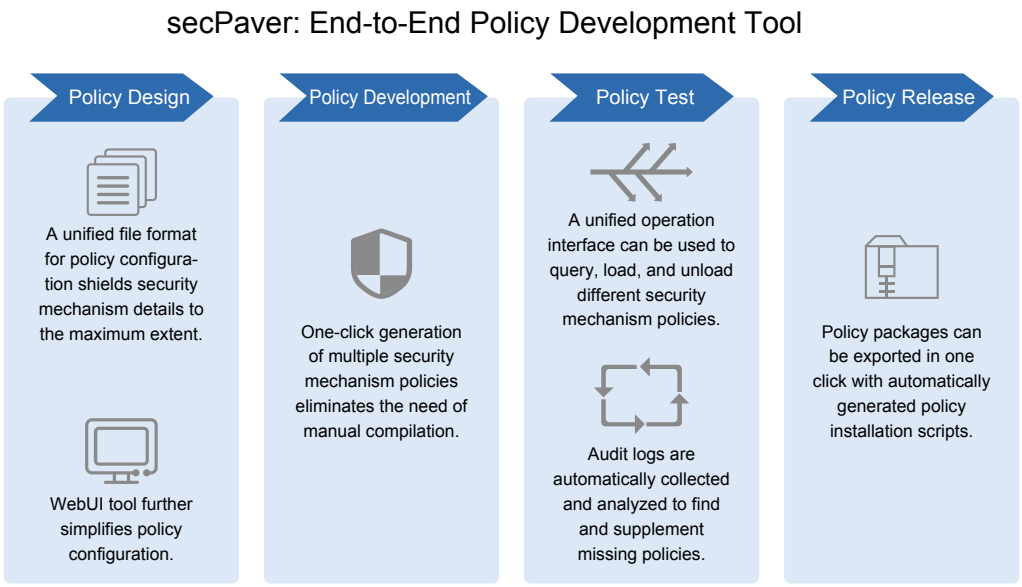


secPaver is written in Go and adopts the client-server architecture. Users can use client tools to interact with server processes to develop security policies. The server is a systemd service process that provides specific policy development functions. The go-plugin mechanism used by secPaver encapsulates function implementations of different security mechanisms in plugins for the server process to load and invoke. Currently, secPaver supports SELinux policy development, and in the future, it will support the policy development of AppArmor and other security mechanisms.



Application Scenarios

secPaver assists developers in security policy generation during application development. From a security policy development cycle perspective, the functions of secPaver cover the policy design, iterative development, and policy release processes.



Repositories

https://gitee.com/src-openEuler/secpaver

https://gitee.com/openeuler/secpaver

sysMaster

Server | Cloud | Edge | Embedded

Dev-utils SIG

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

Challenges

In Linux, PID 1, traditionally the init process, is the parent of all user-mode processes. The init process is the first process that is created when the system is started. It starts and manages all other processes and ends them when the system is shut down. In modern Linux distributions, init is usually replaced by the systemd process. However, the concept of PID 1 (whose minimum functions include system startup and zombie process recycling) still exists.

PID 1 is a key system process and is responsible for system initialization and runtime service management. It faces the following challenges:

- Poor reliability: The function problem of PID 1 has a bigger impact. When PID 1 is faulty, the OS must be restarted to rectify the fault.
- High complexity: systemd becomes the de facto standard for PID 1, introducing many new concepts, tools, and extended components that depend on each other. It is difficult to tailor the components based on actual application scenarios.

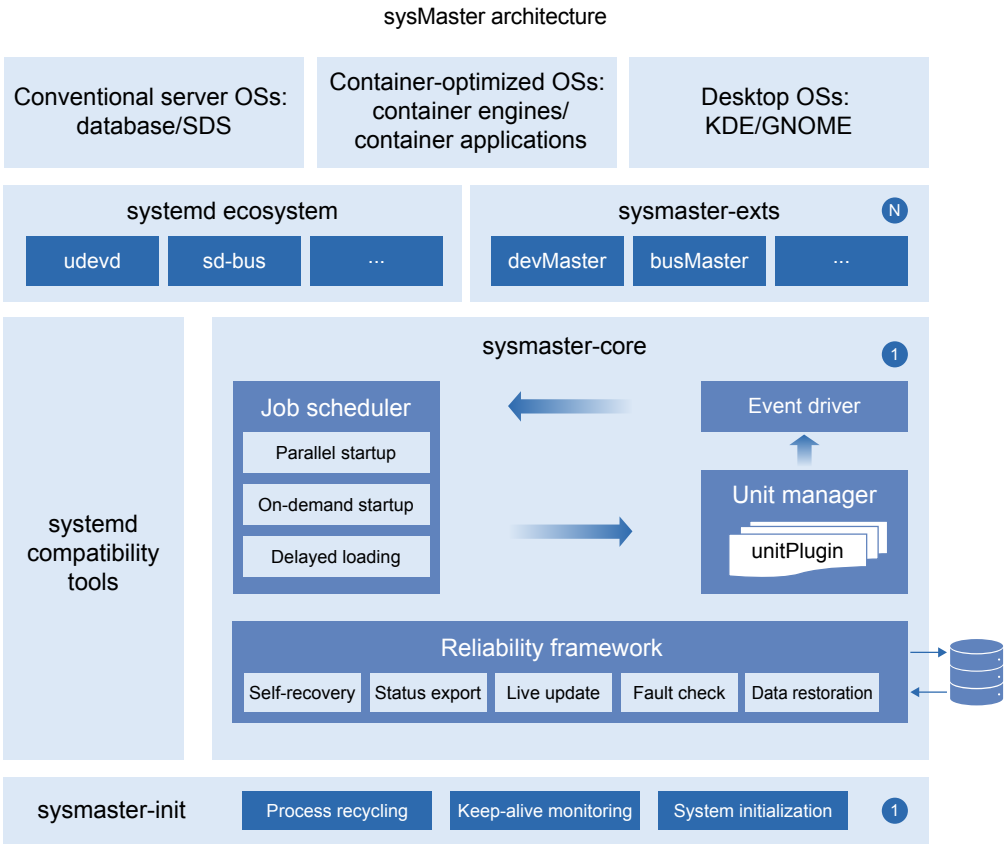
Project Introduction

Features

sysMaster manages processes, containers, and VMs centrally and provides fault monitoring and self-healing mechanisms to help deal with Linux initialization and service management challenges. All these features make sysMaster the ideal choice for server, cloud computing, and embedded scenarios.

sysMaster divides the functions of traditional PID 1 into a 1+1+N architecture based on application scenarios. As shown in the figure, sysMaster consists of three components:

- sysmaster-init, which is a new implementation of PID 1, features simplified functions, a thousand lines of code (KLOC), and ultimate reliability. It is applicable to embedded systems with functions such as system initialization, zombie process recycling, and keep-alive monitoring.
- sysmaster-core undertakes the core service management functions and incorporates the reliability framework to enable live updates and quick self-recovery in the event of crashes, ensuring 24/7 service availability.
- sysmaster-extends offers a collection of components (such as devMaster for device management and busMaster for bus communication) that deliver key system functions, which are coupled in traditional PID 1. You can choose the components to use as required.



Featuring a simple component architecture, sysMaster improves the scalability and adaptability of the overall system architecture while reducing development and maintenance costs. sysMaster provides the following advantages:

- Live updates and self-recovery in seconds in the event of crashes
- Faster startup speed with lower memory overhead
- Plugin-based service types that can be dynamically loaded as required
- Migration tools that provide seamless migration from systemd to sysMaster
- Unified interfaces that work with the iSulad container engine and QEMU for management of container and virtualization instances

In the future, sysMaster will extend to more scenarios and have its architecture and performance further optimized for higher scalability and adaptability. In addition, new features and components will be developed to meet the requirements of container, virtualization, and edge computing scenarios. These features will make sysMaster a powerful, efficient, and user-friendly system management framework.

Application Scenarios

sysMaster can be used in containers, virtualization, servers, and edge devices to deliver a reliable and lightweight experience.

Repositories

https://gitee.com/openeuler/sysmaster

A-Ops

Server | Cloud

Ops SIG

A-Ops is an OS-oriented O&M platform that provides intelligent O&M solutions covering data collection, health check, fault diagnosis, and fault rectification.

Challenges

In recent years, the implementation of cloud-native, serverless, and other technologies has made cloud infrastructure O&M increasingly challenging. Specifically, the characteristics of sub-health problems, such as intermittent emergence, short duration, multiple problem types, and wide involvement, have brought great challenges to cloud infrastructure troubleshooting. The sub-health fault diagnosis in Linux poses even higher requirements for capabilities such as observability, massive data management, and generalization of AI algorithms. In openEuler, the existing O&M methods fail to detect and locate sub-health problems promptly due to insufficient capabilities, including online continuous monitoring, refined observation from the application perspective, and AI automatic analysis based on full-stack observation data. The difficulties in diagnosing sub-health faults are as follows:

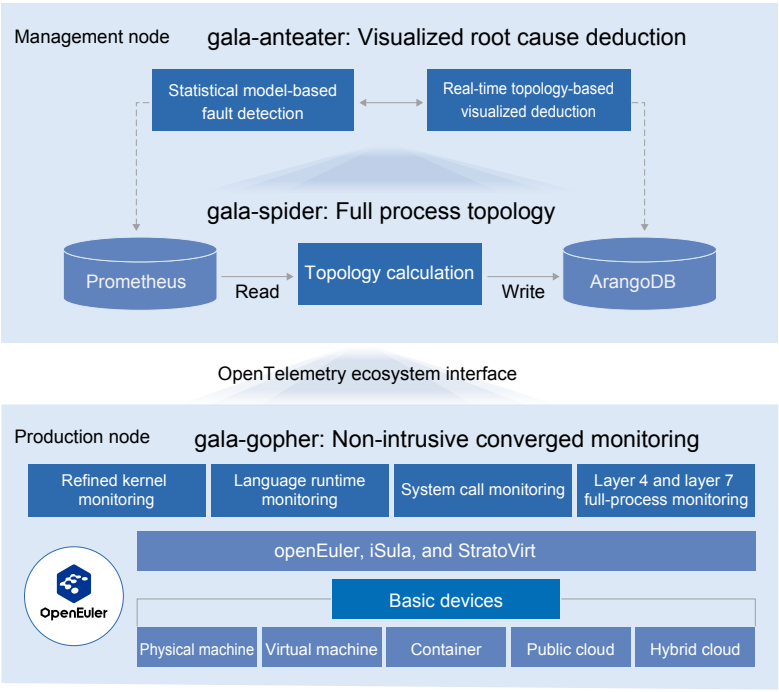
- Full-stack non-intrusive observation
- Continuous, refined, and low-resource consumption monitoring
- Adaptive exception detection and visualized fault deduction for various scenarios
- Patch management and application without affecting services

Project Introduction

Features

The A-Ops project includes the following sub-projects: fault detection (gala), fault locating (X-diagnosis), and defect rectification (Apollo).

gala: gala utilizes a non-intrusive observation technology based on the eBPF + Java agent and provides intelligent assistance to diagnose sub-health faults, such as performance jitter, increased error rate, and slow system response. The figure shows the architecture of gala.



gala has the following features:

- Online application performance jitter diagnosis: Online performance diagnosis for database applications, to identify issues including network issues (packet loss, retransmission, latency, and TCP zero window), I/O issues (slow drives and I/O performance deterioration), scheduling issues (high system CPU usage and deadlock), and memory issues (out of memory and leakage).
- System performance diagnosis: TCP and I/O performance jitter diagnosis in common scenarios.
- System risk inspection: Second-level inspection on kernel protocol stack packet loss, virtualization network packet loss, TCP exceptions, I/O latency, system call exceptions, resource leakage, JVM exceptions, application RPC exceptions (including error rates and latency of eight common protocols), and hardware faults (uncorrectable errors and drive media errors).
- Full-stack I/O monitoring: Full-stack I/O monitoring for the SDS scenario, covering the process I/O and block layer I/O of the guest OS, virtual storage layer frontend I/O, and SDS backend I/O.
- Refined performance profiling: Online real-time continuous collection of performance statistics, including CPU performance, memory usage, resource usage, and system calls in high precision (collected every 10 ms) and multiple dimensions (covering system, process, container, and pod) to generate flame and timeline graphs.
- Full-stack monitoring and diagnosis of Kubernetes pods: Real-time topology of pod cluster service flow, pod performance monitoring, DNS monitoring, and SQL monitoring from the perspective of Kubernetes.

X-diagnosis: X-diagnosis is an O&M suite for Linux, providing a fault locating toolkit, system exception inspection, and enhanced ftrace functions.

X-diagnosis has the following features:

- Various fault locating tools: Locates faults related to the network, I/O, CPU scheduling, file system, and memory, such as, ICMP, TCP, and UDP packet loss and exceptions, read-only system files, and slow I/O.
- Abundant inspection items for system problems: Inspects network, CPU scheduling, drives, services, configurations, and system resources to detect exceptions such as incorrect DNS configurations, high CPU usage, full drive space, time changes, and excessive processes. Inspection results can be output to logs and interfaces.
- eftrace and ntrace for system debugging and analysis: eftrace is an enhanced version of ftrace that supports automatic calculation of structure offsets, making ftrace more accessible. ntrace can quickly output key parameters of the kprobe protocol stack function to assist in locating protocol stack process problems. What's more, the parameters can be filtered by IP address, port, and protocol.

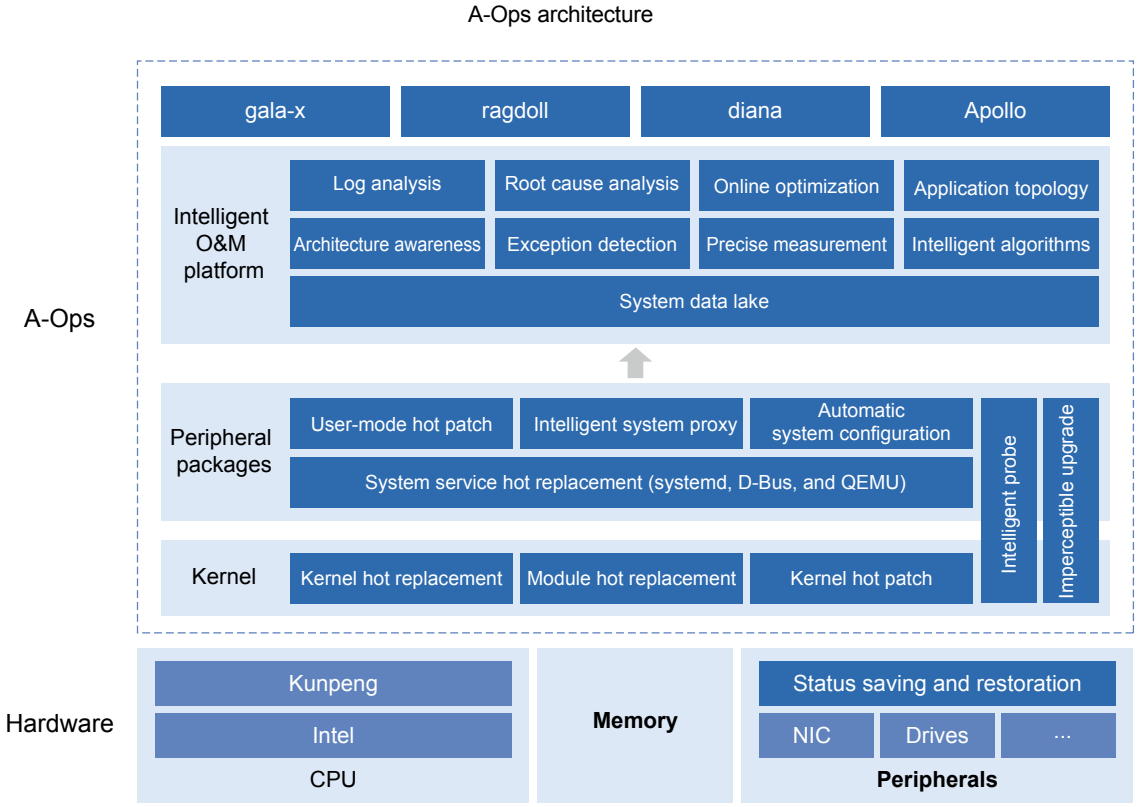
Apollo: The Apollo project is an intelligent patch management framework. It provides real-time inspection of CVEs and bugs and cold and hot patching, in order to implement automatic discovery and zero-interruption fixing.

Apollo has the following features:

- Patch service: Cold and hot patch subscription allows patches to be acquired online.
- Intelligent patch inspection: Supports CVE/bug inspection and notification based on single-node systems and clusters, hybrid management of cold and hot patches, and one-click repair and rollback, significantly reducing patch management costs.

ragdoll: More than 50% of OS faults are caused by incorrect configurations. ragdoll can monitor system configurations to detect real-time configuration changes and quickly locate incorrect configurations.

- Configuration baseline: Configuration file types, including user-defined configuration files, can be added to the cluster baseline through plugins.
- Configuration source tracing: Automatically detects system configuration file changes in the backend and notifies the administrator of the changes through alarms and emails.
- Configuration locating: Quickly locates the cause of configuration file changes by monitoring file operations through the eBPF. (This function is under development.)



Application Scenarios

A-Ops is applicable to openEuler and other Linux distributions in database, SDS, virtualization, and cloud-native scenarios. A-Ops provides full-stack monitoring capabilities for users in industries such as finance, telecom, and Internet to diagnose sub-health faults, and promptly checks configuration errors caused by manual operations in cluster scenarios. In addition, A-Ops manages cold and hot patches in a unified manner to simplify patch management and directly provides hot patches for high-severity kernel CVEs to prevent the system from being restarted when dealing with the emergent kernel issues.

Repositories

https://gitee.com/openeuler/A-Ops
https://gitee.com/openeuler/aops-apollo
https://gitee.com/openeuler/X-diagnosis
https://gitee.com/openeuler/gala-gopher
https://gitee.com/openeuler/gala-spider
https://gitee.com/openeuler/gala-anteater

CPDS

Cloud Edge

CloudNative SIG

CPDS, short for container problem detection system, monitors and detects top and sub-health faults in containers.

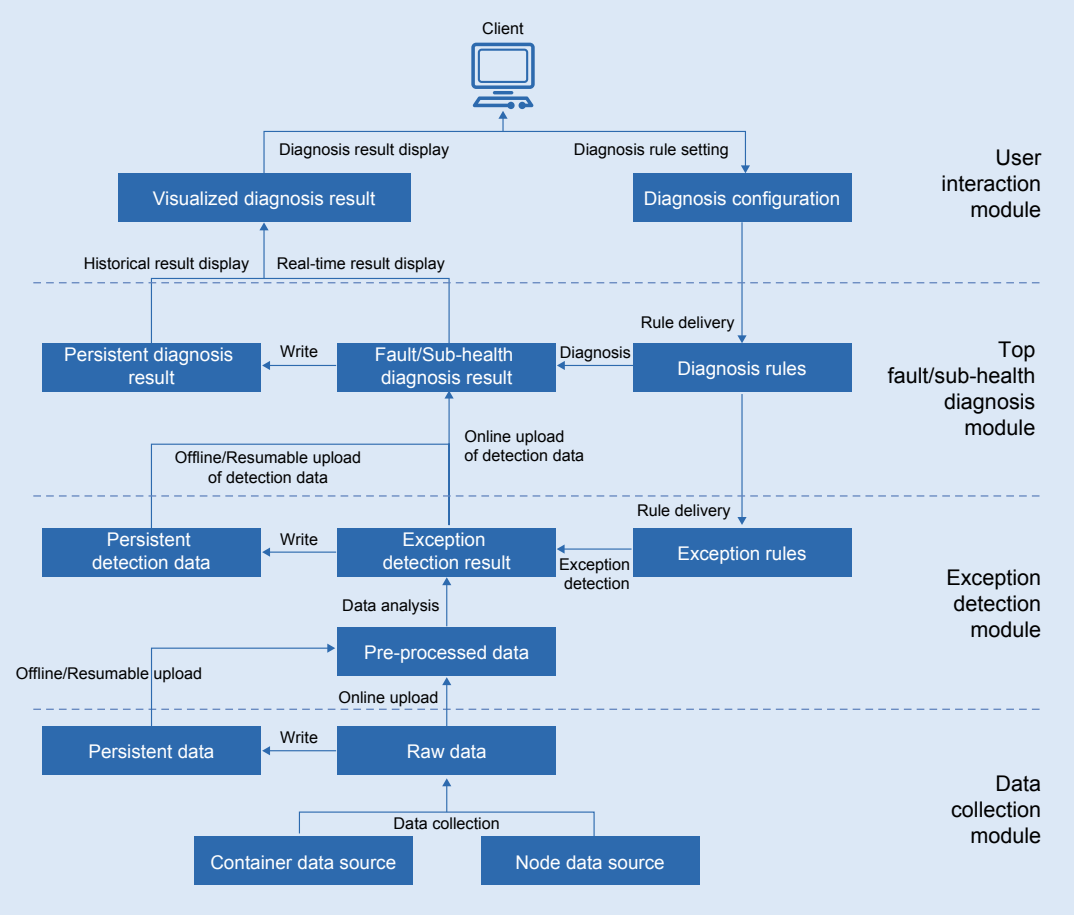
Challenges

As businesses grow larger and larger, and container clusters continue to expand, IT O&M finds itself under more pressure. Service interruptions caused by software and hardware faults have become one of the major factors that affect stability. Most existing fault detection solutions for container clusters in the industry are based on cluster component status detection, service entry monitoring, and user-defined interface liveness probing. These technologies struggle to detect or identify the sub-health status of services and cannot deliver fault diagnosis or execution policies. As a result, key faults cannot be handled.

Project Introduction

Features

CPDS adopts the microservice architecture. The components of its four component groups communicate with each other through APIs.



- cpds-agent, the data collection component, collects raw container and system data from nodes in the cluster.
- cpds-detector, the exception detection component, analyzes the raw data of each node based on configured exception rules to check whether the node is abnormal.
- cpds-analyzer, the fault/sub-health diagnosis component, performs health analysis on abnormal nodes based on configured diagnosis rules and determines the current health status of the nodes.
- cpds-dashboard, the user interaction component, provides a web UI to display the health status of nodes in the cluster and set and deliver diagnosis rules.

Cluster Information Collection

Node agents are implemented on host machines to monitor key container services using systemd, initv, eBPF, and other technologies. These agents also keep track of the application status, resource consumption, key system function execution status, and I/O execution status of containers. The collected information covers network, kernel, and drive LVM of the nodes.

Cluster Exception Detection

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

Fault/Sub-Health Diagnosis on Nodes and Service Containers

Nodes and service containers are diagnosed based on exception detection data. Diagnosis results are saved permanently and can be displayed on the UI for users to view real-time and historical diagnosis data.

Application Scenarios

CPDS is mainly used in the infrastructure of cloud-native scenarios to detect container cluster faults.

Repositories

https://gitee.com/openeuler/Cpds

CPM4OSSP

Server

Ops SIG

Centralized management platform for operating system software packages (CPM4OSSP) is designed by Linx Software for unified management of software package installation, upgrade, and uninstallation on multiple servers.

Challenges

Currently, most Linux distributions run commands on a single host or use a remote tool such as Ansible to download software packages from the source server. Such a method fails to deliver centralized management of software packages and repositories on multiple hosts. To address this issue, CPM4OSSP provides a centralized and unified software package management solution for multiple nodes.

Project Introduction

CPM4OSSP consists of the following modules:

- Software repository management module, which can be used to generate, view, deliver, edit, and delete software repository templates, helping O&M personnel quickly set up dependency environments.
- Software package management module, which provides software package installation, uninstallation, upgrade, and rollback capabilities, simplifying software package maintenance and improving O&M efficiency.
- Audit management module, which audits all user operations to facilitate fault locating.
- User management module, which separates the privileges of system users, security users, and audit users to improve platform security and stability.

Application Scenarios

CPM4OSSP provides users with the following functions:

- Simplified management UI that supports privilege-based user management
- Host software repository updates and rollbacks
- Node software package update check, one-click upgrade, and search
- Software package management by category
- Software package uninstallation
- Operation audit

Repositories

https://gitee.com/openeuler/CPM4OSSP-UI

https://gitee.com/openeuler/CPM4OSSP-SERVER

https://gitee.com/openeuler/CPM4OSSP-PROXY

CTinspector

Server | Cloud

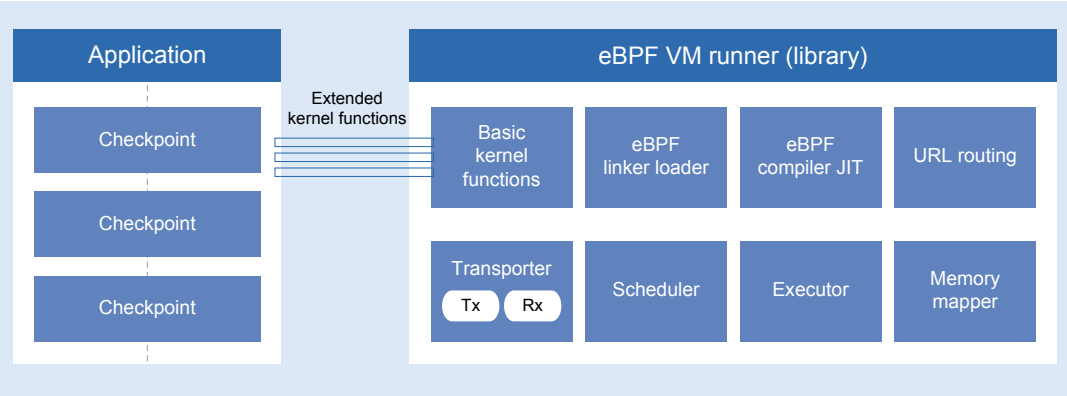
Ops SIG

CTinspector uses the eBPF VM technology to implement multi-node stateful monitoring and network traffic analysis, helping quickly locate network performance bottlenecks.

Challenges

Traditional performance monitoring tools lack the capability of multi-node stateful monitoring and cannot be programmed in real time to cope with rapid requirement changes.

Project Introduction



The access control list (ACL) based on the eBPF VM turns the traditional iptables rules into an eBPF program, simplifying ACL function development and improving delivery speed. In addition, RPC based on the eBPF VM can save network bandwidth, reduce latency, and adaptively diagnose network problems based on scripts compiled by O&M personnel.

The server framework of CTinspector consists of modules including the kernel functions, loader, compiler, router, transporter, scheduler, executor, and memory mapper. When an external task is delivered, the script content, execution context, and calculation results are encapsulated in the eBPF VM. In this way, non-interactive chained propagation can be implemented, greatly improving execution and calculation efficiency.

Application Scenarios

To locate a network fault, O&M personnel often need to dump Open vSwitch flow tables. However, this can be inconvenient due to the large number of flow tables, and it can take a long time to dump them all. Therefore, it is necessary to filter the flow tables. Traditional tools have difficulties in adding filtering fields online or performing stateful filtering, such as identifying the three flows that forward the most packets. To address this issue, O&M personnel can use CTinspector to pass the flows to an eBPF program. This program can then determine which flows to filter and count the number of packets to identify the top three flows that forward the most packets.

Repositories

https://gitee.com/openeuler/CTinspector

eggo

Server | Cloud

CloudNative SIG

eggo helps automatically deploy large Kubernetes clusters in a flexible and traceable manner within production environments.

Challenges

Kubernetes cluster deployment has always been a challenging task, and the industry has provided several solutions to address this issue. Despite this, the openEuler CloudNative SIG is not content with the current solutions and has proposed additional requirements for cluster deployment:

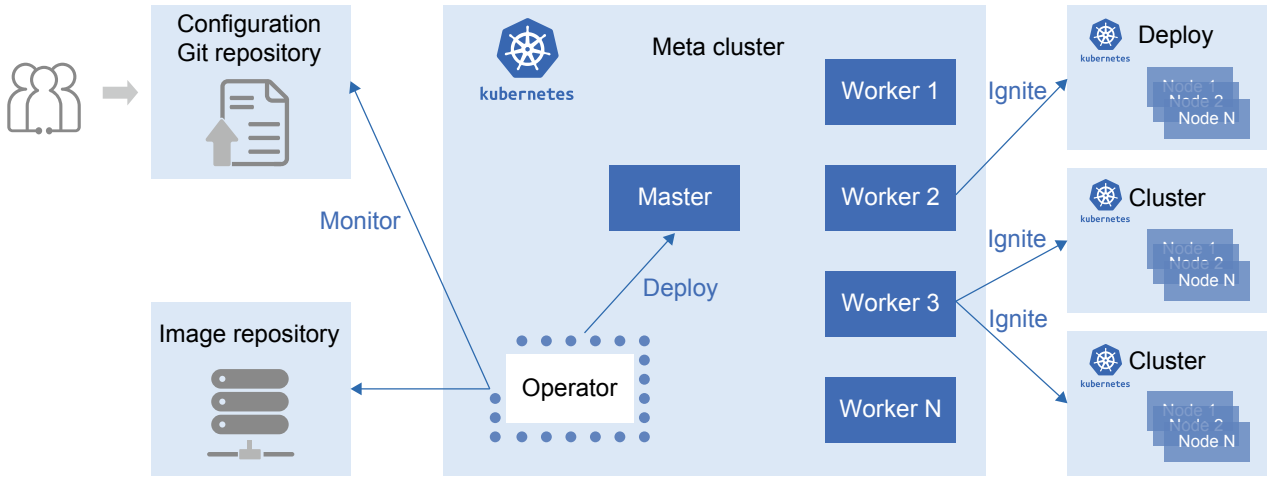
- Multiple deployment methods
- Online and offline deployment
- Heterogeneous nodes
- Traceable cluster configuration management

Project Introduction

Features

eggo can run as an independent component to provide cluster deployment and management for individuals or work with GitOps to manage cluster deployment configurations through a repository. It provides the following functions:

- Deployment of Kubernetes clusters based on common Linux distributions, such as openEuler, CentOS, and Ubuntu
- Deployment of heterogeneous clusters that contain nodes of different architectures (x86_64 and AArch64)
- Kubernetes component deployment using binary files
- Online and offline deployment



The scheduling of deployment tasks is determined by the network affinity of worker nodes in the target cluster and meta cluster

Key components:

- Configuration Git repository: Git repository for storing cluster deployment configurations. A cloud cluster can register a webhook to detect configuration changes in the repository and trigger cluster management operations.
- Image repository: repository for container images used by the cluster.
- Meta cluster: Kubernetes cluster where the eggo management program is deployed to detect changes in the cluster configuration repository set by the user, and manage cluster lifecycles.
- Operator: custom resource definition (CRD) of the meta cluster, which is responsible for cluster configuration awareness and lifecycle management of load clusters.
- Worker nodes: load nodes of the meta cluster, which execute target cluster deployment tasks and are selected based on network affinity.
- Load cluster: cluster managed by eggo, which runs user services. Users can manage the cluster (for example, delete, create, or update nodes) through eggo at any time.

Application Scenarios

eggo can be used to automatically deploy large Kubernetes clusters in production environments and trace cluster lifecycles and changes.

Repositories

https://gitee.com/openeuler/eggo

nvwa

Server | Cloud | Edge | Embedded

Ops SIG

The nvwa kernel live upgrade function upgrades the kernel and fixes kernel vulnerabilities without interrupting services, ensuring stable system operation.

Challenges

The increasing complexity of the Linux kernel leads to the discovery of new CVEs, of which 60% can only be fixed by kernel upgrades rather than hot patches, posing following challenges in fixing kernel vulnerabilities:

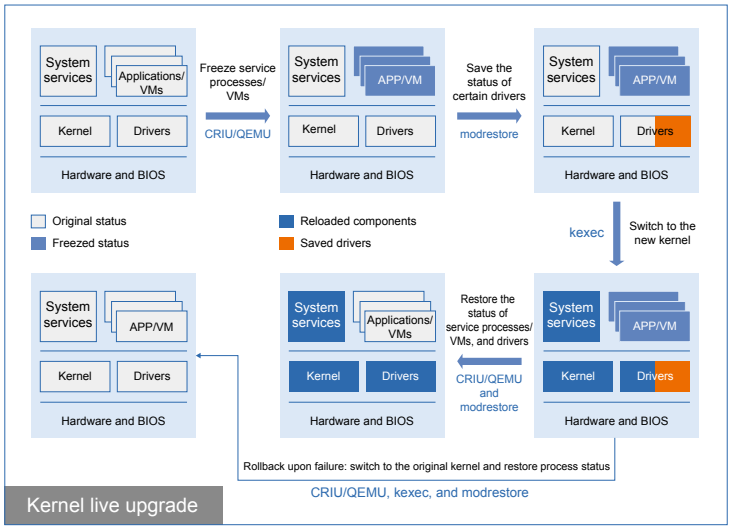
- In most service scenarios, system restart is not allowed because service continuity must be ensured.
- The time and storage costs of service migration are unacceptable.
- A large number of services cannot be migrated.

Project Introduction

nvwa provides a simple **nvwa update** command to perform kernel live upgrades.

nvwa uses Checkpoint/Restore In Userspace (CRIU) to ensure service continuity, kexec as well as some enhanced technologies to switch kernels, and driver and kernel status saving mechanisms to resume services after the system is recovered. nvwa's key technologies are as follows:

- Freezing and restoration of processes and VMs: CRIU and QEMU are improved to freeze the complete status of a process or VM in memory. After the kernel is upgraded, the process or VM is restored seamlessly in the user space.
- Fast kernel reboot: kexec is optimized so that the kernel can be rebooted within 500 ms.



- Hardware status retention: During the kernel upgrade, hardware and BIOS are not operated to ensure their status remains unchanged, while direct memory access (DMA) is allowed.
- Kernel module status retention: The module_suspend and module_resume interfaces are provided to save the status of necessary drivers for restoration after the kernel upgrade.
- Memory pinning: Memory used by user-mode processes and kernel modules can be frozen until the kernel upgrade is complete.

Application Scenarios

nvwa supports the x86 and AArch64 architectures and can be directly used on VMs that run common services such as MySQL, Redis, and Nginx. On physical servers, the framework can be adapted for specific drivers if necessary.

Repositories

https://gitee.com/openeuler/nvwa

PilotGo

Server

Ops SIG

The PilotGo O&M management platform is a plugin-based O&M management tool developed by the openEuler community, providing full-lifecycle management capabilities for server clusters and function expansion based on plugins. PilotGo aims to bridge the gap between different O&M tools to build a complete automatic O&M management process.

Challenges

O&M technologies are used in a wide range of service scenarios, including security, log collection, metric monitoring, fault locating, performance optimization, configuration management, and process automation. Each service O&M scenario may have its own O&M platform, which provides complete but complex functions that require high learning costs. In addition, service scenarios are typically distinct from one another, which requires different O&M tools in a cross-scenario process, complicating the effective implementation of full-process automation. Moreover, O&M platforms may differ in their functions and not provide convenient APIs or CLI interfaces that can be effectively encapsulated, thus streamlining different O&M tools is quite challenging.

Project Introduction

Features

PilotGo is a plugin-based O&M management platform, which only provides basic functions. Functions specific to service scenarios are provided by platform plugins.

Core Functions

The core functions of PilotGo support the basic capabilities of the platform and provide related services for the plugin system, mainly including:

- User management: manages user information.
- Host management: manages host clusters.
- Permission management: manages permissions.
- Audit logs: provides the audit function.
- Batch management: manages batch tasks.
- Configuration management: manages system and software configurations.
- Remote command execution: provides the remote control function.
- Plugin management: manages plugins and plugin extensions.

Most core functions are exposed as interfaces for plugins to implement various service logics. In addition, PilotGo provides a global event notification mechanism to improve the awareness ability of the entire cluster.

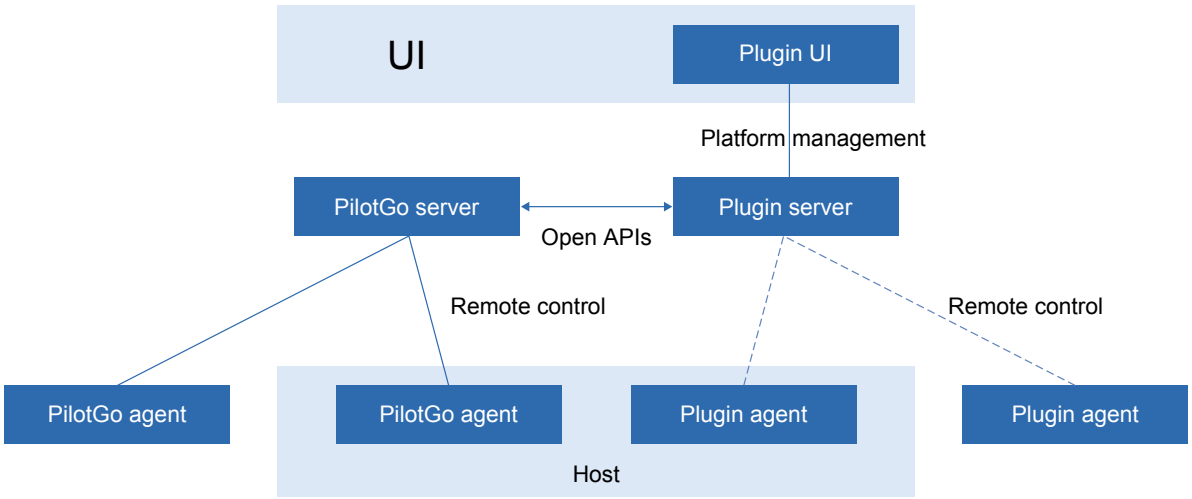
Extended Functions

The PilotGo O&M management platform extends functions through plugins to implement common O&M operations. PilotGo supports the following extension plugins:

- Prometheus: collects and displays monitoring metrics and generates alarms.
- Grafana plugin: visualizes metrics.
- gala plugin: supports the functions of the gala intelligent O&M software, such as advanced metric collection and display, fault diagnosis, and platform topology.

The function interfaces of PilotGo can also be exposed for different service plugins to streamline cooperation, enhancing automation of the entire platform.

Architecture



PilotGo consists of the following independent components:

- PilotGo server: core logical component of the PilotGo platform
- PilotGo UI: provides the frontend framework of the microservice architecture to extend the plugin UI.
- PilotGo agent: provides the remote control capability.
- Plugin server: provides the service expansion capability.
- Plugin UI: provides UIs related to plugin functions and is embedded in the PilotGo UI.
- Plugin agent (optional): works with the plugin server to implement the service expansion capability.

Generally, the PilotGo server and plugin server are deployed on separate servers and communicate with each other through network protocols. The PilotGo agent and plugin agent are deployed on service hosts and communicate with only their corresponding servers.

Application Scenarios

PilotGo can be used to monitor and manage OSs and application running environments on bare metal servers, VMs, and containers, to provide cluster monitoring and O&M capabilities in typical application scenarios, such as container, MySQL, and Nginx.

Repositories

Project repository: https://gitee.com/openeuler/PilotGo

Plugin repository: https://gitee.com/openeuler/PilotGo-plugins

SysCare

Server | Cloud | Edge | Embedded

Ops SIG

SysCare is an OS O&M tool that resolves faults during system operations and provides hot patching services for Linux OSs.

Challenges

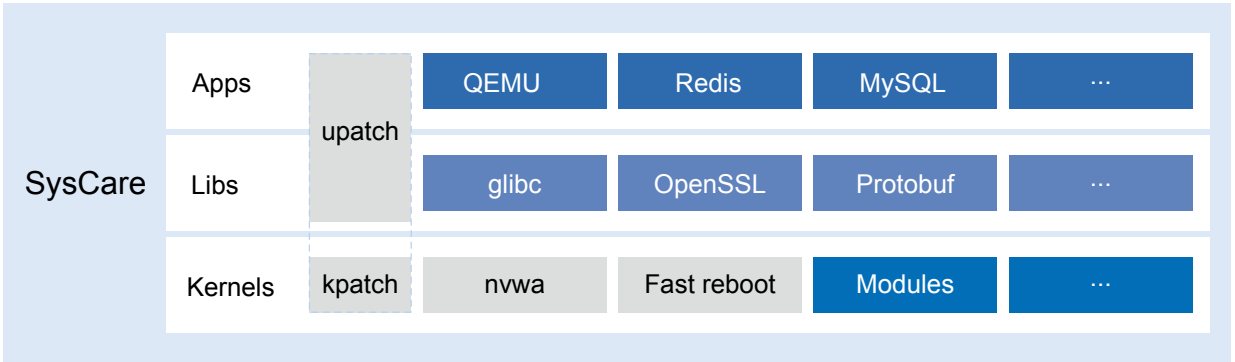
A long-standing problem in Linux is the lack of quick and reliable solution to fix vulnerabilities and faults without affecting services.

Hot patches are typically used to solve such problems, which allows users to repair components failures online by directly changing the code, all without affecting services. However, hot patches are hard to create and manage, especially to match the specific code. Further, there is no simple and unified patch mechanism for the diverse file formats, programming languages, compilation methods, and running modes.

Even using hot patches to repair requires software upgrade to ensure direct use of the software upon service or system restart. In general, software upgrades and hot patching are the two methods to fix vulnerabilities, but due to certain limitations, hot patching can only address 40% of the issues. The remaining 60% of issues can only be resolved by live migration and cold upgrades. However, cold upgrade will lead to slow system restart, resulting in service interruptions. Therefore, Linux requires a fast and secure system restart method.

Project Introduction

SysCare offers a comprehensive solution for hot services. It provides unified hot patching and fast reboot capabilities, including hot patching for applications (C/C++ applications like QEMU, Redis, MySQL), dynamic libraries (glibc, OpenSSL, Protobuf), and kernels. It also integrates kernel hot upgrade capability (nvwa). See figure below.



SysCare provides the following key functions:

User-mode hot patching

There are many mainstream options for kernel hot patching solutions such as kpatch and livepatch. SysCare integrates common hot patching capabilities of openEuler with new features to meet demands of user-mode file formats, programming languages, compilation, and running modes.

- Compares the **.o** object files generated before and after code modifications, then extracts the differences to generate hot patch files.
- Injects code into compilers such as GCC by using uprobe to track the entire compilation process and obtain the essential information and **.o** object files required for creating hot patches.
- Binds the hot patches to the ELF files by using uprobe, in which patches are triggered by uprobe after the ELF file runs, eliminating the need for process monitoring. This enables the patches to take effect after patching or when a new process runs, and also supports hot patching for dynamic libraries.

Patch management

To simplify the complexity associated with typical hot patch management solutions, SysCare shields the differences between kernel and user-mode hot patches. For example, you can run the **syscare build** command to build a hot patch for a specified component. In addition, SysCare provides patch management commands such as **apply**, **active**, **deactive**, **remove**, **status**, **info**, and **list** for applying, activating, deactivating, and removing patches, and can also query the status, information, and lists of patches.

Fast reboot

It uses quick kexec and CPU park technologies to significantly improve system reboot speeds. Moreover, it integrates with the systemd reboot process to ensure secure termination of service and swift recovery.

Application Scenarios

SysCare can be used in kernel, dynamic library, and user mode scenarios that require hot patches to fix bugs and CVEs, or to support software upgrades and system restarts.

Repositories

https://gitee.com/openeuler/syscare

05

Developer Support

Compass-CI

Server | Cloud | Edge | Embedded

CICD SIG

Compass-CI is an open source software platform that supports continuous integration. It provides developers with test, login, assisted fault demarcation, and historical data analysis for upstream open source software (GitHub, Gitee, GitLab, and other host platforms). Compass-CI performs automated testing, including build testing and use case testing, based on PRs to build an open and complete task execution system.

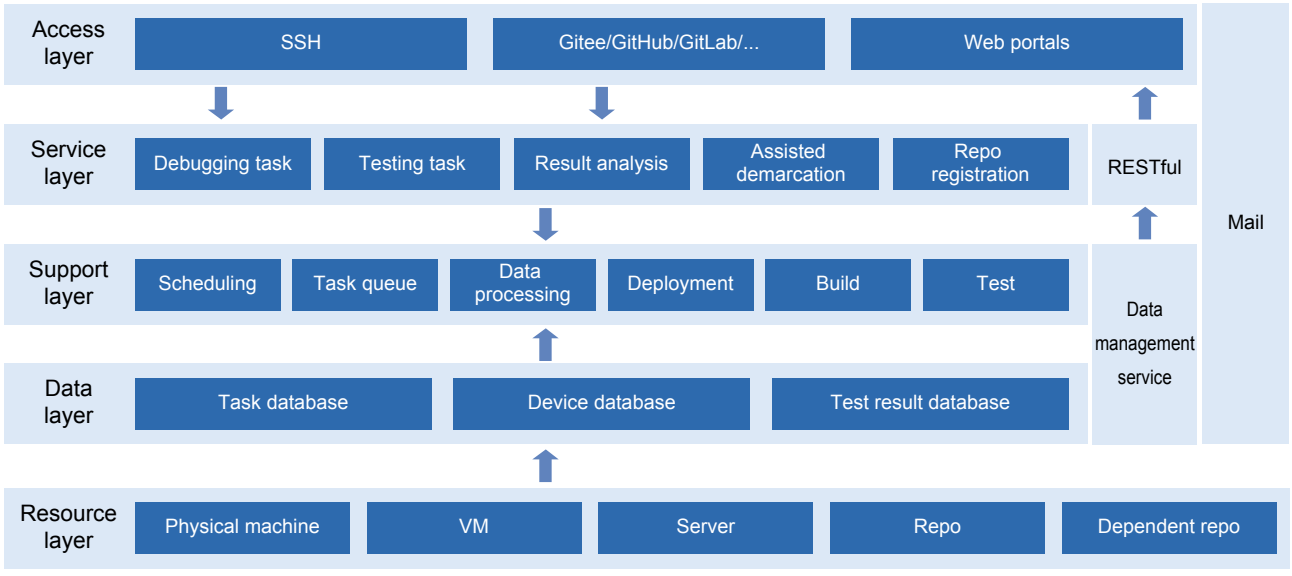
Challenges

The increasing complexity of Linux poses a significant challenge for open source developers due to the limited resources. With the numerous Linux distributions available, developers are looking at ways to quickly introduce, test, and verify the open source software.

Facing diverse application scenarios, limited resources can lead to numerous problems during subsequent use, which require further modifications. Reproducing these issues is challenging, resulting in considerable costs for preparing the environment and making rapid issue localization impractical.

Project Introduction

Compass-CI is a full-stack testing platform that integrates build and test systems with login debugging, test analysis, and error locating services. It actively tests thousands of open source software projects to expose issues related to chips and OSs and automatically locates faults in real-time. Feedback is then sent to software developers, who can perform troubleshooting to ensure software quality. Compass-CI provides an excellent development experience for community developers, and contributes to a thriving open source software ecosystem.



- Testing services: To support local device development, Compass-CI automatically retrieves the code submitted to GitHub for testing and provides feedback on the test results.
- Debugging environment: Supports logging in to the debugging environment via SSH if issues are detected during testing.
- Test result analysis: Analyzes historical test results and provides web and CLI for developers, helping users identify factors that affect the outcomes.
- Assisted locating: Automatically identifies error messages and triggers tests based on Git tree, pinpointing the changes that introduced the problematic modules.

Application Scenarios

Software repository testing & verification: When code, test cases, and tools are submitted on the host platform, Compass-CI automatically retrieves the submitted code for build testing. It also performs automated testing of test cases written in open source software packages and provides feedback on the test results.

On-demand debugging: If a bug is identified during the testing process, on-demand debugging of environment resources is available.

Data analysis and comparison: Compass-CI can be used to monitor system running information (CPU, memory, I/O, network, etc.) and capture snapshots for archiving. It analyzes and compares snapshot data between multiple tests, to assist developers in identifying factors that influence the outcomes from the test results.

Assisted demarcation: If a bug is discovered, the regression mechanism is automatically triggered to identify the initial commit information that introduced the problem.

Repositories

https://gitee.com/openeuler/compass-ci/blob/master/README.en.md

https://gitee.com/compass-ci/lkp-tests

CVE Manager

Server | Cloud | Edge | Embedded

Infrastructure SIG | Security Committee

Vulnerability management integrates processes, tools, and mechanisms of the openEuler community to detect, collect, handle, and disclose security vulnerabilities.

Challenges

There is a demand for timely and effective detection, handling, and disclosure of vulnerabilities of community software packages and distributions, which can directly affect the system security of downstream OSVs and users.

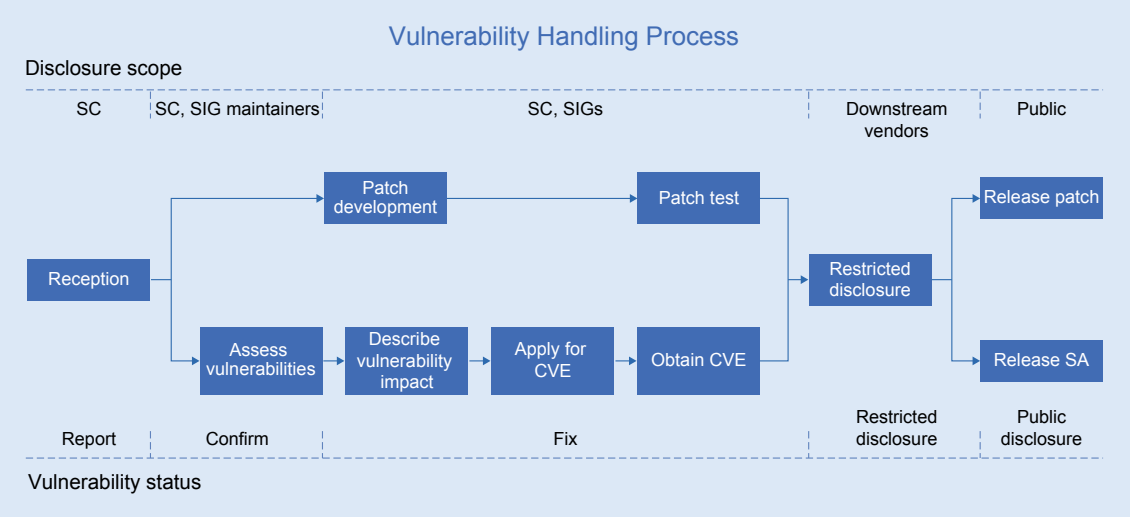
The objectives are as follows:

- Timely detection
- Efficient analysis
- Rapid repair
- Controlled disclosure

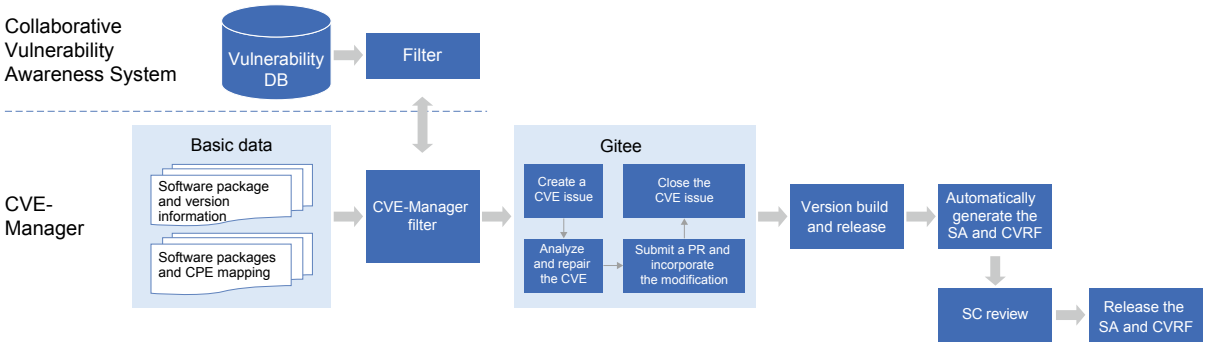
Project Introduction

The openEuler community attaches great importance to the community version security. The openEuler Security Committee (SC) will investigate, analyze, resolve, and disclose security vulnerabilities related to the community. Researchers and industry organizations are encouraged to report the potential vulnerabilities to the SC.

The vulnerability response process is available across the openEuler LTS and its branch versions. See the following flowchart.



The openEuler SC encourages users to report the potential vulnerabilities of openEuler distributions to the openEuler community. Vulnerabilities can be sent to the email of the openEuler SC at openEuler-security@openEuler.org. The SC team will respond within 48 hours and provide updates of the handling progress.



The openEuler community obtains public vulnerability information from the collaborative vulnerability awareness system through the CVE-Manager project, and then creates and maintains records in the software package repository of the corresponding project on Gitee. After the vulnerability is fixed, the general version is built and released with security bulletins.

openEuler uses CVSS v3 to assess vulnerabilities.

For security purposes, the openEuler community will not disclose, discuss, or confirm the security issues of an openEuler distribution until the vulnerability is investigated and resolved and the security bulletin is issued. A security bulletin includes technical details, CVE identifier, CVSS security score, severity level of the vulnerability, and the affected and fixed versions. You can subscribe to security bulletins via email. The community also provides security bulletins in the CVRF format.

Application Scenarios

- Public vulnerabilities of openEuler LTS releases
- Zero-day vulnerabilities of openEuler LTS releases

Repositories

https://www.openeuler.org/en/security/vulnerability-reporting/

https://gitee.com/openeuler/security-committee/blob/master/security-process-en.md

https://gitee.com/openeuler/cve-manager

EUR

The openEuler User Repo (EUR) is a personal software package hosting platform used to upload the third-party packages that have not been introduced to the community or variants of community software packages. Once they are built on EUR, these packages can be conveniently distributed to other users.

Challenges

The openEuler community lacks a robust build-test-distribute platform for third-party packages, covering those still in development, community variant packages, and long-tail packages. For developers who expect the latest Nginx version for the 20.03 LTS release, or adapt their software to community releases, the current community complicates this process.

Project Introduction

The EUR platform enables developers to use SPEC files and source code of various formats to create software packages that are compatible with any openEuler versions. The software packages are automatically signed and a software repository is generated. Developers can access SPEC files via Git/SVN or package software packages from PyPI and RubyGems, and convert them into RPM packages.

Application Scenarios

- Community users with third-party long-tail packages, such as software packages that are no longer available in the community
- Community users with third-party variant packages, such as the latest GCC, Nginx, and GNOME components for earlier versions
- Upstream communities with new third-party software packages, such as Firefox and Chrome Nightly

Repositories

https://github.com/opensourceways/copr_docker/

oepkgs

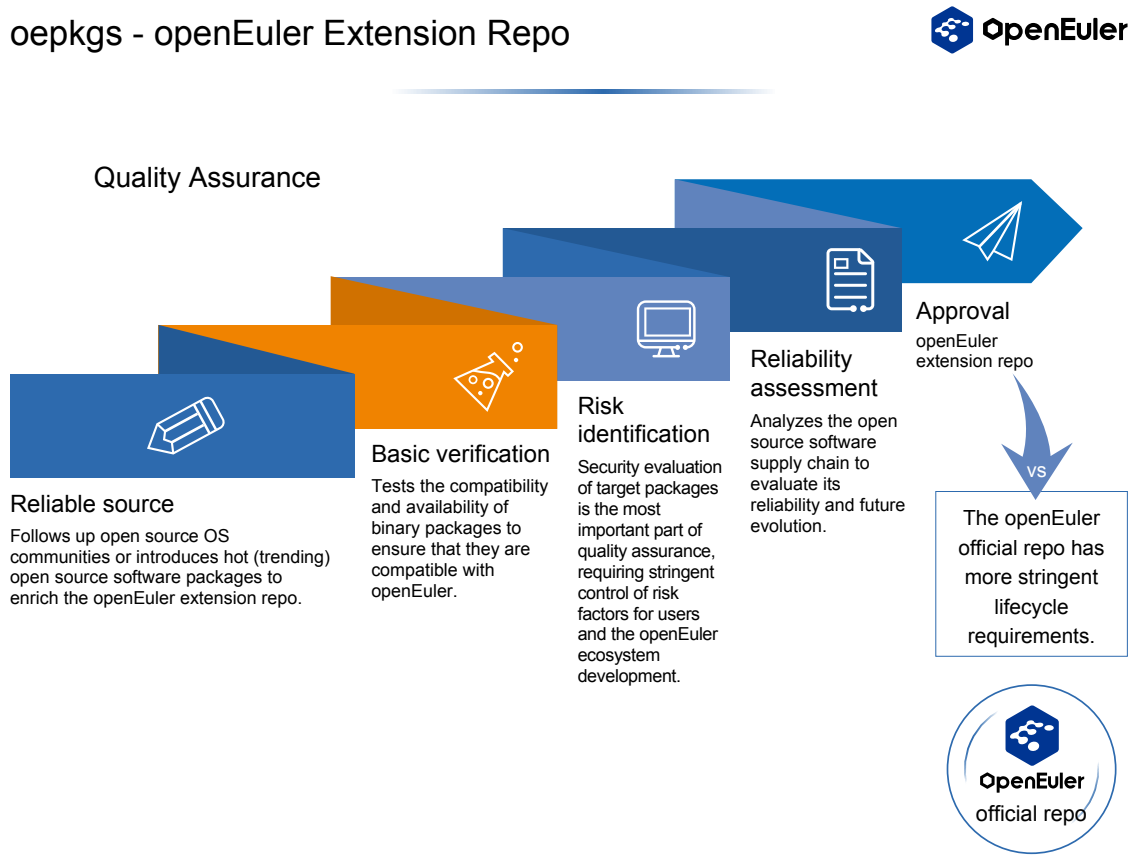
The open external packages service (oepkgs) provides over 30,000 software packages (source code, binary) compatible for the openEuler ecosystem. It provides one-stop software package compatibility, file query, and download, as well as open source software package risk detection services for developers, OSVs, and enterprises who are porting from CentOS and Fedora to openEuler.

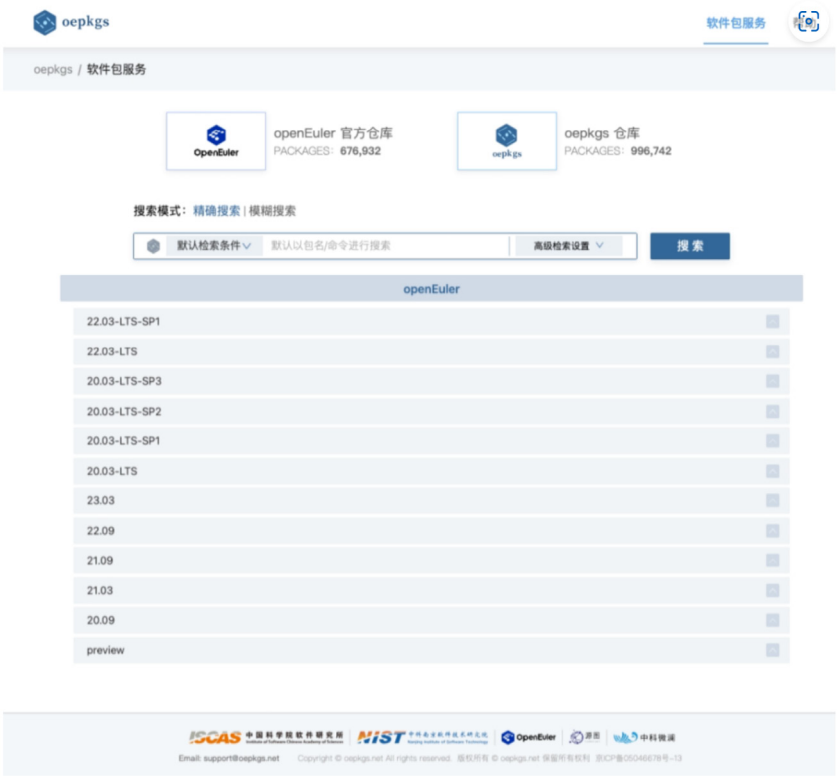
Challenges

- Accurate and fuzzy retrieval of files and software packages
- Access control for software packages in CI/CD, metadata dependency analysis and management, and flexible verification of software packages
- Risk awareness, security, and compliance analysis of open source software

Project Introduction

oepkgs is a collaborative project developed by the Institute of Software at the Chinese Academy of Sciences, the Nanjing Institute of Software Technology, and the openEuler community. To ensure the quality of software repositories and continuous evolution, its extensive packages and mature CI/CD system support source tracing analysis, source code building, binary scanning, basic function verification, vulnerability and compliance risk awareness, and patch and version update. Further, it has excellent capabilities such as RPM software package retrieval, metadata analysis, SBOM and supply chain analysis, and security and compliance risk analysis, providing one-stop access to files, software package queries, risk awareness queries, and download services for incredible UX.





Application Scenarios

- Introduce software packages to convert upstream projects into RPM packages
- Introduce multi-version software packages for porting, to replace conventional Linux OSs
- Query compatibility of and locate software packages on openEuler
- Allow enterprise users to detect open source software risks on a unified platform
- Provide closed-source software distribution channels to enterprise users for direct downloads of software

User Guide
https://docs.openeuler.org/en/docs/22.03_LTS_SP2/docs/oepkgs/overview.html

Repositories

https://search.oepkgs.net/en-US

https://gitee.com/src-oepkgs

openEuler Software Package Contribution Platform

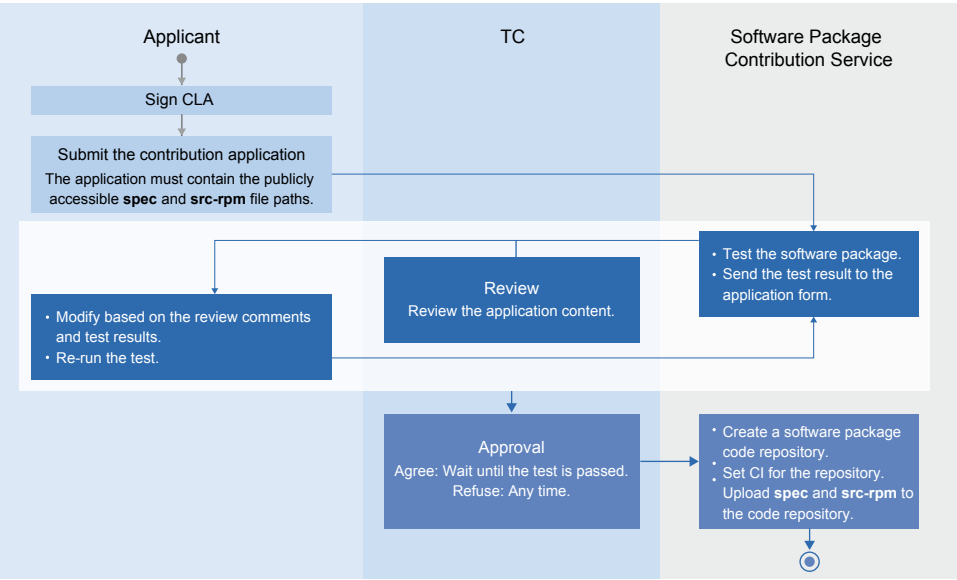
The software package contribution platform is a unified environment for global contributors to contribute software packages to the openEuler community. The platform streamlines the software package contribution pipeline, covering the application, test, approval, and release processes, with full transparency. What's more, the platform supports the release of software package code to Gitee or GitHub, from which contributors all over the world can use in their own environments, feeding into future development and ensuring more software packages will continue to be contributed to the openEuler community.

Challenges

Software packages of the openEuler community are classified into core, extended, and third-party packages. The current workflow to submit software packages is as follows: submitting a PR in the openEuler/community repository, and once the software repository is created, submitting the code of the software package, which after a period of time will be incorporated to the repository. However, the entire process is performed synchronously and can be time-consuming. In addition, contributions can only be made in Gitee, posing restrictions on users outside China.

Project Introduction

This project optimizes the current contribution process for openEuler software packages, streamlining operations for contributors. The following figure shows the optimized contribution process. The test and approval processes are moved forward. After the approval is complete, the subsequent repository creation and code submission will be completed automatically, which changes the workflow from synchronous to asynchronous.



Application Scenarios

A unified platform for developers to contribute software packages to the community.

Repositories

https://software-pkg.openeuler.org/en/package

Signatrust

openEuler Signatrust is a reliable signature service developed by the community infrastructure SIG. It supports key management for OpenPGP and X.509 systems and can seamlessly integrate with various software package forms, including EFI, RPM, KO, and ISO. It enables the signing process for countless software packages, boosting efficiency and enhancing community key management.

Challenges

The existing RPM signature tool experiences a bottleneck when processing a large number of signatures, hindering build efficiency. In addition, the local storage running on pgp-agent has issues with key leakage and poor management usability. As the community grows, there is an increasing need for file signatures, such as for kernel module, EFI, and images. Now, services must support multiple systems and file formats with secure key storage, while ensuring excellent signing and management efficiency.

Project Introduction

Before data is stored, Signatrust encrypts key pairs using systems such as KMS and supports file signature in the TEE or memory. It runs on an asynchronous framework, which enables high-concurrency of signature signing processes. In addition, an independent UI is introduced to facilitate key management.

Application Scenarios

- OpenPGP key pair management in the community versions, to sign files such as RPM/SRPM, ISO, and RepoData
- X.509 key pair management in the community versions, to sign kernel module and EFI files
- Community developers who want to generate their own personal OpenPGP key pairs in EUR, and sign and verify signatures of RPM packages using their personal key pairs

Repositories

https://gitee.com/openeuler/signatrust

EulerLauncher

EulerLauncher is a developer tool kit developed by openEuler's technical operation and infrastructure teams that integrates virtualization technologies, such as LXD, Hyper-V, and virtualization framework, of mainstream desktops. It provides development resources such as VMs and container images for unified provisioning and management, delivering a consistent experience across Windows, macOS, and Linux. It simplifies the building of openEuler development environment on mainstream desktops.

Challenges

Mainstream desktops must ensure convenient and stable resources, such as VMs and containers, to deliver an excellent experience, especially for individuals and university students who have limited development resources. Common VM management platforms have many limitations. For example, a large amount of software needs to be paid. Other issues include VirtualBox, which requires you to download a large ISO image and install an OS, WSL that doesn't provide the openEuler kernel, or other VM management software that typically do not support Apple silicon chips.

Project Introduction

Features

EulerLauncher supports the x86_64 and AArch64 hardware architectures, including Apple silicon chips. It also supports virtual hardware acceleration capabilities for different platforms and provides high-performance development resources. EulerLauncher allows users to use community VMs, daily build images, and other custom images to meet development requirements. Container images will be supported soon.

Application Scenarios

- Build openEuler development environment for Windows, macOS, or Linux desktops.
- Obtain openEuler development tools and simplify the process for submitting dependency configurations.

Repositories

https://gitee.com/openeuler/eulerlauncher

EulerTest

Server | Cloud | Edge | Embedded

QA SIG

The EulerTest management platform carries E2E testing services in the openEuler community. It provides a web-based data middle-end to streamline and make version tests traceable, and provides plugins and automated test services to connect to multiple test engines.

Challenges

The current open source test framework uses automated scripts for version integration tests, however it is largely underused, with a low number of test cases performed in the community.

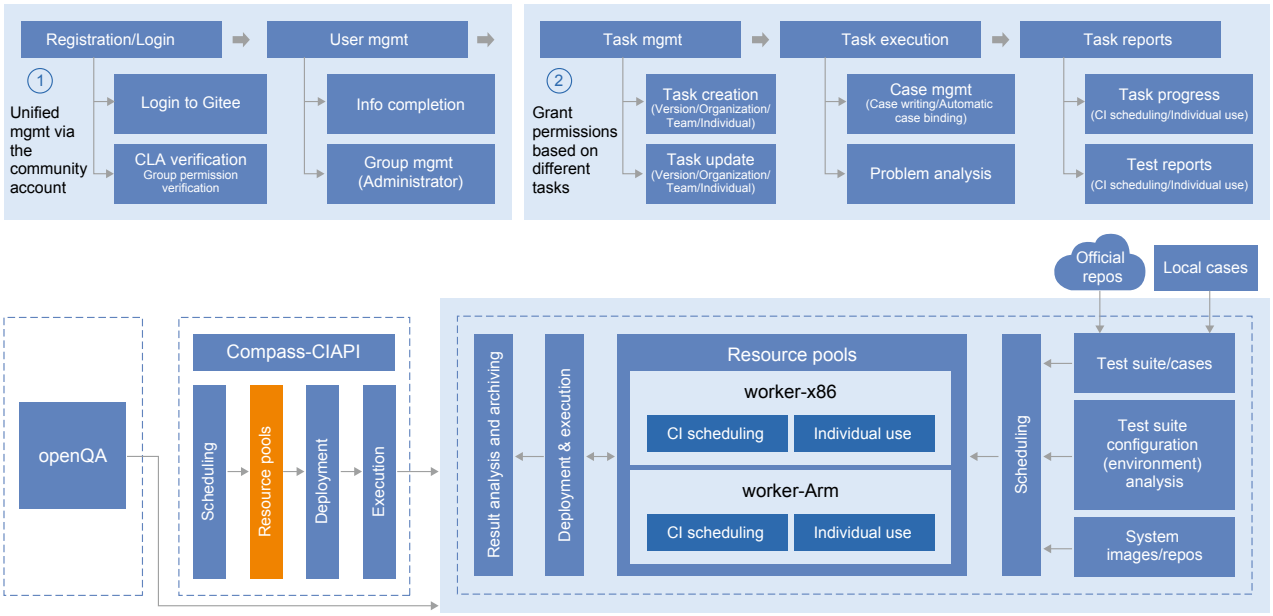
Because there is no unified infrastructure to manage all test activities, assets, or processes, each community team must perform testing using their own means, impacting the reliability of version test results. Furthermore, the community lacks a test hub that can integrate related engineering capabilities for unified scheduling.

Project Introduction

Features

- Static resource management for physical machines, including changing resource keys, releasing occupied resources, and reinstalling the system.
- Dynamic resource management for VMs, providing dynamic configuration of NICs and drives, and a web console.
- Text case management and review and version baseline formulation.
- Delivers trusted tests through product and milestone management, synchronization with openEuler repositories on Gitee, and version quality dashboards.
- Test task management, automated testing, and tracable manual operations; logs can be split, analyzed, and marked by test procedure.
- Automatic reading of test results from platforms such as openQA and Compass-CI, and generating template-based version test reports.

Developer Tool



Application Scenarios

Developer tests

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

Version quality monitoring

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

Repositories

https://gitee.com/openeuler/radiaTest

pkgship

pkgship is a visualized tool used to query the information and dependency tree of RPM packages of openEuler releases or other Linux versions. Developed by the openEuler community, it simplifies the analysis process during software package introduction or upgrade, thereby reducing the time required for analyzing software package dependencies and subsequent O&M.

Challenges

The openEuler OS contains a large number of RPM packages that are connected through complex dependencies. When a package is added, removed, or upgraded, users must first analyze its impact on other RPM packages and its dependencies. To date, the **dnf/yum** command is typically used to query this information, with complex parameters and limited functionality. Furthermore, the corresponding repo file must be loaded in advance, which significantly hinders the developer experience and increases maintenance costs.

Project Introduction

pkgship analyzes the repo file or URL of each release to obtain the basic information and dependencies of software packages, then creates a knowledge map and saves it to Elasticsearch. Basic information includes software package name, version, description, and license, as well as the one-layer or multi-layer compilation dependency information, and that about installations. pkgship provides command lines and RESTful interfaces to query data results within 1 second. The repo configurations can be dynamically expanded, greatly improving analysis efficiency.

Official website: https://pkgmanage.openEuler.org/

Application Scenarios

Query the dependency information of the RPM software packages.

Repositories

https://gitee.com/openeuler/pkgship/blob/master/README.en.md

https://gitee.com/src-openEuler/pkgship

QuickIssue

QuickIssue is a problem tracking system developed by the openEuler infrastructure team to meet community demand for faster submission of issues by category.

Project Introduction

QuickIssue offers following advantages:

- Provides a unified entry for issue submission on the openEuler official website, allowing developers to easily locate the corresponding repository.
- Provides alternative methods for submitting issues to allow developers that don't have a Gitee account to submit issues.
- Provides clear instructions for submitting issues to a repository and a default repository for developers to use.
- Streamlines certain operations on openEuler, including query, search, and filtering.
- Obtains information from other services, such as SIG management and contribution statistics.

QuickIssue provides three main functions: creating an issue, querying an issue, and querying a PR.

Creating an issue

- A unified issue submission process ensures all issues in the openEuler community are submitted through a single entry.
- In the event that the issue creator does not have an account of the code hosting platform, they can still submit their issue by email and a verification code.
- An optimized process for issue creators ensures that users can easily locate the repository or submit the issue to the default repository.

Querying an issue

The QuickIssue service displays all issues in the openEuler community and filters the key information based on search preferences. If you want to search for issues submitted via email, enter the first half of the email address in the **Creator** text box.

Querying a PR

QuickIssue has access to all PR information in the openEuler community. Users can simply enter the filters (status, creator, and labels) and the system will display the relevant results. Furthermore, the system uses cached data, which guarantees a fast query response speed.

Application Scenarios

A portal for community developers to submit issues and search for community issues and PRs.

Repositories

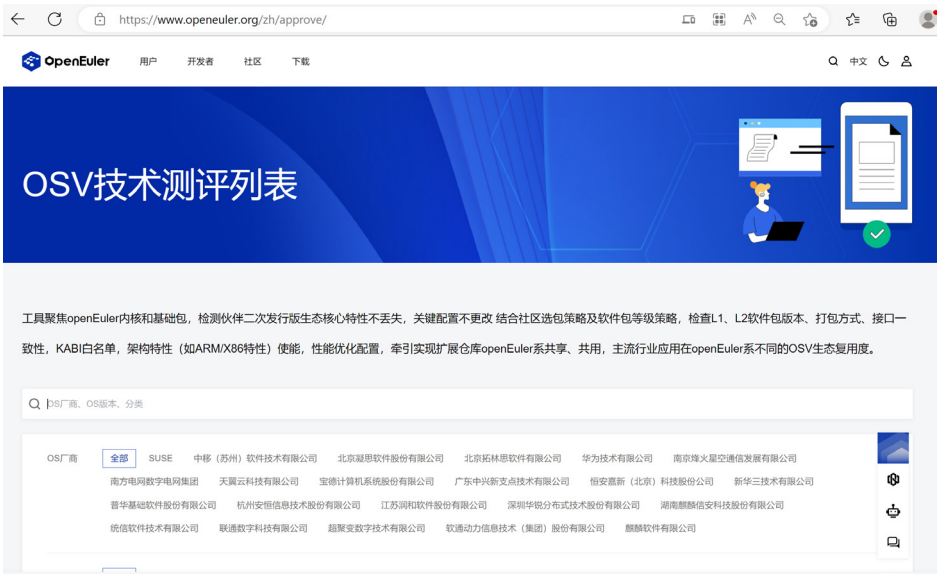
https://github.com/opensourceways/issue_pr_board/blob/main/README.en.md

OSV Technical Assessment

The openEuler OSV technical assessment is a standard set up in coordination with the OpenAtom Foundation, to guide technical assessment on OSVs. Currently, this assessment is being carried out at openEuler Innovation Centers.

Project Introduction

The OSV technical assessment list displays all OS vendors and versions that have been certified with the community's OSV basic assessment standard.



The OSV technical assessment is used to verify the consistency of components like the OS kernel versions and configurations, KABI, software package scope, versions, configurations, services, commands, and files. It ensures the availability of openEuler-based ecosystem software by checking the degree of reuse of repositories like EPOL/oepkgs and runtime consistency.

Application Scenarios

This standard evaluates the consistency of OSVs' technical roadmap with openEuler, to ensure compatibility with community releases and reduce repeated porting and adaptation.

Further, it ensures that OSV distributions remain forward compatible during development.

To accelerate OS porting and iteration, differential databases are created and integrated with the x2openEuler porting tool.

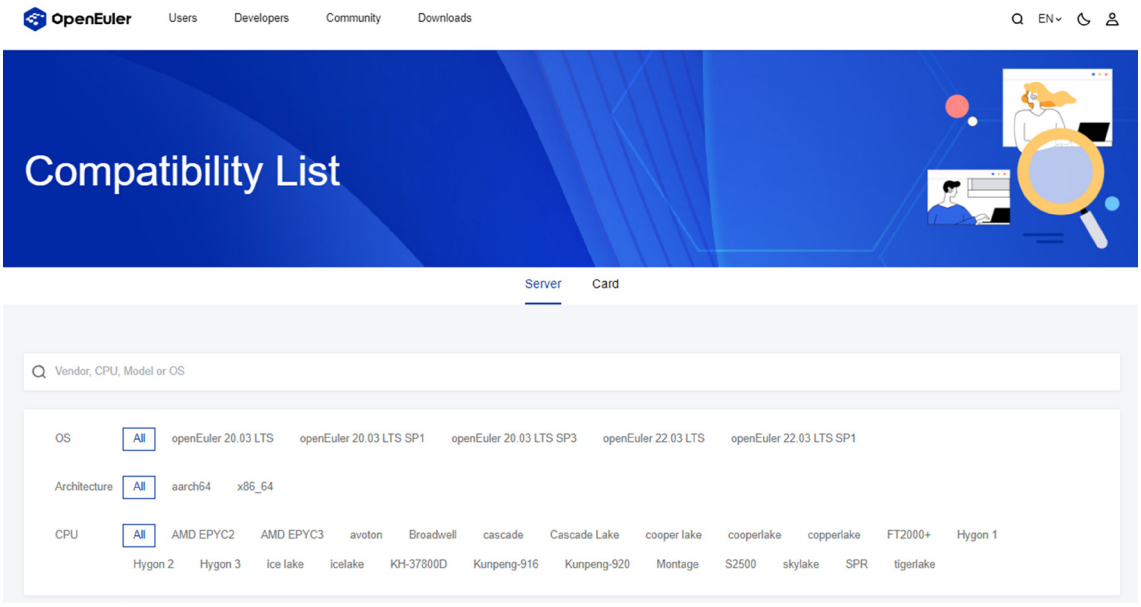
Repositories

https://gitee.com/openeuler/oecp

openEuler Compatibility List

The openEuler Compatibility List provides a platform for users to query servers and cards.

Project Introduction



The openEuler Compatibility List contains information about servers and cards, covering hardware and software.

To guide chip and card vendors to develop independent repository and maintain continuous evolution based on the community infrastructure, openEuler provides a list of standard specifications, processes, and CI/CD processes for hardware, covering CPU architectures and vendors, server models and vendors, and component chip models and vendors. openEuler works with vendors to ensure performance, compatibility, and stability of chips by maintaining their functions, enabling new versions, and adapting to the ecosystem. Companies such as Marvell, NebulaMatrix, and 3SNIC have built repositories in the community and regularly operate with versions. For details about related processes, see:

https://www.openeuler.org/en/compatibility/hardware/

We integrate open source software into openEuler and extended communities in accordance with openEuler package specifications, to meet upstream active industry projects and ensure compatibility with mainstream software. In OS porting and upgrade scenarios, openEuler provides a unified platform that enables users to quickly introduce and acquire software packages of corresponding versions. For details, see:

https://www.openeuler.org/en/compatibility/software/

To query the compatibility of open source software, visit https://search.OEPKGS.net/.

For ISVs, openEuler also provides a testing system to test commercial software, covering testing specifications, processes, solutions, and tool chains. ISVs can conduct testing at the Innovation Centers, with community certificates can be issued for standard releases. For details about related processes, see:

https://certification.openEuler.org/

Application Scenarios

The openEuler Compatibility List allows users to search and query compatibility information for CPU architectures and vendors, server vendors and models, chip models and vendors, and open source and commercial software.

It provides users access to test specifications/schemes, processes, and related tools to ensure releases are compatible with openEuler standards.

Additionally, IHVs, ISVs, and developers can obtain specific processes for adding software and hardware to the compatibility list.

Repositories

https://gitee.com/openeuler/oec-hardware

https://gitee.com/openeuler/oec-application/blob/master/README.en.md

https://gitee.com/openeuler/technical-certification

openEuler Technical Assessment

The openEuler technical assessment is a specification set up in coordination with the OpenAtom Foundation to assess commercial software, hardware, and OSVs. Currently, this assessment is carried out at openEuler Innovation Centers.

Project Introduction

The openEuler technical assessment runs on openEuler OS and aims to develop a unified openEuler ecosystem through unified device assessment tools and detection standards on diversified computing power platforms. For software and hardware, components are tested against community specifications to confirm compatibility with the openEuler OS, whereas for OSs, the assessment tests the consistency with the openEuler technical roadmap.

Typical tests for OSs are plagued by low test efficiency, high costs, and the need for repeated testing due to the lack of an automated platform- and tool-based standard.

The openEuler ecosystem service platform aggregates and streamlines multiple computing resources, and provides a unified environment for resource management, automatic testing, and automatic generation of test reports. Highlights are as follows:

- Unified resource scheduling and E2E automation of environment installation, use case testing, and report generation.
- A full-computing hardware resource platform developed with the openEuler Ecosystem Innovation Center that supports mainstream Kunpeng and x86 architectures, helping slash computing costs for partners in OS porting, adaptation, and assessment.
- Certifications developed with software and server vendors that can certify partners for multiple parties with just one openEuler test, greatly improving the efficiency of building a software partner ecosystem.

Application Scenarios

The compatibility technical assessment defines a unified testing system that helps users verify and find verified solutions, and serves as the basis for a thriving OS technical ecosystem.

Repositories

https://gitee.com/openeuler/technical-certification

Acknowledgment

Every line of code contributed by developers is like a drop of water that merges into the ocean of openEuler community, which has led to the three-year rapid development of the openEuler community. We extend our heartfelt gratitude to the following companies that have contributed to the openEuler community:

Huawei Technologies Co., Ltd.
Kylinsoft Co., Ltd.
UnionTech Software Technology Co., Ltd.
Jiangsu HopeRun Software Co., Ltd.
xFusion Digital Technologies Co., Ltd.
Chinasoft International Technology Services Co., Ltd.
Loongson Technology Co., Ltd.
Hunan Kylinsec Technology Co., Ltd.
TurboLinux (Beijing) Co., Ltd.
Institute of Software, Chinese Academy of Sciences
GBA (Guangdong) National Center of Technology Innovation
SUSE
China Unicom
iSoftStone Information Technology (Group) Co., Ltd.
Intel Asia-Pacific Research And Development Ltd.
iSOFT Infrastructure Software Co. Ltd.
Cloud Computing Branch of China Telecom Co., Ltd.
Beijing XSKY Technology Co., Ltd.
School of Cyber Science and Engineering, HUST
Linaro Limited
China Mobile (Suzhou) Software Technology Co., Ltd
Beijing Netswift Technology Co., Ltd.
Shenzhen Epro Software Co., Ltd.
Beijing Huijun Technology Co., Ltd.
QingCloud Technologies Corp
Sangfor Technologies Inc.
CASIC Network Information Development Co., Ltd.
Shaanxi Gongjin Network Technology Co., Ltd.
H3C Technologies Co., Ltd.
Guangxi Yunyang Software Co., Ltd.
Wuxi Advanced Technology Research Institute
Core Technology Co., Ltd.
Wangsu Science & Technology Co., Ltd.
(In no particular order)

Trademark

All trademarks, product, service, and company names mentioned in this document are the property of their respective owners.

Disclaimer

This document may contain predictive information, including but not limited to information about future finance, operations, product series, and new technologies. There are a number of factors or developments that could cause actual results to differ materially from those expressed or implied in the forward-looking statements. Therefore, the information in this document is for reference only and does not constitute any offer or commitment. openEuler is not liable for any behavior that you make based on this document. openEuler may change the information at any time without notice.