openEuler 24.09 技术白皮书

1. 概述

OpenAtom openEuler（简称“openEuler”） 社区是一个面向数字基础设施操作系统的开源社区。由开放原子开源基金会（以下简称“基金会”）孵化及运营。

openEuler 是一个面向数字基础设施的操作系统，支持服务器、云计算、边缘计算、嵌入式等应用场景，支持多样性计算，致力于提供安全、稳定、易用的操作系统。通过为应用提供确定性保障能力，支持 OT 领域应用及 OT 与 ICT 的融合。

openEuler 社区通过开放的社区形式与全球的开发者共同构建一个开放、多元和架构包容的软件生态体系，孵化支持多种处理器架构、覆盖数字基础设施全场景，推动企业数字基础设施软硬件、应用生态繁荣发展。

2019 年 12 月 31 日，面向多样性计算的操作系统开源社区 openEuler 正式成立。

2020 年 3 月 30 日， openEuler 20.03 LTS （Long Term Support，简写为 LTS，中文为长生命周期支持）版本正式发布，为 Linux 世界带来一个全新的具备独立技术演进能力的 Linux 发行版。

2020 年 9 月 30 日，首个 openEuler 20.09 创新版发布，该版本是 openEuler 社区中的多个企业、团队、独立开发者协同开发的成果，在 openEuler 社区的发展进程中具有里程碑式的意义，也是中国开源历史上的标志性事件。

2021 年 3 月 31 日，发布 openEuler 21.03 内核创新版，该版本将内核升级到 5.10，并在内核方向实现内核热升级、内存分级扩展等多个创新特性，加速提升多核性能，构筑千核运算能力。

2021 年 9 月 30 日，全新 openEuler 21.09 创新版如期而至，这是 openEuler 全新发布后的第一个社区版本，实现了全场景支持。增强服务器和云计算的特性，发布面向云原生的业务混部 CPU 调度算法、容器化操作系统 KubeOS 等关键技术；同时发布边缘和嵌入式版本。

2022 年 3 月 30 日，基于统一的 5.10 内核，发布面向服务器、云计算、边缘计算、嵌入式的全场景 openEuler 22.03 LTS 版本，聚焦算力释放，持续提升资源利用率，打造全场景协同的数字基础设施操作系统。

2022 年 9 月 30 日，发布 openEuler 22.09 创新版本，持续补齐全场景的支持。

2022 年 12 月 30 日，发布 openEuler 22.03 LTS SP1 版本，打造最佳迁移工具实现业务无感迁移，性能持续领先。

2023 年 3 月 30 日，发布 openEuler 23.03 内核创新版本，采用 Linux Kernel 6.1 内核，为未来 openEuler 长生命周期版本采用 6.x 内核提前进行技术探索，方便开发者进行硬件适配、基础技术创新及上层应用创新。

2023 年 6 月 30 日，发布 openEuler 22.03 LTS SP2 版本，场景化竞争力特性增强，性能持续提升。

2023 年 9 月 30 日，发布 openEuler 23.09 创新版本，是基于 6.4 内核的创新版本（参见版本生命周期），提供更多新特性和功能，给开发者和用户带来全新的体验，服务更多的领域和更多的用户。

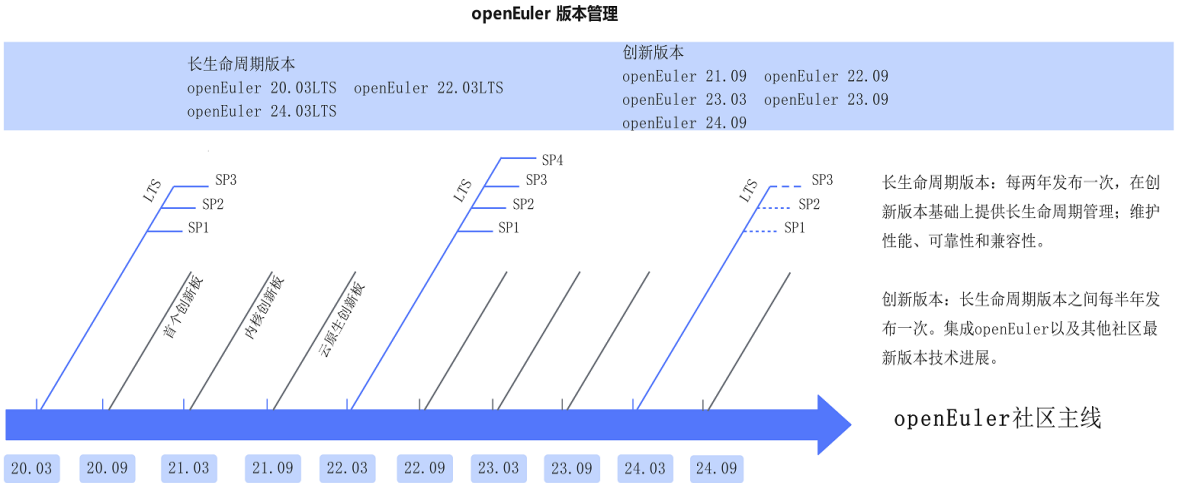
2023 年 11 月 30 日，发布 openEuler 20.03 LTS SP4 版本，其作为 20.03 LTS 版本的增强扩展版本，面向服务器、云原生、边缘计算场景，提供更多新特性和功能增强。

2023 年 12 月 30 日，发布 openEuler 22.03 LTS SP3 版本，是 22.03 LTS 版本增强扩展版本，面向服务器、云原生、边缘计算和嵌入式场景，持续提供更多新特性和功能扩展，给开发者和用户带来全新的体验，服务更多的领域和更多的用户。

2024 年 5 月 30 日，发布 openEuler 24.03 LTS，基于 6.6 内核的长周期 LTS 版本（参见版本生命周期），面向服务器、云、边缘计算、AI 和嵌入式场景，提供更多新特性和功能，给开发者和用户带来全新的体验，服务更多的领域和更多的用户。

2024 年 6 月 30 日，发布 openEuler 22.03 LTS SP4，是 22.03 LTS 版本增强扩展版本，面向服务器、云原生、边缘计算和嵌入式场景，持续提供更多新特性和功能扩展，给开发者和用户带来全新的体验，服务更多的领域和更多的用户。

2024 年 9 月 30 日，发布 openEuler 24.09，基于 6.6 内核的创新版本，提供更多新特性和功能。

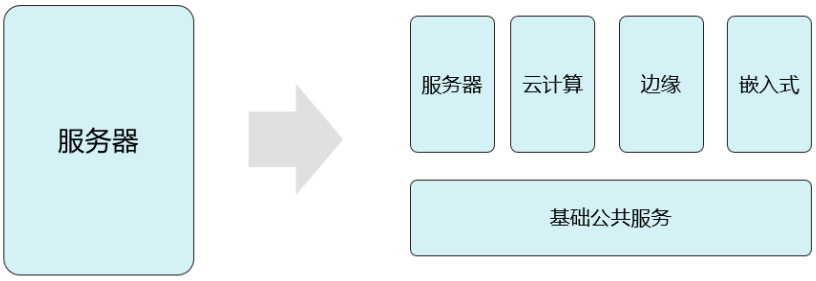


openEuler 作为一个操作系统发行版平台，每两年推出一个 LTS 版本。该版本为企业级用户提供一个安全稳定可靠的操作系统。

openEuler 也是一个技术孵化器。通过每半年发布一个创新版，快速集成 openEuler 以及其他社区的最新技术成果，将社区验证成熟的特性逐步回合到发行版中。这些新特性以单个开源项目的方式存在于社区，方便开发者获得源代码，也方便其他开源社区使用。

社区中的最新技术成果持续合入社区发行版，社区发行版通过用户反馈反哺技术，激发社区创新活力，从而不断孵化新技术。发行版平台和技术孵化器互相促进、互相推动、牵引版本持续演进。

openEuler 覆盖全场景的创新平台



openEuler 已支持 X86、ARM、SW64、RISC-V、LoongArch 多处理器架构及 PowerPC 芯片架构，持续完善多样性算力生态体验。

openEuler 社区面向场景化的 SIG 不断组建，推动 openEuler 应用边界从最初的服务器场景，逐步拓展到云计算、边缘计算、嵌入式等更多场景。openEuler 正成为覆盖数字基础设施全场景的操作系统，新增发布面向边缘计算的版本 openEuler Edge、面向嵌入式的版本 openEuler Embedded。

openEuler 希望与广大生态伙伴、用户、开发者一起，通过联合创新、社区共建，不断增强场景化能力，最终实现统一操作系统支持多设备，应用一次开发覆盖全场景。

openEuler 开放透明的开源软件供应链管理

开源操作系统的构建过程，也是供应链聚合优化的过程。拥有可靠开源软件供应链，是大规模商用操作系统的基础。 openEuler 从用户场景出发，回溯梳理相应的软件依赖关系，理清所有软件包的上游社区地址、源码和上游对应验证。完成构建验证、分发、实现生命周期管理。开源软件的构建、运行依赖关系、上游社区，三者之前形成闭环且完整透明的软件供应链管理。

2. 平台架构

系统框架

openEuler 是覆盖全场景的创新平台，在引领内核创新，夯实云化基座的基础上，面向计算架构互联总线、存储介质发展新趋势，创新分布式、实时加速引擎和基础服务，结合边缘、嵌入式领域竞争力探索，打造全场景协同的面向数字基础设施的开源操作系统。

openEuler 24.09 发布面向服务器、云原生、边缘和嵌入式场景的全场景操作系统版本，统一基于 Linux Kernel 6.6 构建，对外接口遵循 POSIX 标准，具备天然协同基础。同时 openEuler 24.09 版本集成分布式软总线、KubeEdge+边云协同框架等能力，进一步提升数字基础设施协同能力，构建万物互联的基础。

面向未来，社区将持续创新、社区共建、繁荣生态，夯实数字基座。

夯实云化基座

• 容器操作系统 KubeOS：云原生场景，实现 OS 容器化部署、运维，提供与业务容器一致的基于 K8S 的管理体验。

• 安全容器方案：iSulad+shimv2+StratoVirt 安全容器方案，相比传统 Docker+QEMU 方案，底噪和启动时间优化 40%。

• 双平面部署工具 eggo：ARM/X86 双平面混合集群 OS 高效一键式安装，百节点部署时间<15min。

新场景

• 边缘计算：发布面向边缘计算场景的版本，支持 KubeEdge+边云协同框架，具备边云应用统一管理和发放等基础能力。

• 嵌入式： 发布面向嵌入式领域的版本，镜像大小 < 5M，启动时间 < 5s。

• AI 原生 OS：OS 使能 AI 软件栈，开箱即用；异构融合内存，调度，训推场景降本增效；智能化交互平台，赋能开发者及管理员。

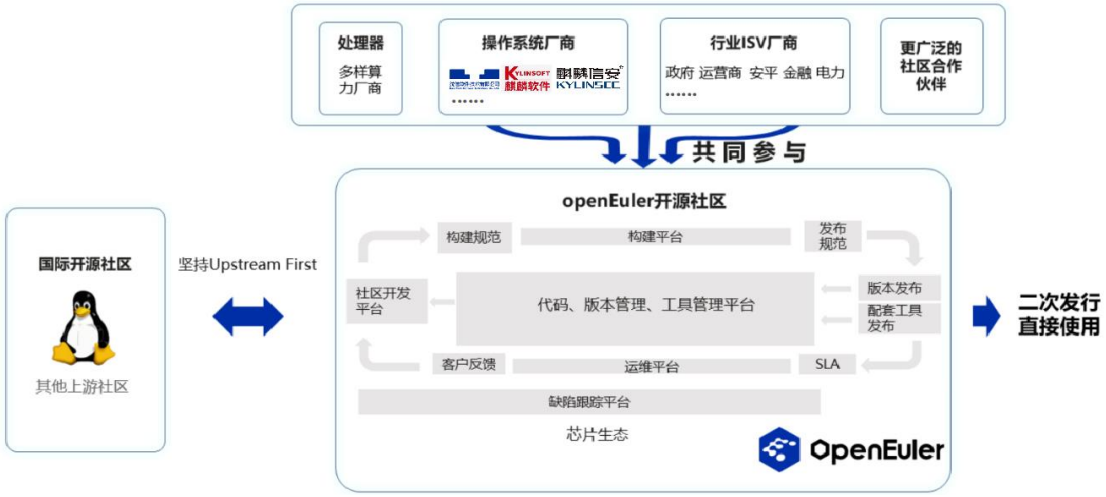
繁荣社区生态

• 友好桌面环境：UKUI、DDE 、Xfce、Kiran-desktop、GNOME 桌面环境，丰富社区桌面环境生态。

• openEuler DevKit：支持操作系统迁移、兼容性评估、简化安全配置 secPaver 等更多开发工具。

平台框架

openEuler 社区与上下游生态建立连接，构建多样性的社区合作伙伴和协作模式，共同推进版本演进。



硬件支持

全版本支持的硬件型号可在兼容性网站查询:
https://www.openeuler.org/zh/compatibility/。

3. 运行环境

服务器

若需要在物理机环境上安装 openEuler 操作系统，则物理机硬件需要满足以下兼容性和最小硬件要求。

硬件兼容支持请查看 openEuler 兼容性列表：https://openeuler.org/zh/compatibility/。

部件名称	最小硬件要求
架构	ARM64、x86_64、riscV
内存	为了获得更好的体验，建议不小于 4GB
硬盘	为了获得更好的体验，建议不小于 20GB

虚拟机

openEuler 安装时，应注意虚拟机的兼容性问题，当前已测试可以兼容的虚拟机及组件如下所示。

1.以 openEuler 24.09 为 HostOS，组件版本如下:

- libvirt-9.10.0-12.oe2409

- libvirt-client-9.10.0-12.oe2409
- libvirt-daemon-9.10.0-12.oe2409
- qemu-8.2.0-17.oe2409
- qemu-img-8.2.0-17.oe2409

2.**兼容的虚拟机列表如下：**

HostOS	GuestOS(虚拟机)	架构
openEuler 24.09	Centos 6	x86_64
openEuler 24.09	Centos 7	aarch64
openEuler 24.09	Centos 7	x86_64
openEuler 24.09	Centos 8	aarch64
openEuler 24.09	Centos 8	x86_64
openEuler 24.09	Windows Server 2016	aarch64
openEuler 24.09	Windows Server 2016	x86_64
openEuler 24.09	Windows Server 2019	aarch64
openEuler 24.09	Windows Server 2019	x86_64

部件名称	最小虚拟化空间要求
架构	ARM64、x86_64
CPU	2 个 CPU
内存	为了获得更好的体验，建议不小于 4GB
硬盘	为了获得更好的体验，建议不小于 20GB

边缘设备

若需要在边缘设备环境上安装 openEuler 操作系统，则边缘设备硬件需要满足以下兼容性和最小硬件要求。

部件名称	最小硬件要求
架构	ARM64、x86_64
内存	为了获得更好的体验，建议不小于 4GB
硬盘	为了获得更好的体验，建议不小于 20GB

嵌入式

若需要在嵌入式环境上安装 openEuler Edge 操作系统，则嵌入式硬件需要满足以下兼容性和最小硬件要求。

部件名称	最小硬件要求
架构	ARM64、ARM32
内存	为了获得更好的体验，建议不小于 512MB
硬盘	为了获得更好的体验，建议不小于 256MB

4. 场景创新

AI

智能时代，操作系统需要面向 AI 不断演进。一方面，在操作系统开发、部署、运维全流程以 AI 加持，让操作系统更智能；另一方面，openEuler 已支持 ARM，x86，RISC-V 等全部主流通用计算架构，在智能时代，openEuler 也率先支持 NVIDIA、昇腾等主流 AI 处理器，成为使能多样性算力的首选。

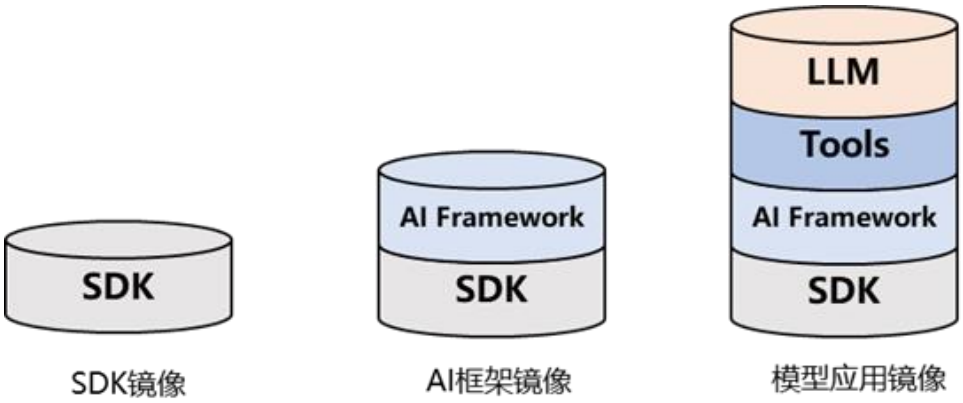
OS for AI

openEuler 兼容 NVIDIA、Ascend 等主流算力平台的软件栈，为用户提供高效的开发运行环境。通过将不同 AI 算力平台的软件栈进行容器化封装，即可简化用户部署过程，提供开箱即用的体验。同时，openEuler 也提供丰富的 AI 框架，方便大家快速在 openEuler 上使用 AI 能力。

功能描述

1. openEuler 已兼容 CANN、CUDA 等硬件 SDK，以及 TensorFlow、PyTorch、MindSpore 等相应的 AI 框架软件，支持 AI 应用在 openEuler 上高效开发与运行。

2. openEuler AI 软件栈容器化封装优化环境部署过程，并面向不同场景提供以下三类容器镜像。



- SDK 镜像：以 openEuler 为基础镜像，安装相应硬件平台的 SDK，如 Ascend 平台的 CANN 或 NVIDIA 的 CUDA 软件。
- AI 框架镜像：以 SDK 镜像为基础，安装 AI 框架软件，如 PyTorch 或 TensorFlow。此外，通过此部分镜像也可快速搭建 AI 分布式场景，如 Ray 等 AI 分布式框架。
- 模型应用镜像：在 AI 框架镜像的基础上，包含完整的工具链和模型应用。

相关使用方式请参考 openEuler AI 容器镜像用户指南。

应用场景

openEuler 使能 AI，向用户提供更多 OS 选择。基于 openEuler 的 AI 容器镜像可以解决开发运行环境部署门槛高的问题，用户根据自身需求选择对应的容器镜像即可一键部署，三类容器镜像的应用场景如下。

- SDK 镜像：提供对应硬件的计算加速工具包和开发环境，用户可进行 Ascend CANN 或 NVIDIA CUDA 等应用的开发和调试。同时，可在该类容器中运行高性能计算任务，例如大规模数据处理、并行计算等。
- AI 框架镜像：用户可直接在该类容器中进行 AI 模型开发、训练及推理等任务。
- 模型应用镜像：已预置完整的 AI 软件栈和特定的模型，用户可根据自身需求选择相应的模型应用镜像来开展模型推理或微调任务。

AI for OS

当前，openEuler 和 AI 深度结合，一方面使用基础大模型，基于大量 openEuler 操作系统的代码和数据，训练出 openEuler Copilot System，初步实现代码辅助生成、智能问题智能分析、系统辅助运维等功能，让 openEuler 更智能。

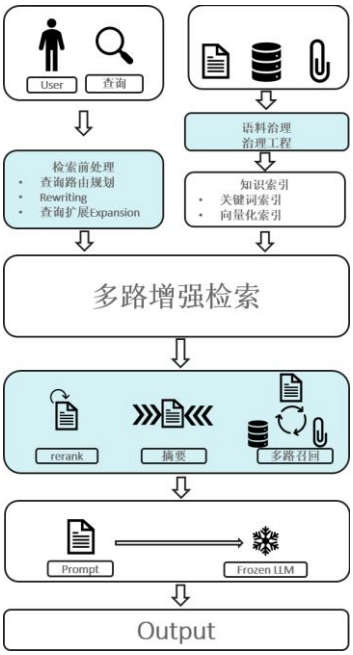
智能问答

功能描述

openEuler Copilot System 智能问答平台目前支持 web 和智能 shell 两个入口。

- Web 入口：操作简单，可咨询操作系统相关基础知识，openEuler 动态数据、openEuler 运维问题解决方案、openEuler 项目介绍与使用指导等等。
- 智能 Shell 入口：自然语言和 openEuler 交互，启发式的运维。

RAG



RAG(检索增强技术)是为了增强大模型长期记忆能力和降低大模型训练成本诞生的技术，相较传统 RAG，openEuler Copilot System 中的 RAG 技术在检索前处理、知识索引和检索增强方面做了改进：

- 检索前处理：对于用户的查询，首先，通过路由规划选择最适合的信息源进行文档

检索，其次，通过拆解用户的复合型查询以增强检索结果的覆盖面，最后，结合用户历史信息对用户当前查询进行改写，增大最终检索结果的命中率。

- 知识索引：对于入库文档，首先，通过语料治理工程从文档中提取片段，其次，对于杂难片段（代码段、流程图经过 ocr 产生的文字内容）构建片段衍生物用于代替片段参与后期检索，最后，基于片段和片段衍生物提取片段的关键字和向量特征，并使用片段特征构对应建数据结构用于检索。
- 多路召回：对于用户被改写过的查询，首先，基于用户查询通过向量化检索、关键字检索和 chat2DB 等能力检索目标片段，接着，基于用户查询对检索结果进行重排。最后，通过摘要和润色等能力聚焦检索结果关键内容、压缩 token 长度和增强自然语言特征（数据库检索结果）。

通过以上能力，相较传统的 RAG 技术，openEuler Copilot System 中的 RAG 技术能更强的适应多种文档格式和内容场景，在不为系统增加较大负担的情况下，增强问答服务体验。

语料治理

语料治理是 openEuler Copilot System 中的 RAG 技术的基础能力之一，其通过片段关系提取、片段衍生物构建和 OCR 等方式将语料以合适形态入库，以增强用户查询命中期望文档的概率：

- 片段关系提取：通过片段关系提取来表征片段和片段之间以及片段和片段衍生物之间的抽象联系，在用户查询命中目标片段之后，通过片段关系搜索强相关片段（上下文等），增强最终检索结果完整性。
- 片段衍生物构建：对于杂难片段（代码等），构建摘要、描述和案例问题（用户可能基于此片段提出的问题）来代替杂难片段参与搜索，可以极大降低杂难片段对于检索过程的干扰，较大增强检索结果的精准性和完整性。
- OCR：对于文档内图片内容，通过图生文的方式将其提取为片段衍生物，用户查询通过检索命中图片的片段衍生来关联原始图片，较大弥补检索结果有文无图的缺陷。

通过以上语料治理手段，可以增强问答服务在多轮对话、内容完整性和图文展示上的体验。

应用场景

- 面向 openEuler 普通用户：深入了解 openEuler 相关知识和动态数据，比如咨询如何迁移到 openEuler。

- 面向 openEuler 开发者：熟悉 openEuler 开发贡献流程、关键特性、相关项目的开发等知识。
- 面向 openEuler 运维人员：熟悉 openEuler 常见或疑难问题的解决思路和方案、openEuler 系统管理知识和相关命令。

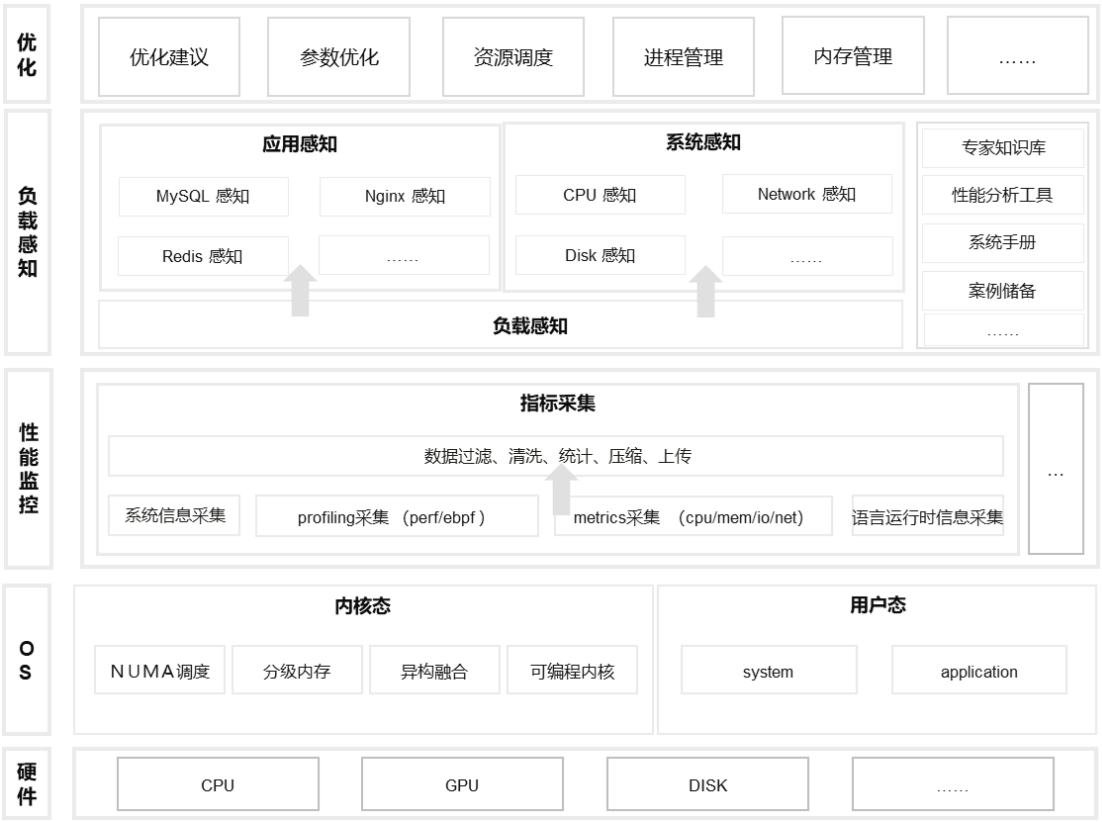
相关使用方式请参考 openEuler Copilot System 智能问答用户指南。

智能调优

功能描述

openEuler Copilot System 智能调优功能目前支持智能 shell 入口。

在上述功能入口，用户可通过与 openEuler Copilot System 进行自然语言交互，完成性能数据采集、系统性能分析、系统性能优化等作业，实现启发式调优。



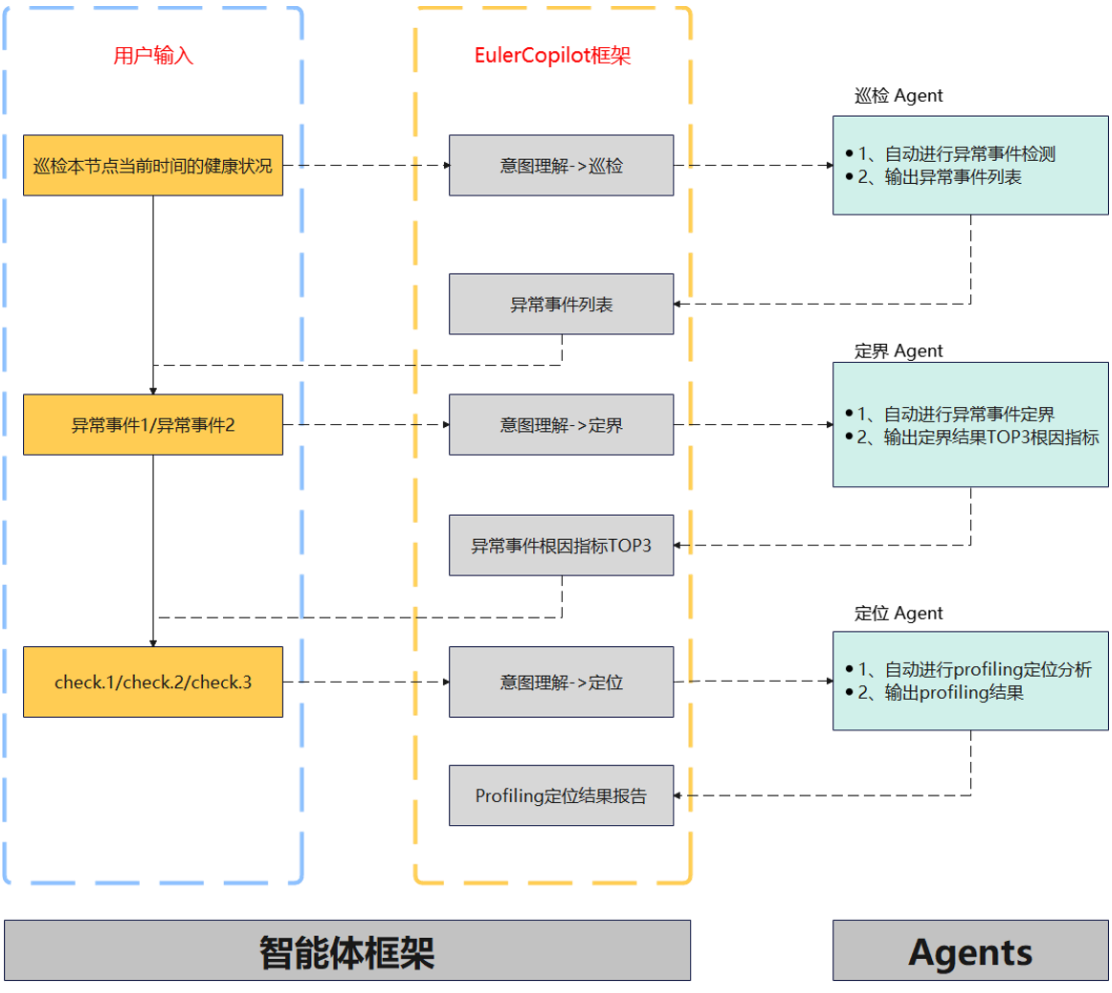
应用场景

- 快速获取系统重要性能指标数据：可快速获取当前系统中 CPU/IO/DISK/NETWORK 等多个重要维度的性能指标以及指定应用的性能指标，帮助用户快速了解系统性能数据。

- 分析系统性能状况：可生成性能分析报告，报告从 CPU/IO/DISK/NETWORK 等多个重要维度分析系统性能状况以及分析指定应用的性能状况，并提示当前系统可能存在的性能瓶颈。
- 推荐系统性能优化建议：可生成一键式执行的性能优化脚本，用户在审核脚本内容后，可执行该脚本，对系统及指定应用的配置进行优化。

智能诊断

功能描述



1. 巡检：调用 Inspection Agent，对指定 IP 进行异常事件检测，为用户提供包含异常容器 ID 以及异常指标（cpu、memory 等）的异常事件列表
2. 定界：调用 Demarcation Agent，对巡检结果中指定异常事件进行定界分析，输出导致该异常事件的根因指标 TOP3
3. 定位：调用 Detection Agent，对定界结果中指定根因指标进行 Profiling 定位分析，为

用户提供该根因指标异常的热点堆栈、热点系统时间、热点性能指标等信息

应用场景

智能诊断接口在本次 openEuler 24.09 版本的功能范围是:具备单机异常事件检测 + 定界 + profiling 定位的能力。

- 其中检测能力指的是：进行单机性能指标采集、性能分析、异常事件检测。
- 其中定界能力指的是：结合异常检测结果进行根因定位，输出 top3 根因指标及其描述。
- 其中 profiling 定位能力指的是：结合 profiling 工具对根因指标进行具体问题模块（代码）定位。

智能部署

功能描述

openEuler Copilot System 目前支持通过自然语言调用环境资源，在本地协助用户基于实际物理资源拉取容器镜像，并且建立适合算力设备调试的开发环境。

当前版本支持三类容器，并且镜像源已同步在 dockerhub 发布，用户可手动拉取运行:

1、 SDK 层：仅封装使能 AI 硬件资源的组件库，例如：cuda、cann 等。

2、 SDK + 训练/推理框架: 在 SDK 层的基础上加装 tensorflow、pytorch 等框架，例如：tensorflow2.15.0-cuda12.2.0、pytorch2.1.0.a1-cann7.0.RC1 等。

3、 SDK + 训练/推理框架 + 大模型：在第 2 类容器上选配几个模型进行封装，例如 llama2-7b、chatglm2-13b 等语言模型。

当前支持的容器镜像汇总:

registry	repository	image_name	tag
docker.io	openeuler	cann	8.0.RC1-oe2203sp4
			cann7.0.RC1.alpha002-oe2203sp2
docker.io	openeuler	oneapi-runtime	2024.2.0-oe2403lts
docker.io	openeuler	oneapi-basekit	2024.2.0-oe2403lts
docker.io	openeuler	llm-server	1.0.0-oe2203sp3
docker.io	openeuler	mlflow	2.11.1-oe2203sp3
			2.13.1-oe2203sp3
docker.io	openeuler	llm	chatglm2_6b-pytorch2.1.0.a1-cann7.0.RC1.alpha002-

			oe2203sp2
			llama2-7b-q8_0-oe2203sp2
			chatglm2-6b-q8_0-oe2203sp2
			fastchat-pytorch2.1.0.a1-cann7.0.RC1.alpha002-oe2203sp2
docker.io	openeuler	tensorflow	tensorflow2.15.0-oe2203sp2
			tensorflow2.15.0-cuda12.2.0-devel-cudnn8.9.5.30-oe2203sp2
docker.io	openeuler	pytorch	pytorch2.1.0-oe2203sp2
			pytorch2.1.0-cuda12.2.0-devel-cudnn8.9.5.30-oe2203sp2
			pytorch2.1.0.a1-cann7.0.RC1.alpha002-oe2203sp2
docker.io	openeuler	cuda	cuda12.2.0-devel-cudnn8.9.5.30-oe2203sp2

应用场景

- 面向 openEuler 普通用户：简化深度学习开发环境构建流程，节省物理资源的调用前提，比如实现在 openEuler 系统上搭建昇腾计算的开发环境。
- 面向 openEuler 开发者：熟悉 openEulerAI 软件栈，减少组件配套试错成本。

openEuler Embedded

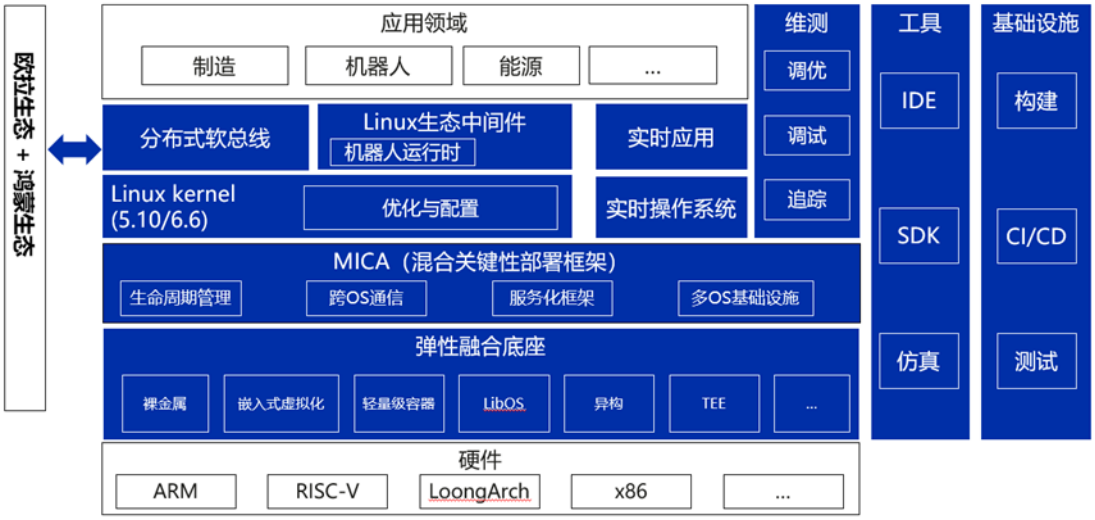
openEuler 发布面向嵌入式领域的版本 openEuler 24.09，构建了一个相对完整的综合嵌入系统软件平台，在南北向生态、关键技术特性、基础设施、落地场景等方面都有显著的进步。

openEuler Embedded 围绕以制造、机器人为代表的 OT 领域持续深耕，通过行业项目垂直打通，不断完善和丰富嵌入式系统软件栈和生态。在内核方面，结合嵌入式场景的实际需求，继续保留双内核版本支持，支持 5.10 和 6.6 双内核，用户可以结合 BSP、应用场景等实际需求在两个版本的内核中选择其一，同时开发了自定义内核的能力。嵌入式弹性底座支持多种解决方案，包括 Jailhouse 分区虚拟化方案、openAMP 裸金属混合部署方案、基于 ZVM 和 Rust-Shyper 的实时虚拟化部署方案，用户可以根据自己的使用场景选择最优的部

署方案。在嵌入式弹性底座之上打造了混合关键性部署框架 MICA，对下屏蔽不同底座的差异，对上为不同运行时提供统一的接口。在北向，目前已经支持 600+软件包，包括支持 ROS humble 版本，集成 ros-core、ros-base、SLAM 等核心软件包，满足 ROS2 运行时要求，针对嵌入式上层用户开发 SDK，加入了 ROS2 的嵌入式特色能力，SDK 支持 ROS2 colcon 交叉编译；支持 BMC 生态，已初步支持 openBMC。在南向，新增飞腾、海思、瑞萨、德州仪器、全志等硬件的支持，特别是提出了面向开发者的硬件开发板概念“欧拉派/Euler Pi”，并具体推出了第一款 openEuler Embedded 原生开发板“海鸥派/HiEuler Pi”。在基础设施，正式发布 openEuler Embedded 元工具 oebuild，构建系统升级到 Yocto 4.0 LTS，工具链新增支持 LLVM 工具链，可以采用 LLVM 工具链来构建镜像，并生成对应的 SDK，相对 GCC 工具链，可以获得在性能、体积、安全性等诸多方面的改进。在落地场景方面，openEuler Embedded 已经有了多个商业发行版/自用版，在 BMC，工业控制器，机器人控制器等领域开始应用。

未来 openEuler Embedded 将协同 openEuler 社区生态伙伴、用户、开发者，逐步扩展支持龙芯等新的芯片架构和更多的南向硬件，完善工业中间件、嵌入式 AI、嵌入式边缘、仿真系统等能力，打造综合嵌入式系统软件平台解决方案。

系统架构图



南向生态

openEuler Embedded Linux 当前主要支持 ARM64、x86-64、ARM32、RISC-V 等多种芯片架构，未来计划支持龙芯等架构，从 24.03 版本开始，南向支持大幅改善，已经支持树莓派、海思、瑞芯微、瑞萨、德州仪器、飞腾、赛昉、全志等厂商的芯片。

嵌入式弹性虚拟化底座

openEuler Embedded 的弹性虚拟化底座是为了在多核片上系统（SoC, System On Chip）上实现多个操作系统共同运行的一系列技术的集合，包含了裸金属、嵌入式虚拟化、轻量级容器、LibOS、可信执行环境（TEE）、异构部署等多种实现形态。不同的形态有各自的特点：

1. 裸金属：基于 openAMP 实现裸金属混合部署方案，支持外设分区管理，性能最好，但隔离性和灵活性较差。目前支持 UniProton/Zephyr/RT-Thread 和 openEuler Embedded Linux 混合部署。
2. 分区虚拟化：基于 Jailhouse 实现工业级硬件分区虚拟化方案，性能和隔离性较好，但灵活性较差。目前支持 UniProton/Zephyr/FreeRTOS 和 openEuler Embedded Linux 混合部署，也支持 openHarmony 和 openEuler Embedded Linux 的混合部署。
3. 实时虚拟化：openEuler 社区孵化了嵌入实时虚拟机监控器 ZVM 和基于 rust 语言的 Type-I 型嵌入式虚拟机监控器 Rust-Shyper，可以满足不同场景的需求。

混合关键性部署框架

openEuler Embedded 打造了构建在融合弹性底座之上混合关键性部署框架，并命名为 MICA（MIxed CriticAlity），旨在通过一套统一的框架屏蔽下层弹性底座形态的不同，从而实现 Linux 和其他 OS 运行时便捷地混合部署。依托硬件上的多核能力使得通用的 Linux 和专用的实时操作系统有效互补，从而达到全系统兼具两者的特点，并能够灵活开发、灵活部署。

MICA 的组成主要有四大部分：生命周期管理、跨 OS 通信、服务化框架和多 OS 基础设施。生命周期管理主要负责从 OS（Client OS）的加载、启动、暂停、结束等工作；跨

OS 通信为不同 OS 之间提供一套基于共享内存的高效通信机制；服务化框架是在跨 OS 通信基础之上便于不同 OS 提供各自擅长服务的框架，例如 Linux 提供通用的文件系统、网络服务，实时操作系统提供实时控制、实时计算等服务；多 OS 基础设施是从工程角度为把不同 OS 从工程上有机融合在一起的一系列机制，包括资源表达与分配，统一构建等功能。

混合关键性部署框架当前能力：

1. 支持裸金属模式下 openEuler Embedded Linux 和 RTOS（Zephyr/UniProton）的生命周期管理、跨 OS 通信。
2. 支持分区虚拟化模式下 openEuler Embedded Linux 和 RTOS（FreeRTOS）的生命周期管理、跨 OS 通信。

北向生态

1. 北向软件包支持：600+嵌入式领域常用软件包的构建。
2. 软实时内核：提供软实时能力，软实时中断响应时延微秒级。
3. 分布式软总线基础能力：集成 OpenHarmony 的分布式软总线和 hichain 点对点认证模块，实现欧拉嵌入式设备之间互联互通、欧拉嵌入式设备和 OpenHarmony 设备之间互联互通。
4. 嵌入式容器与边缘：支持 iSula 容器，可以实现在嵌入式上部署 openEuler 或其他操作系统容器，简化应用移植和部署。支持生成嵌入式容器镜像，最小大小可到 5MB，可以部署在其他支持容器的操作系统之上。支持 Kubeedge，可以更好地实现“云-边-端”协同。

UniProton 硬实时系统

UniProton 是一款实时操作系统，具备极致的低时延和灵活的混合关键性部署特性，可以适用于工业控制场景，既支持微控制器 MCU，也支持算力强的多核 CPU。目前关键能力如下：

- 支持 Cortex-M、ARM64、X86_64、riscv64 架构，支持 M4、RK3568、RK3588、X86_64、Hi3093、树莓派 4B、鲲鹏 920、昇腾 310、全志 D1s。

- 支持树莓派 4B、Hi3093、RK3588、X86_64 设备上通过裸金属模式和 openEuler Embedded Linux 混合部署。
- 支持通过 gdb 在 openEuler Embedded Linux 侧远程调试。
- 支持 890+ POSIX 接口，支持文件系统、设备管理、shell 控制台、网络。

应用场景

openEuler Embedded 可广泛应用于工业控制、机器人控制、电力控制、航空航天、汽车及医疗等领域。

DevStation 开发者工作站支持

DevStation 是一款 openEuler 专为开发者打造的 Linux 桌面发行版，旨在简化软件开发和部署流程。它预装了广泛的开发工具和 IDE，支持多语言编程，适合前后端、全栈开发者使用。DevStation 集成了容器技术（如 Docker、Isula），帮助开发者快速搭建测试环境。针对 AI 开发，它内置了 TensorFlow 和 PyTorch 等框架，并优化了对硬件加速器的支持。DevStation 提供图形化编程环境、调试和自动化测试工具，为开发者提供从初学到高级项目的全方位支持，是一款高效、便捷的开发利器。

功能描述

开发者友好的集成环境：发行版预装了广泛的开发工具和 IDE，如 VS Code 系列等。支持多种编程语言，满足从前端、后端到全栈开发的需求。

软件包管理与自动部署：提供简单便捷的软件包管理工具，支持通过一键安装和更新多种开发环境。同时，内置 Docker、Isula 等容器技术，方便开发者进行应用容器化与自动化

部署，提供新型包管理体系 EPKG，支持多版本部署，大大降低开发者在安装不同开发工具时的使用门槛。

图形化编程环境：集成了图形化编程工具，降低了新手的编程门槛，同时也为高级开发者提供了可视化编程的强大功能。

AI 开发支持：针对 AI 领域的开发者，预装了 TensorFlow、PyTorch 等机器学习框架，同时优化了硬件加速器（如 GPU、NPU）的支持，提供完整的 AI 模型开发与训练环境。同时，openEuler Devstation 集成了 openEuler Copilot System，提供 AI 助手服务，帮助用户解决大部分操作系统使用问题。

调试与测试工具：内置 GDB、CUnit、gtest、perf 等调试工具和测试、调优工具，帮助开发者快速调试和自动化测试，提升开发效率。

版本控制和协作：集成 Git、SVN 等版本控制工具，并支持多种远程协作工具，如 Slack、Mattermost 和 GitLab，使得团队开发和远程协作更加顺畅。

安全与合规检查：提供安全扫描和代码合规性检查工具，帮助开发者在开发阶段就能发现并修复潜在的安全漏洞和代码问题。

应用场景

多语言开发环境：适用于需要同时开发多语言（如 Python、JavaScript、Java、C++）项目的开发者，无需手动配置环境，系统预装各种编译器、解释器和构建工具。

快速部署与测试：通过内置的容器化支持（Docker、Isula）和虚拟化平台，开发者可以快速构建、测试和部署应用，在本地进行多环境的自动化测试，未来 Devstation 会集成 openEuler 部署工具，为用户提供良好的开发环境安装体验。

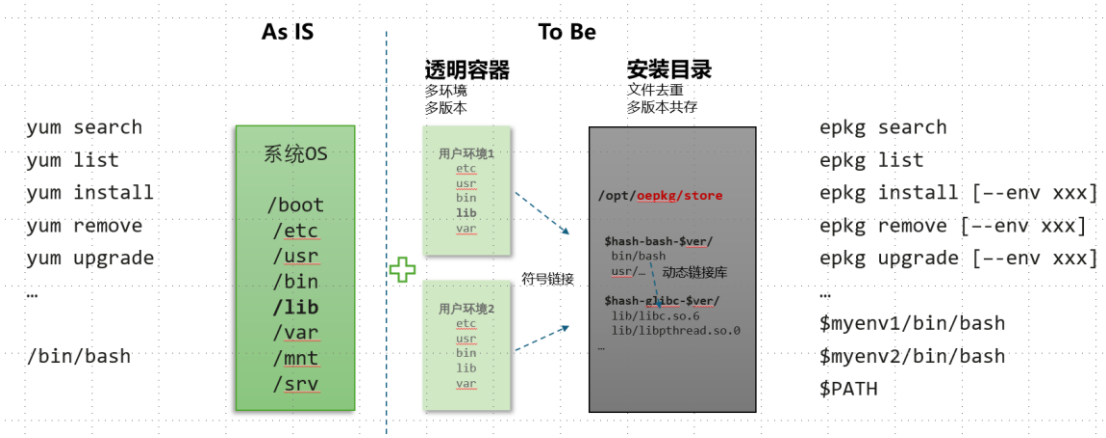
AI 开发和数据科学：为从事人工智能、机器学习、深度学习的开发者准备了完备的开发环境，预装常用数据处理库和模型训练工具，支持异构计算加速，适合进行模型训练和推理。

游戏与多媒体开发：为游戏开发者提供了图形化开发工具，同时也预装了多媒体编辑软件，支持图像、音频、视频处理的全栈开发。

学生和编程初学者：通过图形化编程工具和内置的开发教程，系统也适合计算机科学入门者使用，降低学习编程的门槛，Devstation 自带的 openEuler Copilot System，提供 Linux 操作系统智能问答服务，解决初学者，开发者在操作系统上遇到的问题。

epkg 新型软件包

epkg 是一款新型软件包，支持普通用户在操作系统中安装及使用。新的软件包格式相比现有软件包，主要解决多版本兼容性问题，用户可以在一个操作系统上通过简单地命令行安装不同版本的软件包。同时支持环境环境实现环境的创建/切换/使能等操作，来使用不同版本的软件包。目前 epkg 主要支持非服务类的软件包的安装和使用。



功能描述

多版本兼容：支持普通用户安装，支持安装不同版本的软件包，不同版本的同一软件包安装不冲突。使能用户在同一个节点上，快速安装同一软件包的不同版本，实现多版本软件包的共存。

环境管理：支持环境环境实现环境的创建/切换/使能等操作，用户通过环境的切换，在环境中使用不同的 channel，实现在不同的环境中使用不同版本的软件包。用户可以基于环境，快速实现软件包版本的切换。

普通用户安装：epkg 支持普通用户安装软件包，普通用户能够自行创建环境，对个人用户下的环境镜像管理，无需特权版本。降低软件包安装引起的安全问题。

应用场景

适用于用户希望安装软件包的多个版本的场景，用户能够通过切换环境使用不同的版本的软件包，解决兼容性难题。

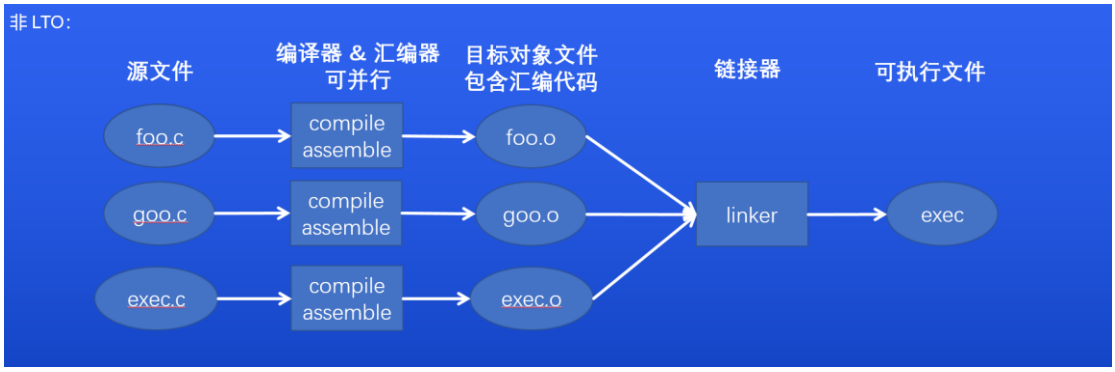
使用文档参考: epkg 使用文档。

LTO for openEuler

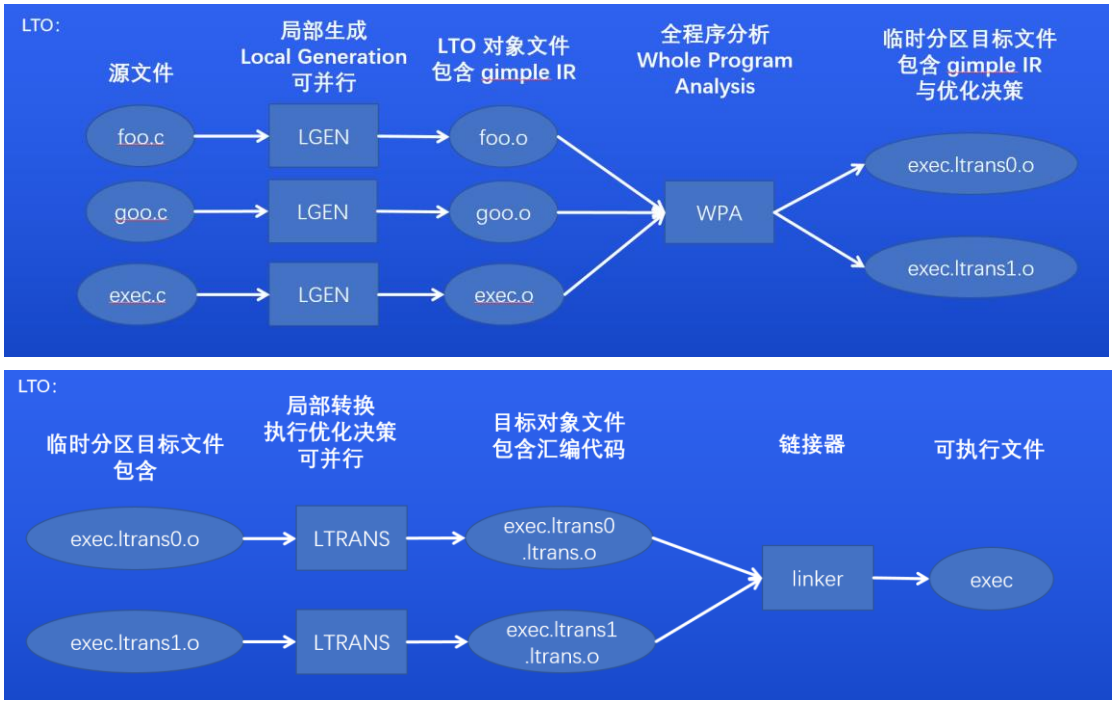
LTO（Link-Time Optimization）全称链接时优化，是一项将编译优化延后到链接时进行的编译技术。该技术由于拥有链接时的全程序视角，相比传统编译流程中的单文件编译优化，往往可以提供更多的优化机会，做出更激进的优化决策，带来更强的性能与更小的二进制体积。

功能描述

在传统编译流程中，gcc 将单个源文件（称谓一个编译单元）直接进行编译优化生成包含汇编代码的`.o`目标对象文件，并由链接器对这些`.o`文件进行符号表解析与重定位，链接成可执行文件。在这个过程中，拥有跨文件函数调用信息的链接器由于操作的是汇编代码，难以进行编译优化，而可以执行编译优化的环节，却没有跨文件的全局信息。这样的编译框架，虽然提高了编译效率，每次重新编译只需要编译修改过的少量编译单元，但也丢失了许多跨文件的优化机会。



LTO 设计的初衷是希望能够在链接时，即拥有跨编译单元的调用信息时，进行编译优化，提供更多的优化机会。为了达到这个目的，LTO 需要将编译优化所需的 IR 信息保留到链接时。在链接时，链接器会调用 LTO 插件，执行全程序分析，生成更加有效的优化决策，再经由编译优化生成更高效的 IR，进一步转成包含汇编代码的目标对象文件，最后由链接器完成常规的链接工作。



应用场景

openEuler 24.09 创新版本通过修改 openEuler-rpm-config 中的默认编译选项，在白名单机制下，在版本构建时为 532 个应用使能了链接时优化，削减生成的链接产物（可执行文件与动态库）体积约 300MB，占这 532 个应用的链接产物总体积的 14%。

使能 LTO 的应用列表可以在 LTO 补丁中找到。

GCC 14 多版本编译工具链支持

openEuler GCC Toolset 14 编译工具链，为用户提供了更加灵活且高效的编译环境选择。用户可以轻松地在不同版本的 GCC 之间进行切换，以便充分利用新硬件特性，同时享受到 GCC 最新优化所带来的性能提升。

功能描述

为了使能多样算例新特性，满足不同用户对不同硬件特性支持的需求，在 openEuler 24.09 版本推出 openEuler GCC Toolset 14 编译工具链，该工具链提供一个高于系统主 GCC 版本的副版本 GCC 14 编译工具链，为用户提供了更加灵活且高效的编译环境选择。通过使用 openEuler GCC Toolset 14 副版本编译工具链，用户可以轻松地在不同版本的 GCC 之间进行切换，以便充分利用新硬件特性，同时享受到 GCC 最新优化所带来的性能提升。

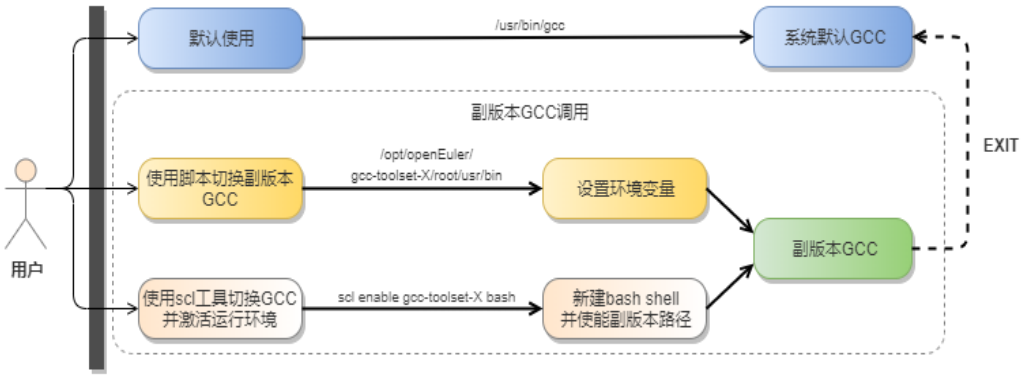
为了与系统默认主版本 GCC 解耦，防止副版本 GCC 安装与主版本 GCC 安装的依赖库产生冲突，openEuler GCC Toolset 14 工具链的软件包名均以前缀“gcc-toolset-14-”开头，后接原有 GCC 软件包名。

此外，为便于版本切换与管理，本方案引入 SCL 版本切换工具。SCL 工具的核心就是会在/opt/openEuler/gcc-toolset-14 路径下提供一个 enable 脚本，通过注册将 gcc-toolset-14 的环境变量注册到 SCL 工具中，从而可以使用 SCL 工具启动一个新的 bash shell，此 bash shell 中的环境变量即为 enable 脚本中设置的副版本环境变量，从而实现主副版本 GCC 工具链的便捷切换。

应用场景

主版本场景：正常编译使用系统默认的 gcc-12.3.1。

副版本场景：需要使用 GCC 14 高版本特性构建相关应用，使用 SCL 工具将构建环境切换为 gcc-toolset-14 编译工具链的编译环境。



GCC 主副版本切换示意图

5. 内核创新

openEuler 内核中的新特性

openEuler 24.09 基于 Linux Kernel 6.6 内核构建，在此基础上，同时吸收了社区高版本的有益特性及社区创新特性。

- 内存管理 folio 特性：Linux 内存管理基于 page（页）转换到由 folio（拉丁语 foliō，对开本）进行管理，相比 page，folio 可以由一个或多个 page 组成，采用 struct folio 参数的函数声明它将对整个（1 个或者多个）页面进行操作，而不仅仅是 PAGE_SIZE

字节，从而移除不必要复合页转换，降低误用 tail page 问题；从内存管理效率上采用 folio 减少 LRU 链表数量，提升内存回收效率，另一方，一次分配更多连续内存减少 page fault 次数，一定程度降低内存碎片化；而在 IO 方面，可以加速大 IO 的读写效率，提升吞吐。全量支持匿名页、文件页的 large folio，提供系统级别的开关控制，业务可以按需使用。对于 ARM64 架构，基于硬件 contiguous bit 技术（16 个连续 PTE 只占一个 TLB entry），可以进一步降低系统 TLB miss，从而提升整体系统性能。24.09 版本新增支持 anonymous shmem 分配 mTHP 与支持 mTHP 的 lazyfree，进一步增加内存子系统对于 large folio 的支持；新增 page cache 分配 mTHP 的 sysfs 控制接口，提供系统级别的开关控制，业务可以按需使用。

- MPTCP 特性支持：MPTCP 协议诞生旨在突破传统 TCP 协议的单一路径传输瓶颈，允许应用程序使用多个网络路径进行并行数据传输。这一设计优化了网络硬件资源的利用效率，通过智能地将流量分配至不同传输路径，显著缓解了网络拥塞问题，从而提高数据传输的可靠性和吞吐量。

  目前，MPTCP 在下述网络场景中已经展现出了其优秀的性能：

  1) 网络通路的选择：在现有的网络通路中，根据延迟、带宽等指标评估，选择最优的通路。
  2) 无缝切网：在不同类型网络之间切换时，数据传输不中断。
  3) 数据分流：同时使用多个通道传输，对数据包进行分发实现并发传输，增加网络带宽。

在实验环境中，采用 MPTCP v1 技术的 RSYNC 文件传输工具展现出了令人满意的效率提升。具体而言，传输 1.3GB 大小的文件时，传输时间由原来的 114.83 s 缩短至仅 14.35s，平均传输速度由原来的 11.08 MB/s 提升至 88.25 MB/s，可以极大程度的缩减文件传输时间。同时，实验模拟了传输过程中一条或多条路径突发故障而断开的场景，MPTCP 在此种场景下可以将数据无缝切换至其他可用的数据通道，确保数据传输的连续性与完整性。

在 openEuler 24.09 中，已经完成了对 linux 主线内核 6.9 中 MPTCP 相关特性的全面移植与功能优化。

- ext4 文件系统支持 Large folio：iozone 性能总分可以提升 80%，iomap 框架回写流程支持批量映射 block。支持 ext4 默认模式下批量申请 block，大幅优化各类 benchmark 下 ext4 性能表现（华为贡献）。ext4 buffer io 读写流程以及 pagecache 回

写流程弃用老旧的 buffer_head 框架，切换至 iomap 框架，并通过 iomap 框架实现 ext4 支持 large folio。24.09 版本新增对于 block size < folio size 场景的小 buffered IO（<=4KB）的性能优化，性能提升 20%。

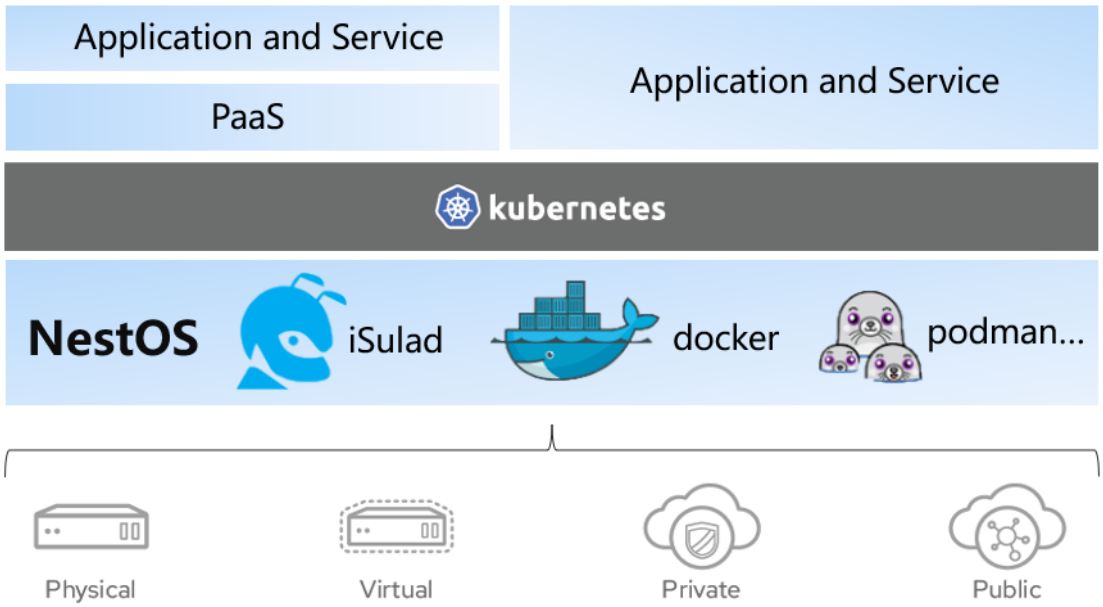
- 按需加载支持 failover 特性 ： cachefiles 在按需模式下，如果守护进程崩溃或被关闭，按需加载相关的读取和挂载将返回 -EIO。所有挂载点必须要在重新拉起 daemon 后重新挂载后方可继续使用。这在公共云服务生产环境中发生时是无法接受的，这样的 I/O 错误将传播给云服务用户，可能会影响他们作业的执行，并危及系统的整体稳定性。cachefiles failover 特性避免了守护进程崩溃后重新挂载所有挂载点，只需快速重新拉起守护进程即可，用户和服务是不感知守护进程崩溃的。
- 支持 CLANG 编译的 PGO 优化：PGO(Profile-Guided Optimization)是一种编译器反馈优化技术，通过收集程序运行时的信息，指导编译器的优化决策。根据业界经验，对于数据中心大型应用（如 MySQL、Nginx、Redis 等）通过反馈优化技术优化应用本身和 Linux kernel 有较好优化效果。经过测试，LLVM PGO 可以在 Nginx 上有 20%+ 性能提升，其中优化 kernel 提升 10+%。

6. 云化基座

NestOS 容器操作系统

NestOS 是在 openEuler 社区孵化的云底座操作系统，集成了 rpm-ostree 支持、ignition 配置等技术。采用双根文件系统、原子化更新的设计思路，使用 nestos-assembler 快速集成构建，并针对 K8S、OpenStack 等平台进行适配，优化容器运行底噪，使系统具备十分便捷的集群组建能力，可以更安全的运行大规模的容器化工作负载。

功能描述



1. 开箱即用的容器平台：NestOS 集成适配了 iSulad、Docker、Podman 等主流容器引擎，为用户提供轻量级、定制化的云场景 OS。
2. 简单易用的配置过程：NestOS 通过 ignition 技术，可以以相同的配置方便地完成大批量集群节点的安装配置工作。
3. 安全可靠的包管理：NestOS 使用 rpm-ostree 进行软件包管理，搭配 openEuler 软件包源，确保原子化更新的安全稳定状态。
4. 友好可控的更新机制：NestOS 使用 zincati 提供自动更新服务，可实现节点自动更新与重新引导，实现集群节点有序升级而服务不中断。
5. 紧密配合的双根文件系统：NestOS 采用双根文件系统的设计实现主备切换，确保 NestOS 运行期间的完整性与安全性。

应用场景

NestOS 适合作为以容器化应用为主的云场景基础运行环境，引入社区孵化项目 NestOS-Kubernetes-Deployer，辅助 NestOS 解决在使用容器技术与容器编排技术实现业务发布、运维时与底层环境高度解耦而带来的运维技术栈不统一，运维平台重复建设等问题，保证了业务与底座操作系统运维的一致性。

7. 特性增强

syscare 特性增强

SysCare 是一个系统级热修复软件，为操作系统提供安全补丁和系统错误热修复能力，主机无需重新启动即可修复该系统问题。 SysCare 将内核态热补丁技术与用户态热补丁技术进行融合统一，用户仅需聚焦在自己核心业务中，系统修复问题交予 SysCare 进行处理。后期计划根据修复组件的不同，提供系统热升级技术，进一步解放运维用户提升运维效率。

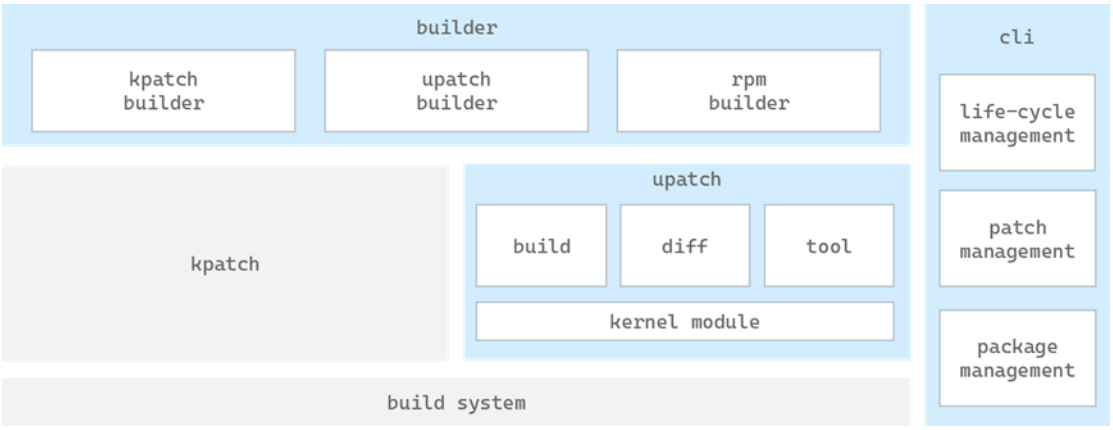
功能描述

1. 热补丁制作

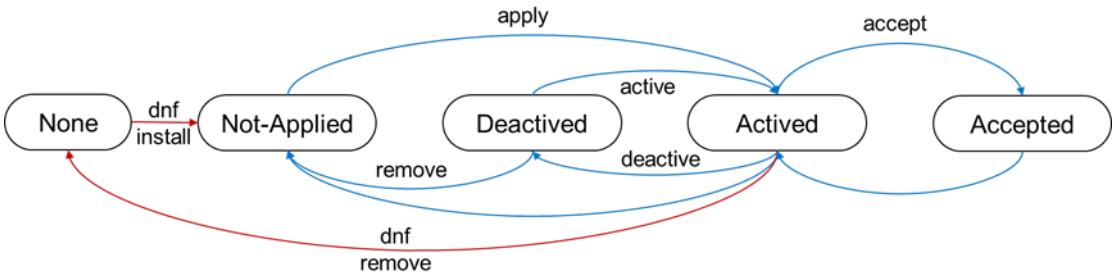
用户仅需输入目标软件的源码 RPM 包、调试信息 RPM 包与待打补丁的路径，无需对软件源码进行任何修改，即可生成对应的热补丁 RPM 包。

2. 热补丁生命周期管理

SysCare 提供一套完整的，傻瓜式补丁生命周期管理方式，旨在减少用户学习、使用成本，通过单条命令即可对热补丁进行管理。依托于 RPM 系统，SysCare 构建出的热补丁依赖关系完整，热补丁分发、安装、更新与卸载流程均无需进行特殊处理，可直接集成放入软件仓 repo。



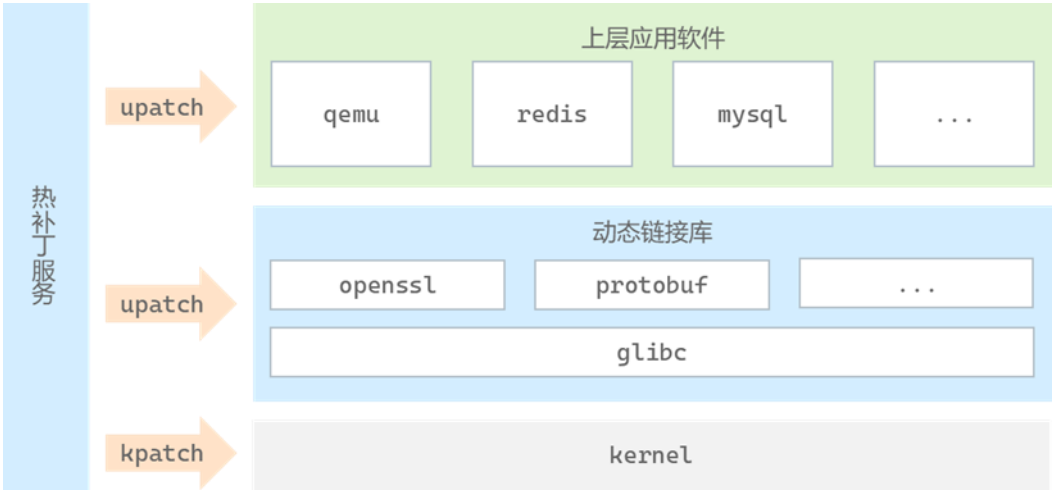
SysCare 整体架构



热补丁生命周期

3. 内核热补丁与用户态热补丁融合

SysCare 基于 upatch 和 kpatch 技术，覆盖应用、动态库、内核，自顶向下打通热补丁软件栈，提供用户无感知的全栈热修复能力。



热补丁应用范围

新增特性

- 用户态热补丁制作工具支持通过配置交叉编译器的方式初步适配嵌入式场景。
- 适配容器内热补丁制作需求，移除用户态热补丁制作相关内核模块。
- 新增用户态进程栈检查，避免多线程场景不一致引起的业务崩溃、异常等。

约束限制

- 仅支持 64 位系统；
- 仅支持 ELF 格式的热修复，不支持解释型语言，不支持纯汇编修改；
- 仅支持 GCC / G++编译器，且不支持交叉编译；
- 暂不支持修改 TLS 变量；
- 暂不支持 LTO 优化；
- 仅 upatch-build 实现编译器配置接口，syscare-build 暂未提供。

应用场景

应用场景 1： CVE 补丁快速修复。

应用场景 2： 现网问题临时定位。

iSula 支持 NRI 插件式扩展

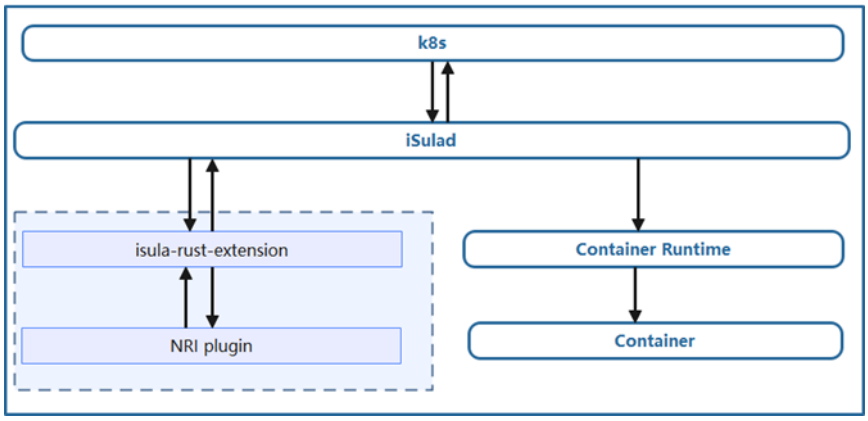
NRI (Node Resource Interface)，是用于控制节点资源的公共接口，是 CRI 兼容的容器运行时插件扩展的通用框架。它为扩展插件提供了跟踪容器状态，并对其配置进行有限修改的基本机制。允许将用户某些自定的逻辑插入到 OCI 兼容的运行时中，此逻辑可以对容器进行受控更改，或在容器生命周期的某些时间点执行 OCI 范围之外的额外操作。iSulad 新增对 NRI 插件式扩展的支持，减少 k8s 场景下对于容器资源管理维护成本，消除调度延迟，规范信息的一致性。

功能描述

NRI 插件通过请求 isula-rust-extension 组件中启动的 NRI runtime Service 服务与 iSulad 建立连接后，可订阅 Pod 与 Container 的生命周期事件：

1. 可订阅 Pod 生命周期事件，包括：creation、stopping 和 removal。
2. 可订阅 Container 生命周期事件，包括 creation、post-creation、starting、post-start、updating、post-update、stopping 和 removal。

iSulad 在接收到 k8s 下发的 CRI 请求后，对于所有订阅了对应生命周期事件的 NRI 插件都发送的请求，NRI 插件可在请求中获得 Pod 与 Container 的元数据与资源信息。之后 NRI 插件可根据需求更新 Pod 与 Container 的资源配置，并将更新的信息传递给 iSulad，iSulad 将更新后的配置传递给容器运行时，生效配置。



约束限制

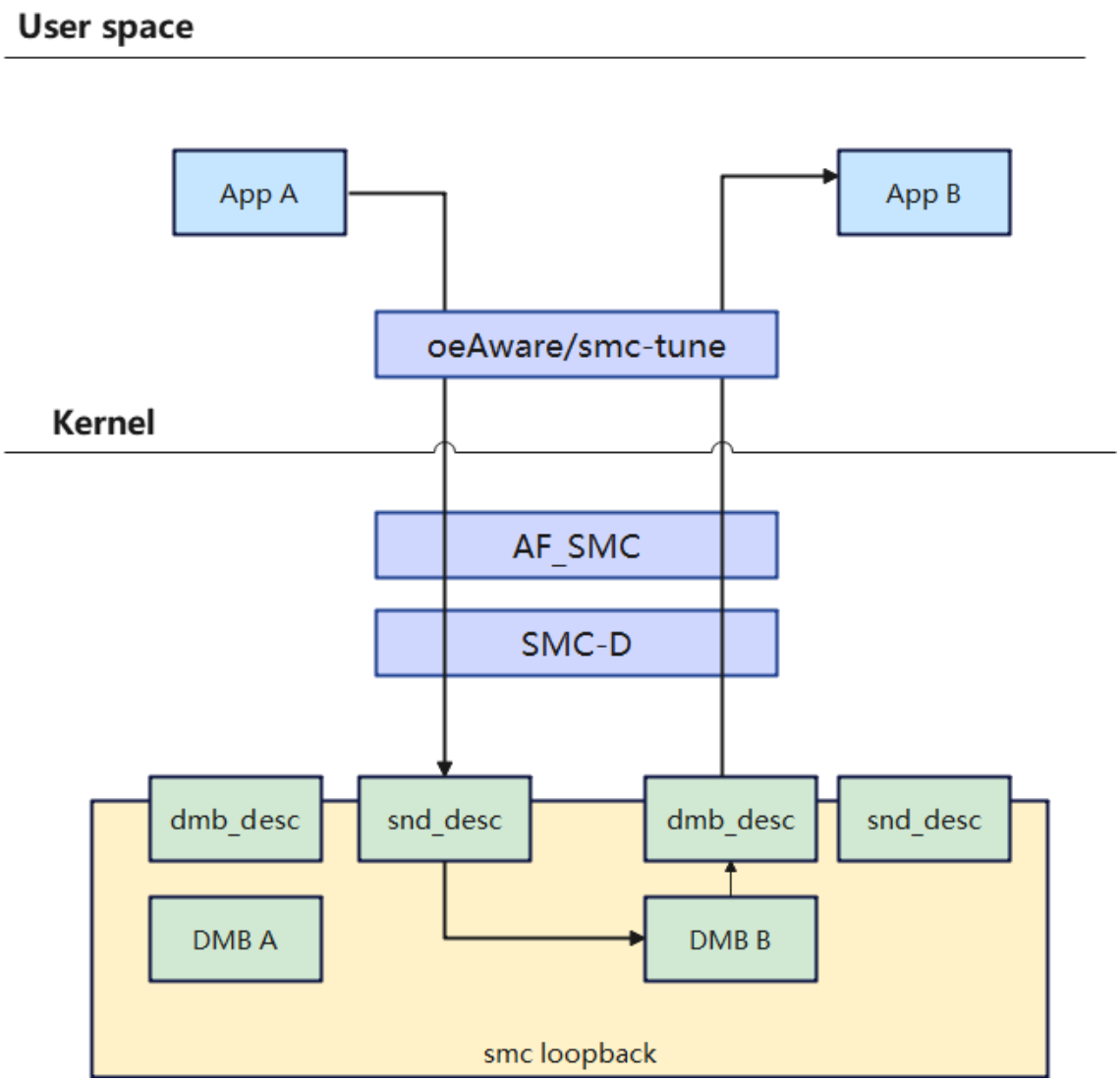
- 仅支持在 k8s CRI V1 中使用 NRI 特性
- 支持的 NRI API 版本为 0.6.1

应用场景

iSulad 的 NRI 特性可用于 k8s 场景下资源的管理。用户可通过实现 NRI 插件，订阅容器生命周期事件跟踪容器资源状态，并根据资源管理逻辑对 NRI API 范围内允许的容器配置进行修改。例如，可以利用 NRI 实现以标准化插件的形式解决在线与离线业务混部场景中资源利用率与优先级调度的问题。

oeAware 支持本地网络加速

Linux 的 SMC-D 内核特性主要涉及共享内存通信（SMC）在操作系统内部的优化和加速，特别是在进程间通信（IPC）方面的应用。oeAware 在这特性基础上做到无感使能，使得 tcp 传输能够无感加速到 smc-d 上。



功能描述

SMC-D 引入了 loopback 设备，作为系统全局设备，可以被所有容器访问，从而加速本地进程间的 TCP 通信。这种设计使得 SMC-D 能够在不同的虚拟环境中提供一致的性能表现。

1. 高效通信：SMC-D 利用内部共享内存（ISM）技术，避免了传统网络通信的开销，从而实现低延迟和高吞吐量的通信。

2. 透明替换：SMC-D 可以透明地替换现有的 TCP 协议栈，无需对应用进行大规模修改。

3. 自动协商：SMC-D 具备自动协商和动态回退到 TCP 的能力，确保在不同环境下的兼容性和稳定性。

oeAware 在这特性基础上做到无感使能，使得 tcp 传输能够无感加速到 smc-d 上。

约束限制

- 需要在服务端客户端建链前，完成使能 smc 加速。

- 比较适用于长链接多的场景。

应用场景

SMC-D 特别适用于需要高吞吐量和低延迟的应用场景，如高性能计算（HPC）、大数据处理和云计算平台。通过直接内存访问（DMA），SMC-D 能够显著减少 CPU 负载并提升交互式工作负载的速率。

1. 高性能计算：在需要高吞吐量和低延迟的高性能计算环境中，SMC-D 可以显著提升数据传输效率。

2. 数据中心：在数据中心内部，SMC-D 可以用于加速进程之间的通信，减少网络延迟和 CPU 负载。

3. 容器通信：在同一操作系统内的容器之间，SMC-D 可以提供快速的进程间通信，适用于微服务架构。

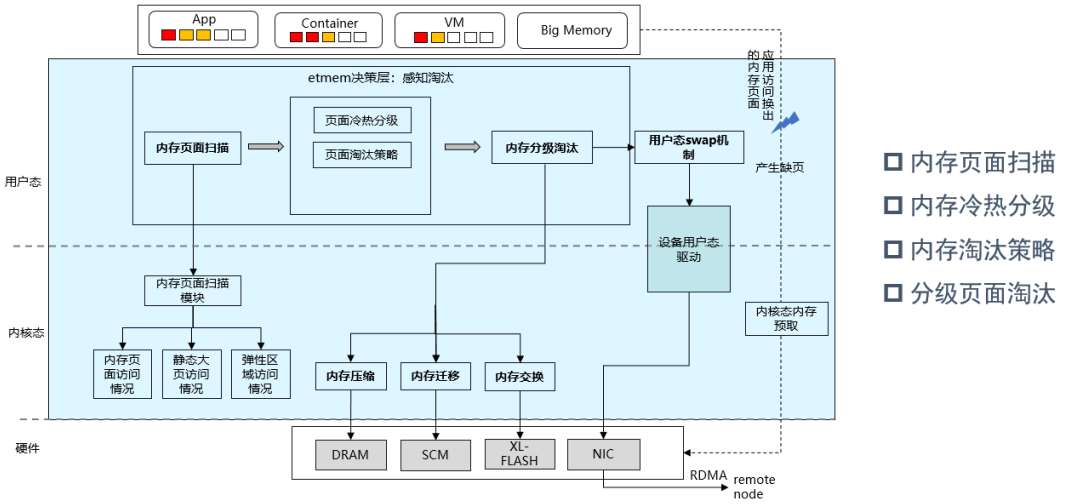
etmem 内存扩展，实现虚机内存超分

随着 CPU 算力的发展，尤其是 ARM 核成本的降低，内存成本和内存容量成为约束业务成本和性能的核心痛点，因此如何节省内存成本，如何扩大内存容量成为存储迫切要解决的问题。

etmem 内存分级扩展技术，通过 DRAM+内存压缩/高性能存储新介质形成多级内存存储，对内存数据进行分级，将分级后的内存冷数据从内存介质迁移到高性能存储介质中，达到内存容量扩展的目的，从而实现内存成本下降。

功能描述

etmem 内存分级扩展通过 DRAM+SCM+XL-FLASH 换入换出形成多级存储，并对应用透明，兼容原有应用生态，实现本地分级内存，达到节省内存成本的目的。



1. 内存页面扫描：通过内核态的内存页面扫描模块遍历页表，获取页面的 access bit，用于判断页面在扫描周期内是否被访问，可以通过配置控制页面扫描的周期，扫描次数，扫描的内存范围等。
2. 内存冷热分级：根据内存页面扫描的结果，统计识别内存冷热分布，如果存在多级存储的情况，对页面进行多级冷热分级。当前只交付冷热两级分级，多种分级的实现做技术预研。
3. 内存淘汰策略：根据页面冷热分级和迁移策略确定要淘汰的内存页面。etmem 提供默认策略，也可支持用户自定义淘汰策略，即第三方策略。
4. 分级页面淘汰：通过内核态的内存压缩机制或 swap 机制或 move_pages 搬迁机制，将页面分级淘汰到不同的 backend 介质中。

内存交换与内存压缩都由 etmem 淘汰机制指定页面，通过指定内存页面淘汰机制入口而非 OS 的 LRU 淘汰机制，交换、压缩和搬迁操作本身复用开源实现。

5. 虚机内存扩展：支持虚机场景下的内存扩展，针对虚机内存超分场景，通过扫描虚机二级页表的方式判断虚机中内存的冷热信息，从而实现虚机内存冷热识别，达到虚机内存超分的目的。

应用场景

etmem 内存扩展技术相较于 kswap 等被动式内存扩展技术，可以实现进程级别主动内存换出，内存提前换出等，实现对于关键业务进程的性能保障。该特性适用于在服务器内存受限，业务具有一定优先级情况下，例如数据库场景、虚拟化场景、管理容器等场景下，能够在降低业务内存占用的同时，尽量较少业务关键进程的性能波动。

1. 可以应用在传统业务，当前内存工艺达到瓶颈，内存成本占比高，通过内存扩展将不同热度的内存数据调度到不同的分级介质中，节省 DRAM 的使用量，降低整体的内存成本。

2. 虚拟化场景下内存超分，针对云服务器场景、多虚机场景等可以利用 etmem 内存扩展技术实现内存成本的降低。

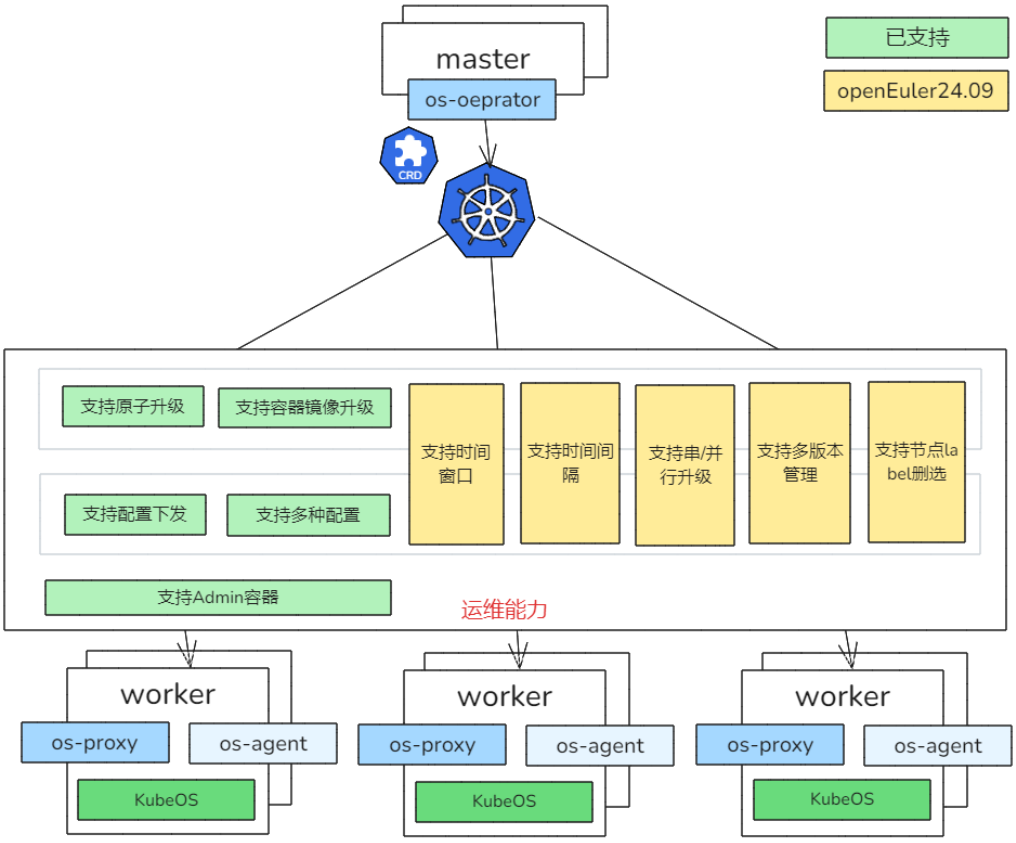
仓库地址

https://gitee.com/openeuler/etmem。

KubeOS 支持串、并行多种升级策略、状态更新等声明式 API

KubeOS 是针对云原生场景而设计、轻量安全的云原生操作系统及运维工具，提供基于 kubernetes 的云原生操作系统统一运维能力。KubeOS 设计了专为容器运行的云原生操作系统，通过根目录只读，仅包含容器运行所需组件，dm-verity 安全加固，减少漏洞和攻击面，提升资源利用率和启动速度，提供云原化的、轻量安全的操作系统。KubeOS 支持使用 kubernetes 原生声明式 API，统一对集群 worker 节点 OS 的进行升级、配置和运维，从而降低云原生场景的运维难度、解决用户集群节点 OS 版本分裂，缺乏统一的 OS 运维管理方案的问题。

功能描述



KubeOS 基于统一运维框架新增运维策略如图所示：

1. KubeOS 支持配置运维时间窗口和时间间隔：

(1) KubeOS 支持在升级/配置时设置时间窗，在指定时间内升级/配置。

(2) KubeOS 支持在升级/配置时设置时间间隔，一次升级/配置完成后，间隔指定时间再次下发升级/配置。

2. KubeOS 支持配置串行或者并行运维策略：并行升级/配置为每次升级/配置会有 n 个节点（用户指定）同时被下发升级/配置任务（已支持），串行升级/配置为每次升级/配置会有 n 个节点（用户指定）串行被下发升级/配置任务，当所有节点完成升级/配置后，下发下一批节点。

3. KubeOS 支持集群 OS 多版本管理：

(1) 支持单个集群内可以有多个 os cr，指定多个 OS 的版本，根据 os cr 中的 nodeselector 字段指定的 node label 和节点进行对应。

(2) 支持在升级/配置的时候根据 node label 进行升级/配置。

应用场景

降低云原生场景下操作系统运维管理难度，改善人工介入多，OS 升级时间长等运维痛点，基于 KubeOS 统一运维框架功能，提供灵活、多维度的运维策略，满足用户定时、串行、集群 OS 多版本等多种场景下运维需求。

IMA 摘要列表支持三方 PKI 证书导入

IMA，全称 Integrity Measurement Architecture（完整性度量架构），是内核中的一个安全子系统，能够基于策略对通过 execve()、mmap()和 open()等系统调用访问的文件进行度量，度量结果可被用于本地校验或远程证明，从而实现对系统的文件完整性保护。

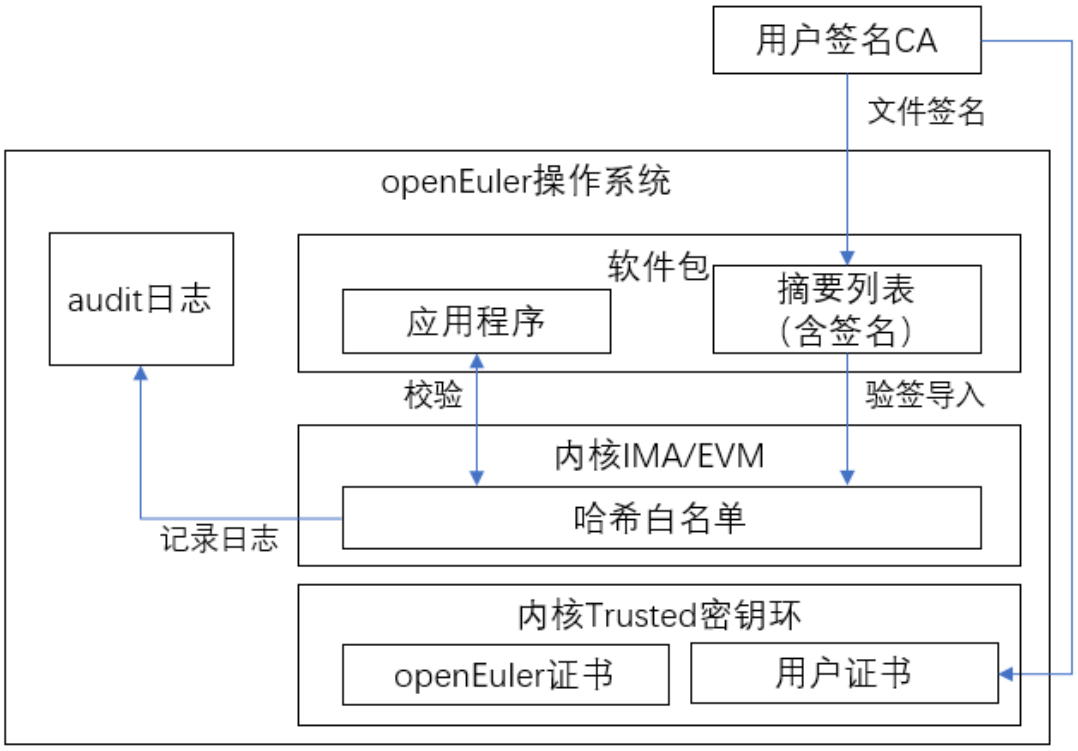
IMA Digest Lists（IMA 摘要列表）是 openEuler 对内核原生 IMA 机制的增强，从 openEuler 21.03 版本开始支持，通过 RPM 包构建预生成摘要列表并签名的方式，对整体的性能、安全性和易用性进行提升。

功能描述

IMA 评估模式（IMA appraisal）与安全启动类似，旨在对执行的应用程序或访问的关键文件执行完整性校验，如果校验失败，则拒绝访问。

在开启 IMA 摘要列表功能后，执行校验的基准数据来源于内核维护的摘要值白名单，该白名单可以通过内核接口从用户态导入摘要列表文件实现添加/删除。为保证白名单的真实性，需要对导入的文件执行签名验证。

openEuler 24.09 版本实现了对导入用户证书的支持，即支持用户在编译内核阶段，预置自定义的签名证书，并在后续的流程中，支持使用自定义证书对摘要列表文件执行签名验证。



应用场景

用户可基于该特性，实现用户态应用程序安全启动，确保只有预先经过认证的应用程序才可在当前系统中运行，从而抵御病毒植入。

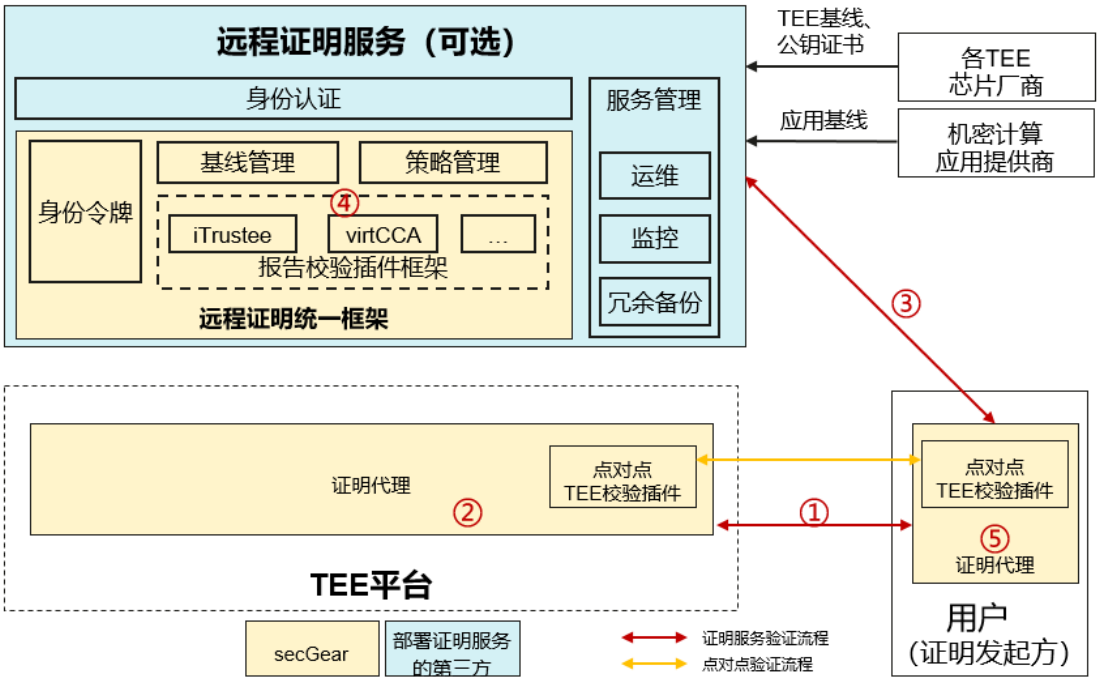
构建远程证明统一框架，支持部署远程证明服务

secGear 远程证明统一框架是机密计算远程证明相关的关键组件，屏蔽不同 TEE 远程证明差异，提供 Attestation Agent 和 Attestation Service 两个组件，Agent 供用户集成获取证明报告，对接证明服务；Service 可独立部署，支持 iTrustee、virtCCA 远程证明报告的验证。

功能描述

远程证明统一框架聚焦机密计算相关功能，部署服务时需要的服务运维等相关能力由服务部署第三方提供。远程证明统一框架的关键技术如下：

- **报告校验插件框架**：支持运行时兼容 iTrustee、vritCCA、CCA 等不同 TEE 平台证明报告检验，支持扩展新的 TEE 报告检验插件。
- **证书基线管理**：支持对不同 TEE 类型的 TCB/TA 基线值管理及公钥证书管理，集中部署到服务端，对用户透明。
- **策略管理**：提供默认策略（易用）、用户定制策略（灵活）。
- **身份令牌**：支持对不同 TEE 签发身份令牌，由第三方信任背书，实现不同 TEE 类型相互认证。
- **证明代理**：支持对接证明服务/点对点互证，兼容 TEE 报告获取，身份令牌验证等，易集成，使用户聚焦业务。



根据使用场景，支持点对点验证和证明服务验证两种模式。

证明服务验证流程如下：

① 用户（普通节点或 TEE）对 TEE 平台发起挑战。

② TEE 平台通过证明代理获取 TEE 证明报告，并返回给用户。

③ 用户端证明代理将报告转发到远程证明服务。

④ 远程证明服务完成报告校验，返回由第三方信任背书的统一格式身份令牌。

⑤ 证明代理验证身份令牌，并解析得到证明报告校验结果。

点对点验证流程(无证明服务)如下：

① 用户向 TEE 平台发起挑战，TEE 平台返回证明报告给用户。

② 用户使用本地点对点 TEE 校验插件完成报告验证。

注意：点对点验证和远程证明服务验证时的证明代理不同，在编译时可通过编译选项，决定编译有证明服务和点对点模式的证明代理。

应用场景

在金融、AI 等场景下，基于机密计算保护运行中的隐私数据安全时，远程证明是校验机密计算环境及应用合法性的技术手段，远程证明统一框架提供了易集成、易部署的组件，帮助用户快速使能机密计算远程证明能力。

容器干扰检测，分钟级完成业务干扰源（CPU/IO）识别与干扰源发现

gala-anteater 是一款基于 AI 的操作系统灰度故障的异常检测平台，集成了多种异常检测算法，通过自动化模型预训练、线上模型的增量学习和模型更新，可以实现系统级故障发现和故障点上报。

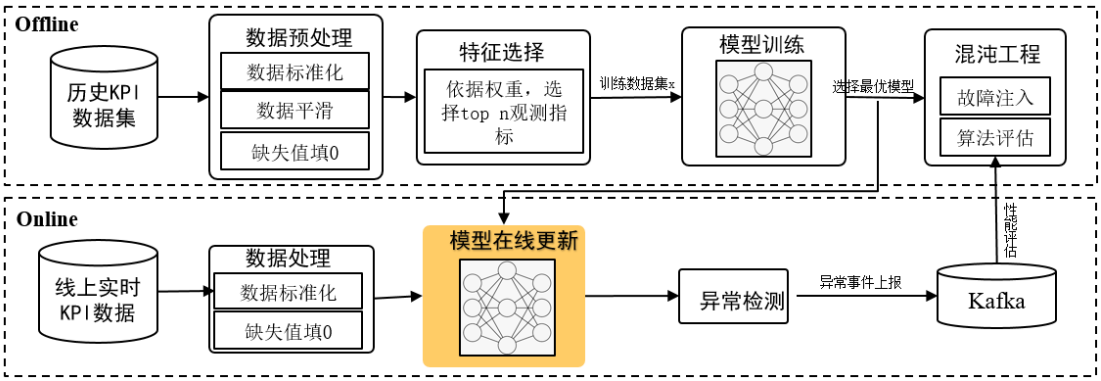
在线容器高密部署场景下，存在资源无序竞争现象，导致容器实例间相互干扰，使用 gala-anteater 可以分钟级完成业务干扰源（CPU/IO）识别与干扰源发现，辅助运维人员快速跟踪并解决问题，保障业务 Qos。

功能描述

gala-anteater 通过线上线下相结合，利用在线学习技术，实现模型的线下学习，线上更新，并应用于线上异常检测。

Offline: 首先，利用线下历史 KPI 数据集，经过数据预处理、特征选择，得到训练集；然后，利用得到的训练集，对无监督神经网络模型（例如 Variational Autoencoder）进行训练调优。最后，利用人工标注的测试集，选择最优模型。

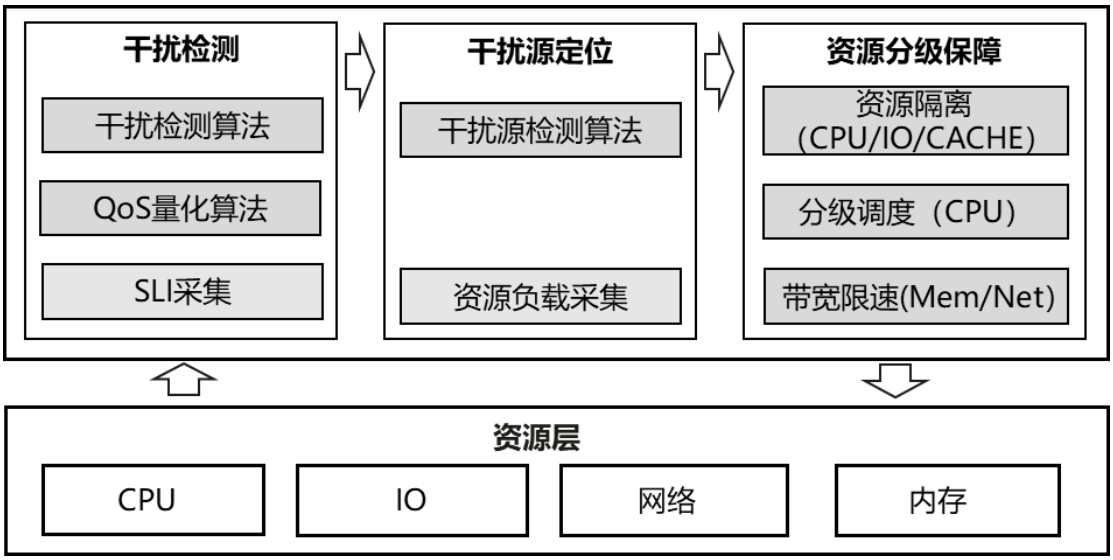
Online：将线下训练好的模型，部署到线上，然后利用线上真实的数据集，对模型进行在线训练以及参数调优，然后利用训练好的模型，进行线上环境的实时异常检测。



应用场景

gala-anteater 支持应用级和系统级的异常检测和上报。

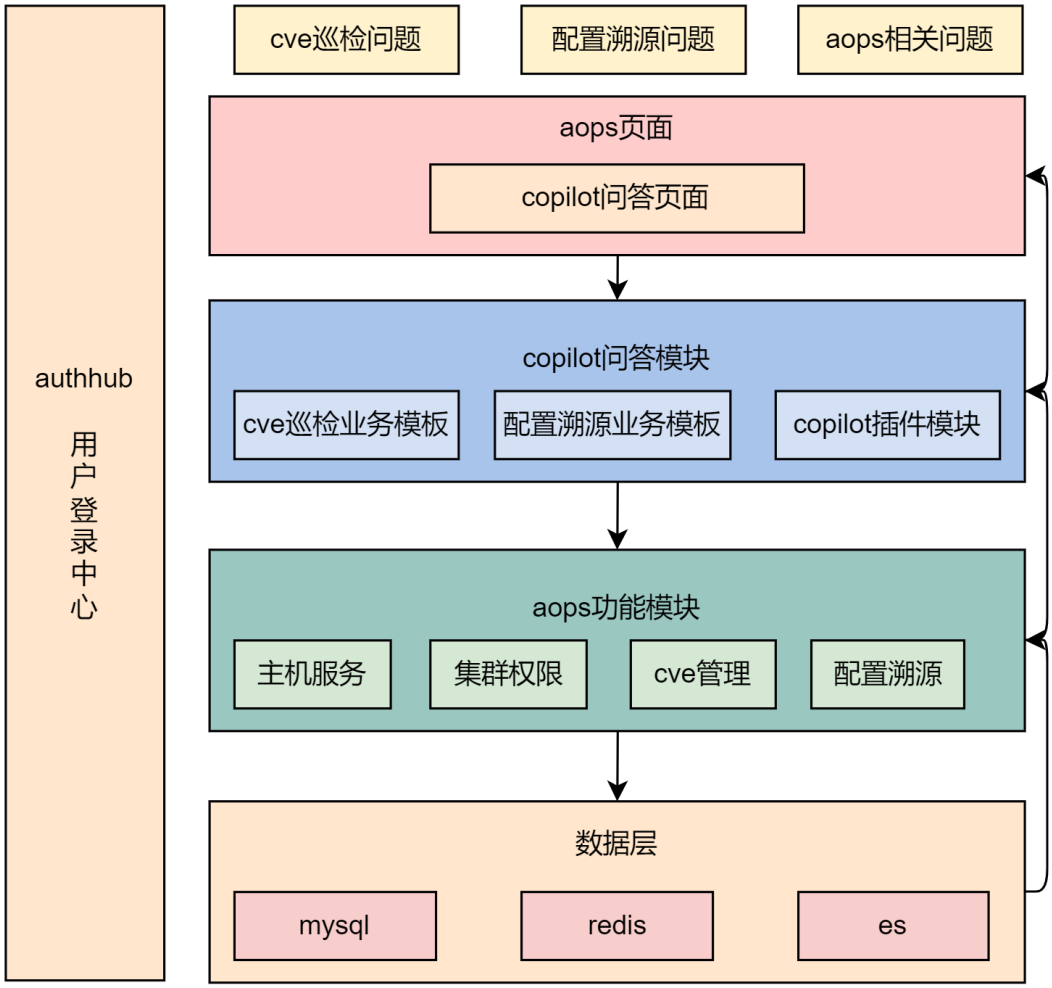
在实际应用场景中可以结合 gala-gopher 和 rubik 来实现实时干扰检测，定位以及自愈。gala-anteater 利用 gala-gopher 实时采集的资源层指标进行干扰检测和定位，上报的检测结果通过 rubik 实现资源分级保障对干扰进行快速恢复。



Aops 支持智能运维助手

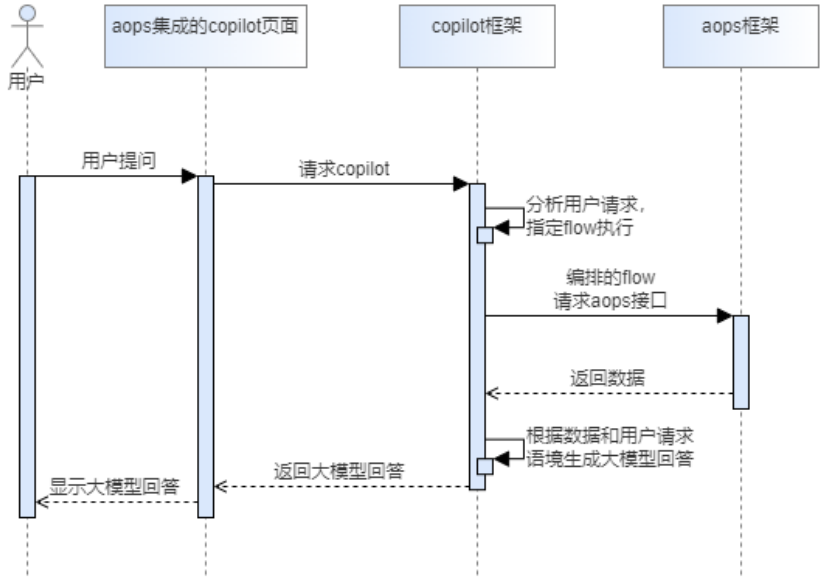
Aops 基于 EulerCopilot plugins 功能，实现智能运维助手，通过自然语言交互实现 cve 巡检和配置溯源的功能。

功能描述



Aops 基于 EulerCopilot plugins 功能，实现智能运维助手，智能运维助手功能包括：

- 统一用户登录中心：Aops 和 EulerCopilot 实现统一用户登录中心 authhub，统一用户身份鉴权。
- cve 巡检和配置溯源：通过配置 cve 巡检和配置溯源的业务场景 flow 模板，支持自然语言对接，实现启发式 cve 巡检和配置溯源。

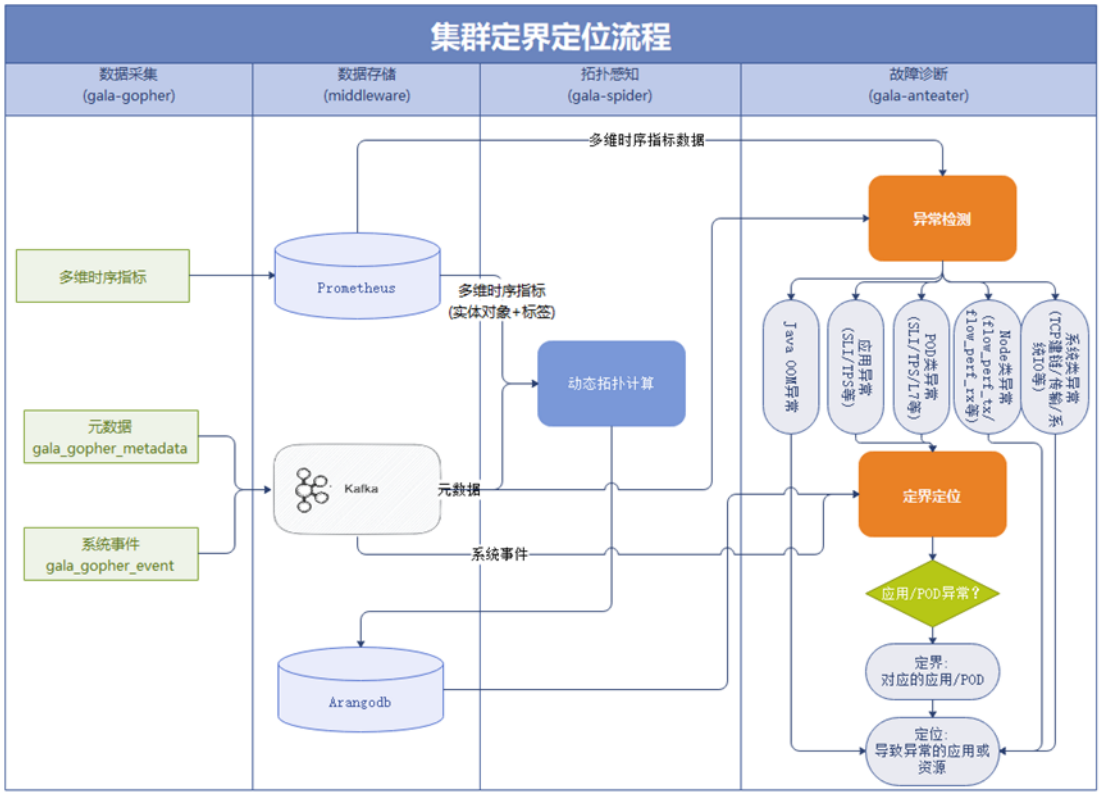


应用场景

智能运维助手支持用户用自然语言交互使用 cve 巡检和配置溯源功能，简化用户操作流程，提升用户体验。

微服务性能问题分钟级定界/定位（TCP，IO，调度等）

功能描述



基于拓扑的根因推导技术提供了大规模集群情况下的故障检测、根因定位服务，新增云原生场景故障定界定位能力，特别是针对网络 L7 层协议，如 HTTPS、PGSQL、MYSQL 等。这项技术通过 gala-gopher 和 gala-anteater 实现，新增内容提供了更细粒度的故障定界能力，通过这种能力运维团队可以快速定位问题源头，从而提高系统的稳定性和可靠性。该服务主要功能如下：

- 指标采集服务：gala-gopher 通过 ebpf 技术提供网络、IO 相关的指标采集上报功能。
- 集群拓扑构建服务：gala-spider 通过接受指标采集服务上报的数据信息，构建容器、进程级别的调用关系拓扑图构建。
- 故障检测服务：gala-anteater 通过故障检测模型对应用上报的指标采集进行分类，判断是否发生异常。
- 根因定位服务：gala-anteater 通过节点异常信息和拓扑图信息， 定位导致此次异常的根因节点。

应用场景

分钟级别应用问题定界/定位场景主要包括以下几种情况：

- 云 K8S POD 部署场景：拥有云环境的企业可以根据自身的需求和 IT 资源的现状，选择 gala-anteater 技术，gala-gopher 上报数据支持 K8S 容器、进程粒度的指标采集，通过这些指标采集可以实现调用管理拓扑图构建以及故障发生时的故障检测与根因节点根因发现功能。
- 裸金属部署场景：拥有裸金属部署场景的企业可以根据自身的需求和 IT 资源的现状，选择 gala-anteater 技术，gopher 同样可以支撑进程级别的数据采集和容器粒度指标采集， 同样可以实现拓扑图构建以及故障发生时的故障检测与根因节点根因发现功能。
- 大规模虚拟机场景：拥有大规模虚拟机场景的企业可以根据自身的需求和 IT 资源的现状，选择 gala-anteater 技术， gala-gopher 对虚拟机场景也提供支持，通过对虚拟机相关数据的采集上报， gala-spider 和 gala-anteater 可以完成虚拟机之间的拓扑图调用关系构建和虚拟机之间的故障检测与根因定位。

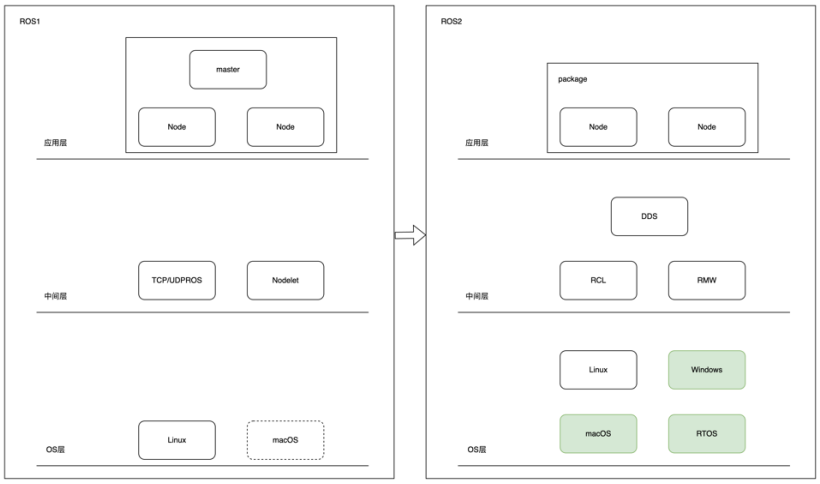
ROS2-humble 和 ROS1-noetic 基础版本发布

ROS（Robot Operating System），作为一套专为机器人应用开发设计的软件框架与工具集，其核心使命在于简化并加速机器人应用程序的构建过程。从底层的硬件驱动程序到高层的复杂算法实现，再到各类开发工具，ROS 通过其独特的通信架构，将原本各自为政的软硬件组件紧密地集成在一起，为开发者打造了一个高效协同的工作平台。

ROS 发展至今，有 ROS1 和 ROS2 两大版本，ROS SIG 选取 humble 和 noetic 版本做为长期维护版本，旨在填补 openEuler 在支持 ROS 应用方面的空白。

功能描述

ROS1 与 ROS2 功能架构图：



1. OS 层：

ROS1 并不是一个传统意义上的操作系统，它无法像 Windows、Linux 等直接运行在计算机硬件上，而是需要依托于 Linux 系统（如 Ubuntu）或其他操作系统（如 macOS、Windows、RTOS 等）来运行。这一层主要提供基础的操作系统支持，包括硬件驱动、进程管理等。

2. 中间层：

- ROS1 在中间层封装了大量的机器人开发中间件，其中最核心的是基于 TCP/UDP 封装的 TCPROS/UDPROS 通信系统。这个通信系统使用发布/订阅、客户端/服务器等模型，实现多种通信机制的数据传输。
- 除了 TCPROS/UDPROS，ROS1 还提供了一种进程内的通信方法——Nodelet，它适用于对数据传输实时性有较高要求的应用场景。
- ROS1 还提供了大量机器人开发相关的库，如数据类型定义、坐标变换、运动控制等，这些库可以被应用层直接使用。
- ROS2 的中间层是 ROS2 的核心部分，它主要由数据分发服务（DDS, Data Distribution Service）和 ROS2 封装的关于机器人开发的中间件组成。DDS 是一种去中心化的数据通讯方式，由对象管理组（OMG）发布，它允许节点在不需要中心管理节点（如 ROS 1 中的 Master 节点）的情况下进行通信，从而提高了系统的鲁棒性和可扩展性。
- 中间层可以进一步细分为以下几层：

- DDS Implementation 层：该层是 DDS 实现层，由各个厂商提供相应的 DDS 软件产品。用户可以根据自己的实际需求选择不同厂商的 DDS 实现，如开源的 Fast-DDS、Cyclone-DDS 等。
- Abstract DDS 层（RMW 层）：称为 ROS Middleware Interface（ROSMiddleWare Interface）层，是相对底层的接口层，由 C 语言实现，直接和 DDS 交互，并向 ROS2 Client 层提供 API 接口。该层的目的是让用户程序可以在不同厂商的 DDS 之间进行无感移植。
- ROS2 Client 层（RCL 层）：包含 ROS2 核心概念如 Node、Action 等的具体实现，以及不同编程语言的 API 接口，如 C++、Python 和 Java 等。在 ROS2 中，不同编程语言的 RCL 库是一致的，都是 C 实现的，而编程语言 API 接口是独立的，从而保证了不同编程语言下程序运行的一致性。

3. 应用层：

- 在应用层，ROS1 需要运行一个管理者——Master，它负责管理整个系统的正常运行。Master 通过远程过程调用（RPC）提供登记列表和对其他计算图表的查找功能，帮助 ROS 节点之间相互查找、建立连接，并管理全局参数。
- ROS1 社区内共享了大量的机器人应用功能包，这些功能包内的模块以节点为单位运行，通过 ROS 标准的输入输出作为接口，开发者可以方便地复用这些功能包，提高开发效率。
- 在 ROS2 中，应用程序以功能包（package）为核心，功能包中包含源码、数据定义、接口等内容。开发者通过实现具有特定功能的功能包来构建机器人应用程序。

约束限制

ROS1 的限制：

1. **单点故障**:

ROS1 中存在一个 master 节点，负责管理所有其他节点的通信和状态。一旦 master 节点出现故障，整个系统将面临宕机的风险，因为所有节点间的通信都将中断。

2. **平台依赖**:

ROS1 主要设计在 Linux 平台上运行，对其他操作系统的支持不够友好，这限制了 ROS1 在不同机器人平台上的广泛应用。

3. **安全性问题**:

ROS1 的通信协议（如 TCP/IP）在数据加密和安全性方面存在不足，可能无法满足一些对安全性要求较高的应用场景。

4. **实时性不足**:

对于需要高实时性处理的机器人应用，ROS1 可能无法提供足够的性能保障，因为它在设计时并未特别强调实时性。

5. **编译和构建系统**:

ROS1 使用的编译和构建系统（如 rosbuild 和 catkin）在处理大型项目或 Python 编写的项目时，可能会遇到编译、链接错误等问题，影响开发效率。

ROS2 的限制：

尽管 ROS2 在设计和实现上针对 ROS1 的许多限制进行了改进，但它也面临一些新的挑战和限制:

1. **复杂性增加：**

ROS2 采用了更复杂的通信协议（DDS）和编程模型，这使得 ROS2 在功能上更加强大，但同时也增加了系统的复杂性和学习曲线。

2. **兼容性问题：**

由于 ROS2 在 API 和通信协议上进行了重大更改，导致 ROS1 的代码无法直接在 ROS2 中运行。这要求开发者在迁移现有项目到 ROS2 时需要进行大量的重构工作。

3. **DDS 实现的多样性：**

DDS 作为一种国际标准，存在多个不同的实现版本。虽然 ROS2 通过 RMW 层提供了对多种 DDS 实现的支持，但不同 DDS 实现之间的性能差异和接口差异仍然可能给开发者带来一定的困扰。

4. **资源消耗：**

相比 ROS1，ROS2 在提供更高可靠性和扩展性的同时，也可能带来更高的资源消耗（如 CPU、内存和网络带宽）。这可能对资源有限的嵌入式系统构成一定的挑战。

5. **生态系统成熟度：**

尽管 ROS2 在不断发展和完善中，但其生态系统相比 ROS1 仍然不够成熟。一些在 ROS1 中广泛使用的工具和库可能尚未在 ROS2 中得到充分支持或优化。

应用场景

ROS 的应用场景广泛，主要涵盖工业自动化、农业自动化、医疗（如手术机器人）、家庭服务机器人（如扫地机器人）、娱乐（如机器人竞赛）、安全保障（如灾害应对）以及无人驾驶等领域，为机器人技术的多样化应用提供了强大支持。

utsudo 项目发布

Sudo 是 Unix 和 Linux 操作系统中常用的工具之一，它允许用户在需要超级用户权限的情况下执行特定命令。然而， 传统 Sudo 在安全性和可靠性方面存在一些缺陷，为此 utsudo 项目应运而生。

utsudo 是一个采用 Rust 重构 Sudo 的项目，旨在提供一个更加高效、安全、灵活的提权工具，涉及的模块主要有通用工具、整体框架和功能插件等。

功能描述

基本功能：

- 访问控制：可以根据需求，限制用户可以执行的命令，并规定所需的验证方式。
- 审计日志：可以记录和追踪每个用户使用 utsudo 执行的命令和任务。
- 临时提权：允许普通用户通过输入自己的密码，临时提升为超级用户执行特定的命令或任务。
- 灵活配置：可以设置参数如命令别名、环境变量、执行参数等，以满足复杂的系统管理需求。

增强功能：

utsudo 在 openEuler 24.09 版本中是 0.0.2 版本，当前版本主要功能有：

- 提权流程：把普通用户执行命令的进程，提权为 root 权限。
- 插件加载流程：实现了对插件配置文件的解析，以及对插件库的动态加载。

应用场景

通过部署 utsudo，管理员能够更好地管理用户权限，确保合适的授权级别，防止未经授权的特权操作，从而降低安全风险。

常见的应用场景有：系统管理维护、用户权限控制、多用户环境等。

utshell 项目发布

utshell 是一个延续了 bash 使用习惯的全新 shell，它能够与用户进行命令行交互，响应用户的操作去执行命令并给予 反馈。并且能执行自动化脚本帮助运维。

功能描述

基本功能：

- 命令执行：可以执行部署在用户机器上的命令，并将执行的返回值反馈给用户。
- 批处理：通过脚本完成自动任务执行。
- 作业控制：能够将用户命令作为后台作业，从而实现多个命令同时执行。并对并行执行的任务进行管理和控制。
- 历史记录：记录用户所输入的命令。
- 别名功能：能够让用户对命令起一个自己喜欢的别名，从而个性化自己的操作功能。

增强功能：

utshell 在 openEuler24.09 版本中是 0.5 版本，当前版本主要功能有：

- 实现对 shell 脚本的解析。
- 实现对第三方命令的执行。

应用场景

utshell 适用于传统服务器系统以及云原生环境，有人值守或无人值守下自动化脚本运维的生产环境。也适用于桌面命 令行爱好者的日常使用。

GCC for openEuler

GCC for openEuler 基线版本已经从 GCC 10.3 升级到 GCC 12.3 版本，支持自动反馈优化、软硬件协同、内存优化、SVE 向量化、矢量化数学库等特性。

1. GCC 版本升级到 12.3，默认语言标准从 C14/C++14 升级到 C17/C++17 标准，支持 Armv9-a 架构，X86 的 AVX512 FP16 等更多硬件架构特性。

	GCC 10.3.0	GCC 11.3.0	GCC 12.3.0
发布时间	2021/4/8	2022/4/21	2023/5/8
C标准	默认c17 支持c2x	默认c17 支持c2x	默认c17 支持c2x
C++标准	默认c++14 支持c++17 实验性C++2a改进 支持部分C++20	默认c++17 实验性C++2a改进 支持部分C++20	默认c++17 实验性C++2a改进 支持部分C++20
架构 新特性	armv8.6-a (bfloat16 extension/Matrix Multiply extension) SVE2 Cortex-A77 Cortex-A76AE Cortex-A65 Cortex-A65AE Cortex-A34	armv8.6-a, +bf16, +i8mm armv8.6-r Cortex-A78 Cortex-A78AE Cortex-A78C Cortex-X1	armv8.7-a, +ls64 atomic load and store armv8.8-a, +mop, accelerate memory operations armv9-a Ampere-1 Cortex-A710 Cortex-X2 AVX512-FP16 SSE2-FP16

2. 支持结构体优化，指令选择优化等，充分使能 ARM 架构的硬件特性，运行效率高，在 SPEC CPU 2017 等基准测试中性能大幅优于上游社区的 GCC 10.3 版本。

3. 支持自动反馈优化特性，实现应用层 MySQL 数据库等场景性能大幅提升。

功能描述

- 支持 ARM 架构下 SVE 矢量化优化，在支持 SVE 指令的机器上启用此优化后能够提升程序运行的性能。
- 支持内存布局优化，通过重新排布结构体成员的位置，使得频繁访问的结构体成员放置于连续的内存空间上，提升 Cache 的命中率，提升程序运行的性能。
- 支持 SLP 转置优化，在循环拆分阶段增强对连续访存读的循环数据流分析能力，同时在 SLP 矢量化阶段，新增转置操作的分析处理，发掘更多的矢量化机会，提升性能。

- 支持冗余成员消除优化，消除结构体中从不读取的结构体成员，同时删除冗余的写语句，缩小结构体占用内存大小，降低内存带宽压力，提升性能。
- 支持数组比较优化，实现数组元素并行比较，提高执行效率。
- 支持 ARM 架构下指令优化，增强 ccmp 指令适用场景，简化指令流水。
- 支持自动反馈优化，使用 perf 收集程序运行信息并解析，完成编译阶段和二进制阶段反馈优化，提升 MySQL 数据库等主流应用场景的性能。

应用场景

通用计算领域，运行 SPECCPU 2017 测试，相比于上游社区的 GCC 10.3 版本可获得 20%左右的性能收益。

其他场景领域，使能自动反馈优化后，MySQL 性能提升 15%以上；使能内核反馈优化后，实现 Unixbench 性能提升 3%以上。

麒麟可信镜像构建管理工具

KTIB（Kylin Trust Image Builder）是一个包含静态合规扫描、单步构建镜像、一步构建镜像、镜像管理等组件的自研开源镜像构建工具，为用户提供了全方位的镜像构建解决方案，帮助用户确保镜像的安全性和合规性。

功能描述

1. 快速构建容器镜像 rootfs 文件系统

   (1) KTIB 支持自定义构建容器镜像 rootfs 文件系统。

   (2) 用户可以通过配置文件轻松地定制和扩展 rootfs，满足不同的业务需求。

2. 静态合规扫描

   (1) 支持在构建镜像前对 Dockerfile 进行静态合规扫描。

   (2) 提供默认的扫描策略文件，策略基于 CIS 规范制定。

   (3) 策略文件支持自定义，用户可以根据独立需求，按照官方文档修改策略文件内容，自定义扫描项。

   (4) 支持输出 JSON 格式的扫描报告和相应的修改建议。

3. 单步构建镜像

(1) KTIB 支持在 shell 终端通过命令行逐条执行 Dockerfile 中的指令,如 FROM、COPY、RUN 等。

(2) 这种单步构建的方式方便用户查看和调试每一个构建步骤，有助于排查问题和优化构建过程。

(3) 用户可以针对性地执行特定的构建指令，灵活地控制镜像的构建。

4. 一步构建镜像

支持一键式的镜像构建，自动执行 Dockerfile 中的所有指令，生成镜像。

5. 镜像管理

KTIB 支持命令行管理镜像，目前提供推送、拉取、列表、删除、登录等命令。

应用场景

1. 适用于需要定制化文件系统的场景，如特殊应用的开发环境、特殊依赖的环境构建，或者需要遵循特定安全策略的生产环境。
2. 适用于需要遵循安全和合规标准的企业或组织。在开发和生产环境中，确保 Dockerfile 符合安全和合规性要求，减少潜在的安全漏洞和配置问题。
3. 适用于开发阶段需要频繁调整构建步骤的场景，单步构建帮助开发者对镜像构建过程进行详细调试，快速定位和解决构建过程中出现的问题。
4. 适用于生产环境和自动化部署场景。

NDK

NKD（NestOS kubernetes deployer）是一款面向容器云场景的 kubernetes 集群部署工具，功能涵盖基础设施创建、Kubernetes 集群部署以及配置管理等多个方面。

功能描述

1.支持使用通用 OS 在 libvirt、openstack 虚拟化场景下自动化部署基础设施与 kubernetes 集群、扩展工作节点、销毁基础设施与 kubernetes 集群。

2.支持使用通用 OS 在裸金属场景下自动化安装操作系统与部署 kubernetes 集群、扩展工作节点。

3.支持 containerd、cri-o、docker、iSulad 容器运行时。

应用场景

适用于企业内部快速部署容器化平台，无缝支持微服务架构的应用开发与管理。该方案助力软件研发团队高效构建开发、测试环境，深度集成 CI/CD 流程，显著加速应用迭代周期。同时，为高校及研究机构量身打造易于操控的 Kubernetes 集群管理方案，比较契合教学实验与科研项目的需求。

QSemOS-plugin

QSemOS-plugin 是国创开发的一套适用于嵌入式 OS 开发场景插件，包括软实时、硬实时、混合部署三个插件；面向企业和个人开发者提供工程创建、开发、编译、调试、仿真、烧录等嵌入式开发全流程全家桶，帮助开发者迅速构建自定义开发环境和简化开发过程。

功能描述

1、**嵌入式软实时插件：**

软实时插件是一个辅助嵌入式软实时操作系统内核、Yocto 工程及应用开发的可视化开发工具，可一键实现嵌入式软实时 OS（linux 内核、bootloader、rootfs、OpenEuler Yocto）和应用（C/C++）开发的可视化辅助工具。同时，用户根据实际场景，通过组合各种工具，形成向导化、可视化集成开发的工具链，提高开发效率的效果。

2、**嵌入式混合部署插件：**

混合部署插件主要实现 OS 的部署业务，为了快速部署基于 Jailhouse 和 xvisor 框架下的虚拟化环境，在部署前准备好 host 和 guest 的配置文件及其相关镜像文件，用户可以通过 IDE 直接使用。

3、**嵌入式硬实时插件：**

硬实时插件能够在可视化图形界面下实现 UniProton 和 zephyr 实时操作系统的编译、烧录，调试等功能，达到利用工具链结合业务使用需求，提高开发效率。

应用场景

面向机器人、工控系统、高端数控机床的嵌入式系统开发工具链，为工业领域操作系统相关开发、安全、测试、验证、迁移、调优、运维管理的全套工具链支持。

8. 著作权说明

openEuler 白皮书所载的所有材料或内容受版权法的保护，所有版权由 openEuler 社区拥有，但注明引用其他方的内容除外。未经 openEuler 社区或其他方事先书面许可，任何人不得将 openEuler 白皮书上的任何内容以任何方式进行复制、经销、翻印、传播、以超级链路连接或传送、以镜像法载入其他服务器上、存储于信息检索系统或者其他任何商业目的的使用，但对于非商业目的的、用户使用的下载或打印（条件是不得修改，且须保留该材料中的版权说明或其他所有权的说明）除外。

9. 商标

openEuler 白皮书上使用和显示的所有商标、标志皆属 openEuler 社区所有，但注明属于其他方拥有的商标、标志、商号除外。未经 openEuler 社区或其他方书面许可，openEuler 白皮书所载的任何内容不应被视作以暗示、不反对或其他形式授予使用前述任何商标、标志

的许可或权利。未经事先书面许可，任何人不得以任何方式使用 openEuler 社区的名称及 openEuler 社区的商标、标记。

10.附录

附录 1：搭建开发环境

环境准备	地址
下载安装 openEuler	https://openeuler.org/zh/download/
开发环境准备	https://gitee.com/openeuler/community/blob/master/zh/contributors/prepare-environment.md
构建软件包	https://gitee.com/openeuler/community/blob/master/zh/contributors/package-install.md

附录 2：安全处理流程和安全批露信息

社区安全问题披露	地址
安全处理流程	https://gitee.com/openeuler/security-committee/blob/master/security-process.md
安全披露信息	https://gitee.com/openeuler/security-committee/blob/master/security-disclosure.md