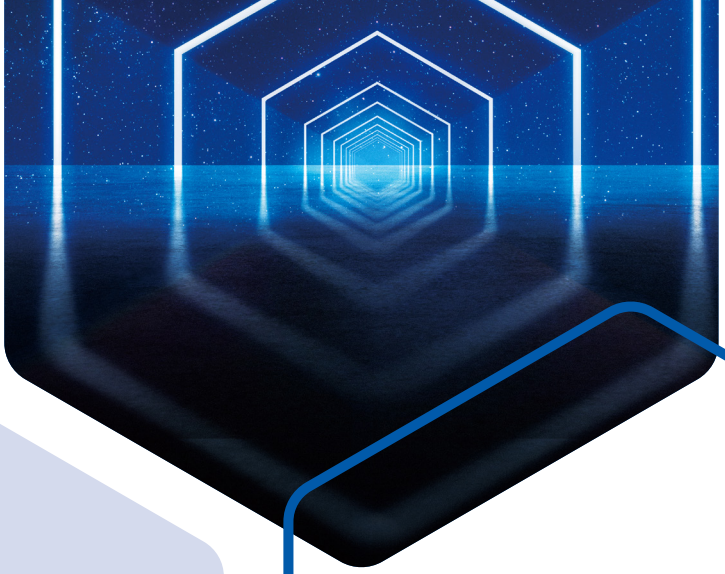
# openEuler 25.09

# 技术白皮书



# CONTENTS

# 01 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。

2024 年 12 月 30 日，发布 openEuler 24.03 LTS SP1，基于 6.6 内核的 24.03-LTS 版本增强扩展版本（参见版本生命周期），面向服务器、云、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。

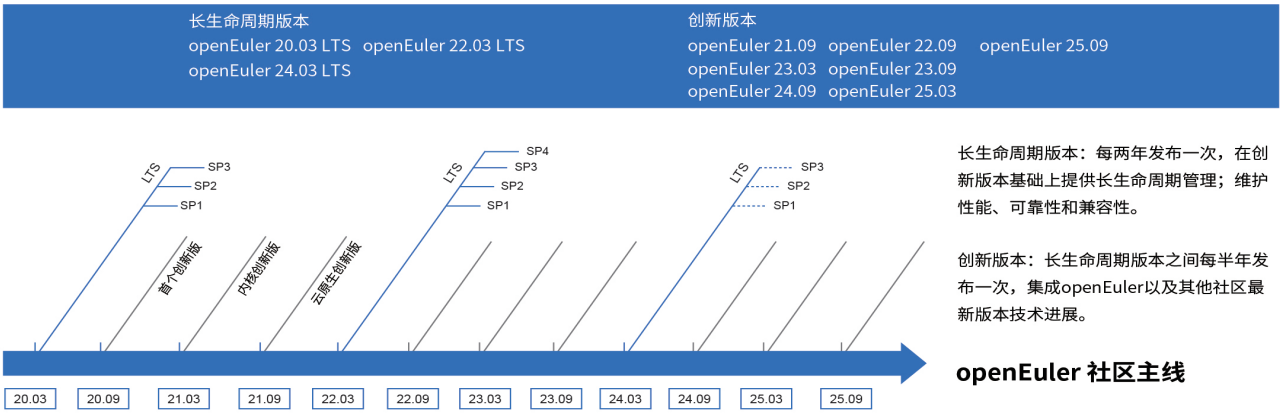
2025 年 3 月 30 日，发布 openEuler 25.03 是基于 6.6 内核的创新版本，面向服务器、云、边缘计算和嵌入式场景，提供更多新特性和功能，给开发者和用户带来全新的体验，服务更多的领域和更多的用户。

2025 年 6 月 30 日，发布 openEuler 24.03 LTS SP2，基于 6.6 内核的 24.03-LTS 版本增强扩展版本（参见版本生命周期），面向服务器、云、AI 和嵌入式场景，持续提供更多新特性和功能扩展，包括内核优化、异构协同推理、众核高密、机密容器、多核多实例混部等，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。

2025 年 9 月 30 日，发布 openEuler 25.09，基于 6.6 内核创新版本，面向服务器、云、AI 和嵌入式场景，持续提供更多新

特性和功能扩展，包括内核优化、异构协同推理、众核高密、机密容器、机密虚机、多核多实例混部、编译器、可信计算、密码加速等，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。

openEuler 版本管理

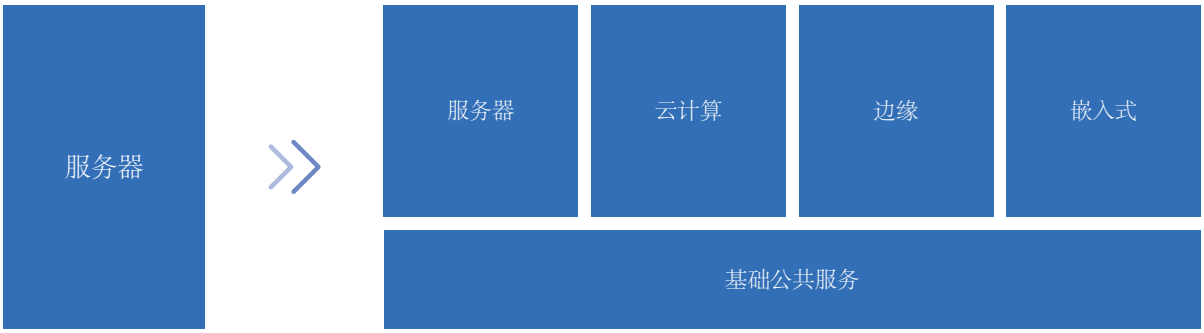



openEuler 作为一个操作系统发行版平台，其 LTS 版本为企业级用户提供一个安全稳定可靠的操作系统。

openEuler 也是一个技术孵化器。通过发布创新版，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。

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。

# 02 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 25.09 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 25.09 版本集成分布式软总线、K8s 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%。
- 机密容器方案：iSulad+Kuasar+secGear 机密容器方案，提供兼容云原生生态的隐私数据保护方案。
- 机密计算方案：基于 ARM CCA 机密计算架构规范，首个支持 CCA 机密虚机的社区版本，提供机密虚机内数据和代码的机密性和完整性保护，即使面对拥有特权的基础设施软件或云服务提供商，也能得到有效保护。

### 全场景

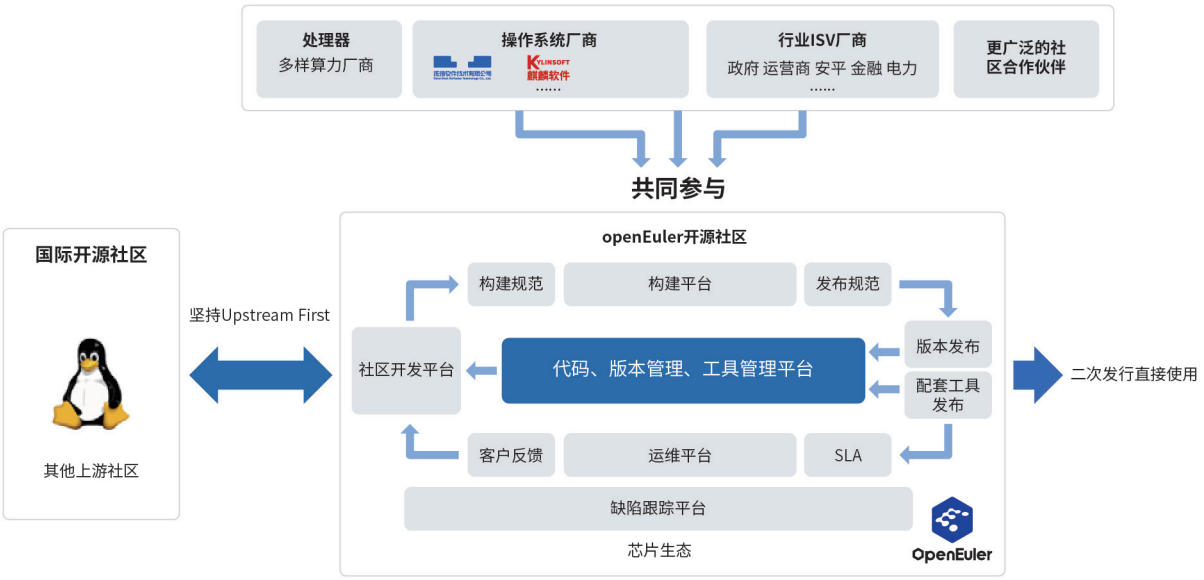
- 边缘计算：发布面向边缘计算场景的版本，支持 K8s 边云协同框架，具备边云应用统一管理和发放等基础能力。
- 嵌入式： 发布面向嵌入式领域的版本，该版本镜像大小 < 5M，启动时间 < 5s；弹性虚拟化底座以及混合关键性部署框架，支持多核多实例同时部署，跨 OS 通信为不同 OS 之间提供一套基于共享内存的高效通信机制。
- AI 原生 OS: OS 使能 AI 软件栈，开箱即用; 异构融合内存，调度，训推场景降本增效; 智能化交互平台，赋能开发者及管理员。

### 繁荣社区生态

- 友好桌面环境：UKUI、DDE 、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。
- DevStation 开发者工作站: 基于 openEuler 的智能 LiveCD 桌面版开发者工作站，旨在提供开箱即用、高效安全的开发环境，打通从部署、编码、编译、构建到发布的全流程，通过新增 MCP AI 智能引擎，快速完成社区工具链调用，实现从基础设施搭建到应用开发的效率飞跃。

## 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

openEuler 社区当前已与多个设备厂商建立丰富的南向生态，比如 Intel、AMD 等主流芯片厂商的加入和参与，openEuler 全版本支持 x86、Arm、RISC-V 三种架构，并支持多款 CPU 芯片，包括兆芯开先 / 开胜系列、Intel Sierra Forest/Granite Rapids、AMD EPYC 4/5 等芯片系列，支持多个硬件厂商发布的多款整机型号、板卡型号，支持网卡、RAID、FC、GPU&AI、DPU、SSD、安全卡七种类型的板卡，具备良好的兼容性。

支持的 CPU 架构如下

| 硬件类型 | x86_64 | arm64 | RISC-V |
| --- | --- | --- | --- |
| CPU | Intel、AMD、Hygon、兆芯 | 鲲鹏、飞腾 | Sophgo、THead 等 |

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

# 03 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 25.09 为 HostOS，组件版本如下：

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

2. 兼容的虚拟机列表如下：

| HostOS | GuestOS(虚拟机) | 架构 |
| --- | --- | --- |
| openEuler 25.09 | Centos 6 | x86_64 |
| openEuler 25.09 | Centos 7 | aarch64 |
| openEuler 25.09 | Centos 7 | x86_64 |
| openEuler 25.09 | Centos 8 | aarch64 |
| openEuler 25.09 | Centos 8 | x86_64 |
| openEuler 25.09 | Windows Server 2016 | x86_64 |
| openEuler 25.09 | 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 |

# 04 场景创新

# AI 场景创新

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

## sysHAX 大语言模型异构协同加速运行时

### 功能描述

sysHAX 大语言模型异构协同加速运行时专注于单机多卡环境下大模型推理任务的性能提升，针对鲲鹏 +xPU（GPU、NPU 等）的异构算力协同，显著提升大模型的吞吐量和并发量：

- 异构融合调度：支持在 GPU 侧任务满载时，动态将推理请求的 prefill 阶段在 GPU 上执行，decode 阶段放在 CPU 上执行。

### 应用场景

sysHAX 大语言模型推理优化方案当前支持 DeepSeek、Qwen、baichuan、Llama 等 transformer 架构的模型。主要适用于以下典型场景：

- 数据中心场景：sysHAX 通过上述技术，利用 CPU 填充推理任务，充分利用 CPU 资源，增加大模型并发量与吞吐量。
- sysHAX 使用方式请参考 sysHAX 部署指南：https://gitee.com/openeuler/sysHAX/blob/dev/docs/sysHAX_online_deployment_guide.md

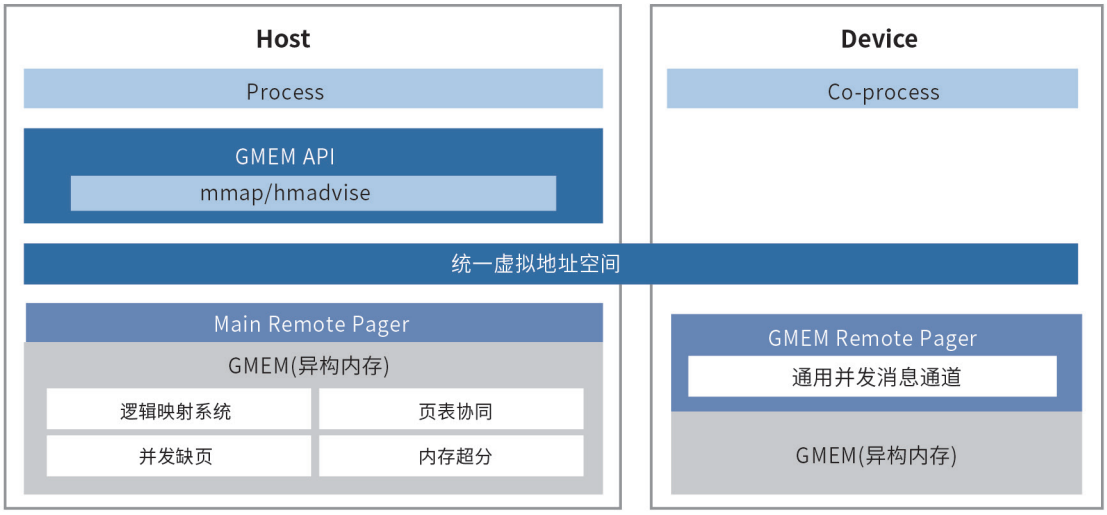
# 异构融合 GMem

在后摩尔时代，GPU、TPU 和 FPGA 等专用异构加速器设备正不断涌现，它们与 CPU 类似，需要将数据放在本地内存（例如 LPDDR 或 HBM）中以提高计算速度。加速器厂商们也不可避免地需要开发复杂的内存管理系统。 现行加速器内存管理方案存在诸多缺陷：

- CPU 侧内存管理与加速器侧分离，数据显式搬移，加速器内存管理的易用性和性能难以平衡。
- 大模型场景下加速器设备 HBM 内存（Hign BandWidth Memory）严重不足，现有的手动 swap 方案性能损耗大且 通用性差。
- 搜推、大数据场景存在大量无效数据搬移，缺少高效内存池化方案。

Linux 现有的 HMM 框架，编程复杂度高且依赖人工调优，性能和可移植性差，引发 OS 社区反弹，最终导致 HMM 方案搁浅。异构加速器领域亟需高效的统一内存管理机制。异构通用内存管理框架 GMem （Generalized Memory Management），提供了异构内存互联的中心化管理机制，且 GMem API 与 Linux 原生内存管理 API 保持统一，易用性强，性能与可移植性好。加速器使用 GMem API 将内存接入统一地址空间后，可自动获得 GMem 面向异构内存编程优化的能力。与此同时，加速器驱动无需重复实现内存管理框架，大幅降低开发维护带来的成本。开发者使用一套统一申请、释放的 API，即可完成异构内存编程，无需处理内存搬移等细节。在加速器 HBM 内存不足时，GMem 可将 CPU 内存作为加速器缓存，透明地超分 HBM，无需应用手动 swap。GMem 提供高效免搬移的内存池化方案，当内存池以共享方式接入后，可解决数据反复搬移的痛点。

## 功能描述




GMem 革新了 Linux 内核中的内存管理架构，其中逻辑映射系统屏蔽了 CPU 和加速器地址访问差异，remote_pager 内存消息交互框架提供了设备接入抽象层。在统一的地址空间下，GMem 可以在数据需要被访问或换页时，自动地迁移数 据到 OS 或加速器端。

### 异构内存特性

为了结合加速器算力与 CPU 通用算力，实现统一的内存管理和透明内存访问，GMem 设计了统一虚拟内存地址空间机制，将原本的 OS 与加速器并行的两套地址空间合并为统一虚拟地址空间。

GMem 建立了一套新的逻辑页表去维护这个统一虚拟地址空间，通过利用逻辑页表的信息，可以维护不同处理器、不同微架构间多份页表的一致性。基于逻辑页表的访存一致性机制，内存访问时，通过内核缺页流程即可将待访问内存在主机与加速器进行搬移。在实际使用时，加速器可在内存不足时可以借用主机内存，同时回收加速器内的冷内存，达到内存超分的效果，突破模型参数受限于加速器内存的限制，实现低成本的大模型训练。

通过在内核中提供 GMem 高层 API，允许加速器驱动通过注册 GMem 规范所定义的 MMU 函数直接获取内存管理功能， 建立逻辑页表并进行内存超分。逻辑页表将内存管理的高层逻辑与 CPU 的硬件相关层解耦，从而抽象出能让各类加速器复用的高层内存管

理逻辑。加速器只需要注册底层函数，不再需要实现任何统一地址空间协同的高层逻辑。

- **remote Pager 内存消息交互框架**

Remote Pager 作为 OS 内核外延的内存管理框架，设计并实现了主机和加速器设备之间协作的消息通道、进程管理、 内存交换和内存预取等模块，由独立驱动 remote_pager.ko 使能。通过 Remote Pager 抽象层可以让第三方加速器很容易地接入 GMEM 系统，简化设备适配难度。

- **用户 API**

用户可以直接使用 OS 的 mmap 分配统一虚拟内存，GMem 在 mmap 系统调用中新增分配统一虚拟内存的标志（MMAP_ PEER_ SHARED）。

同时 libgmem 用户态库提供了内存预取语义 hmadvise 接口，协助用户优化加速器内存访问效率（参考 https:// gitee.com/ openeuler/libgmem） 。

- **约束限制**

目前仅支持 2M 大页，所以 host OS 以及 NPU 卡内 OS 的透明大页需要默认开启。

通过 MAP_PEER_SHARED 申请的异构内存目前不支持 fork 时继承。

GMem 使用方法可参考以下链接：

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

## 应用场景

- **异构统一内存编程**

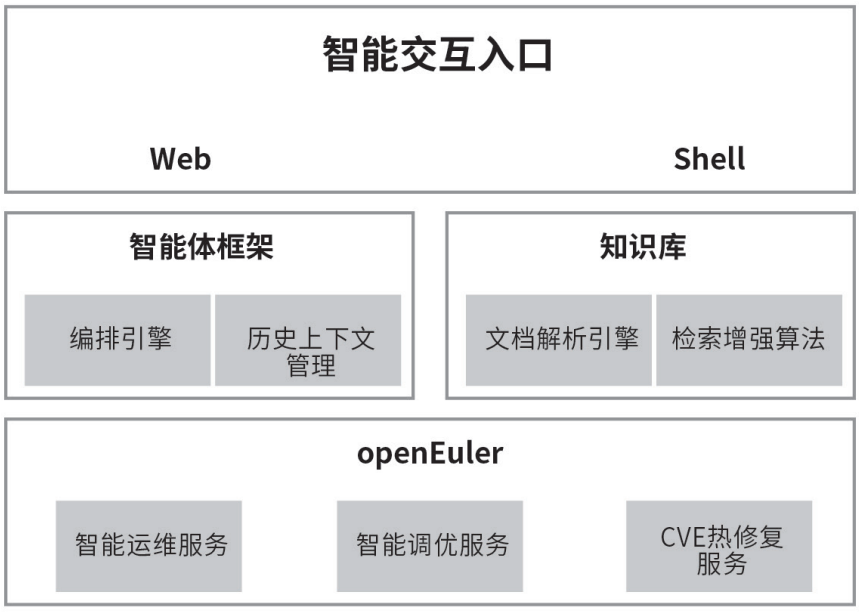
在面向异构内存编程时，使用 GMEM 可分配 CPU 和加速器之间的统一虚拟内存，CPU 内存与加速器内存可共享一个指针，显著降低了异构编程复杂度。当前基于 NPU 试点，驱动仅需百行修改即可接入 GMEM，替换原有约 4000 行内存管理框架代码。

- **加速器内存自动超分**

使用 GMEM 接口分配内存时，将不受加速器的物理内存容量所限制，应用可以透明地超分内存（当前上限为 CPU 的 DRAM 容量）。GMEM 将较冷的设备内存页换出到 CPU 内存上，拓展了应用处理的问题规模，实现高性能、低门槛训推。 通过 GMEM 提供的极简异构内存管理框架，在超大模型训练中，GMEM 性能领先 NVIDIA-UVM。随着内存使用量增长，领先比例不断提升，在超分两倍以上时可领先 NVIDIA-UVM 60% 以上（数据基于 NPU-Ascend910 与 GPU-A100 硬件，在 相同 HBM 内存条件下测试）。

## AI for OS

当前，openEuler 和 AI 深度结合，一方面，基于 openEuler 操作系统，研发出了 openEuler Intelligence，初步实现基于知识库的智能问答、基于语义接口的工作流编排、基于 mcp 的 Agent 构建等功能，在此基础上，openEuler Intelligence 集成了部分系统服务，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：用户可以通过 Web 入口以可视化的形式进行知识库的构建和使用、知识库准确率的自动化测试与评估结果获取、openapi 形式的语义接口注册、基于语义接口、工作流应用的构建和使用、mcp 的注册安装和激活和基于 mcp 的 Agent 应用的构建和使用，Web 入口便携了新手用户对 openEuler 知识的获取和对 openEuler AI 能力的使用。
- 智能 Shell 入口：用户可以通过 Shell 入口调用智能体框架中的智能问答或者是预集成的 Agent 应用，Shell 入口便携了运维人员对操作系统的使用，能通过亲和 openEuler 的形式进行智能交互，降低运维成本。

#### 智能规划、调度和推荐

- 智能规划：openEuler Intelligence 的 Agent 应用可以基于用户的输入和当前可用的工具实时规划运行步骤，直至完成用户目标或者达到步骤执行上限。
- 智能调度：openEuler Intelligence 支持用户在一个工作流应用中定义多个工作流，基于用户的查询，openEuler Intelligence 会自动的提取参数且选择最为合适的工作流进行工作。
- 智能推荐：openEuler Intelligence 基于用户的查询和工作流的运行结果，推荐用户接下来可能会使用的工作流，增加任务的完成概率，简便应用的使用。

#### 工作流应用

- 语义接口：语义接口是指含有自然语言注释的接口形式，openEuler Intelligence 提供了两种语义接口注册方式：首先，

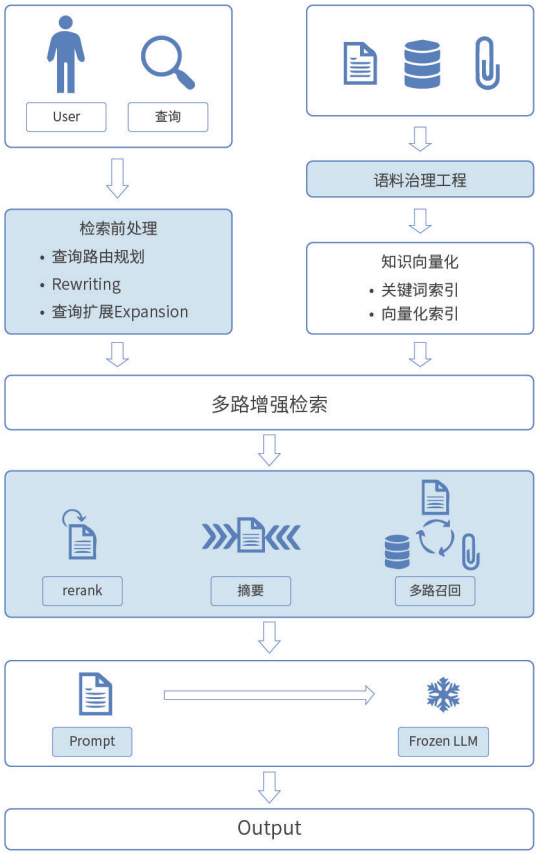
openEuler Intelligence 允许用户将 api 接口以 openapi（3.0+）yaml 文件形式注册到系统中，用户需要在编写 openapi yaml 文件时，对接口添加自然语言注释，后续在这些接口调用过程中，大模型会根据接口的注释对接口进行选择和填参；其次，openEuler Intelligence 也允许用户将功能以 python 编码的形式注册到 openEuler Intelligence 中，这种形式对于 openEuler Intelligence 而言更为亲和。以上两种形式产生的语义接口最终都可以通过可视化形式编排为工作流。

- 工作流编排及调用：openEuler Intelligence 允许用户将系统提供的语义接口以及用户注册的语义接口以可视化的形式连线成工作流，并支持用户对工作流进行调试且以应用的形式进行发布和使用，工作流在调试和使用过程中会展示中间结果，降低用户调试成本，提升用户交互体验。

### Agent 应用

- mcp 注册、安装和激活：mcp 是当前比较主流的一种 AI 相关协议，它支持用 sdk 将复杂多样服务统一封装，带有天然的语义信息，并支持 AI 对基于 mcp 改造后服务下的工具进行比较便捷的调用。
- Agent 构建和应用：当前 intelligence 支持以 mcp 和不同的大模型结合构建 Agent，这些构建完成的 Agent 可以基于配置的大模型信息以及后续用户输入的目标，将用户的目标拆解成阶段性需求，最后使用 mcp 服务下的工具将阶段性需求完成，直至达成用户的目标。

## RAG




- 检索前处理：对于信息缺失的用户查询，基于历史上下文和用户意图，提供查询改写的能力，提升检索准确率；
- 知识索引：对于格式和内容多样化的文档，提供摘要、文本特征提取、树形解析、ocr 等文档解析能力，建立片段特征索引，提升检索增强命中率；

- 检索增强算法：对于多样化的检索场景，提供 8 种检索方法，其中基于动态关键字权重的检索方法和基于大模型过滤的检索方法能极大提升开箱及用的准确率。
- 检索后处理：对于上下文缺失的片段，提供均匀随机化上下文补全的方法，增加片段信息完整度，降低最终模型拟合幻想程度；对于 token 阈值上限的片段（集合），提供基于杰卡德距离的 rerank 方法、基于通用词去除及随机丢弃的 token 压缩方法和基于二分的 token 截断方法，允许一些资源受限的场景也能正常进行智能问答。

通过以上能力，相较传统的 RAG 技术，openEuler Intelligence 中的 RAG 技术能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 Intelligence 中的 RAG 技术的基础能力之一，其通过上下文位置信息提取、文本摘要和 OCR 增强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：

- 上下文位置信息提取：对于文档内容，留存文档相对位置关系，包过片段的全局相对偏移和局部相对偏移，为上下文补全提供基础数据；
- 文本摘要：对复杂文档或者片段，通过滑动窗口 + 大模型对文本进行摘要，为后续多层次检索的方式提供基础数据；
- OCR 增强：对图文混合的文档，基于图片文字内容 + 图片上下文进行摘要，为后续针对图片的提问提供基础数据。
- 文档溯源：openEuler Intelligence 在生成答案的过程中，基于返回的片段和相关的文档信息，在对应的句子右下方生成角注，点击角注可以获取文档的名称和摘要，增加问答生成的可信度。

## 自动化测试

自动化测试是 openEuler Intelligence 中的 RAG 技术的基础能力之一，其通过自动化数据集生成和测试评估识别知识库和检索增强算法等配置的不足：

- 数据集生成：用户选择解析完成的文档，基于用户选择的文档，随机选择部分解析结果，并将这些解析结果发送给大模型，让大模型生成及过滤问答对，最终形成含有查询、标准答案和原始片段的高质量测试数据集；
- 测试评估：基于用户生成的数据集和配置的检索增强算法、大模型、topk 片段限制等参数展开测试，最终基于测试集中的查询、标准答案、原始片段、关联到的片段和大模型拟合结果进行打分，最终使用精确率、召回率、忠实值、可解释性、最长公共子串得分、编辑距离得分和杰卡德相似系数对测试结果进行评估。

## 应用场景

- 面向 openEuler 普通用户：基于 openEuler 社区文献例如白皮书或者用户本地文档，可以构建定制化智能助手。
- 面向 openEuler 开发者: 基于 web 端，开发者可以构建自己的工作流或者 Agent 应用，降低一些场景（代码审计等）重复劳动。
- 面向 openEuler 运维人员：基于 shell 端，运维人员可以通过自然语言与 os 交互，降低复杂命令构造和修改成本。

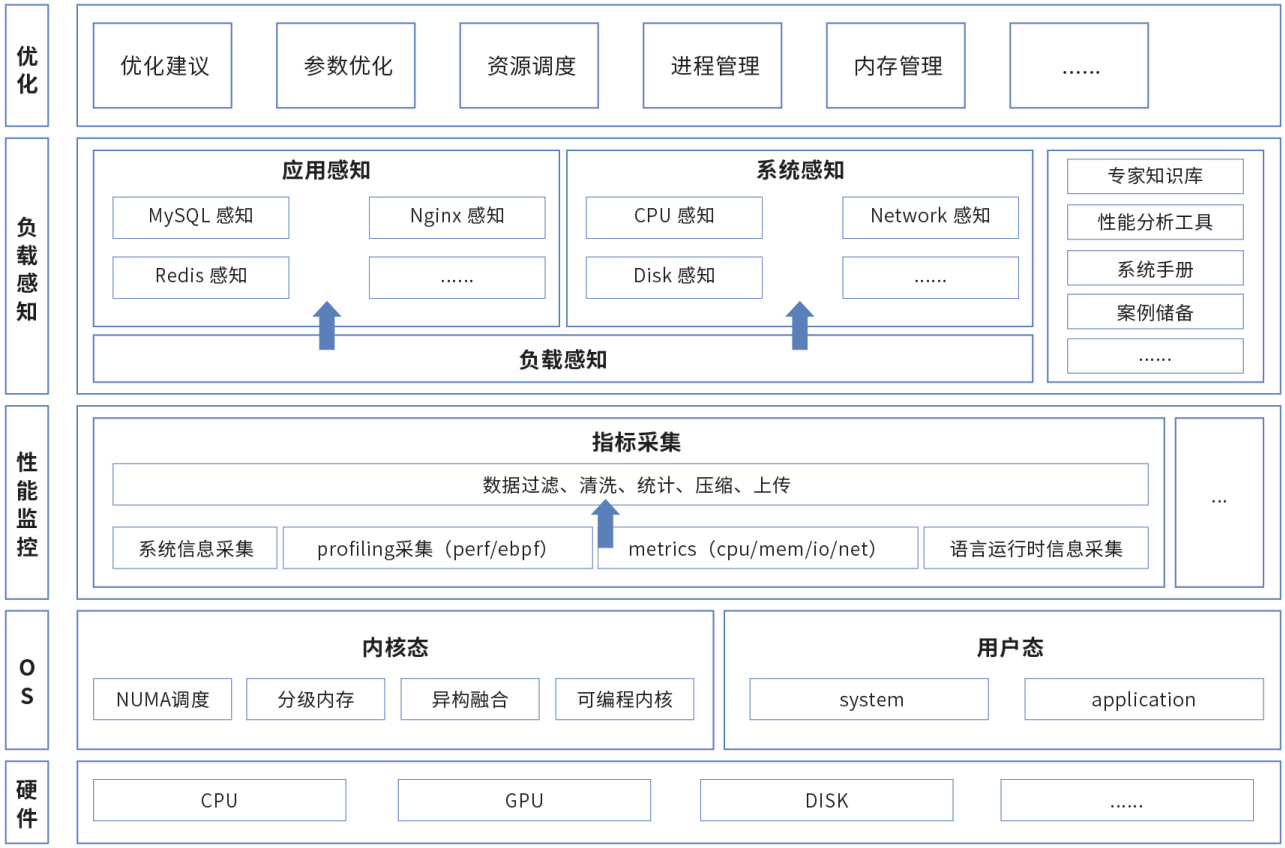
相关使用方式请参考 openEuler-intelligence- 专区：https://www.openeuler.org/zh/projects/intelligence/

# 智能调优

## 功能描述

openEuler Intelligence 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 Intelligence 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。

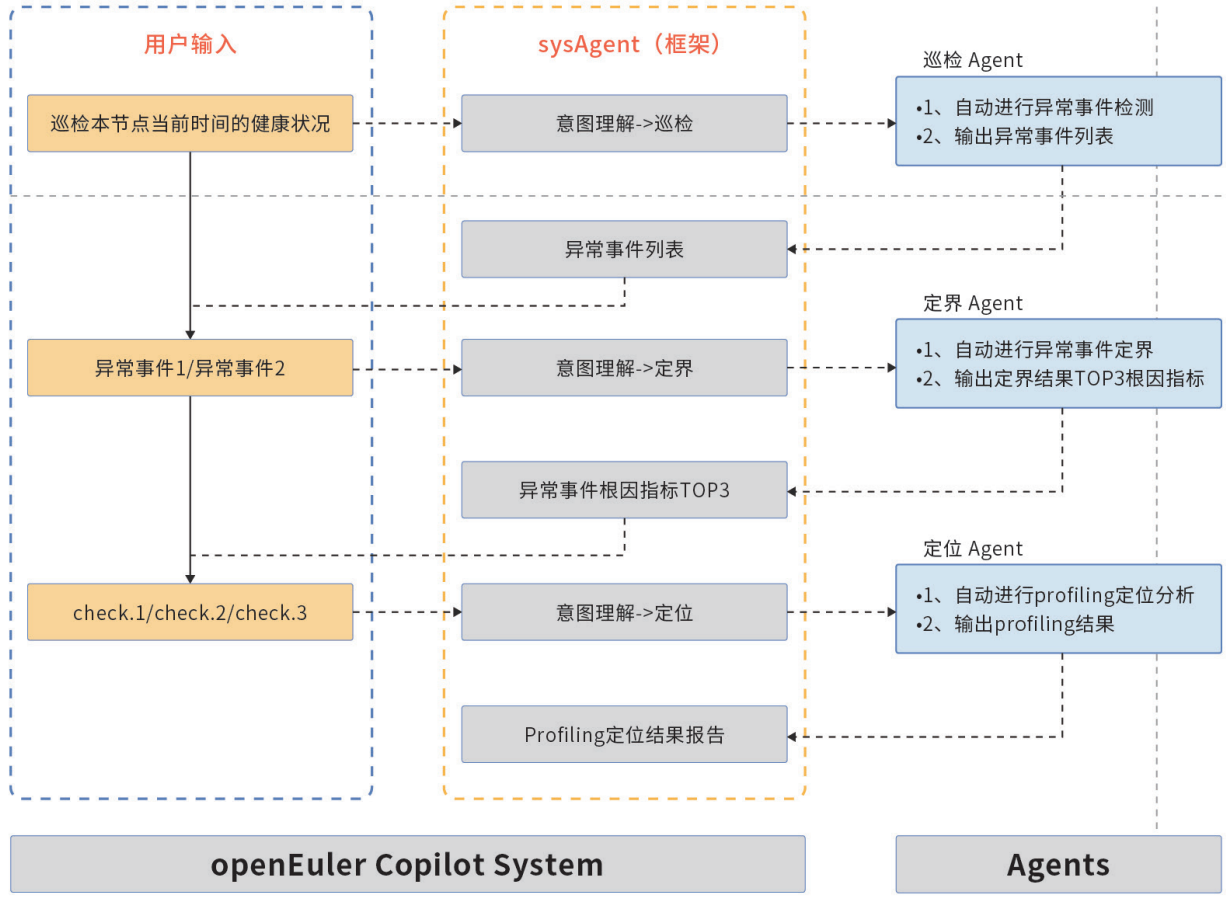



## 应用场景

- 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. 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。
3. 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。
4. 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。

### 应用场景

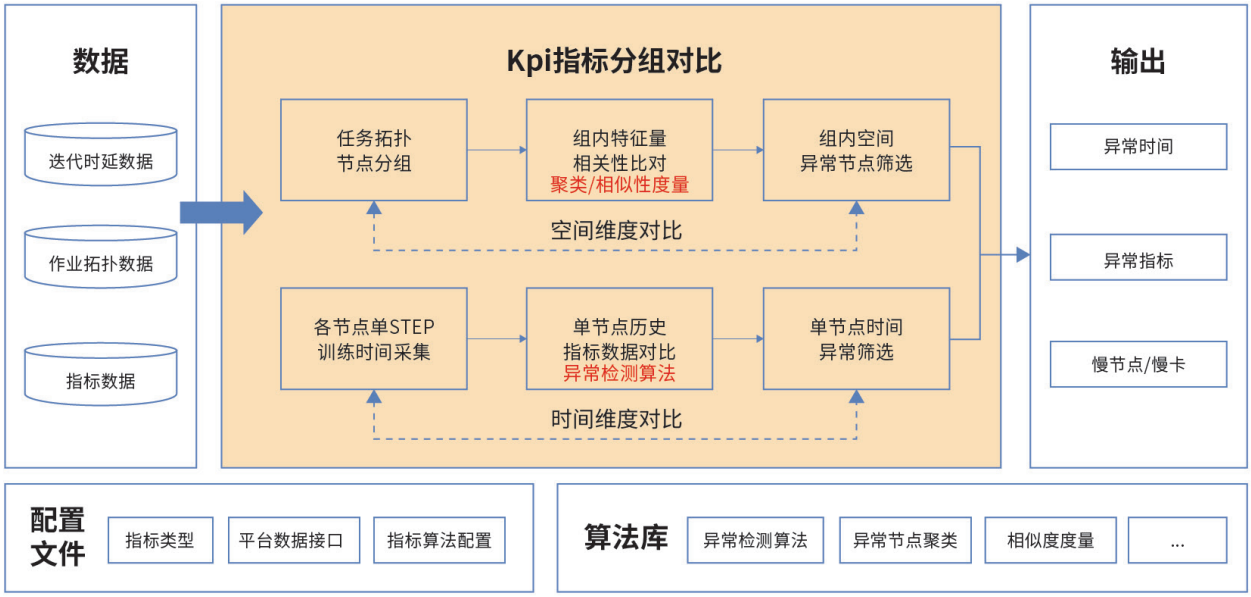
智能诊断接口在本次 openEuler 25.09 版本的功能范围是：具备单机异常事件检测 + 定界 + profiling 定位的能力。

- 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。

# AI 集群慢节点定界

AI 集群在训练过程中不可避免会发生性能劣化，导致性能劣化的原因很多且复杂。现 有方案是在发生性能劣化之后利用日志分析，但是从日志收集到问题定界根因诊断以及现 网闭环问题需要长达 3-4 天之久。基于上述痛点问题，我们设计了一套在线慢节点定界方 案，该方案能够实时在线观测系统关键指标，并基于模型和数据驱动的算法对观测数据进 行实时分析给出劣慢节点的位置，便于系统自愈或者运维人员修复问题。

## 功能描述




基于分组的指标对比技术提供了 AI 集群训练场景下的的慢节点 / 慢卡检测能力。这项技术通过 sysTrace 实现，新增内容包括配置文件、算法库、慢节点空间维度对比算法和慢节点时间维度对比，最终输出慢节点异常时间、异常指标以及对应的慢节点 / 慢卡 ip, 从而提高系统的稳定性和可靠性。该特性主要功能如下：

- 配置文件：主要包括待观测指标类型、指标算法配置参数以及数据接口，用于初始化慢节点检测算法。
- 算法库：包括常用的时序异常检测算法 spot 算法，k-sigma 算法，异常节点聚类算法和相似度度量算法。
- 数据：采集到的各个节点的指标数据，以时序序列表示。
- 指标分组对比：包括组内空间异常节点筛选和单节点时间异常筛选。组内空间异常节点筛选根据异常聚类算法输出异常节点；单节点时间异常筛选根据单节点历史数据进行时序异常检测判断节点是否异常。

## 应用场景

sysTrace 支持慢节点异常检测，告警展示，异常信息落盘。

AI 模型训练场景 : 适用于 AI 模型大规模集群训练任务，通过该功能可快速发现慢节点，便于系统自愈或者运维人员修复问题。

AI 模型推理场景：适用于单一模型的多实例性能劣化检测，通过多实例间应用资源的对比情况，可快速发现性能劣化的实例，便于推理作业的调度和资源利用率的提升。

## 智能容器镜像

### 功能描述

openEuler Intelligence 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。

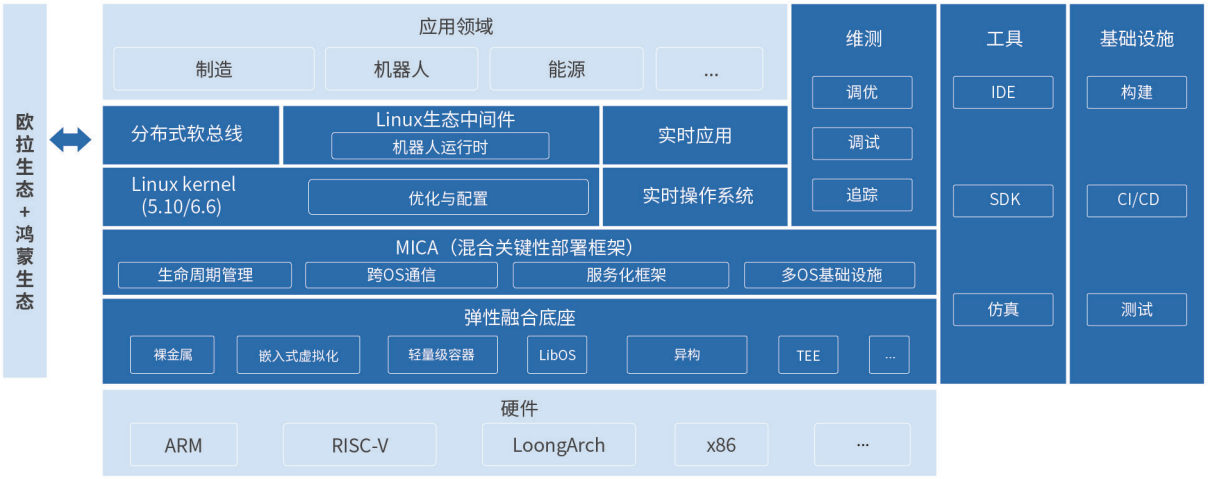
# 嵌入式场景创新

openEuler 发布面向嵌入式领域的版本 openEuler 25.09，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。在软件包生态方面，回合了 oebridge 特性，支持在线一键安装 openEuler 镜像仓软件包，并支持在 Yocto 镜像构建时通过 oebridge 直接安装 openEuler RPM 包快速实现镜像定制。此外，还扩展支持了 oeDeploy 特性，能够快速完成 AI 软件栈、云原生软件栈的部署。在内核支持方面，持续完善了 meta-openEuler 的内核配置，配合 oeAware 实时调优功能实现干扰控制以增强系统实时性。

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/Zephyr）的生命周期管理、跨 OS 通信。

## 北向生态

1. 北向软件包支持：700+ 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 openEuler 嵌入式设备之间互联互通、 openEuler 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。

## 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。

## 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。

# 05 内核创新

# openEuler 内核中的新特性

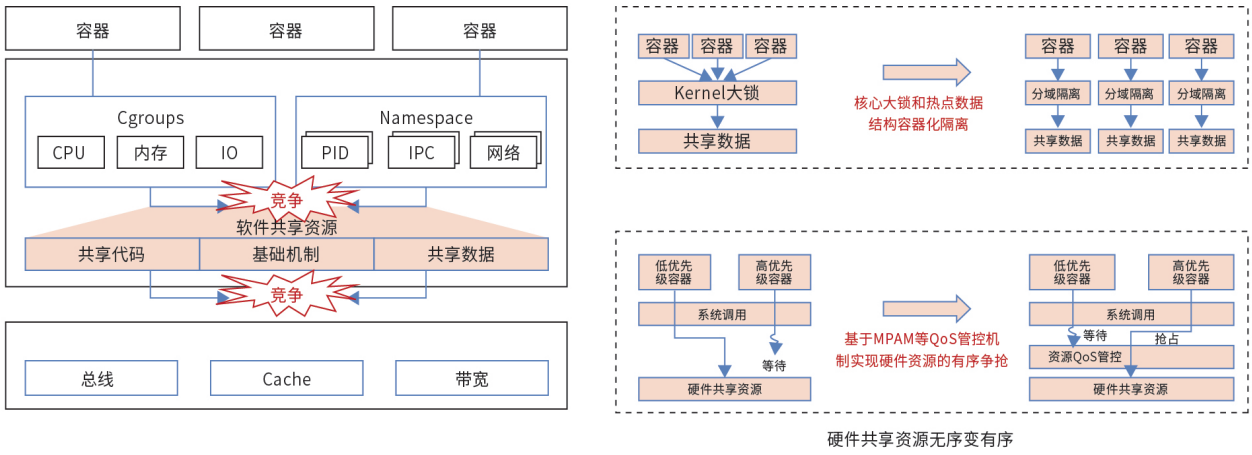
openEuler 25.09 基于 Linux Kernel 6.6 内核构建，在此基础上，同时吸收了社区高版本的有益特性及社区创新特性。

- fuse passthrough 特性支持：当前 fuse 在分布式存储、AI 中广泛使用。在直通场景中 fuse 用户态文件系统没有对读写 IO 进行额外处理，仅仅是记录元数据，向后端文件系统发起 IO。此时 fuse 的处理流程成为了整个系统的 IO 瓶颈。 fuse passthrough 特性旨在 fuse 直接对接后端文件系统（透传）的场景下，消除数据面 fuse 上下文切换、唤醒、数据拷贝开销，允许应用程序直接将读写 IO 在内核中发给后端文件系统，进而大幅提升读写性能。在实验室环境中，fuse passthrough 特性展现出了令人满意的性能提升。具体来说，在 fio 测试中，4K-1MB 粒度的读写测试均有 100%+ 的性能提升。同时通过了故障注入测试和稳定性测试。业务可以按需使用。
- MPAM 增强特性支持：新增 QoS 增强特性，拓展内存带宽和 L3 缓存控制方式，可按照使用量上限 / 保底 / 优先级方式配置，为混部场景动态调控共享资源提供端到端能力。新增 IO QoS 管控特性，联动 SMMU 对外围硬件设备或异构加速器的 IO 带宽流量进行隔离配置，支持 iommu_group 粒度级别监控，为异构场景下 IO QoS 管控方案提供控制侧新方案。此外，新增 L2 缓存隔离配置，提供 L2C 占用量和带宽流量监控能力，为混部场景下系统性能提供物理核级别的优化和分析手段。以上 MPAM 特性在业务实测场景展现出明显的性能提升，其中在混部场景下，Specjbb 作为在线业务的混部干扰率从 25.5% 降低至 5% 以下。

# 06 云化基座

# 众核高密

服务器芯片由多核进入众核时代（>256C），对操作系统提出新的挑战。提升 Rack 计算密度、降低数据中心 TCO，众核服务器已成为互联网行业主流选择，随着云技术和业务规模发展，容器化部署成为互联网行业的主流业务部署形态，在这种场景下，系统串行开销和同步开销限制可扩展性，干扰问题凸显，资源利用率低，影响容器部署扩展性的串行访问开销和同步开销主要来自软硬共享资源争用。




硬件共享资源无序变有序

## 功能描述

本期主要采用轻量虚拟化按 NUMA 分域拆分资源、域内实现资源容器级隔离增强，降低因软硬件资源争用导致的性能干扰，提升容器部署扩展性。关键技术特性如下：

虚拟机内存 QoS 控制：当多个租户的虚拟机（VM）部署于同一物理主机时，若内存密集型虚拟机占用大量内存带宽，可能引发资源争用，导致其他虚拟机无法获得足够的内存带宽以满足其业务性能需求，进而影响整体系统服务质量。基于鲲鹏处理器提供的内存带宽监控与调控能力（MPAM, Memory Power and Access Monitoring），结合操作系统层面的 resctrl（Resource Control）机制，系统可实现对最多 30 个虚拟机的内存带宽使用情况进行精细化监测与动态控制。该能力支持对虚拟机内存带宽的上限、下限及优先级策略进行配置，从而构建多租户环境下的内存带宽资源隔离与保障体系。具体而言：内存带宽上限控制：通过为各虚拟机配置最大内存带宽使用阈值，有效防止单个虚拟机过度占用内存带宽资源，避免对其他租户虚拟机造成性能干扰；内存带宽下限保障：支持设定最低带宽保障值，确保在资源竞争时，实际分配给虚拟机的带宽低于设置的最低带宽时需要提升优先级，实现资源的动态优化与高效利用；优先级调度策略：支持基于业务重要性配置虚拟机的内存带宽优先级，优先保障关键业务虚拟机的带宽稳定供给，提升高优先级工作负载的可用性与服务质量。

虚拟设备 NUMA 亲和：PCI 设备也具有 NUMA 亲和性，在主机侧直接访问，OS 调度系统会根据设备亲和性进行调度优化，以防止跨 NUMA 访问 PCI 设备而造成性能损耗。对于虚拟机设备直通来说，当前还不具备在虚机内部呈现 PCI 设备亲和的 NUMA 节点。本功能基于 PCI 扩展桥 (PXB) 扩展虚拟机 PCI 设备拓扑结构，支持在虚拟机内呈现虚拟设备所在 NUMA，便于系统 OS 优化调度或者用户根据虚拟设备所在 NUMA 部署业务应用，减少跨 NUMA 资源访问导致到性能损耗，提高虚拟机内业务应用性能；

CPU 分域调度：CPU 基于硬件拓扑划分子域部署容器，一个容器一个独立子调度域，实现容器之间干扰隔离，降低跨 cluster cache 同步次数和 cache/NUMA 内存等硬件资源争抢，减轻容器之间的相互干扰。对于 redis 多并发场景性能提升 10%+；

文件系统块分配干扰隔离：优化 ext4 块分配释放流程中的 group lock 和 s_md_lock 两个主要争抢的锁，以提高 EXT4 块分配流程的可扩展性。通过允许在当前目标块组被占用时尝试使用其他空闲块组进行分配，从而减少了多个容器争抢同一个块组造成的 CPU 浪费，并充分利用了 ext4 多块组的优势，缓解了 group lock 的竞争。其次通过将流式配的全局目标拆分为多个，从而减少了全局锁 s_md_lock 的竞争文件数据也更加聚集。在 64 容器并发场景下，块分配和块释放混合场景 OPS 提升 5 倍以上，单块分配场景提升 10 倍以上；

高效 slab 回收：将 slab 内存回收的读写锁，优化为 RCU 无锁化 slab 回收，不同 slab 之间的内存回收互不干扰，回收效率显著增加，多容器并发场景下，系统调用性能显著增强；

网络 tcp hash 干扰隔离：tcp_hashinfo bash、ehash 存在锁竞争，ehash 计算频繁，导致高并发下带宽下降，时延变大。将 tcp_hashinfo bash、ehash 的自旋锁改为 rcu，ehash 计算方式改为 lport 递增减少查询时间和计算次数，减少 tcp connect hash 的锁竞争；

Cgroup 隔离增强：user namespace 通过 percpu counter 替换原来的原子操作，避免不同 namespace 相同父节点竞争访问，消除容器间 rlimit 计数干扰。 解决 will-it-scale/ signal1 用例线性度问题，64 个容器并发吞吐性能提升 2 倍。通过对 memcg 实现批量释放处理，避免大量的小内存释放对于相同父节点计数竞争，提升内存计数的可扩展性，tlb-flush2 测试用例 64 容器吞吐提升 1.5 倍；基于 eBPF 可编程内核能力，提供主机容器信息隔离与过滤机制，高效实现容器资源视图。相较业界 LXCFS 方案，本方案避免了内核态 - 用户态切换开销，消除了 LXCFS 进程的性能与可靠性瓶颈，单容器内资源视图吞吐量在单容器场景下性能提升 1 倍，在 64 容器场景下提升 10 倍；

干扰监测：干扰监测回答的是容器有没有被干扰、是什么干扰、干扰程度如何这三个问题，从结果上看干扰可以分为干扰导致指令得不到执行、干扰导致指令执行变慢和干扰导致指令执行变多三类，干扰监测从内核角度，针对每一类的典型干扰在运行时进行统计，当前支持在线统计 schedule latency、throttling、softirq、hardirq、spinlock、mutex 和 smt 干扰，性能开销在 5% 以内；

鲲鹏内存 /Cache QoS 管控机制 MPAM：内存带宽流量和各级缓存占用量，可按照使用量上限 / 保底 / 优先级方式进行配置，根据不同业务，以线程为粒度部署不同隔离策略。支持业务资源实时监控，在客户业务层面和线程级别，实时对共享资源的使用情况进行跟踪监控，将资源使用情况反馈给控制策略，形成闭环控制效果。此外，MPAM 联动 SMMU 扩展外设 IO QoS 方案，支持对外围设备和异构加速器 IO 带宽流量进行隔离配置，按设备粒度级别进行资源监控。

QoS 策略动态配置：提供集群级别的 mpam Qos 管理插件，基于 mpam 提供的 Qos 接口，在插件中根据用户定义，实现为所有节点自动分解各级别优先级，并根据用户的声明自动设置在离线任务 mpam Qos 优先级，从而实现在混部场景下对资源的充分利用：在线业务繁忙时自动抢占离线任务的 llc 及内存带宽，在线业务空闲时自动释放 llc 及内存带宽资源提升离线业务处理性能。

## 应用场景

众核服务器场景下，业务容器高密度部署场景，通过降低容器间干扰，提升容器部署密度，进而提升资源利用率。

内存密集型场景：适用于多台虚拟机同时竞争带宽的场景，通过该功能可对每台虚拟机内存带宽占用进行监测和控制，保障对性能要求高的虚拟机业务能正常进行。

# 07 特性增强

# LLVM for openEuler 编译器

LLVM for openEuler 编译器基于开源 LLVM 软件进行深度打造，是面向服务器互联网行业、数据中心新应用、通算 AI 新场景和视频编解码等高算力场景的高性能编译器。同时在 openEuler 社区打造国内的 LLVM 基线，提供高性能、安全可靠、易创新的 LLVM 下游社区稳定的发行版，已支持主流的系统语言（C/C++）和芯片架构（X86/AArch64/RISCV/LoongArch/ 申威等）。

## 功能描述

LLVM for openEuler 编译器在 openEuler 25.09 版本引入以下编译特性，优化数据库、大数据相关应用的运行效率，释放软件的极致性能。

**ICP 增强优化：**

ICP 优化通过反馈信息将间接函数调用优化为直接函数调用，增加潜在的内联优化机会，并降低函数调用开销。

**智能哈希预取优化：**

识别应用中多层间接嵌套访存场景，自动计算实际访存地址并插入数据预取指令，降低数据缓存未命中的概率。

**自适应内存拷贝优化技术：**

通过识别内存拷贝时源指针和目标指针特征，增加运行时检查对内存拷贝方式进行特化优化（如生成 memset、memmove 等）。

**动态库快速访问技术：**

传统动态库函数调用需通过 PLT（过程链接表）跳转，导致额外的内存访问和跳转指令，优化为直接使用 GOT（全局偏移表）中的函数地址调用，消除 PLT 跳转开销。

## 应用场景

通过 LLVM for openEuler 编译器的编译优化能力，能够帮助提升数据库、大数据场景中主流应用（如 MySQL、Clickhouse、Doris 等）性能。

# oeDeploy 特性增强

oeDeploy 是一款轻量级的软件部署工具，旨在帮助开发者快速、高效地完成各类软件环境部署，对单节点与分布式场景均可适配。

## 功能描述

**多场景支持 & 主流软件一键部署：**

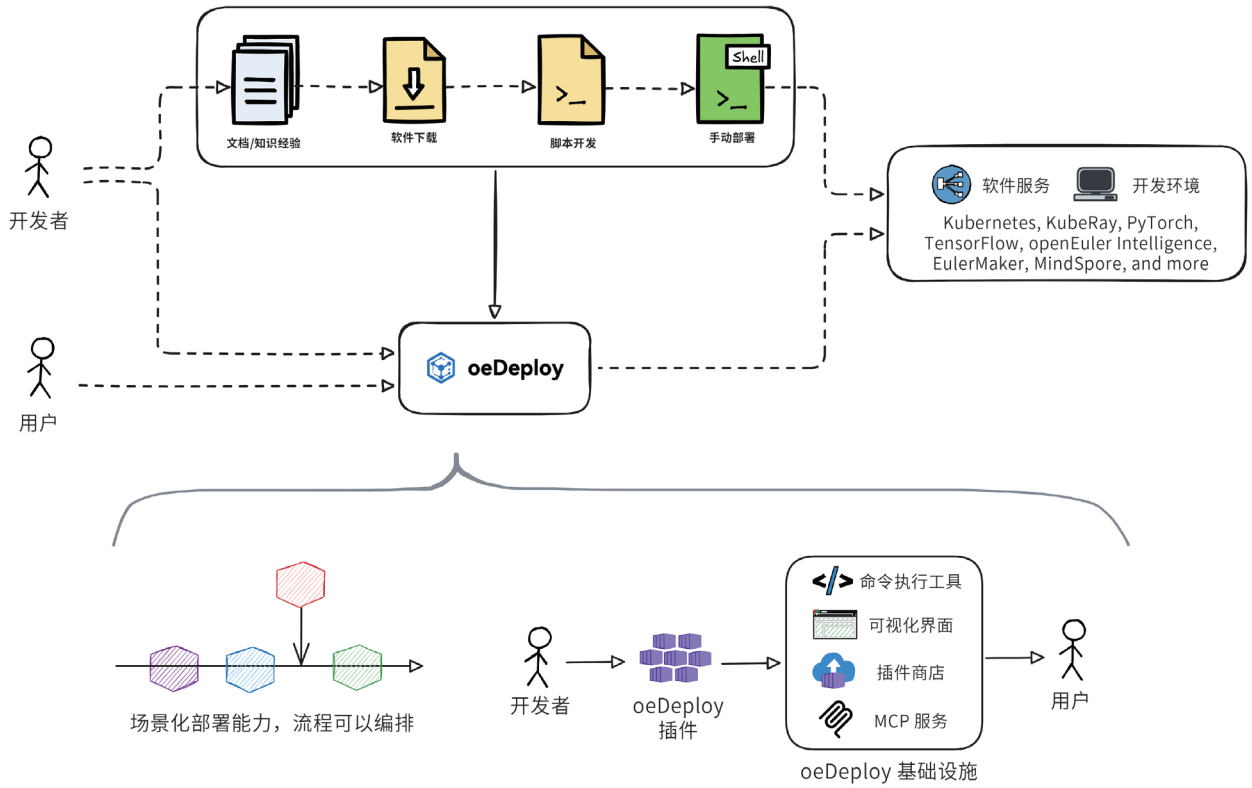
支持单节点应用与集群软件环境的一键部署，新版本 oeDeploy 增加了对多 master 节点的 Kubernetes 环境的快速部署，新增支持了 openEuler Intelligence、Devkit-pipeline 等社区工具链，以及 RAGFlow、anythingllm、Dify 等主流 RAG 软件。

**灵活的插件化管理 & 优秀的部署体验：**

oeDeploy 提供可扩展的插件架构，灵活管理多种部署能力，开发者也可以快速发布自定义部署插件。新版本 oeDeploy 支持了插件源的管理，支持一键更新插件版本、一键完成插件初始化。oeDeploy 支持极简的命令行操作方式，也即将上线可视化工具与插件商店，用更少的代码，实现更高效的软件部署体验。

**高效部署 & 智能开发：**

新版本 oeDeploy 发布了 MCP 服务，在 DevStation 中实现开箱即用，借助大模型的推理能力，支持用自然语言完成各类软件的一键部署，部署效率提升 2 倍；支持将用户文档快速转换成可以直接运行的 oeDeploy 插件，开发效率提升 5 倍。



## 应用场景

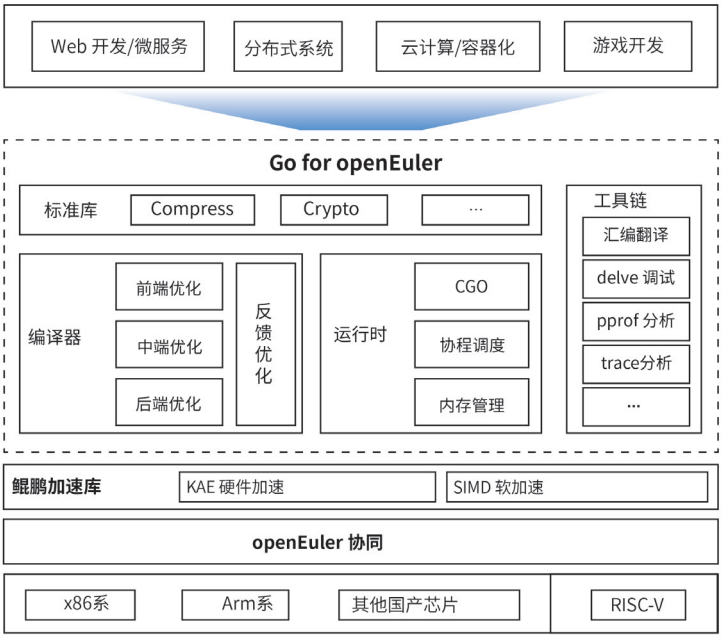
ISV 与开发团队可以将 oeDeploy 作为向客户交付软件产品的统一形式。借助 oeDeploy 提供的命令行工具与插件框架，开发团队只需较少的开发工作量就可以实现优秀的部署效果，降低用户的额外学习成本，提高客户满意度。

面向开发者与维护人员，oeDeploy 提供了复杂软件环境的快速部署能力，可以在几分钟内部主流的 AI 训推框架，大幅降低软件开发门槛，避免了重复操作。同时，开发者可以基于 oeDeploy 发布自定义的部署能力，让更多用户受益于一键部署的便利。借助大模型与 MCP 的能力，oeDeploy 可以让部署流程更加高效，开发过程更加智能。

# Go for openEuler 编译器

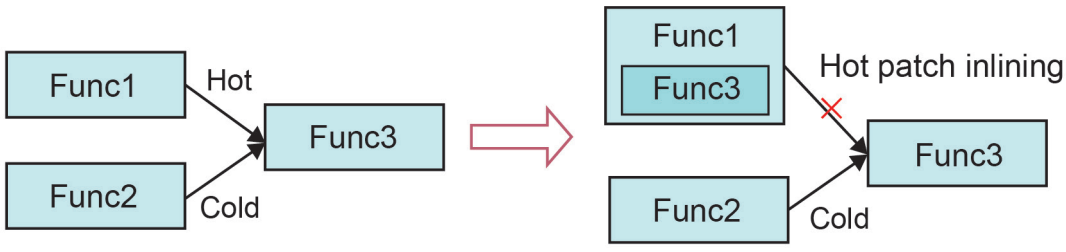
Go for openEuler 是基于开源 Golang 开发，是一款高性能、高可靠、易开发的 Golang 发行版，致力于打造生态兼容、极致体验、亲和 openEuler 的高性能编译器。主要面向云原生、微服务应用等对敏捷开发和运行性能都有要求的容器云场景，围绕业界主流 Go 业务负载进行优化，解决实际业务中由原生 Golang 能力不足导致的性能问题，并适配国产龙芯、鲲鹏等硬件平台，充分释放国产硬件算力。

## 功能描述




### 功能描述 1：CFGO 反馈优化

在保证程序功能不变的前提下，通过收集程序运行时信息，指导编译优化进行更准确的优化决策，获得性能更优的目标程序。基于程序局部性原理，使热指令紧密排布，优化 cache/TLB 命中，有效降低程序前端瓶颈，提升程序性能。



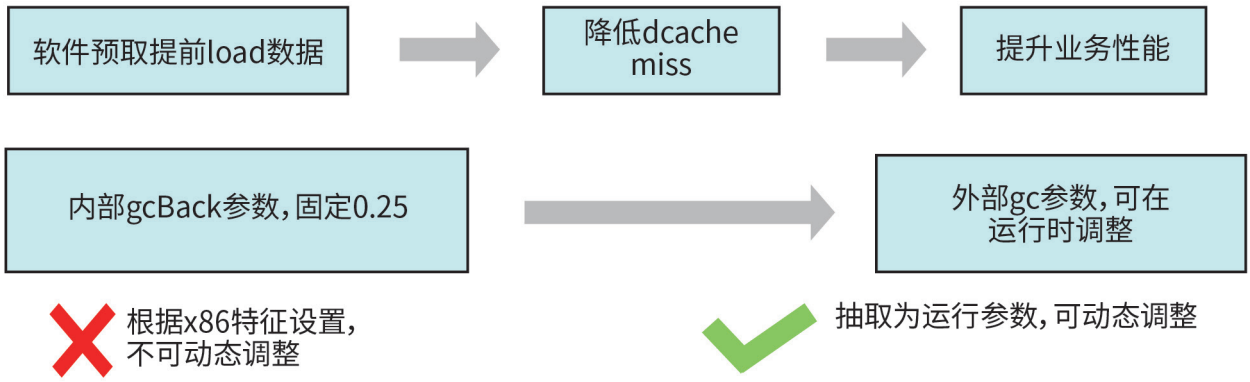

函数内联

### 功能描述 2：ARM 原子指令优化

在部分业务场景中，Golang 运行时调用 CAS 锁、LD/ST 指令开销较大，改为 ARM 亲合指令序列实现，可实现性能提升。

功能描述 3：运行时 GC 优化

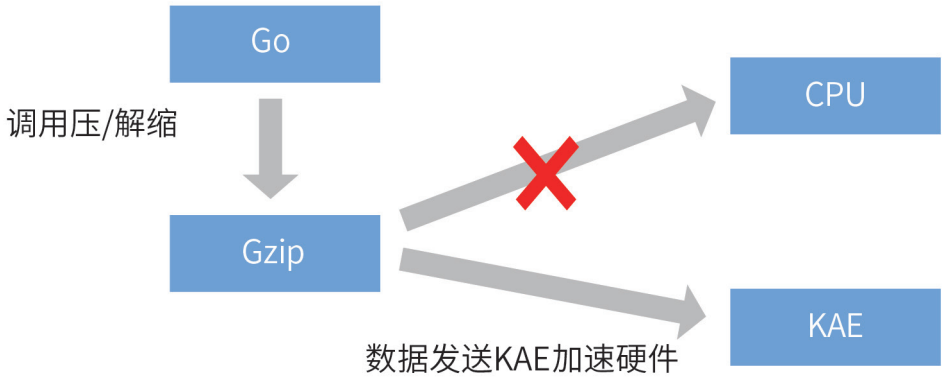
结合特征，插入软件预取；抽取 GC 协程开销资源参数为运行时参数，支持根据不同业务特征进行动态调整。




运行时 GC 优化

功能描述 4：结合鲲鹏 KAE 使能底层硬件加速

改造 Golang 自身 Compress 库 Gzip 的压缩 / 解压缩逻辑实现使能底层硬件加速。




鲲鹏 KAE 使能

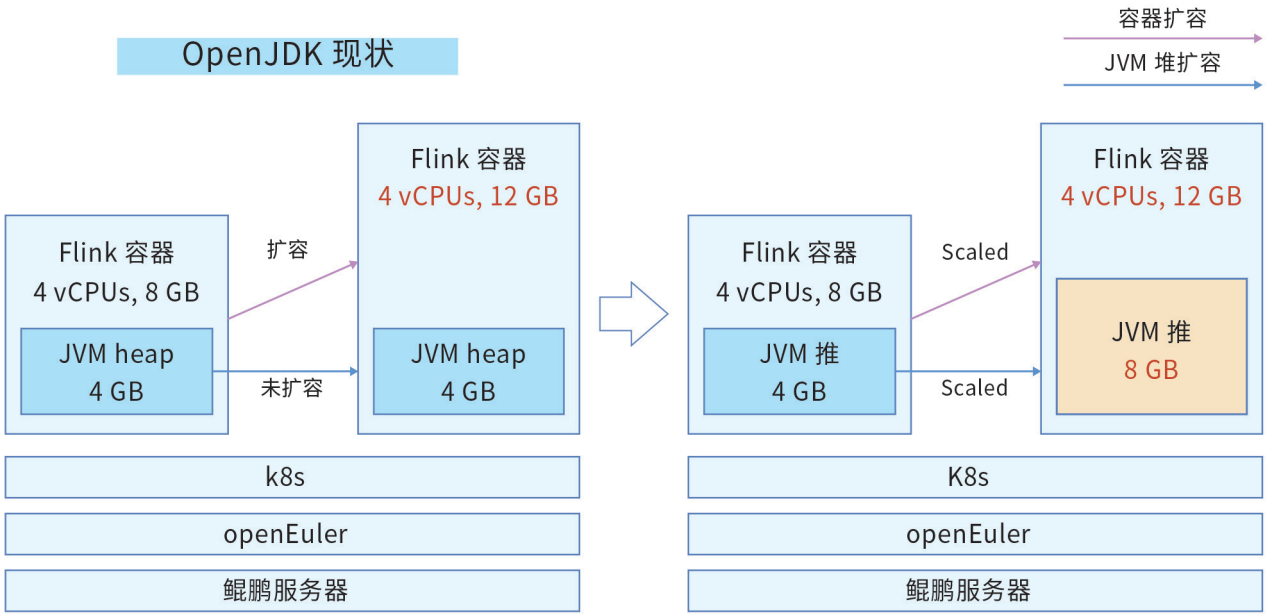
## 应用场景

互联网容器化部署应用的模式下，大部分客户容器场景下容器资源支持垂直伸缩，当前 OpenJDK 的最大堆只能在启动时支持修改，无法支持在线动态扩缩，java 应用无法在线使用到容器扩容出的内存，需要 java 应用启动时重新设置最大堆；鉴于此问题，毕昇 JDK 在 G1GC 实现堆内存上限在线伸缩能力，允许用户在应用运行时动态更新 Java 堆内存的上限，而无需重启 JVM。

# 毕昇 JDK 支持堆内存扩容

## 功能描述

互联网容器化部署应用的模式下，大部分客户容器场景下容器资源支持垂直伸缩，当前 OpenJDK 的最大堆只能在启动时支持修改，无法支持在线动态扩缩，java 应用无法在线使用到容器扩容出的内存，需要 java 应用启动时重新设置最大堆；鉴于此问题，毕昇 JDK 在 G1GC 实现堆内存上限在线伸缩能力，允许用户在应用运行时动态更新 Java 堆内存的上限，而无需重启 JVM。



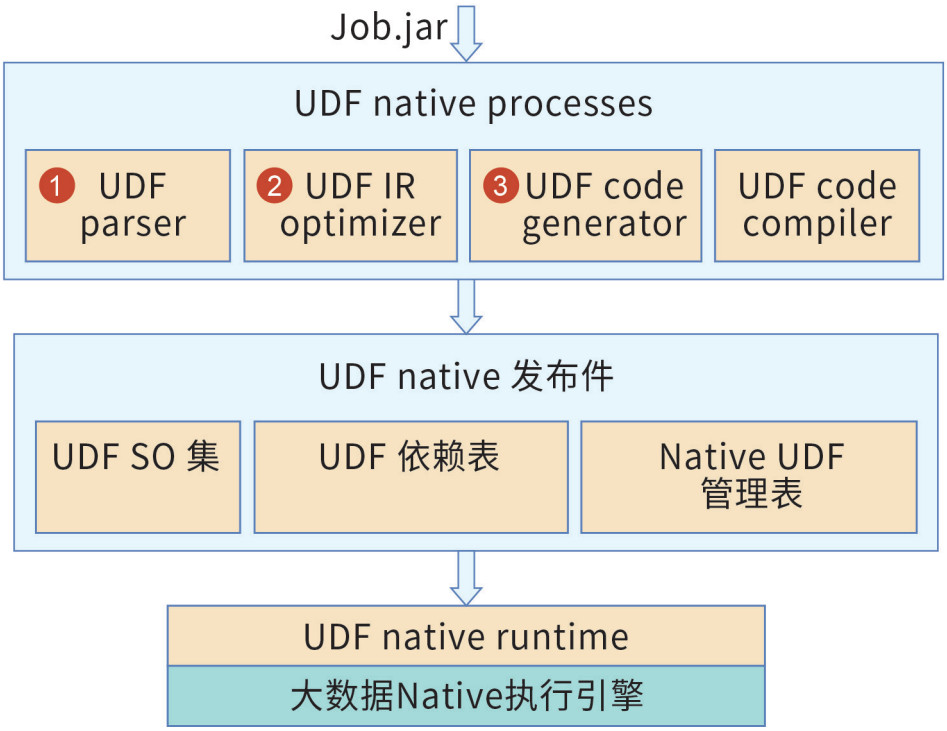

## 应用场景

互联网等容器场景业务，在容器在线扩容后需要 java 业务的堆内存大小也支持在线扩容的场景。

# 编译器 UDF 自动 native 框架

## 功能描述

针对开源大数据 JVM 执行效率低的缺点，UDF 自动 native 框架负责将 Java UDF 自动转换为 C/C++ Native UDF，并进一步从内存高效管理、硬件亲和等维度提升大数据处理性能。UDF 自动 native 框架致力于实现用户无感知、全自动的 Java UDF native 加速机制。UDF 自动 native 框架主要由 UDF parser、UDF IR Optimizer、UDF code Generator、UDF code compiler 等模块组成：




UDF parser 将业务 jar 包字节码自动转换为 IR 代码，并基于 UDF 特征提取出 UDF 代码；UDF IR Optimizer 从内存对象自动管理、硬件亲和加速等维度对 UDF IR 进行优化；UDF Code Generator 将 UDF IR 对等转换为 native 代码；UDF code compiler 将 UDF native 代码在线编译为 native 二进制。最后，UDF 的 native 二进制发布到大数据执行节点上，由大数据系统 native 执行引擎动态加载执行，提升大数据系统处理性能。

## 应用场景

当前版本的 UDF 自动 native 框架适用于对接 Flink 大数据 native 执行引擎，通过配套 Flink Datastream 场景 native 基础库，可以在用户无感知的情况下实现 Flink Datastream UDF 自动 native 加速。

# 毕昇 JDK17 支持退优化可观测

JDK17 的 JFR Streaming API 功能，是 JFR 从“事后静态分析”迈向“实时监控”的关键特性。

在传统的 JFR 使用模式中，流程是：记录 -> 停止记录 -> 转储为 .jfr 文件 -> 用 JMC 离线分析。这种模式是“事后分析”，对于排查已经发生的问题非常有效。

Streaming API 引入了一种全新的模式：它允许 Java 应用程序在不中断 JFR 记录、不生成完整 .jfr 文件的情况下，实时、持续地从 JVM 内部订阅和消费 JFR 事件流。

## 功能描述

在使用 Streaming API 的时候，通过本功能可以在 ****** 处获取当前时间之前的一段 jfr event，例如退优化事件。

```
// 1. 创建一个 RecordingStream
RecordingStream rs = new RecordingStream();
// 2. 启用我们感兴趣的事件并配置设置
rs.enable("jdk.GCPhasePause").withPeriod(Duration.ofSeconds(1));
rs.enable("jdk.Deoptimization").withPeriod(Duration.ofSeconds(1));
// 3. 订阅特定事件并设置事件处理器（回调函数）
rs.onEvent("jdk.GCPhasePause", event -> {
// 从事件中读取字段
Duration duration = event.getDuration("duration");
String name = event.getString("name"); // 例如 "GC Pause"
**********
shell:  jcmd JFR.start delay=-1 filename=xxx.jfr
**********
});
// 4. 启动流（这是一个非阻塞调用）
rs.startAsync();
```

## 应用场景

在线监控和事后分析配合的模式，在事后分析的 jfr 文件中，能够包含当前时间之前的一段时间内的所有 jfr 事件。

# GCC for openEuler CFGO 反馈优化特性增强

日益膨胀的代码体积导致当前处理器前端瓶颈成为普遍问题，影响程序运行性能。编译器反馈优化技术可以有效解决此类问题。

CFGO（Continuous Feature Guided Optimization）是 GCC for openEuler 的反馈优化技术名，指多模态（源代码、二进制）、全生命周期（编译、链接、链接后、运行时、OS、库）的持续反馈优化，主要包括以下两类优化技术：

- 代码布局优化：通过基本块重排、函数重排、冷热分区等技术，优化目标程序的二进制布局，提升 i-cache 和 i-TLB 命中率。
- 高级编译器优化：内联、循环展开、向量化、间接调用等提升编译优化技术受益于反馈信息，能够使编译器执行更精确的优化决策。

## 功能描述

- GCC CFGO 反馈优化共包含三个子特性：CFGO-PGO、CFGO-CSPGO、CFGO-BOLT，通过依次使能这些特性可以缓解处理前端瓶颈，提升程序运行时性能。为了进一步提升优化效果，建议 CFGO 系列优化与链接时优化搭配使用，即在 CFGO-PGO、CFGO-CSPGO 优化过程中增加 -flto=auto 编译选项。
- CFGO-PGO
- CFGO-PGO 在传统 PGO 优化的基础上，利用 AI4C 对部分优化遍进行增强，主要包括 inline、常量传播、去虚化等优化，从而进一步提升性能。
- CFGO-CSPGO
- PGO 的 profile 对上下文不敏感，可能导致次优的优化效果。通过在 PGO 后增加一次 CFGO-CSPGO 插桩优化流程，收集 inline 后的程序运行信息，从而为代码布局和寄存器优化等编译器优化遍提供更准确的执行信息，实现性能进一步提升。
- CFGO-BOLT
- CFGO-BOLT 在基线版本的基础上，新增 aarch64 架构软件插桩、inline 优化支持等优化，进一步提升性能。

## 应用场景

CFGO 通用性较好，适用于整机性能瓶颈在 CPU 上，且 CPU 瓶颈在前端的 C/C++ 应用，如数据库、分布式存储等场景，一般可以取得 5~10% 性能提升。

# DevStation 特性增强

DevStation 是基于 openEuler 的智能开发者工作站，专为极客与创新者而生。旨在提供开箱即用、高效安全的开发环境，打通从部署、编码、编译、构建到发布的全流程。它融合了一键式运行环境与全栈开发工具链，支持从系统启动到代码落地的无缝衔接。无需复杂安装，即可体验开箱即用的开发环境，通过新增 MCP AI 智能引擎，快速完成社区工具链调用，实现从基础设施搭建到应用开发的效率飞跃。

## 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 Codium 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。

社区原生工具生态：新增 oeDeploy（一键式部署工具）、epkg（扩展软件包管理器）、devkit 和 openEuler Intelligence，实现从环境配置到代码落地的全链路支持。 oeDevPlugin 插件 +oeGitExt 命令行工具支持：专为 openEuler 社区开发者设计的 VSCodium 插件，提供 Issue/PR 可视化管理面板，支持快速拉取社区代码仓、提交 PR，并实时同步社区任务状态。 openEuler Intelligence 智能助手：支持自然语言生成代码片段、一键生成 API 文档及 Linux 命令解释。

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，预装 Thunderbird 等办公效率工具。

MCP 智能应用生态构建：DevStation 深度集成 Model Context Protocol (MCP) 框架，构建完整的智能工具链生态，预装 MCP 智能工具链，支持 oeGitExt、rpm-builder 等核心 MCP Server，提供社区事务管理、RPM 打包等能力，将传统开发工具（如 Git、RPM 构建器）通过 MCP 协议进行智能化封装，提供自然语言交互接口。

系统部署与兼容性增强：广泛的硬件支持，特别优化对主流笔记本 /PC 硬件的兼容性（触摸板、Wi-Fi 、蓝牙），重构内核构建脚本（kernel-extra-modules），确保裸机部署体验。灵活部署形态，支持 LiveCD（一键运行无需安装）、裸机安装、虚拟机部署。

全新安装工具 heolleo：heolleo 是一款专为 DevStation 设计的现代化客户端工具。其核心使命是简化 DevStation 的安装流程。采用模块化设计使其可以轻松扩展以支持不同的硬件架构（如 x86/ARM）、文件系统或引导加载器（GRUB 等）。支持从本地 ISO 镜像、网络地址（HTTP/FTP）等快速获取系统文件，提供灵活的安装方式：

1. 本地 ISO 安装：对于追求极致稳定、速度或需要在无网络、受限环境中部署系统的用户，heolleo 提供本地 ISO 安装模式。充分利用已有的系统镜像文件，提供一个高速、可靠且完全离线的安装体验，当前已实现自动化分区安装。

2. 网络安装：heolleo 的网络安装模式适应现代系统部署的趋势。可直接从互联网上的服务器获取最新系统文件，省去了手动下载镜像的步骤，能以最便捷的方式触及最新的 DevStation 版本。

PS：DevStation 930 版本默认不集成 heolleo，该版本仅在社区提供软件安装包，有需尝鲜用户可直接通过“dnf install heolleo-frontend”安装。

## 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。

快速安装部署平台：DevStation 集成了 oeDeploy，可以对 kubeflow、k8s 等分布式软件实现分钟级部署，大幅减少开发者部署时间。oeDeploy 同时提供了统一的插件框架与原子化的部署能力，开发者只需遵循简单的开发规范，就可以快速发布自定义的安装部署插件，帮助更多用户解决安装部署问题。

面向测试 / 南向兼容性开发人员，提供硬件兼容性保障（支持主流笔记本 / 服务器），支持裸机部署测试驱动兼容性。

提升开发者效率：MCP RPM-builder 工具链落地，提升 MCP 易用性，支持 mcp servers 自动化打包构建 rpm 包并发布到社区，确保 mcp 一键安装功能 100% 可用，构建完整的 MCP 智能应用生态 REPO，覆盖部署、测试、性能调优等应用场景，包括：查询分配的社区 issue，创建 PR 提交代码变更，通过 CI/CD 自动构建验证。

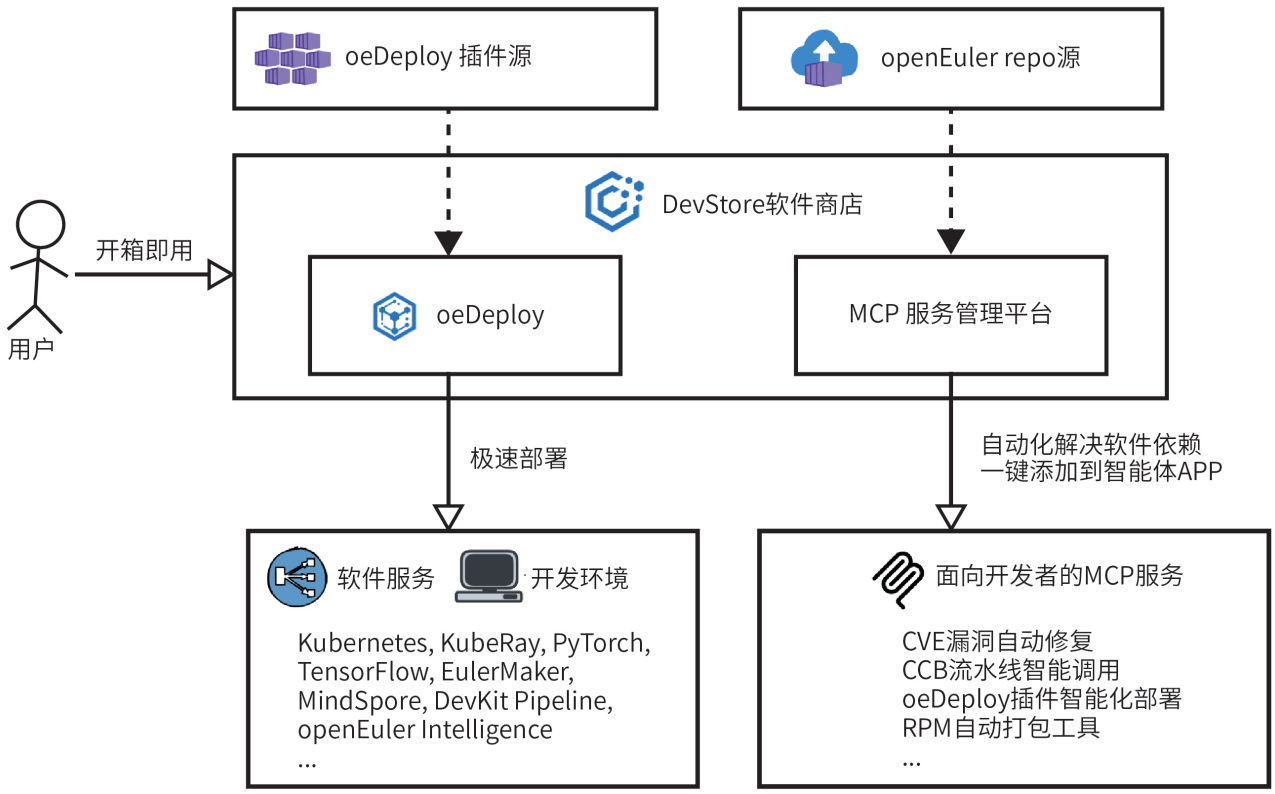
# DevStore 开发者软件商店

DevStore 是 openEuler 桌面版本的应用商店，是面向开发者的软件分发平台，支持 MCP 服务、oeDeploy 插件的检索与快捷部署功能。在 DevStation 平台上实现开箱即用。

## 功能描述

MCP 服务一键安装：DevStore 借助 openEuler 社区丰富的软件生态，以 rpm 软件包的形式处理 MCP 运行所需的软件依赖，并通过内置的服务管理工具，在智能体应用中快速部署 MCP 服务。自动帮助用户解决软件依赖与 MCP 配置问题，大幅提升用户体验。目前已支持 80+MCP 服务。

oeDeploy 插件快速部署：DevStore 借助 oeDeploy 工具实现主流软件的快速部署，大幅度降低开发者部署软件的时间成本。包括 Kubernetes、Kuberay、Pytorch、TensorFlow、DeepSeek 等 AI 软件，EulerMaker、openEuler Intelligence 等社区工具链，以及 RagFlow、Dify、AnythingLLM 等主流的 RAG 工具。




## 应用场景

DevStore 作为 openEuler 桌面端的软件商店，帮助开发者与初学者快速获取主流开发工具、AI 软件、MCP 服务等等，在详情页提供了丰富的用户文档与操作入口。所有复杂软件的部署操作都可以在一个操作界面完成，大幅降低学习成本、提升用户体验。

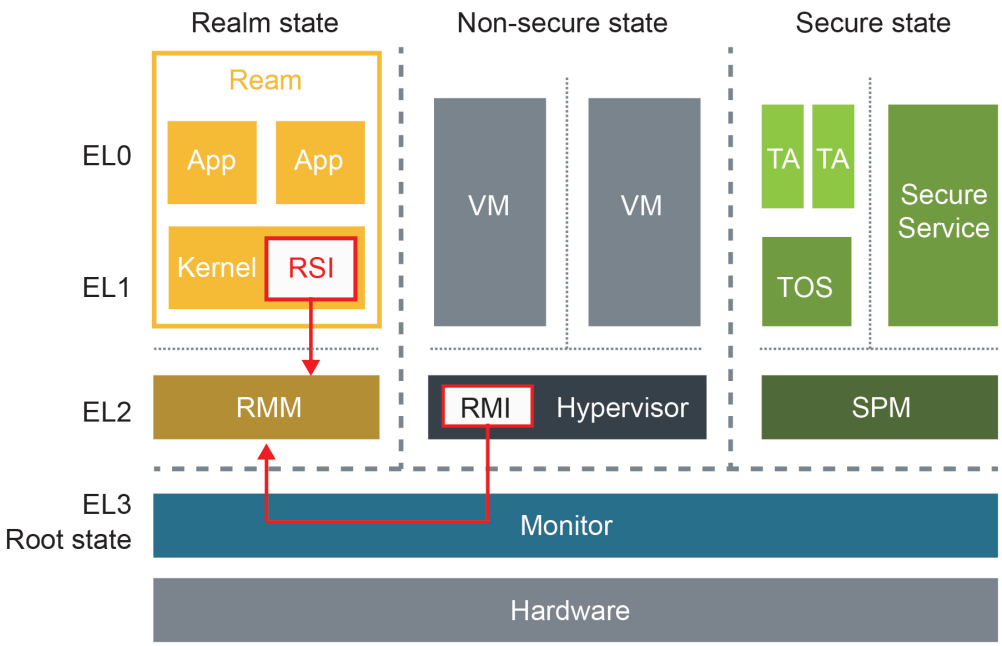
# CCA 机密计算

ARM CCA（Confidential Computing Architecture）是 ARM v9 新引入的机密计算架构规范，旨在为下一代计算设备定义标准化的机密计算解决方案。CCA 引入了全新的 Realm 域作为可信执行环境，来保护正在使用中的数据和代码的机密性和完整性，即使面对拥有特权的基础设施软件或云服务提供商，也能得到有效保护。

openEuler 基于 ARM CCA 机密计算架构规范，实现了 OS 相关组件（KVM、QEMU、libvirt、Guest kernel）对 CCA 的支持，提供了原生支持 Realm 机密虚机的社区版本，满足基于 Realm 机密虚机保护使用中数据的安全诉求，同时提供了兼容传统应用生态及虚机管理软件的易用性。

## 功能描述

ARM CCA 通过以下核心组件协同工作，构建一种隔离的、受保护的执行空间，在代码执行和数据访问方面与正常世界完全隔离，成为 Realm 机密域。




- **Realm 机密域：**Realm 是 CCA 的核心抽象，它是一种与正常世界（Non-secure）和安全世界（Secure，原来的 Trustzone）并行的新类型执行环境。Realm 是硬件隔离的，专为托管敏感代码和数据而设计。它独立于主机操作系统和 Hypervisor，它们可以管理 Realm 但无法访问其内部内容。
- **动态管理：**Hypervisor 可以应客户要求动态创建 Realm，并为其分配内存和 CPU 资源。但在 Realm 初始化后，Hypervisor 会将其控制权移交给一个受保护的安全虚拟化模块 RMM（Realm Management Monitor），此后 Hypervisor 便无法访问 Realm 内的秘密。
- **内存管理：**CCA 扩展了系统内存管理单元（MMU），使其能够识别和隔离 Realm 内存。任何从 Realm 外部（包括 Hypervisor）发起的访问尝试都会被硬件阻断，从而确保数据的机密性。
- **远程证明：**每个支持 CCA 的处理器都有一个基于硬件的唯一身份标识。当 Realm 启动时，它可以生成一份由硬件密码学签名的证明报告（Attestation Token）。用户可以获得这份报告，并验证其签名及组件度量值，从而确信他们的工作负载正在一个真实的、未被篡改的 ARM CCA 环境中运行。

## 应用场景

机密虚机主要应用于云计算环境，为核心工作负载提供高级别安全隔离。它使得用户能在不受信的公有云上处理敏感数据（如金融记录、医疗健康信息或知识产权），并确保其机密性和完整性连云服务商也无法窥探。这有效满足了数据主权、监管合规和跨机构隐私数据协作（如联合建模与分析）的关键需求。

# secGear 特性增强

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。

当前版本新增支持新版 virtCCA 机密虚机的 Platform token 报告的远程证明，补齐 virtCCA 的信任链及远程证明服务配套。完善远程证明流程的同时，依然保持兼容旧版 virtCCA 机密虚机的不含 Platform token 报告的远程证明。

## 功能描述

新版 virtCCA 机密虚机提供 Platform Token，Attestation Agent 在向虚机请求数据时，获得的 virtCCA Token 较旧版新增了 Platform Token 的部分。Attestation Service 在远程证明部分也有所变化：

1. 根证书、二级证书的证书算法由 RSA 算法更换为 ECCP521。
2. 远程证明流程新增 Platform Token 的验签与 Cvm Token 的公钥验证，完善证明链。
3. 支持使用策略对 virtCCA 的 Platform 的 Software 组件版本、HASH 进行校验。
4. 证明报告（根据策略）输出 Platform Software 组件校验结果。
5. 证明报告输出当前 virtCCA 是否支持 Platform Token。

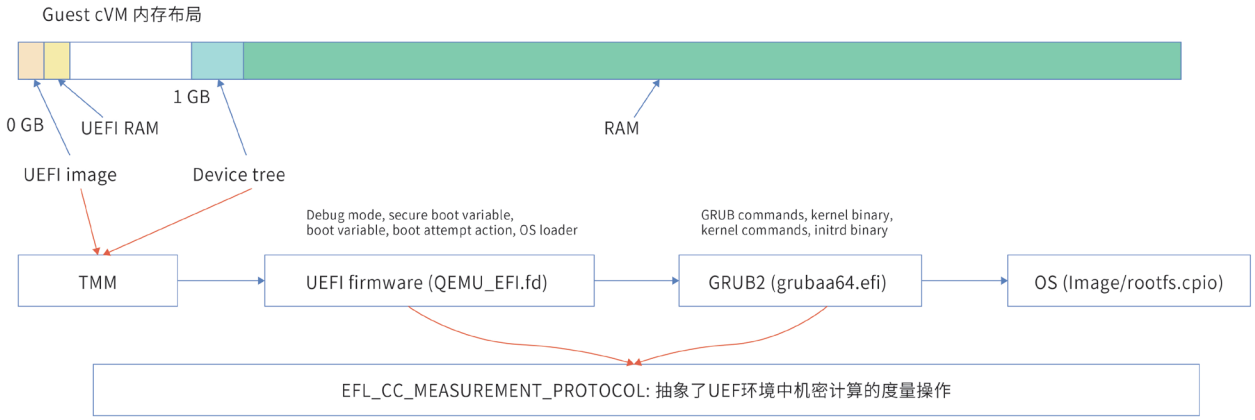
当前 secGear 远程证明统一框架仍保留了对旧版 virtCCA 远程证明的支持。主要根据 Attestation Agent 向 Attestation Service 发送的数据是否包含 Platform Token。如果不包含，则按照旧版 virtCCA 远程证明（证书）流程进行，对应证明策略无法校验相关组件，报告输出 vcca.is_platform 为 False，表示当前平台不支持 Platform Token。

## 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。

# virtCCA 机密计算特性增强

## 功能描述




当前 virtCCA 架构在启动方式上存在特定约束：其仅支持 **kernel 与 rootfs 分离的启动模式**（即内核镜像与根文件系统分别挂载）。然而，在主流云平台环境中，虚拟机的启动流程普遍依赖 **GRUB 引导机制**，这要求将 UEFI 固件（如 EDK2）、内核（Kernel）及初始内存文件系统（initramfs）整合至单一磁盘镜像（如 QCOW2 格式）中。功能要点包括：

1. 单镜像封装

(1) 将 EDK2 固件、GRUB 引导程序、内核（Kernel） 及 initramfs 整合至单一 QCOW2 磁盘镜像，形成完整启动栈。

(2) GRUB 通过配置文件（grub.cfg）定位内核路径，要求内核与 initramfs 必须位于同一文件系统（如 EXT4/XFS）。

2. 安全信任链传递

(1) Secure Boot 机制：EDK2 验证 GRUB 及内核的数字签名，确保启动组件未被篡改。

(2) 硬件资源协同：依赖 UEFI 运行时服务枚举硬件设备，为虚拟机管理程序（如 KVM）提供虚拟化资源池。

3. 云原生优化

(1) 支持快照克隆、根文件系统动态扩容（依赖 initramfs 中的 cloud-init 工具）特性。

## 应用场景

云环境中采用 UEFI（Unified Extensible Firmware Interface）启动模式，已成为现代虚拟化架构的核心技术，virtCCA 架构支持 UEFI 启动，扩展了机密虚机的应用场景：

1. 快速实例启动与弹性伸缩

(1) UEFI 通过并行硬件初始化（如 CPU、内存、存储设备同步检测）显著缩短启动时间。

2. 大容量磁盘支持

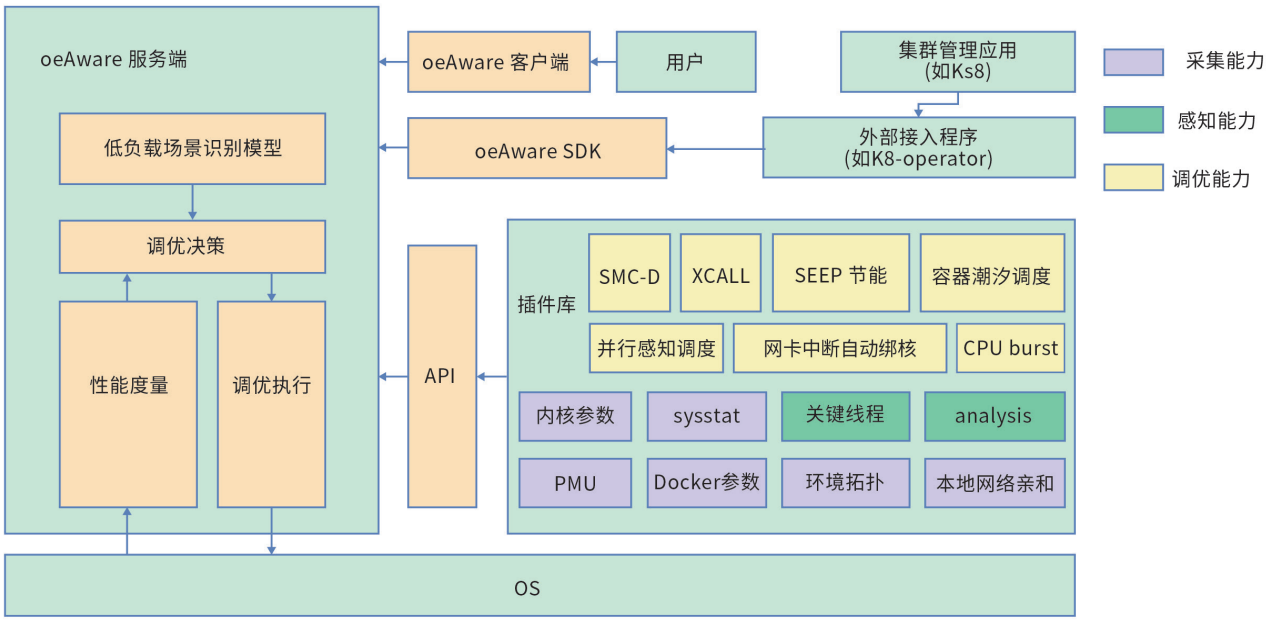
(1) UEFI 依赖 **GPT 分区表**，突破传统 MBR 的 2TB 磁盘限制，支持**百 TB 级云盘**（如阿里云 ESSD 云盘），满足大数据存储（如 HDFS）、AI 训练等场景需求。

3. 自动化运维与批量部署

(1) **镜像标准化：**云服务商提供的 UEFI 兼容镜像默认启用 GPT 分区，简化用户部署流程。

# oeAware 采集、调优插件等功能增强

oeAware 是在 openEuler 上实现低负载采集感知调优的框架，目标是动态感知系统行为后智能使能系统的调优特性。传统调优特性都以独立运行且静态打开关闭为主，oeAware 将调优拆分为采集、感知和调优三层，每层通过订阅方式关联，各层采用插件式开发尽可能复用。




## 功能描述

oeAware 的每个插件都是按 oeAware 标准接口开发的动态库，包含若干个实例，每个实例可以是一个独立的采集、感知或调优功能集，每个实例包含若干个 topic，其中 topic 主要用于提供采集或者感知的数据结果，这些数据结果可供其他插件或者外部应用进行调优或分析。

- SDK 提供的接口可以实现订阅插件的 topic，回调函数接收 oeAware 的数据，外部应用可以通过 SDK 开发定制化功能，例如完成集群各节点信息采集，分析本节点业务特征。
- PMU 信息采集插件：采集系统 PMU 性能记录。
- Docker 信息采集插件：采集当前环境 Docker 的一些参数信息。
- 系统信息采集插件：采集当前环境的内核参数、线程信息和一些资源信息（CPU、内存、IO、网络）等。采集本地进程间 TCP 网络亲和关系。
- 线程感知插件：感知关键线程信息。
- 评估插件：分析业务运行时系统的 NUMA 和网络信息，给用户推荐使用的调优方式。
- 系统调优插件：（1）stealtask：优化 CPU 调优 （2）smc_tune（SMC-D）：基于内核共享内存通信特性，提高网络吞吐，降低时延 （3）xcall_tune：跳过非关键流程的代码路径，优化 SYSCALL 的处理底噪 （4）realtime_tune：提供深度隔离、自动完成实时性能优化配置。 （5）net_hard_irq_tune：动态修改网络中断亲和性，提高网络业务性能。 （6）并行感知调度 : 增强系统的 NUMA 调度性能。
- Docker 调优插件：(1) cpu_burst：利用 cpuburst 特性在突发负载下环境 CPU 性能瓶颈。 （2）容器潮汐调度：根据业务负载特征调整容器业务的 CPU 亲和，提高业务的 QoS。

### 约束限制

- SMC-D：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。
- Docker 调优：暂不适用于 K8s 容器场景。
- xcall_tune : 内核配置选项 FAST_SYSCALL 打开。
- realtime_tune ： 需要配合 Preempt-RT 内核使用。

## 应用场景

stealtask 适用于希望提高 CPU 利用率的场景，例如 Doris，该调优实例可以有效提高 CPU 利用，避免 CPU 空转。

XCALL 适用于应用程序的 SYSCALL 开销较大的场景，XCALL 提供一套跳过非关键流程的代码路径，来优化 SYSCALL 的处理底噪，这些被跳过的非关键流程会牺牲部分维测和安全功能。

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。

cpuburst 适用于高负载容器场景，例如 Doris，能够缓解 CPU 限制带来的性能瓶颈。

realtime 适用于需要低时延、低抖动、有严格时延限制的场景，如工业制造行业。

net_hard_irq_tune 适用于业务性能受 TCP 网络影响较大的业务，例如 redis、nginx 等。

并行感知调度适用于跨 NUMA 访存对业务性能比较大且不好固定绑核的场景，例如 Spark。

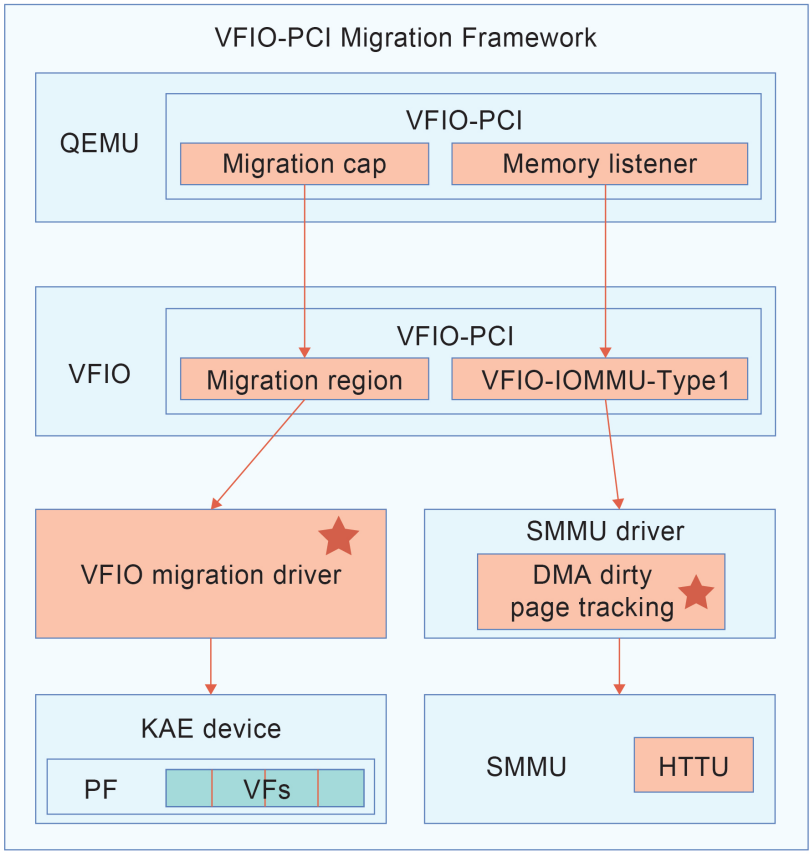
容器潮汐调度适用于容器化部署的业务，例如容器化 mysql，当业务负载低时，在满足 QoS 的基础上，使用最少的 CPU 资源，减少 CPU 迁移、idle 切换、cache miss 和跨 NUMA 访问，增强资源的局部性。当业务负载高时，又可以突破绑核约束，使用更多 CPU 资源来提升 QoS 质量，提高 CPU 资源利用率。

# 虚拟化支持 vKAE 直通设备热迁移

## 功能描述

KAE 是基于鲲鹏 920 新型号处理器提供的硬件加速解决方案，包括 HPRE、SEC、ZIP 设备，可用于加解密和压缩解压缩，能够显著降低处理器消耗，提高处理器效率。KAE 直通热迁移是指虚拟机在配置 KAE 直通设备时，进行热迁移的能力，可以为 KAE 设备的使用提供更强的灵活性和业务不中断的保障。

smmu 脏页跟踪是实现直通设备高效、可靠的热迁移的关键技术。在 ARM 架构中，通过纯软件方式进行脏页跟踪，会带来了较大的性能损耗。HTTU(Hardware Translation Table Udate) 允许硬件自动更新 smmu 页表状态，在进行写操作时会自动置对应页表项的写权限位，热迁移时扫描页表的写权限位进行脏页统计。




## 应用场景

为虚拟机中使用 KAE 直通设备的场景提供热迁移支持，适用于对数据安全性和处理性能有较高要求的领域，如金融、云计算和大数据处理等，有效增强业务连续性与运行稳定性。

# Global Trust Authority 远程证明增强

## 功能描述

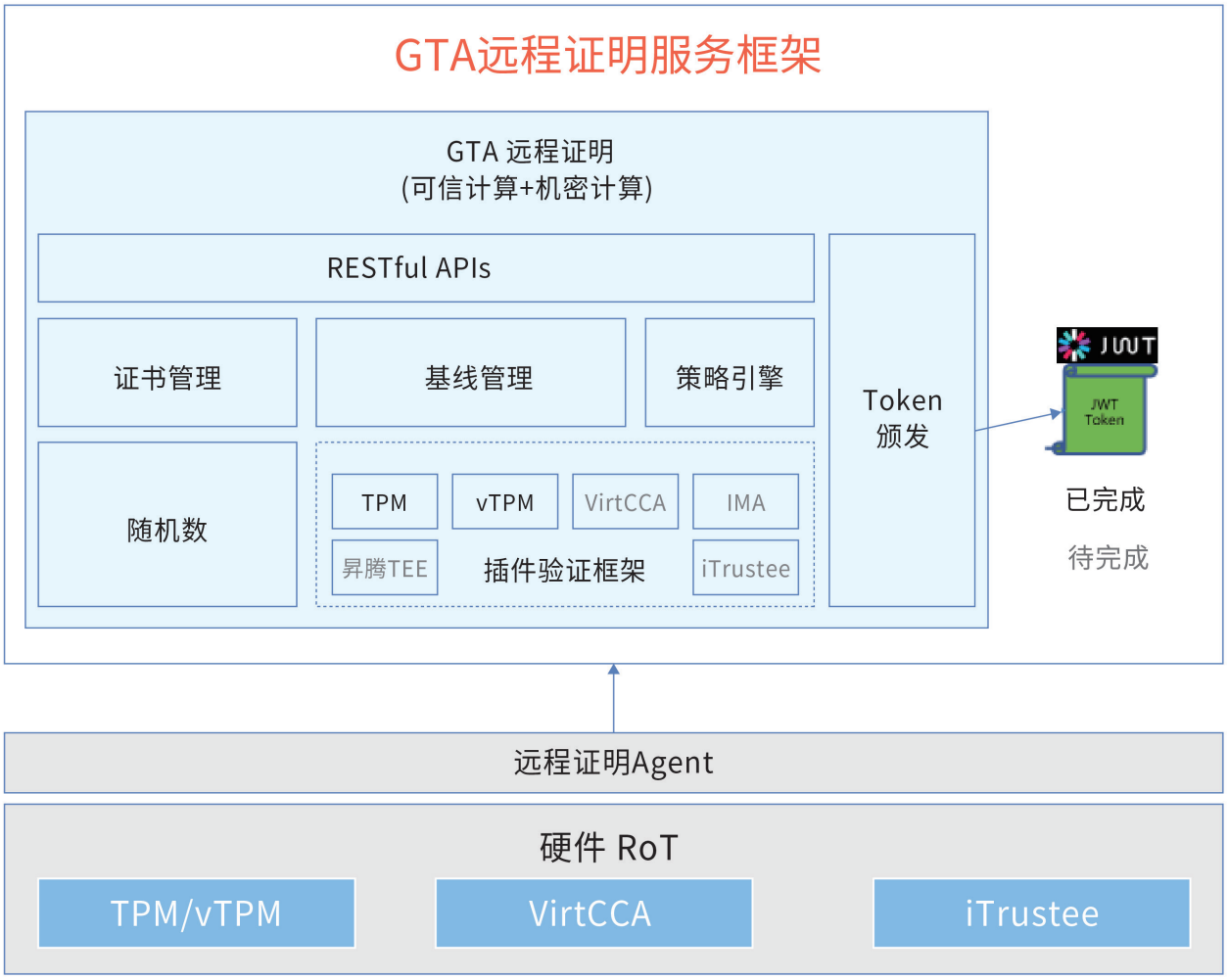
GTA 远程证明服务组件支持 TPM/vTPM、VirtCCA 及其 IMA 的远程证明，分为客户端和服务端。

- 服务端提供了远程证明服务框架兼容可信计算及机密计算，支持证书、策略等的增删改查，Quote 验证，随机数，JWT Token 生成等能力。
- 客户端支持采集本地 TPM 证据，并可与服务端交互，验证 Quote。

本组件在安全性及易用性上也提供了多种能力。

安全性上支持数据库完整性保护、数据链路加密，验证防重放，SQL 防注入，用户隔离，密钥轮换机制等一系列差异化安全竞争力。

易用性上支持护照模式及背调模式。 客户端支持定时上报，响应挑战等多种验证模式。客户端及服务端支持 rpm 包及 docker 安装部署。



## 应用场景

远程证明是开启机密计算及可信计算的前置条件。只有当运行环境在密码学意义上被严格证明安全可信后方可进行后续运算。所有涉及机密计算的端到端解决方案都应将远程证明作为整体解决方案中的核心一环，一旦运行环境被证明不可信，则应立即中止后续任务。在 AI 模型保护，用户隐私保护，密钥管理等多场景均有广泛应用。

# Kuasar 机密容器

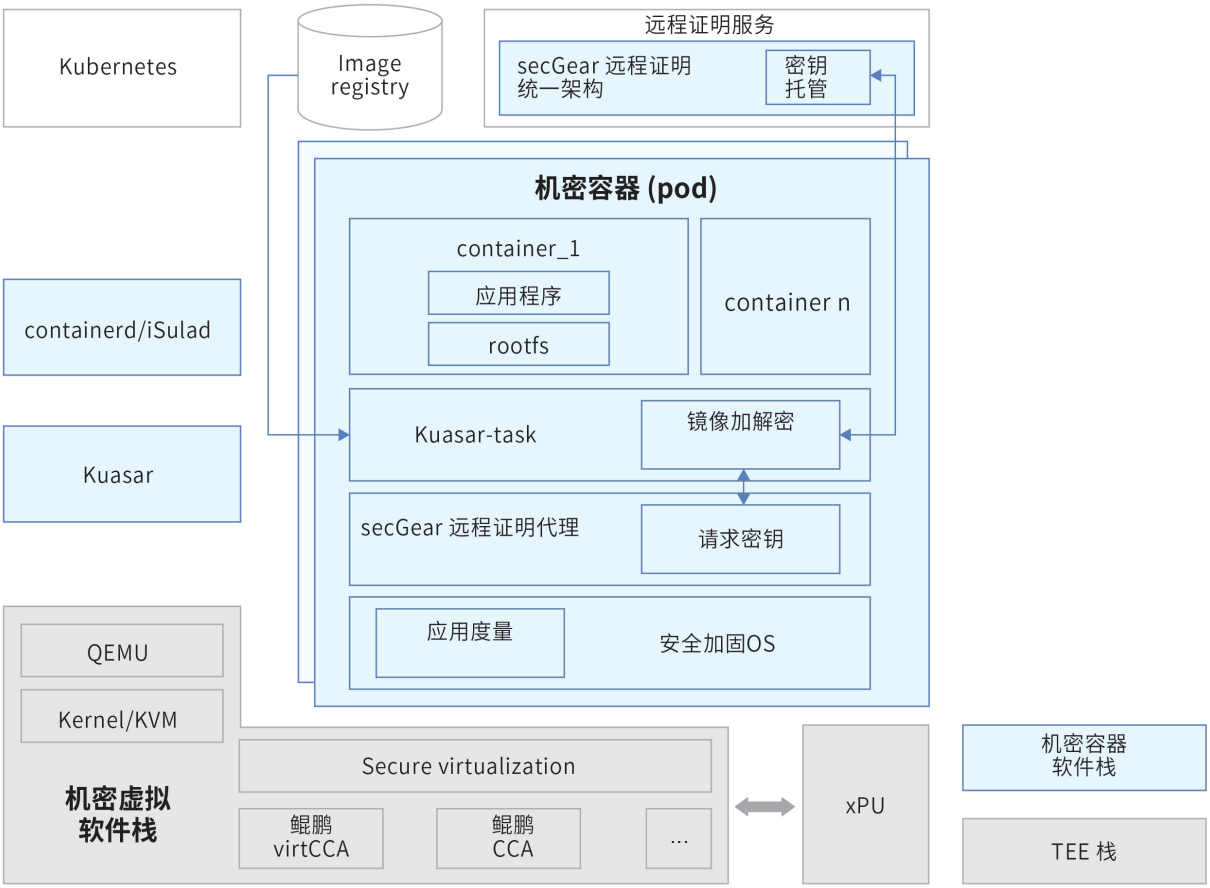
## 功能描述

Kuasar 统一容器运行时在支持安全容器的基础上添加了对机密容器的支持。用户可以通过配置 iSulad 的运行时参数，完成对 Kuasar 机密容器的纳管。

当前 Kuasar 机密容器使用 iSulad+Kuasar 方案，提高了启动速度，极大降低了内存底噪。一方面 Sandbox API 的实现，使得创建容器不再单独创建 pause 容器，节省了准备 pause 容器镜像快照的时间；另一方面得益于 1:N 的管理模型，Sandboxer 进程常驻，从而节省了冷启动 Shim 进程的时间，这使得容器的启动速度大大提升，带来与 Pod 数成正比的内存收益。最后，Kuasar 使用 rust 实现，相比 golang，内存更安全，语言本身也带来了一些内存收益。

**支持功能特性：**

- 支持 iSulad 容器引擎对接 Kuasar 机密容器运行时，兼容 Kubernetes 云原生生态。
- 支持基于 virtCCA 的机密硬件，允许用户在鲲鹏 virtCCA 可信执行环境中部署机密容器。
- 支持 secGear 远程证明统一框架，遵循 RFC9334 RATS 标准架构，允许在机密计算环境中运行的容器向外部的受信任服务证明其可信性。
- 支持在机密容器内部拉取并解密容器镜像，保护容器镜像的机密性和完整性。



## 应用场景

Kuasar 机密容器在满足客户数据安全的诉求下，兼容云原生生态，确保用户的机密应用具备高可用性、弹性伸缩、快速交付等云原生优势，在 AI 保护、数据可信流通、隐私保护等机密计算的业务中有广泛的应用场景。

# NET Framework 应用原生开发能力

Mono 是一套跨平台的兼容于微软 .NET Framework 的完整开发工具和运行时环境，让开发者能够使用 C# 语言和 .NET Framework 框架的类库；.NET 是一套跨平台开源开发人员平台，用于构建多种应用程序 ，它提供了一个统一的生态系统，包含了编程语言、运行时环境、庞大的代码库和丰富的开发工具。openEuler 提供了 .NET Framework、Mono 以及 .NET 应用原生开发的能力。

## 功能描述

- openEuler 支持 monoDevelop 及依赖组件：monoDevelop 是一款面向 .NET 开发者的强大的、开源的集成开发环境，通过配置 Mono 运行时，支持 .NET Framework 应用在 openEuler 上开发调试、代码管理、编译构建、集成测试等
- monoDevelop 支持一键快速部署：借助 oeDeploy 快速部署平台，开发者可在该平台上一键部署和卸载 monoDevelop。
- openEuler 支持 .NET SDK 及其依赖组件：适配并引入 .NET SDK 及其依赖组件到 openEuler，目前最高支持到 .NET 9，可在 openEuler 上基于 .NET SDK 开发 .NET 应用
- .NET SDK 支持一键快速部署：借助 oeDeploy 快速部署平台，开发者可在该平台上一键部署和卸载 .NET SDK。

## 应用场景

openEuler .NET 原生开发组件套用于 .NET 应用开发场景，使得原本强依赖 windows 开发环境的开发者可以在 openEuler 上新开发或者迭代开发新的 .NET 应用特性 ， 保留熟悉的 .NET 开发范式，同时获得 Linux 生态的灵活性与强大工具链，降低对单一平台的依赖 。

# 支持树莓派

Raspberry Pi（树莓派）是由 Raspberry Pi 基金会与 Broadcom 公司合作开发的一系列小型单板计算机。凭借其价格低、体积小、能耗低、高可编程性以及丰富的生态系统等特点，树莓派在工业自动化、机器人技术、物联网、教育以及业余爱好者项目等领域得到了广泛应用。树莓派 4B 和树莓派 5 作为树莓派产品线的经典代表，采用 ARM 架构的处理器。其中树莓派 4B 是极具性价比的普及型单板计算机，树莓派 5 凭借其显著的性能突破和扩展能力成为一款在高性能边缘计算领域颇具竞争力的创新产品。

## 功能描述

作为开源硬件领域的一个较为高阶的硬件产品，树莓派 4B 和树莓派 5 支持 Raspberry Pi OS、Ubuntu、openEuler 等多种 Linux 发行版，外设丰富，具有较强的视频编解码能力，以及板载网络等功能，完全可以作为独立计算机系统使用。

## 应用场景

树莓派 4B 和树莓派 5 凭借其强大的性能和丰富的扩展能力，广泛应用于多个领域：

1. 教育与学习：学习 Python 等编程语言、借助外设接口进行电子实验；
2. 多媒体与娱乐：作为媒体中心或游戏机；
3. 物联网和智能家居：作为传感器节点或智能家居中枢，用于环境监测、家庭自动化控制和边缘计算；
4. 服务器与网络应用：用于家庭服务器、轻量级 Web 服务及容器化应用；
5. 创客与 DIY 项目：用于机器人控制、3D 打印管理和无人机飞控；
6. 科研与开发：用于 AI 实验、嵌入式开发原型验证；
7. 工业与自动化：实现设备监控、人机界面和机器视觉。

# 08 著作权说明

# 09 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。

# 附录 10

## 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/en/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-en.md |
| 安全披露信息 | https://gitee.com/openeuler/security-committee/blob/master/security-disclosure-en.md |
| 安全保障策略总纲 | https://gitee.com/openeuler/security-committee/blob/master/security-strategy-overview.md |