OpenEuler

操作系统技术白皮书

创新项目总览

（2023 年 6 月）

OpenAtom openEuler 社区

目 录

目 录

01

社区简介

openEuler 社区，全称为 OpenAtom openEuler 社区，是一个面向数字基础设施操作系统的开源社区，简称 openEuler 或者 openEuler 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。

openEuler 发展历程大事件

2023.06
openEuler 项目群与四个项目签署捐赠意向，项目名称是：Vtopia agent、secScanner、KsPack、secShield

2023.04
openEuler 项目群接收由社区成员贡献的 5 个项目：GoStone、Ctinspector、ZVM、Rust-Shyper、QuickPool

2022.12
成立 openEuler 项目群，首届 openEuler 委员会成立，openEuler 社区可单独接收项目捐赠

2022.11
荣获世界互联网大会 2022 世界互联网领先科技成果奖

2022.09
openEuler 首次亮相全球顶级开源峰会 Open Source Summit Europe 2022

2022.03
发布首个数字基础设施全场景版本 openEuler 22.03 LTS
9 大伙伴发布基于 openEuler 22.03 LTS 构建的商业发行版，累计装机 170+ 万套

2021.12
openEuler 技术路线操作系统超过 100 万套装机

2021.11
华为携手社区伙伴捐赠 openEuler
300+ 家企业加入社区

2021.09
升级为数字基础设施开源操作系统

2021.08
100+ 家企业加入社区

2020.12
社区理事会正式成立
3 万套商业装机

2020.03
首个社区 LTS 版本 20.03 发布
麒麟软件、普华基础软件、统信软件、中科院软研所发布商业版

2019.12
openEuler 开源代码正式上线
openEuler 社区成立

2019.09
华为宣布 openEuler 开源

02

技术生态

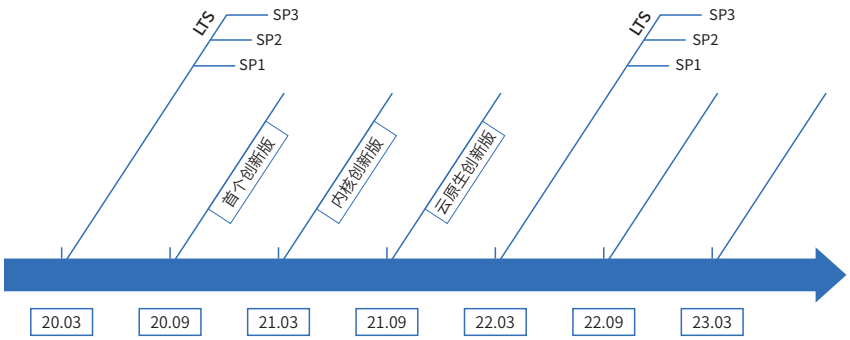
openEuler 作为一个操作系统发行版平台，每两年推出一个 LTS 版本。该版本为企业级用户提供一个安全稳定可靠的操作系统。

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 版本管理

长生命周期版本	创新版本
openEuler 20.03 LTS openEuler 22.03 LTS	openEuler 20.09 openEuler 21.03 openEuler 21.09 openEuler 22.09 openEuler 23.09



长生命周期版本：每两年发布一次，在创新版本基础上提供长生命周期管理，维护性能，可靠性和兼容性

创新版本：长生命周期版本之间每半年发布一次，集成 openEuler 以及其他社区最新版本技术进展。

openEuler 社区主线

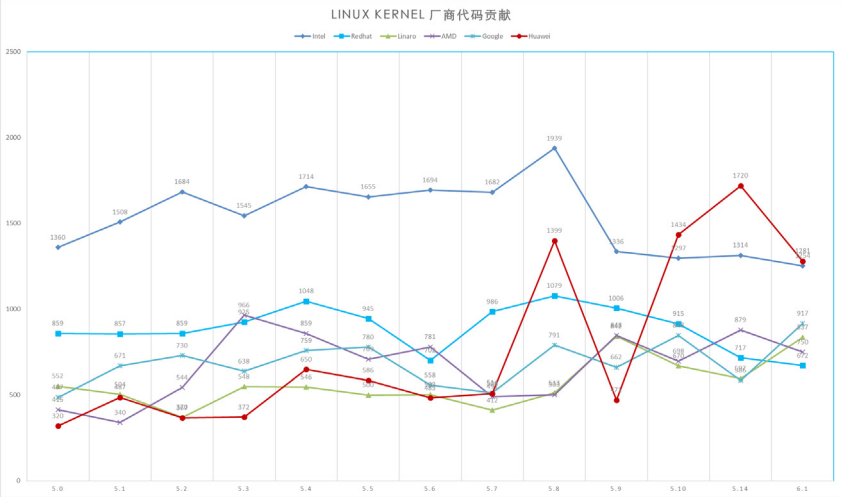
openEuler 覆盖全场景的创新平台

openEuler 是面向数字基础设施开源操作系统。通过一套操作系统架构，南向支持多样性设备，北向覆盖全场景应用，横向对接 OpenHarmony 等其他操作系统，通过能力共享实现生态互通。openEuler 突破性的实现了一套 OS 架构下，100% 支持主流计算架构，是最佳支持多样性算力的开源操作系统。openEuler 开创性的提出全场景操作系统理念，通过全栈原子化解耦和榫卯架构，实现版本灵活构建、服务自由组合。通过一套操作系统架构，实现了对服务器、云计算、边缘计算和嵌入式等场景的支持。

openEuler 对 Linux Kernel 的持续贡献

openEuler 社区成员（华为、龙芯中科、麒麟软件、统信软件、飞腾、成都菁蓉联创、中国电信、中国移动等）持续贡献 Linux Kernel 上游社区，回馈主要集中在：芯片架构、ACPI、内存管理、文件系统、Media、内核文档、针对整个内核质量加固的 bugfix 及代码重构等内容。

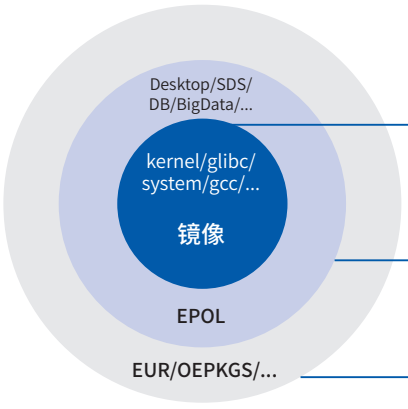
openEuler 社区拥有社区伙伴超过 905 家，贡献者 1.4 万，累计代码合入超 PR 10 万次，支持的软件包达到 3.4 万，全球下载量达 138 万。



华为在 Linux Kernel 5.10、5.14、6.1 贡献排名第一

openEuler 软件包仓库

openEuler 社区及其他第三方开发者共同提供了丰富易用的软件包，在最新的版本中，软件包总数已超过 3.4 万。openEuler 社区根据这些软件包的来源、质量属性、维护方式等不同维度划分为三类 openEuler 社区软件仓库，openEuler 社区版本使用者可以根据自己的需求配置不同的软件仓库；软件包可能会随其用量、稳定性、维护状态等在不同的软件仓库中依据社区规则重新分布。



核心包 / 基础包
openEuler LTS 和创新版正式发布的软件包集合，该仓库所有软件包根据 openEuler 社区软件质量属性规范，均完成了 openEuler 社区全流程质量保证。

扩展包
作为 openEuler LTS 和创新版本软件包仓库的补充，为 openEuler 社区提供尽可能丰富的软件包。该仓库软件包源码均需要来源于 openEuler 社区，同时因受社区软件包质量、技术成熟度、社区参与者投入等原因暂时无法完全满足 openEuler 社区软件包发布质量及维护支持要求。

第三方包：开源 / 闭源
openEuler 社区或第三方提供的软件仓库及构建系统，可以兼容 openEuler 各版本，并且不破坏 openEuler 各版本软件包原有兼容性和依赖关系，作为 openEuler 软件包仓库的第三方软件包补充，为 openEuler 社区使用者提供尽可能丰富的软件包选择。

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、运行依赖关系和上游社区，三者之前形成闭环且完整透明的软件供应链管理。

通过社区认证的 openEuler 发行版

（按版本认证时间排序）

更多信息请访问：https://www.openeuler.org/zh/approve/

伙伴名称	系统名称
超聚变数字技术有限公司	超聚变服务器操作系统 FusionOS 22
统信软件技术有限公司	统信服务器操作系统 V20（1050e）
湖南麒麟信安科技股份有限公司	麒麟信安操作系统 V3（openEuler 版）
SUSE	SUSE Euler Linux 2.0
新华三技术有限公司	H3Linux 2.0.2
江苏润和软件股份有限公司	HopeEdge V1.0 HopeStage V1.0
南京烽火星空通信发展有限公司	fitstarryskyos 22.06
北京凝思软件股份有限公司	linxos 6.0.99
天翼云科技有限公司	CTyunOS
北京拓林思软件有限公司	TurboLinux Enterprise Server 16
麒麟软件有限公司	银河麒麟高级服务器操作系统 V10
深圳华锐分布式技术股份有限公司	ArchforceEuler 22.09
软通动力信息技术（集团）股份有限公司	ISSEL 22 LTS
南方电网数字电网集团	pegaspegasus server v1.0
杭州安恒信息技术股份有限公司	dasos e2.1.0
中移（苏州）软件技术有限公司	BC-Linux for Euler 21.10
普华基础软件股份有限公司	普华服务器操作系统 V5.1
恒安嘉新（北京）科技股份公司	EversecOS 20.03
宝德计算机系统股份有限公司	RedderStar V1.0
联通数字科技有限公司	CULinux
广东中兴新支点技术有限公司	NewStartOS Server V6.02

openEuler 开源操作系统架构图

03

场景化创新

DPUDirect 直连聚合

服务器 云计算

DPU SIG

直连聚合特性（DPUDirect）旨在为业务提供协同运行环境，允许业务在 HOST 和 DPU 之间灵活卸载及迁移。当前已实现进程级别无感卸载功能，提供跨 HOST 和 DPU 的协同框架，支持管理面进程无需改造进行拆分，并近无感知卸载到 DPU 上运行，卸载后进程同时保持对 HOST 侧业务进程的管理能力。直连聚合特性能够极大降低业务在 DPU 场景下的卸载成本，简化运维，大大降低后期维护成本。

技术挑战

随着智能网卡演进到 DPU/IPU 的形态，这一新兴计算单元正逐渐成为云与数据中心基础设施的重要组成部分。DPU/IPU 除了承担 IO 数据面加速的需求外，也在逐渐增加对管理面及控制面组件卸载的支持。数据中心基础设施相关管控组件全卸载到 DPU 能够带来更优的架构和更灵活的部署方式。当前主流卸载方案大都通过组件拆分完成卸载，拆分方案存在以下问题：

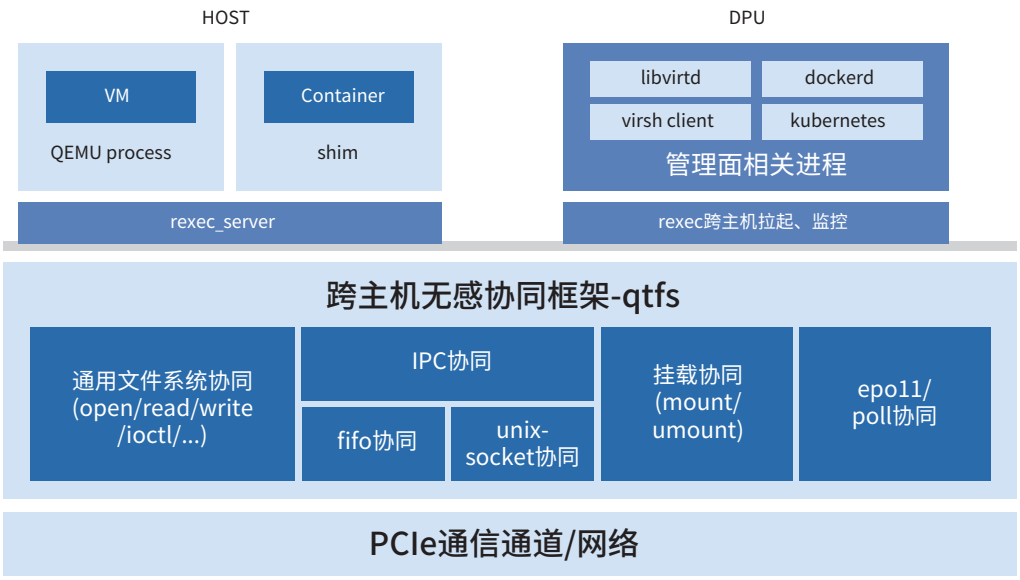
- 组件拆分要求开发者对卸载组件具备代码层级的了解。云厂商维护组件较多，相关组件卸载至 DPU 带来的拆分工作量巨大。
- 拆分工作在组件升级时很难继承，组件升级维护的成本较高，需要移植适配相关拆分代码。

项目介绍

由如下架构图所示， 直连聚合特性在 HOST 和 DPU 的操作系统层面构建了一个跨主机无感协同框架，该框架为卸载到 DPU 侧的管理面进程和 HOST 侧的业务进程提供一致的运行时视图，达到应用对卸载低感知或零感知的效果。用户只需要少量适配管理面业务代码，保证业务的软件兼容性和演进性，降低组件维护成本。

功能描述

DPU 管理面无感卸载框架架构图如下图所示，通过在 HOST 及 DPU 操作系统层面构建一个跨主机无感协同框架，为卸载到 DPU 侧的管理面进程和 HOST 侧的业务进程提供一致的运行时视图，达到应用对卸载低感知或零感知的效果。



从实现层面，本方案提供的机制可以结合定制策略实现不同场景下的进程无感卸载目标，方案包含的协同机制介绍如下：

- 文件系统协同：支持跨主机文件系统的访问，为 HOST 和 DPU 进程提供一致的文件系统视图；除通用文件系统外还包括对 proc、sys、dev 等特殊文件系统的支持。
- IPC 协同：实现跨 HOST 和 DPU 进程间的无感通信，当前支持进程无感使用 fifo 及 unix domain socket 进行跨主机通信。
- 挂载协同：对特定目录下的挂载操作拉远至 HOST 端，可用于适配容器 overlay 镜像构造场景；支持卸载后的管理面进程为 HOST 侧业务进程构造工作目录，提供跨节点的统一文件系统视图。
- epoll 协同：支持远程普通文件及 fifo 类文件跨主机访问的 epoll 操作，支持阻塞读写操作。
- 进程协同：通过 rexec 工具进行远程可执行文件拉起，能够接管远端拉起进程的输入输出流并进行状态监控，保证两端进程生命周期一致性。

通过以上机制的结合，在不同场景下使用定制策略，可满足管理面进程接近无感、无需过多业务拆分改造的业务要求，并可达成卸载到 DPU 的业务目标。

应用场景

DPU 管理面无感卸载方案可应用于云计算中全卸载场景的容器或虚拟化管理面进程的 DPU 卸载；使用无感卸载方案，云上容器管理面进程（kubelet、dockerd）及虚拟化管理面进程（libvirtd）的卸载分别能够减少 10K+ 代码拆分工作量，业务卸载适配和维护的工作量降低近 20 倍，并且无需修改管理面业务逻辑，从而保证业务的软件兼容性和演进性。

仓库地址

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

eNFS 多路径

服务器 云计算

Kernel SIG

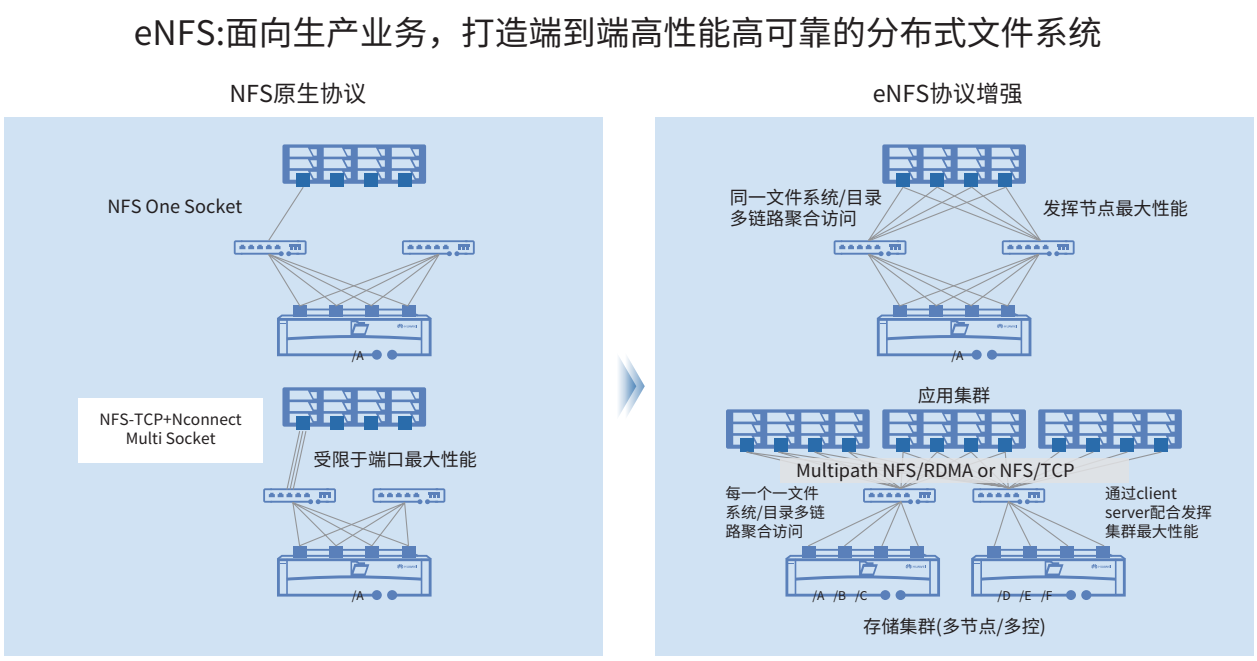
非结构化数据高速增长，NAS 成为海量非结构化生产业务的最佳选择。eNFS 通过建立多个链路进行 IO 负载均衡，并且在链路发生故障时将 IO 切换到其他可用路径，提供高性能高可靠的服务。

技术挑战

随着应用场景扩张，数据重要性不断提高，各行业对 NAS 存储的可靠性和性能提出了更高的诉求。传统 NFS 单个挂载点仅指定一个服务端 IP 地址，在使用过程中面临以下挑战：

- 在网口故障或者链路故障场景下，挂载点无法访问，导致业务 IO 挂死，可靠性不足；
- 单个挂载点性能受限于单个物理链路性能，重要业务存在性能瓶颈。
- NAS 存储部署于公共区，主机访问需要跨三层组网，一端故障时 IP 无法感知，当前依靠应用层手动挂载文件系统，双活链路无法自动切换。

项目介绍

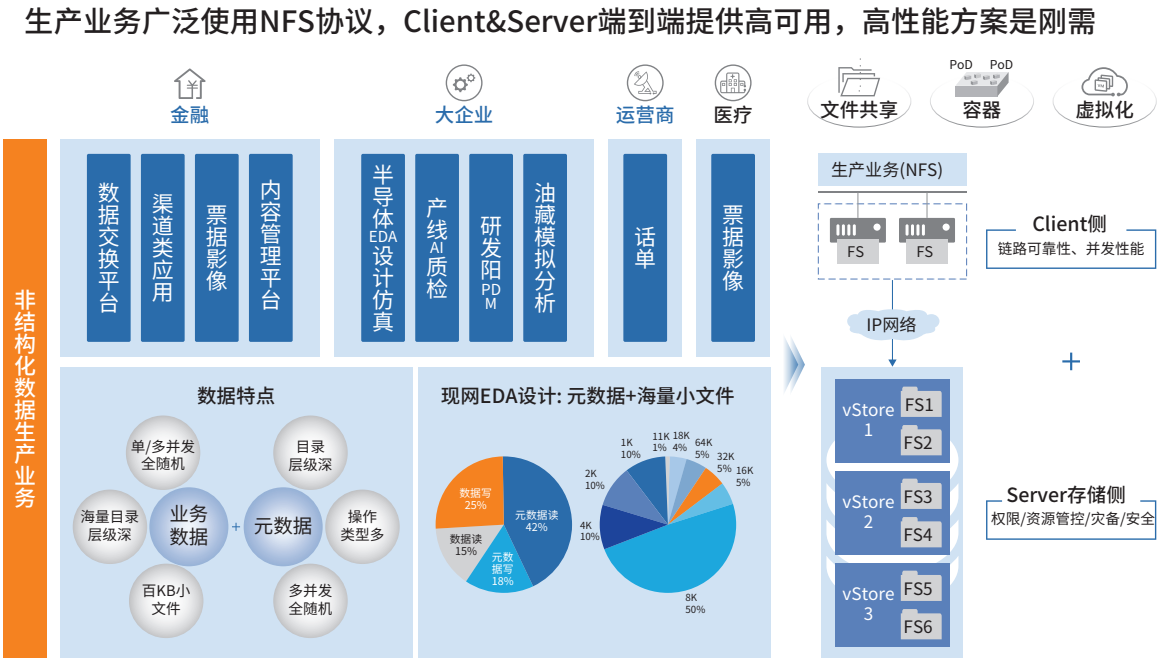


eNFS 协议是运行在 openEuler 操作系统内核中的驱动模块，包含 NFS 协议层的挂载参数管理模块和传输层多路径管理模块。eNFS 通过指定多个本地 IP 地址和多个服务端 IP 地址，实现不同 IP 地址建立多条 TCP/RDMA 链路，实现多路径建链、故障恢复和倒换、负载均衡等特性。

相对于原生 NFS，eNFS 的三大创新是：

- IO 路径软硬件故障，秒级切换。单个 NFS 挂载点使用多个 IP 进行访问，客户端和服务端之间建立多条链路，解决跨控制器及站点的可靠性问题。所有配置仅用一个文件记录，HPC 应用部署到不同的机器仅需修改配置文件。
- 多链路聚合，提升主机并发访问能力。网卡端口 / 多网卡 / 多节点聚合，大幅提升主机访问性能。
- 业界首创三层网络双活路径自动切换，下层存储故障或主机侧 IO 超时，跨站点 AA 双活主动切换，解决跨引擎失效、主机无感知问题。

应用场景



eNFS 提供超越本地文件系统的高性能数据共享能力，并且为 client 到 Server 之间的故障提供解决方案，确保业务不中断无感知，全面替代原生 NFS。

仓库地址

https://gitee.com/openeuler/kernel

hpcrunner 贾维斯智能助手

服务器

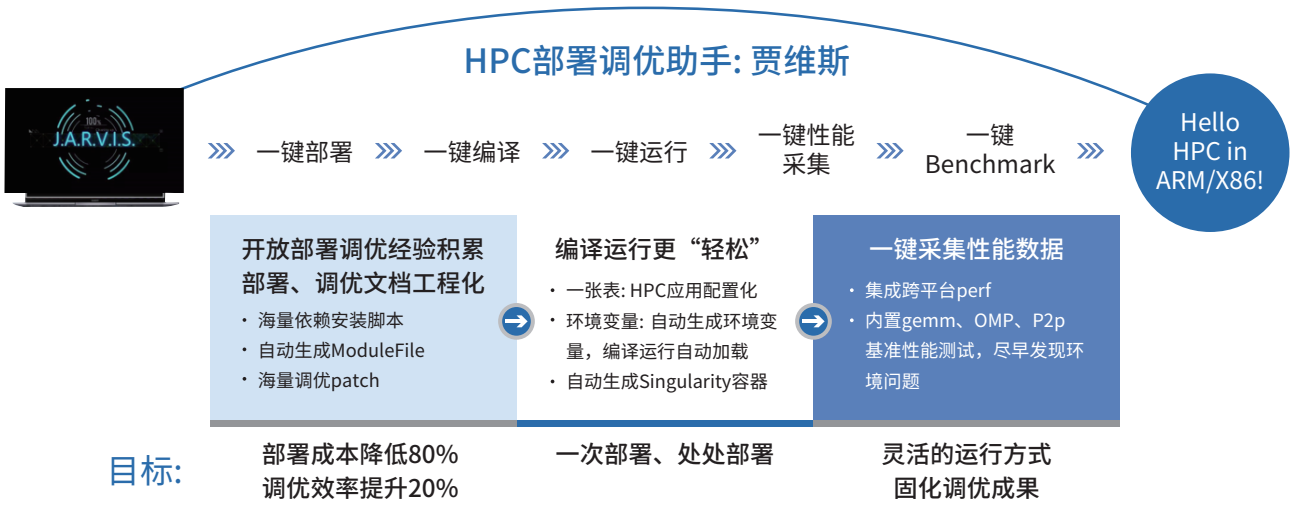
sig-HPC

hpcrunner 秉承着“让人人都懂 HPC”的使命，致力于通过“依赖管理 + 性能分析 + 应用编译 + 自动容器”的全闭环生态模式，为开发者、超算中心、科研机构等用户提供一站式应用部署调优解决方案。

技术挑战

HPC 应用的部署过程极为繁琐，动辄十几个依赖，每个依赖都要手工安装，且需要不同的编译器加上不同的并行通信库配合编译，而且普遍使用了大量硬件相关的并行技术和通信技术，导致不同架构之间迁移的复杂性陡增，部署和优化 HPC 应用会耗费大量精力，为应对这种挑战，设计并实现 HPC 部署调优助手，大幅降低部署成本，提高调优效率。

项目介绍



hpcrunner 的特性可以由两大部分组成：HPC 依赖管理和 HPC 应用管理。HPC 的所有三方依赖统一使用 module 管理，只要提供一个安装脚本，贾维斯会自动完成依赖的下载、解压、编译、安装和包配置生成等工作，以非常优雅的方式解决了 yum 无法实现的多版本并存问题，而且也支持闭源组件的一键部署；HPC 应用管理则是第二项强大功能，贾维斯将所有 HPC 应用抽象为一个单体配置文件，只需要完成配置文件中编译、环境、运行、批量运行的命令编写，就可以使用贾维斯驱动应用的自动编译和运行。具体来说贾维斯支持以下功能：

- 支持 Arm/x86 架构，已支持 100+ 依赖，60+ 应用一键部署安装，采用业界权威依赖目录结构管理海量依赖，自动生成 module file。
- 根据 HPC 配置实现一键编译运行、一键 CPU/GPU 性能采集、一键 Benchmark。
- 所有配置仅用一个文件记录，HPC 应用部署到不同的机器仅需修改配置文件。

- 日志管理系统自动记录 HPC 应用部署过程中的所有信息。
- 软件本身无需编译开箱即用，仅依赖 Python 环境。
- HPC 应用容器化 - 自动对接 Singularity 容器。

应用场景

应用场景 1：超算中心应用管理

贾维斯可以方便地管理不同软件包之间的依赖关系，并确保这些软件包与所需的库和工具链版本兼容，同时可以非常方便的实现编译器和并行库的升级。可以自动化和简化在多台计算机上部署相同版本的软件包，减少手动操作和避免出错。

应用场景 2：科学计算实验

贾维斯可以灵活地配置编译选项，以满足特定应用程序的需要，并优化性能和资源利用率。允许用户记录软件包的版本和构建选项，以确保在相同芯片架构的不同平台上生成相同的二进制文件，从而保证结果可重复性；另外特别针对鲲鹏平台引入鲲鹏软件全栈 (编译器 + 数学库 + 通信库 + 迁移调优工具链)，大幅提升鲲鹏平台的应用运行效率。

仓库地址

https://gitee.com/openeuler/hpcrunner

WayCa scheduler

服务器

WayCa SIG

WayCa scheduler 提供了一套软件库和工具，在开源 Linux 基础上，优化并完善了鲲鹏平台的硬件拓扑导出和任务调度。

技术挑战

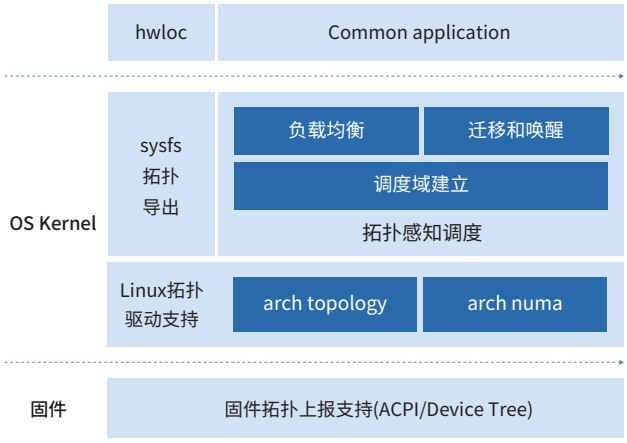
当前服务器核数规模越来越大，缓存和互联结构越来日趋复杂，不同厂商的服务器设计存在较大差异。虽然 Linux 作为通用操作系统能够支持不同厂商的硬件，但是对于鲲鹏新引入的硬件拓扑架构支持不够完善。如何在现有软件平台，充分发挥鲲鹏硬件性能是一个挑战。

项目介绍

WayCa scheduler 基于鲲鹏硬件拓扑结构，通过结合和完善固件拓扑描述，并在 Linux 内核中完善硬件拓扑的建立和拓扑信息的导出，优化内核调度算法，让应用程序能够充分利用 CPU， Cache，内存，IO 外设等组件，提升系统硬件的利用率和内存带宽，降低内存、Cache 及外设的访问延迟，从而提升应用在鲲鹏服务器上的性能。

当前，WayCa scheduler 主要包含如下特性：

- 拓扑发现和导出： 支持 Linux 的 ACPI 及拓扑驱动对于硬件 CPU、Cache、NUMA、设备拓扑的枚举和建立，通过 sysfs 等内核接口实现硬件拓扑信息的检索。
- 调度支持和优化：基于硬件拓扑结构建立 Cluster 和 NUMA 调度域，修复调度器对于鲲鹏服务器支持的缺失和缺陷。任务调度能够基于硬件的 Cluster 及 NUMA 实现负载均衡和迁移，充分利用 L3 Cache 和内存资源，优化系统的延迟和吞吐量。
- 用户态拓扑支持：考虑 Linux 内核主要面向通用场景，而特定场景及应用需要根据自身的业务特点和需求基于硬件进行优化，比如实现特定的 CPU 或设备的绑定策略。考虑这部分应用的需求，通过对 hwloc 的适配为应用提供硬件拓扑信息的支持。



应用场景

WayCa scheduler 功能已经合入 openEuler 内核及相关用户态工具，运行于鲲鹏服务器和 openEuler 操作系统的应用程序，诸如通用数据库等场景，通过内核实现硬件拓扑信息的感知和调度优化，提升性能；在特定应用场景，如 HPC 应用，可以通过 hwloc 等工具获取拓扑信息，满足应用本身的优化策略需求，从而提升应用性能。

仓库地址

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

HybridSched 虚拟化混合调度

云计算

Virt SIG

云数据中心资源利用率低是行业普遍存在的问题，提升资源利用率已成为一个重要的技术课题。将业务区分优先级混合部署（简称混部）运行是典型有效的资源利用率提升手段。

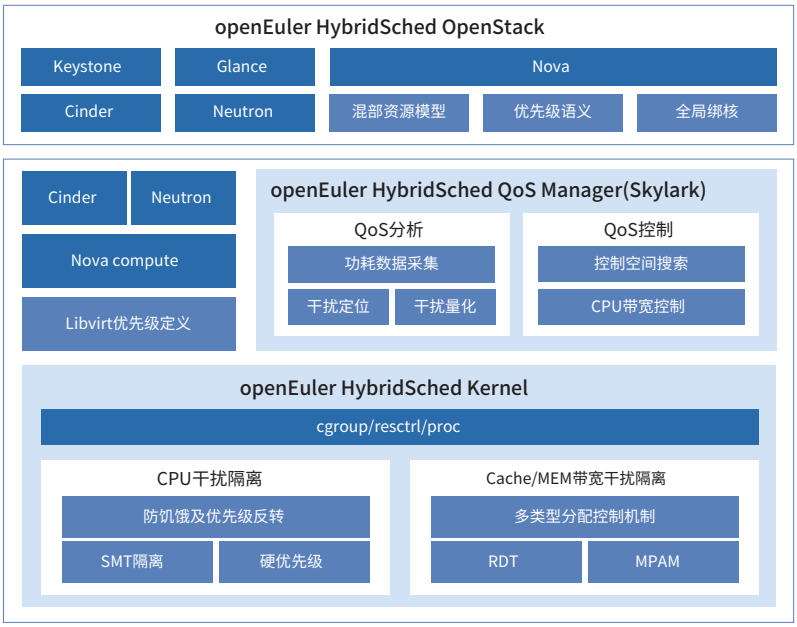
HybridSched 是虚拟机混部全栈解决方案，包括增强的 OpenStack 集群调度能力、全新单机 QoS 管理组件 Skylark 以及内核态基础资源隔离能力。其中 Skylark 是一种高低优先级虚拟机混部场景下的 QoS 感知资源调度器，在保障高优先级虚拟机 QoS 前提下提升物理机资源利用率。

技术挑战

混部的核心技术是资源隔离控制。服务器资源是多维度的，上层业务对硬件资源的需求处于动态变化中。此外虚拟机内部业务不可见，底层调度逻辑难以感知业务受损程度。这两方面现状对虚拟化平台混合调度技术提出了很大挑战。

项目介绍

- 集群调度增强：增强 OpenStack Nova 能力，支持优先级语义调度。
- 功耗控制：通过对低优先级虚拟机的 CPU 带宽进行限制，以此达到降低整机功耗的同时保障高优先级虚拟机 QoS。
- Cache 及内存带宽控制：支持对低优先级虚拟机的 LLC 和内存带宽进行限制，当前仅支持静态分配。
- CPU 干扰控制：支持 CPU 时间片 us 级抢占及 SMT 干扰隔离。同时具有防优先级反转能力。



应用场景

业务可根据时延敏感性分为高优先级业务和低优先级业务，将业务区分优先级混合部署以提高资源利用率。高优先级虚拟机业务推荐：时延敏感类业务，如 web 服务、高性能数据库、实时渲染、机器学习推理等。低优先级虚拟机业务推荐：非时延敏感类业务，如视频编码、大数据处理、离线渲染、机器学习训练等。

仓库地址

https://gitee.com/openeuler/skylark

KubeOS 容器操作系统

服务器 云计算 边缘计算

sig-CloudNative

KubeOS 是面向云原生场景的容器操作系统，通过 Kubernetes 统一纳管容器和节点 OS，提供 API 运维化、原子化、轻量安全的云原生场景下集群 OS 的运维方案。

技术挑战

云原生场景中，容器和 kubernetes 的应用越来越广泛，然而随之而来的就是云原生场景下的 OS 的管理问题：

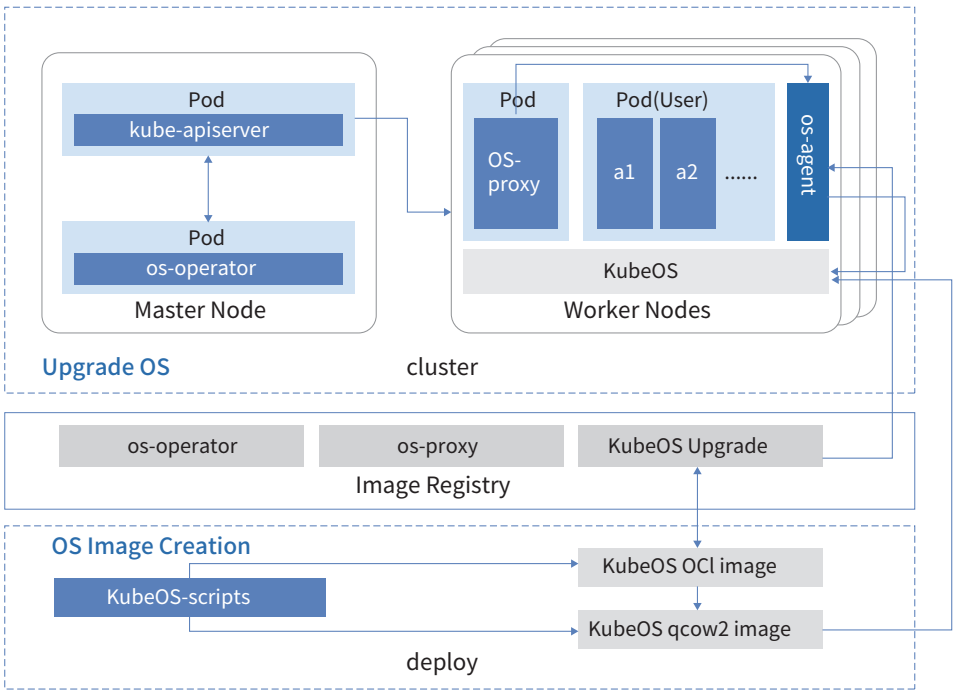
- 云原生场景下应用纷纷容器化，对 OS 有着新的挑战，传统的 OS 形态过重，不再完全适用。
- 容器和 OS 的运维管理的分别独立进行，往往会出现管理功能冗余，两套调度系统协调困难的问题。
- 单独的包管理会导致集群容器 OS 版本零散状态不统一等问题，缺乏统一的容器 OS 管理方式，容器 OS 的版本管理困难。

为解决以上问题，我们提出了 KubeOS，一套通过 Kubernetes 统一纳管容器和 OS 的 OS 运维方案。

项目介绍

功能描述

KubeOS 基于 openEuler 提供了一套云原生场景下 OS 运维管理的解决方案。基于 openEuler 构建轻量化的容器操作系统，并将 OS 作为组件接入到 Kubernetes 中，使得可以通过 Kubernetes 统一对容器和 OS 进行原子化运维管理。



KubeOS OS 统一运维管理

KubeOS 的主要特性如下：

- 统一管理：KubeOS 将 OS 作为组件接入到集群中，使用 kubernetes 统一管理 OS 和业务容器，统一管理所有节点 OS。
- 协同调度：OS 变更前感知集群状况，实现业务容器和 OS 的协同调度。
- API 运维：使用 kubernetes 原生的声明式 API 管理运维 OS，运维通道标准化。
- 原子管理：结合 kubernetes 生态，实现 OS 的原子升级 / 回滚能力，保证集群节点一致性。
- 轻量安全：仅包含容器运行所需组件，减少攻击面和漏洞，提高安全性，降低 OS 运行底噪和重启时间，只读根文件系统，保证系统不被攻击和恶意篡改。

应用场景

KubeOS 主要应用在云原生场景的基础设施中，提供云服务的基础运行环境，助力云厂商，通信行业客户解决云原生场景 OS 运维难题。

仓库地址

https://gitee.com/openeuler/KubeOS

NestOS 云底座操作系统

服务器 云计算 边缘计算 嵌入式

sig-K8sDistro

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件能力，可以更安全的运行大规模的容器化工作负载。

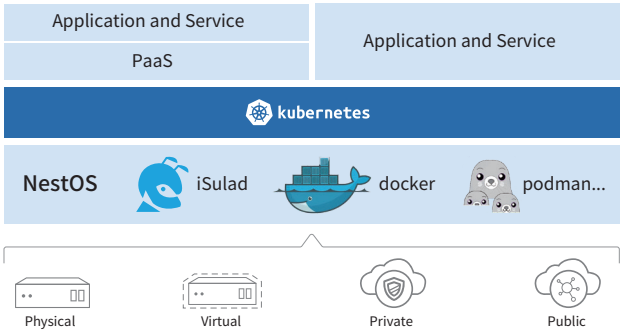
技术挑战

云原生场景中，容器和 kubernetes 等相关技术的应用越来越广泛，随之而来出现了多种多样的容器运行时及相关管理软件，因此在实际使用中容易出现，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。

项目介绍

功能描述

- 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。
- 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。
- 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。
- 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。
- 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，解决了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。

仓库地址

https://gitee.com/openeuler/NestOS

Rubik 容器混部引擎

服务器 云计算 边缘计算

sig-CloudNative

Rubik 是一个自适应单机算力调优和服务质量保障的容器混部引擎，通过对资源进行合理调度与隔离，在保障关键业务服务质量的前提下实现节点资源利用率提升。

技术挑战

当前全球云基础设施服务支出费用庞大，然而数据中心用户集群的平均 CPU 利用率却很低 (仅为 10%~20%)，存在巨大的资源浪费，带来了极大的额外运维成本，成为制约各大企业提升计算效能的关键问题。因此，提升数据中心资源利用率是当前急需解决的一个重要问题。将在线作业与离线作业混合部署，以空闲的在线集群资源满足离线作业的计算需求能够有效提升数据中心资源利用率，成为当今学术界和产业界的研究热点。

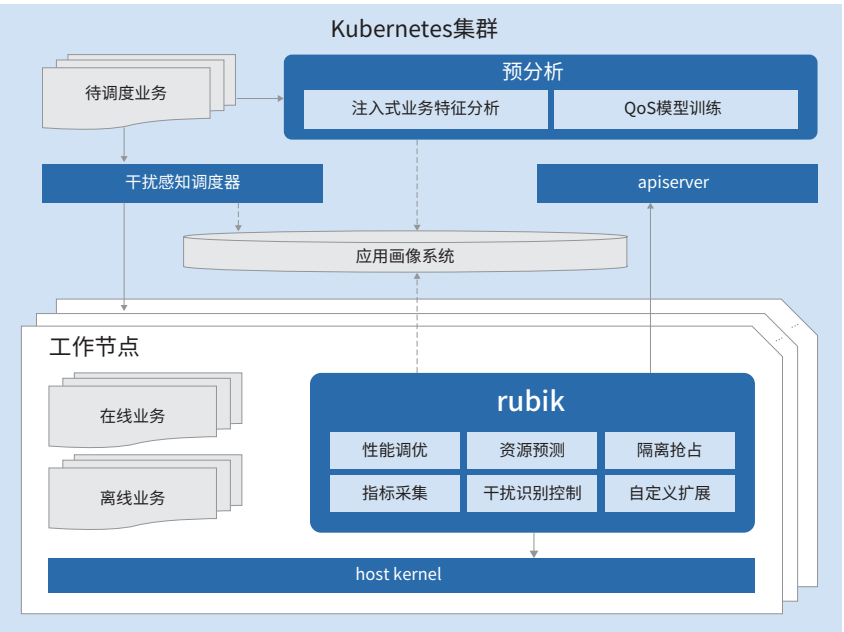
然而，将多种类型业务混合部署能够显著提升集群资源利用率，也带来了共峰问题，会导致关键业务服务质量（QoS）受损。

因此，如何在提升资源利用率之后，保障业务 QoS 不受损是技术上的关键挑战。

项目介绍

功能描述

Rubik 是 openEuler 提供的容器混部引擎，提供一套自适应的单机算力调优和服务质量保障机制，旨在保障关键业务服务质量的前提下，提升节点资源利用率。Rubik 字面意思为魔方，魔方由 Rubik 在 1974 年发明，故 Rubik 既是人名也指代魔方，在我们的解决方案中，Rubik 象征着能够将服务器管理的有条不紊。



Rubik 混部解决方案

包括的能力如下：

- 兼容原生 kubernetes 系统：基于原生 kubernetes 的扩展接口进行能力扩展。
- 兼容 openEuler 系统：自动使能 openEuler 提供的增强特性（如内核分级资源隔离技术），对于其他 Linux 发行版，由于存在部分内核特性缺失，仅提供受限管理能力。
- 运行时干扰识别控制：提供对关键业务性能干扰实时检测能力、干扰源快速定位能力以及干扰快速控制能力。
- 自适应动态调优：例如对关键业务性能优化，使其能能更高效稳定的运行；动态在离线资源配比调优，减少关键业务 QoS 违规等等。
- 支持自定义扩展：支持高级用户针对特定业务场景开发自定义扩展插件。

应用场景

Rubik 在云业务容器混合部署应用比较广泛，应用场景涵盖 Web 服务、数据库与大数据、AI 等混部场景。助力互联网、通信等行业客户实现数据中心资源利用率突破 50%。

仓库地址

https://gitee.com/openeuler/rubik

GearOS 齿轮操作系统

边缘计算　嵌入式

sig-industrial-control

GearOS，即齿轮操作系统，是由 openEuler 开源社区 Industrial-Control SIG 孵化的一款面向工业控制领域的实时增强操作系统，专注于操作系统实时性、可靠性，基于 openEuler Embedded 开源操作系统，使用 Yocto 构建，可应用于汽车控制、机器人控制、PLC 控制、机床控制等领域。

技术挑战

信息技术与工业系统全方位融合并形成产业及应用生态是工业向智能化发展的必由之路。二者的融合将带来智能化提升，如智能化生产、网络化协同、个性化定制、服务化转型；智能制造便是二者融合的产物。

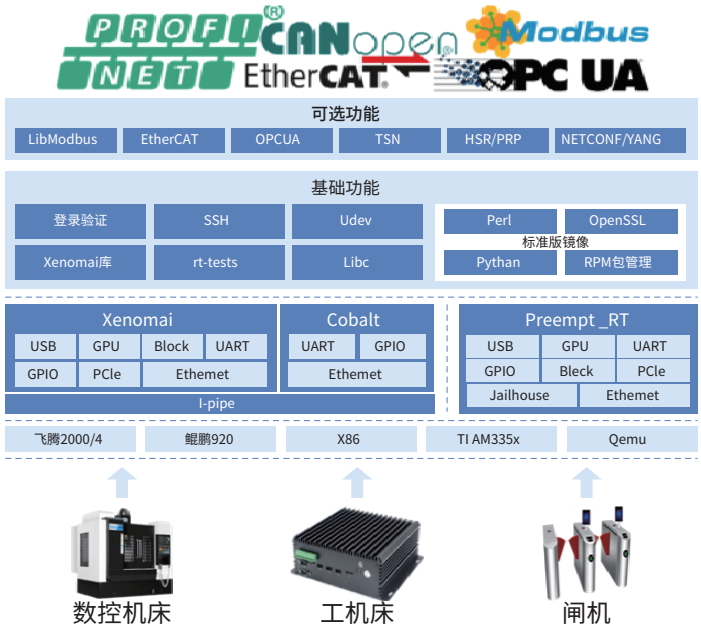
为实现信息技术与工业系统的融合，大数据、AI、机器人控制、端云协同等技术不可或缺。这些技术深度依赖于 Linux 生态，而工业系统由于其独特的应用场景，对实时性、确定性以及功能安全方面有着强烈的需求，则更加偏向于使用 RTOS 系统。而 Linux 系统设计之初更偏向于系统吞吐率、公平调度等特性，对实时性等方面考虑欠佳。

工业系统与信息技术融合在技术实现上成为一个难题。而 GearOS 系统在某些程度上能够解决这个问题。

项目介绍

GearOS 版本主要基于 Arm64 架构，主要包含两个内核和两个文件系统镜像。

- 两个内核：分别为支持 Preempt_RT 实时特性与 Jailhouse 虚拟化的内核和支持 Xenomai 实时特性的内核，均基于 openEuler 4.19 内核改造而来，大小为 8MB。
- 两个文件系统镜像：分别为紧凑型文件系统镜像和标准文件系统镜像。其中紧凑型文件系统镜像使用 BusyBox 制作，大小为 5.4MB；标准文件系统镜像未使用 BusyBox。



系统主要特性

- 支持飞腾 2000/4、鲲鹏 920、TI AM335X、Qemu-Arm64、x86 等平台
- 内核最低可做到 3.3MB，
- 支持串口、网络、块设备、USB、PCIe 等驱动 文
- 件系统最低可做到 5.4MB
- 启动时间小于 5S
- 支持 Preempt_RT 和 Xenomai 实时方案
- 支持 Jailhouse 虚拟化
- 紧凑型文件系统镜像包含登录验证、Udev、SSH、Xenomai 库、rt-tests 工具集
- 标准型文件系统镜像增加 Python、Perl、OpenSSL、Sqlite、RPM 包管理等

工业相关附加特性

- 支持 LibModbus 协议
- 支持 EtherCAT 协议
- 支持 OPC UA 协议
- 支持 TSN
- 支持 HSR/PRP
- 支持 NETCONF/YANG

实时相关特性

在 FT-2000/4、鲲鹏 920 硬件设备，使用 openEuler 4.19 内核，使用 cyclictest 测试工具对比测试结果。

平台	测试环境（空载）			非CPU隔离	CPU隔离
	测试参数：cyclictest -m -h 100 -q -l 10000000 -i100 -t 1 -n -p 99				
鲲鹏 920	openEuler 20.03 LTS sp1	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

虚拟化相关特性

1. 对宿主机影响评估结果：

- 使用 Unixbench、lmbench、iozone、netperf 工具进行裸机测试和开启 Jailhouse 虚拟空载 Linux 的对比测试，测试结果虚拟化对 CPU、内存、存储、网络、IO 几乎没有性能损耗
- 使用飞腾 2000PC 平台测试 Jailhouse 对 Guest Linux 总线数据穿透的性能影响
- Ethernet 网卡穿透使用 iperf 和 netperf 多维度测试，几乎没有性能损耗
- PCI-USB 设备穿透使用 dd、hdparm、iozone 多维度测试，几乎没有性能损耗
- Ivshmem 网络传输使用 iperf 测试，性能优于千兆网卡

2. 对 Guest 系统影响

- 中断响应，测试 500 万次外部中断，avg 从 290ns 增加到 1200ns，max 从 2400ns 增加到 2500ns
- 中断抖动约 1000ns
- Ethernet 网卡穿透使用 iperf 测试 10Mbit 链接模式，几乎没有性能损耗（采用 lwip，只能使用 10Mbit 带宽模式）
- I2C 总线数据穿透，测试 1000 次取平均值，几乎没有性能损耗
- SPI 总线数据穿透，测试 1000 次取平均值，几乎没有性能损耗
- CAN 总线数据穿透，测试 1000 次取平均值，几乎没有性能损耗

应用场景

目前该项目在南方电网项目的工控机设备、智能交通项目的闸机设备、某公司数控机床进行积极推广。

仓库地址

https://gitee.com/openeuler/GearOS

MICA 混合关键性部署框架

边缘计算 嵌入式

sig-embedded

MICA 是一个面向多核 SoC 依托硬件辅助虚拟化、TEE、异构等技术的支持以实时与非实时 OS，安全与非安全 OS 为代表的多 OS 高效混合部署的框架，可以充分发挥各个 OS 的特点以满足嵌入式系统以安全、实时、富功能为代表的多目标约束。

技术挑战

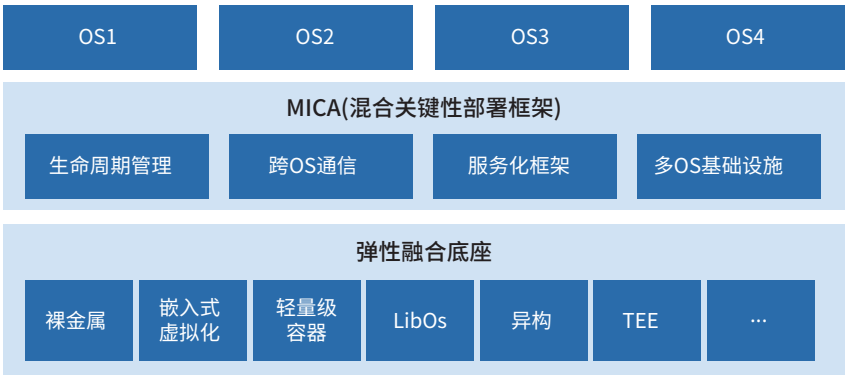
对于当前的嵌入式系统，一方面由于硬件越来越强大，可以有力支撑运行像 Linux 这样复杂的操作系统；另一方面应用也变得越发复杂，包含了越来越多的需求，例如互联的需求、AI 的需求、迭代升级的需求等等。面对复杂而繁多的需求，实践中也往往需要像 Linux 这样强大的操作系统来满足。但同时也必须认识到，嵌入式系统与一般的计算机系统的不同之处在于，其往往有资源限制、功耗限制、实时性、可靠性、安全性等方面的约束。 这些约束并没有随着系统的复杂化而变化。对于这些约束，受限于自身的复杂架构 Linux 并不能很好的满足， 而往往是以实时操作系统乃至裸金属运行时为代表的相对精简的专用系统的用武之地。

对于嵌入式系统而言，实现包含 Linux 和实时操作系统在内的多个 OS 混合部署以满足以安全、实时、富功能为代表的多目标约束，主要面临的挑战有三类：

- 部署：多个 OS 如何高效地部署在同一个多核 SoC 上，可能是同构多核，也可能是异构多核，彼此间高效协同工作，共同实现全系统功能
- 隔离：多个 OS 间彼此间互补不影响，一个 OS 出现问题，如崩溃、故障等不会影响到其他 OS，特别具有高安全、高可靠、高实时要求的 OS
- 调度：多个 OS 能够充分利用硬件资源，有较高的资源利用率

项目介绍

MICA 是一个面向多核 SoC 依托硬件辅助虚拟化、TEE、异构等技术的支持以实时与非实时 OS，安全与非安全 OS 为代表的多 OS 高效混合部署的框架，可以充分发挥各个 OS 的特点以满足嵌入式系统以安全、实时、富功能为代表的多目标约束。MICA 的总体架构图如下图所示：



MICA 需要与弹性融合底座配套使用，融合弹性底座是为了在多核 SoC 上实现多个操作系统 / 运行时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）、异构等多种实现形态。不同的形态有各自的特点，例如裸金属可以得到最佳的性能、嵌入式虚拟化可以实现更好的隔离与保护、轻量级容器则有更好的易用性与灵活性等等。

构建在融合弹性底座之上，通过一套统一的框架屏蔽下层融合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。

混合关键性部署框架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。

应用场景

MICA 项目正在孵化中，其主要应用场景是制造、能源、机器人等领域中的中高端复杂嵌入式系统。MICA 当前已经支持基于 openAMP 的裸金属形态，和基于 jailhouse 的分区虚拟化形态，未来将与 ZVM 和 Rust-Shyper 项目配合。

仓库地址

https://gitee.com/openeuler/mcs

Rust-Shyper 嵌入式 Type-1 型虚拟机监视器

嵌入式

Virt SIG

Rust-Shyper 是一款基于 AArch64 架构、用 Rust 编写、面向无人车、机器人等嵌入式场景的 Type-1 型虚拟机监控器。其设计目标是在提高资源利用率的同时，保障虚拟机实时性、隔离性与内存安全，同时支持虚拟机迁移和监控器动态升级两种热更新机制，能够在不影响虚拟机运行的情况下，动态修复软件漏洞。

技术挑战

物联网的不断发展使得现代嵌入式系统正在朝着通用系统和混合关键系统的方向演化，其承载的任务往往有着不同的可靠性、实时性和验证级别，如何保证不同关键性任务之间的相互隔离以及实时性成为了一个难题。虚拟化技术提供的资源隔离手段成为了解决上述问题的关键，但嵌入式虚拟化也面临一些挑战：

- 如何保证虚拟机之间的「隔离性和安全性」，防止恶意攻击；
- 如何保证虚拟机之间的通信效率和「实时性」，避免延迟或者抖动；
- 如何保证「Hypervisor 本身的稳定性和可靠性」，防止出现故障。

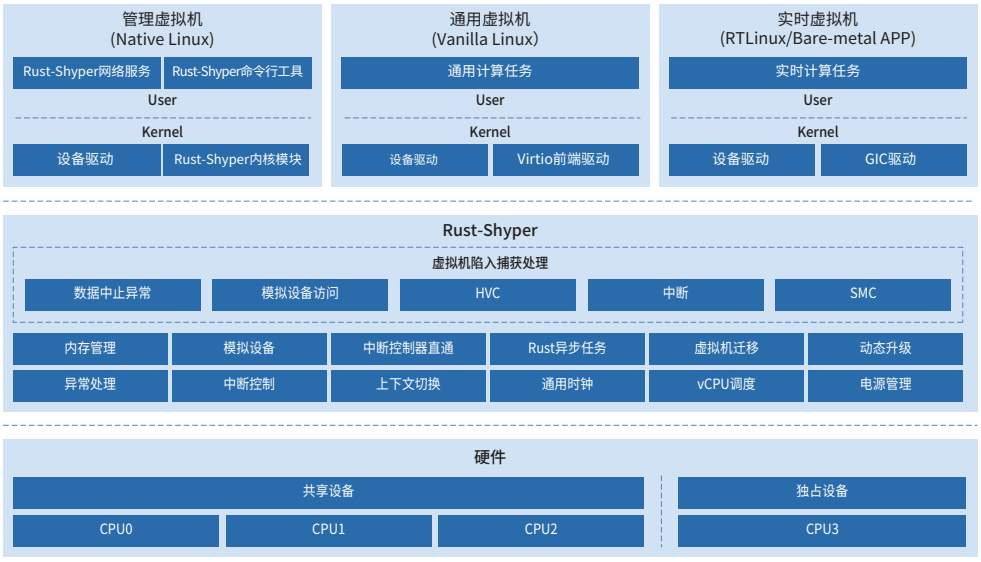
项目介绍

Rust-Shyper 由北京航空航天大学王雷教授团队开发，2023 年 4 月捐献给 openEuler 社区，在 SIG-Virt 持续孵化。

Rust-Shyper 是一款基于 AArch64 体系结构的 Type-1 虚拟机监控器，整个结构包含三个层级：

- 最底层为硬件层级，对应 Armv8 EL3 固件层级；
- 中间层为虚拟机监控器层，对应 Armv8 EL2 虚拟化层级，该层级也是 Rust-Shyper 代码所处的特权层级；
- 最上层为虚拟机层级，对应 Armv8 EL1 和 EL0 层级。

为了符合嵌入式应用的需求，Rust-Shyper 通过提供不同的虚拟机类型，来提供差异化的虚拟化服务，Rust-Shyper 中支持管理虚拟机（MVM）、客户虚拟机（GVM）、实时虚拟机（RTVM）等三类虚拟机。



Rust-Shyper 的设计理念和特点：

- 内存安全：利用 Rust 语言类型系统和内存安全模型，保证 Hypervisor 的内存安全；
- 强隔离性：利用硬件辅助虚拟化，实现虚拟机间的安全隔离和故障隔离；
- 丰富的设备模型：为提高资源利用率，实现了直通设备、中介传递和全模拟等多种设备模型；
- 实时虚拟化：针对性能需求，实现资源直通以及实时虚拟化技术；
- 虚拟机监控器热更新技术：实现了虚拟机迁移和监控器动态升级两类视器热更新机制。

应用场景

Rust-Shyper 项目正在孵化中，其主要应用场景是制造、能源、机器人、汽车电子等领域中的中高端复杂嵌入式系统。

仓库地址

https://gitee.com/openeuler/rust_shyper

UniProton 硬实时操作系统

边缘计算 嵌入式

sig-embedded

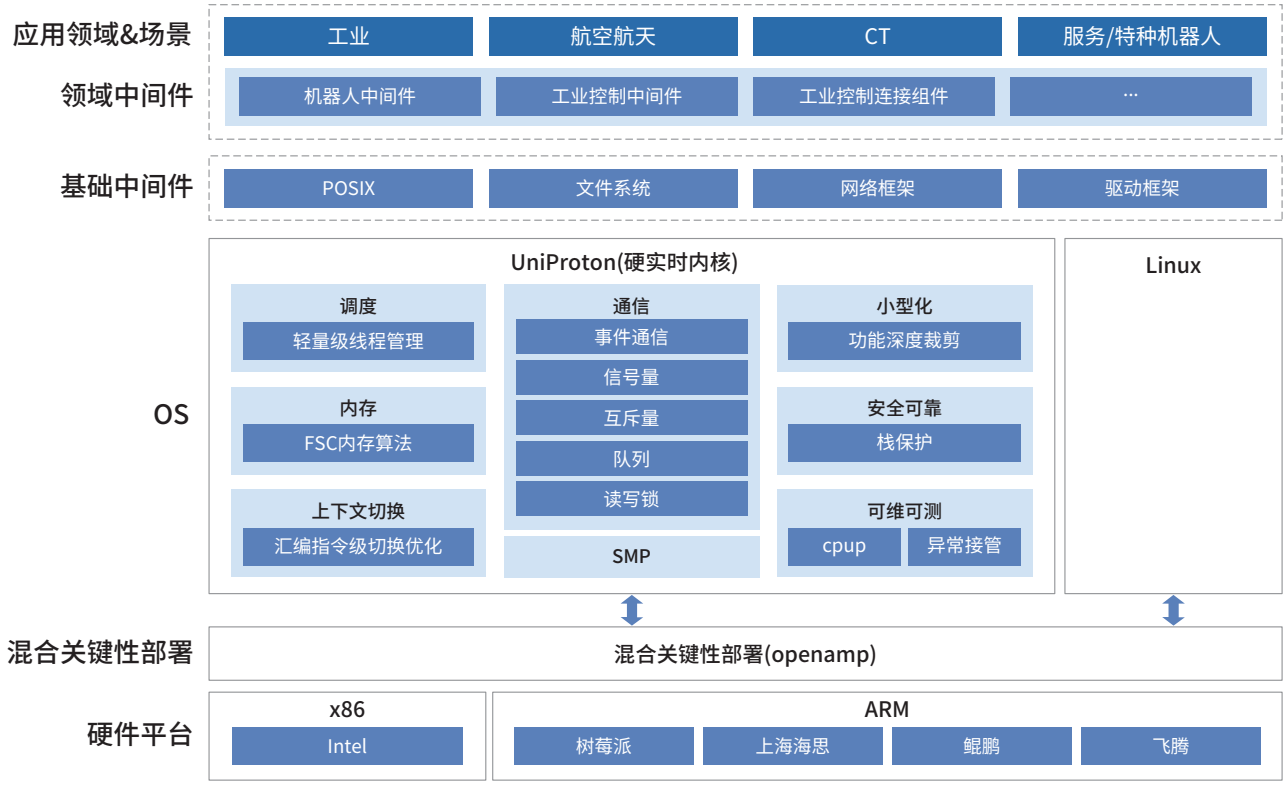
UniProton 是一款硬实时操作系统，具备极致的 us 级别的低时延和灵活的混合关键性部署特性，可以高效地与以 openEuler Embedded 为代表的通用 OS 混合部署，其主要适用于工业控制场景，既支持微控制器 MCU，也支持算力强的多核 CPU。

技术挑战

工业控制应用场景对操作系统有着强烈的确定性时延要求，Linux 由于其体量大、功能复杂等原因无法达成该目标，所以业界一直在推出小型的嵌入式操作系统以满足工业场景诉求。UniProton 凭借其轻量级内核、极致性能优化、功能可裁剪以及混合关键性部署，能够适应多种场景，并在各场景下达成极致低时延目标。

项目介绍

UniProton 结合了华为长期在 CT 领域硬实时场景的经验，于 2022 年 6 月开源到 openEuler 社区，在 sig-embedded 孵化。



UniProton 的主要架构如上图所示，其关键特性如下：

低时延：

- 确定性时延：最大调度时延满足业务需求
- 极致性能：us 级任务调度、中断时延
- 轻量化：可在几十 KB 内存环境上运行

通用性：

- POSIX 兼容：遵循 POSIX 标准 IEEE Std 1003.1 ™ -2008 提供接口
- 主流架构支持：支持 x86、ARM64、Cortex-M 等多种指令集架构，可在 Intel、树莓派等典型芯片上运行
- 多核处理器支持：高效使用多核算力，同时仍具备低延时能力

易用性：

- 功能定制化：功能可裁剪
- 维测功能：cpup、异常接管
- 混合关键部署：复用 Linux 能力，并提供 RTOS 能力

丰富中间件：

- 驱动框架：提供统一的标准化驱动开发框架，提升驱动开发效率
- 网络框架：支持丰富的网络协议栈，并提供标准网络接口
- 工业中间件：支持对接多种工业协议和开发标准，可提供 EtherCAT 等总线通讯能力

应用场景

UniProton 提供硬实时解决方案，广泛适用制造、医疗、能源、电力、航空航天等应用场景，满足工业生产、机器人控制等确定性时延诉求。

仓库地址

https://gitee.com/openeuler/UniProton

ZVM 嵌入式实时虚拟机

边缘计算 嵌入式

sig-Zephyr

ZVM 是一款基于顶级开源实时操作系统 Zephyr 结合硬件辅助虚拟机化技术的嵌入式实时虚拟机，支持包含 Linux、实时操作系统和 baremetal 程序在内的多个运行时混合部署以及混合关键性调度。

技术挑战

嵌入式实时虚拟化技术是一种允许在单个硬件平台上同时运行多个操作系统、并保持确定性和时间关键性能的技术，该技术可为嵌入式系统开发带来许多好处，例如硬件整合、系统隔离、系统灵活可靠性、安全性和可扩展性等。嵌入式实时虚拟化可支持智能汽车、数控机床及 5G 设备等高级应用。

开发嵌入式实时虚拟化软件面临着一些挑战。第一个挑战是如何确保不同 Guest OS 间的隔离和安全性，尤其是当它们具有不同级别的关键性和可信度时。第二个挑战是如何在不同 Guest OS 间有效地共享或分配 I/O 设备，这可能需要设备模拟或直通机制。第三个挑战是如何确保作为 Guest OS 运行的 RTOS 具备低延迟和高吞吐量。

嵌入式实时虚拟化软件需要通过提供强制的隔离和安全、高效的中断处理、灵活的 I/O 设备管理机制及硬件支持来应对这些挑战。

项目介绍

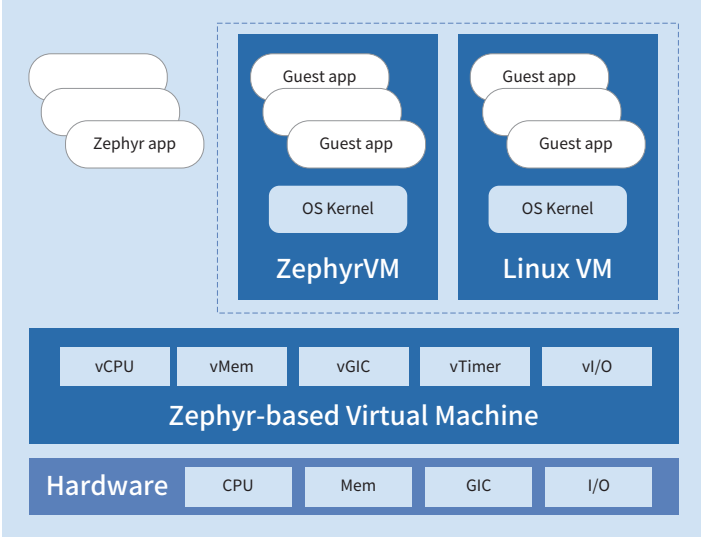
ZVM 由湖南大学嵌入式与网络计算湖南省重点实验室主任谢国琪教授团队开发，2023 年 4 月捐献给 openEuler 社区，在 SIG-Zephyr 持续孵化。

基于架构硬件虚拟化支持与虚拟化主机拓展支持，ZVM 实现了 Guest OS 间的隔离、设备分配及中断处理，保证了系统的安全与实时。

ZVM 总体功能有三个部分：安全隔离、设备管理和系统性能提升。

- 安全隔离：利用虚拟化技术实现不同特权级的应用支持，确保不同 Guest OS 间的隔离和安全，尤其是当它们具有不同级别的关键级时。为每个 Guest OS 分配不同的虚拟地址空间和虚拟设备，实现虚拟机间的隔离以保证系统安全。
- 设备管理：使用支持设备模拟和直通机制的管理程序，在不同 Guest OS 之间有效共享或分配 I/O 设备。对于中断控制器需独占的设备，用完全虚拟化的方式进行分配，对于 UART 等非独占的设备，使用设备直通的方式进行分配。
- 系统性能提升：在处理器方面，使用支持 Arm64 硬件辅助虚拟化拓展技术来减少上下文开销；在内存管理方面，使用基于硬件的两阶段地址转换地址转换性能开销；在中断方面，使用基于硬件的中断注入机制来减少上下文开销和中断时延。

ZVM 整体系统架构如下图所示，通过在 Zephyr RTOS 中加入虚拟化模块，实现 CPU 虚拟化、内存虚拟化、中断虚拟化、定时器虚拟化和 I/O 虚拟化。ZVM 当前支持两种类型的 Guest OS，即通用的 Linux 操作系统和 Zephyr RTOS。



- CPU 虚拟化：CPU 虚拟化模块的主要功能是为每个 Guest OS 的 vCPU 虚拟出一个单独的隔离上下文。每个 vCPU 均作为一个线程存在，由 ZVM 统一调度。为了提高 vCPU 的性能，Arm64 架构为 ZVM 提供了 VHE 支持，VHE 可以使 Host OS 迁移到 EL2 特权模式，而无需改变操作系统原有代码。VHE 主要实现了 Arm 寄存器重定向，可以在不修改 Zephyr RTOS 内核代码的情况下，将其迁移 EL2 层开发 ZVM，既降低了系统冗余，又提高了系统性能。
- 内存虚拟化：内存虚拟化模块的主要作用是实现 Guest OS 间内存地址的隔离。系统需要隔离不同 Guest OS 的内存空间，监控 Guest OS 对实际物理内存的访问，以保护物理内存。为了实现该功能，Arm64 提供了两阶段的地址查找策。第一阶段是从 Guest OS 的虚拟地址到 Guest OS 的物理地址转换，第二阶段是从 Guest OS 的物理地址到 Host OS 的物理地址转化。Arm 专门为第二阶段转换提供单独的硬件，以提高地址翻译性能。
- 中断虚拟化模块：中断虚拟化使用 Arm 的通用中断控制器（GIC）设备，并基于该设备实现虚拟中断配置。Guest OS 的中断统一路由到 ZVM，然后 ZVM 会将它们分配给不同的 vCPU。虚拟中断的注入通过 GIC 中的 Virtual CPU 接口或 List Register 具体实现。
- 定时器虚拟化：定时器虚拟化为每个 CPU 定义了一组虚拟定时器寄存器，它们在预定时间后单独计数并抛出中断，由 Host OS 转发给 Guest OS。同时，在 Guest OS 切换过程中，虚拟定时器会计算 Guest OS 的实际运行时间，并对 Guest OS 退出的时间进行补偿，为 Guest OS 提供定时器服务。
- 设备虚拟化：在设备虚拟化方面，ZVM 采用 Arm 中的 Memory-Mapped I/O (MMIO) 方法将设备地址映射到虚拟内存地址，构建虚拟设备空间，实现 Guest OS 对设备地址的访问。在具体实现上，ZVM 统一构建一个虚拟的 MMIO 设备，在 Guest OS 创建过程中将该设备分配给指定的 Guest OS，实现 I/O 虚拟化。此外，对一些非独占设备，ZVM 使用设备直通的方式实现设备的访问。

应用场景

ZVM 项目正在孵化中，其主要应用场景是制造、能源、机器人、汽车电子等领域中的中高端复杂嵌入式系统。

仓库地址

https://gitee.com/openeuler/zvm

dsoftbus 分布式软总线

边缘计算 嵌入式

sig-distributed-middleware

openEuler 秉承打造“数字化基础设施操作系统”的愿景，为实现端边领域的互通和协同，首次在服务器 & 边缘 & 嵌入式领域引入分布式软总线技术。分布式软总线作为分布式设备通信基座，为设备之间的互通互联提供统一的分布式协同能力，实现设备无感发现和数据高效传输。

技术挑战

边端设备之间的互联是实现边端设备协同工作的基石，涉及边端设备的发现、连接、组网、传输等环节。当前边端设备的互联存在以下难点：

- 端设备形态不一：硬件能力各异，支持的连接方式参差不齐，比如 WiFi、蓝牙、NFC 等多种方式，缺少统一的方案覆盖各种连接方式。
- 稳定快速组网难：如何在边端设备间自动构建和分配组网管理角色，实现网络的鲁棒性，在设备退出、掉电、故障后仍能保持组网的稳定。
- 传输性能差：如何达成边端设备间最佳传输性能，特别是当部分端设备对功耗有一定的约束时。
- 接口适配难：对于上层应用开发者，如何做到提供统一的接口，屏蔽底层硬件、组网的差异，能够让应用开发者不用关心底层实现，聚焦业务流程，实现一次开发、边端复用。

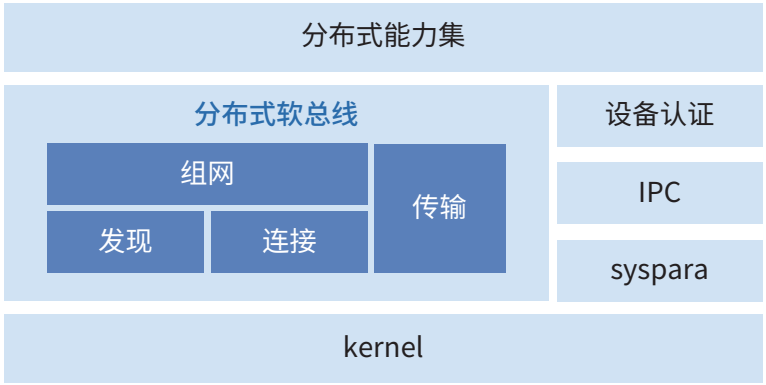
项目介绍

主要能力

- 发现连接：提供基于 Wifi、有线网络及蓝牙等通信方式的设备发现连接能力。
- 设备组网：提供统一的设备组网和拓扑管理能力，为数据传输提供已组网设备信息。
- 数据传输：提供数据传输通道，支持字节、流、文件的数据传输能力。

模块架构

软总线主体功能分为发现、组网、连接和传输四个基本模块。



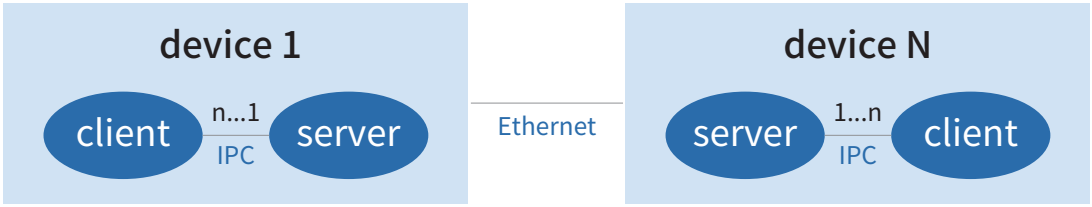
软总线与外部模块架构关系图

软总线南向支持 Wifi、有线网络及蓝牙等通信方式。并为北向的分布式应用提供统一的 API 接口，屏蔽底层通信机制。

软总线依赖于设备认证、IPC、日志和系统参数（SN 号）等周边模块，在嵌入式场景将这些依赖模块进行了样板性质的替换，以实现软总线基本功能。实际的周边模块功能实现，还需要用户根据实际业务场景进行丰富和替换，以拓展软总线能力。

部署示意：

- 软总线支持局域网内多设备部署，设备间通过以太网通信。
- 单设备上分为 server 和 client，二者通过 IPC 模块进行交互。
- 单节点上支持多 client 同时接入单一 server。



部署示意图

如部署模型，软总线通过独立进程部署的方式对外提供服务，通过执行服务端主程序可拉起软总线进程提供对外服务。

应用场景

分布式软总线主要适用于 openEuler 边缘服务器、嵌入式设备及 OpenHarmony 嵌入式设备之间的自发现、互联互通。

多用于工业产线、园区设备管理场景，通过软总线统一接口和协议标准，让不同厂家、不同类型硬件设备之间自连接、自组网、新设备即插即用，实现数据与外设互通访问。

仓库地址

https://gitee.com/openeuler/dsoftbus_standard

openEuler Edge

云计算 边缘计算

sig-Edge

openEuler 发布面向边缘计算的版本 openEuler Edge，集成 KubeEdge 边云协同框架，KubeEdge 是 CNCF 基金会孵化的开源项目，具备边云应用统一管理和发放等基础能力，并将通过增强智能协同提升 AI 易用性和场景适应性，增强服务协同实现跨边云服务发现和流量转发，以及增强数据协同提升南向服务能力。

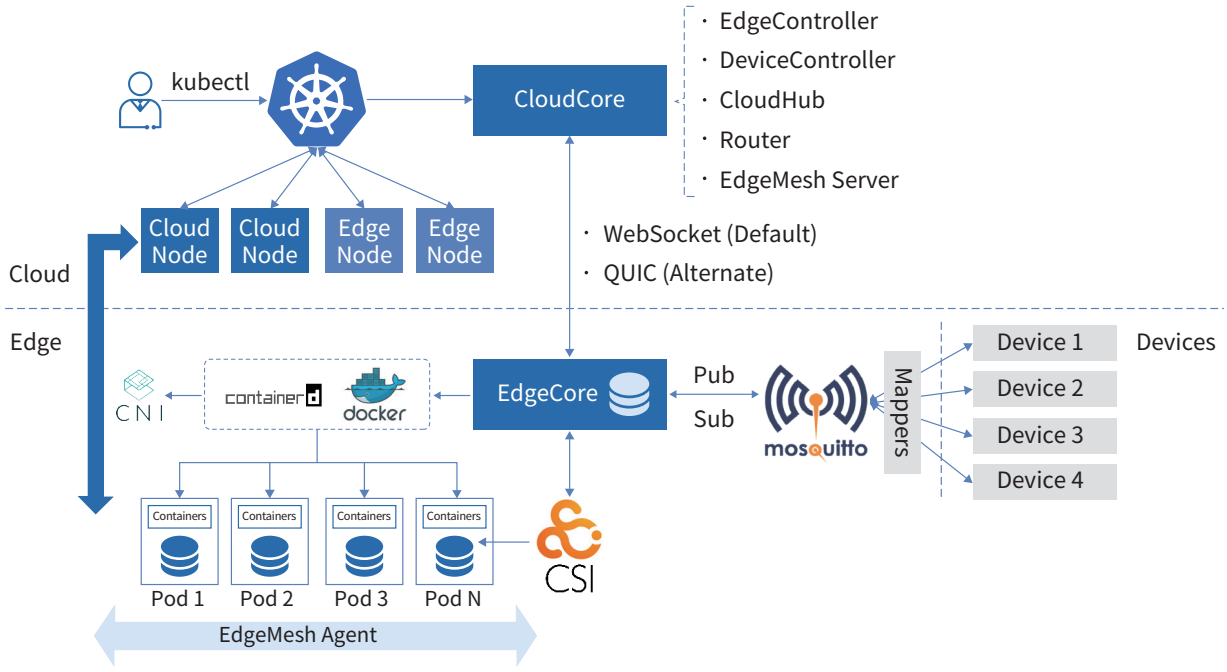
技术挑战

边缘计算是未来 10 大战略技术趋势。随着智慧城市、自动驾驶、工业互联网等应用落地，海量数据将在边缘产生，IDC 预测中国 2025 年每年产生的数据将达 48.6ZB，集中式云计算在带宽负载、网络延时、数据管理成本等方面愈发显得捉襟见肘，难以适应数据频繁交互需求，边缘计算价值凸显。

项目介绍

openEuler 发布面向边缘计算的版本 openEuler Edge，集成 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，并将通过增强智能协同提升 AI 易用性和场景适应性，增强服务协同实现跨边云服务发现和流量转发，以及增强数据协同提升南向服务能力。

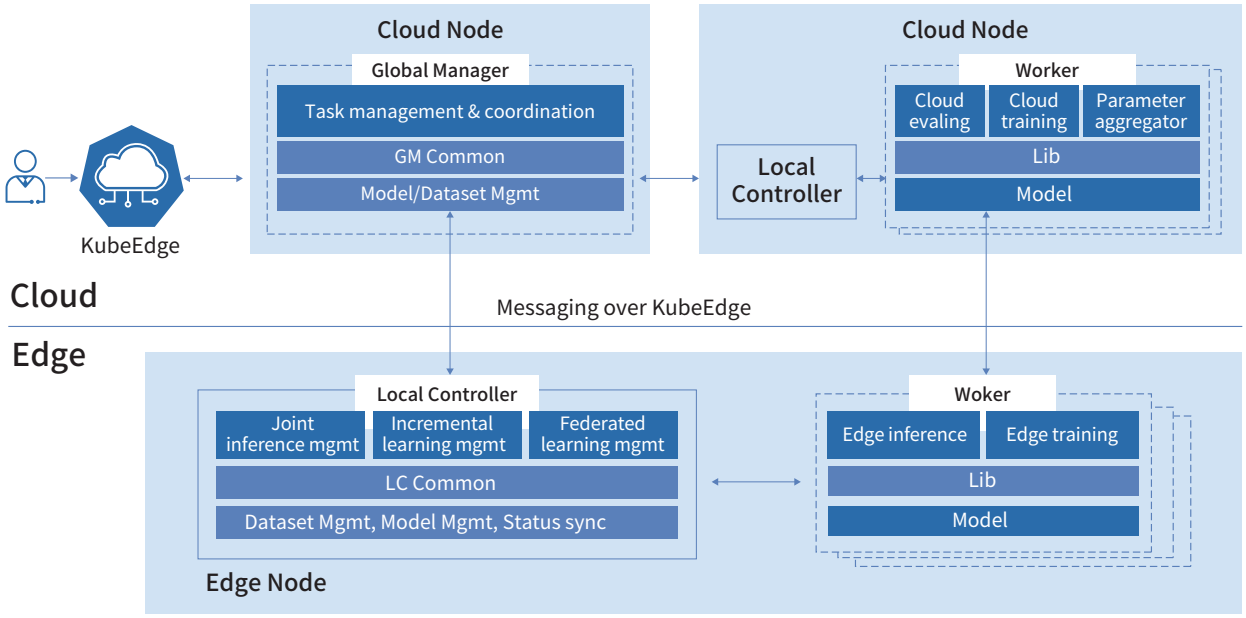
功能描述



KubeEdge 边云协同基础框架

边云协同基础框架功能如下：

- 边云管理协同：实现边云之间的应用管理与部署、跨边云的通信，以及跨边云的南向外设管理等基础能力。
- 边云服务协同：边侧部署 EdgeMesh Agent，云侧部署 EdgeMesh Server 实现跨边云服务发现和服务路由。
- 边缘南向服务：南向接入 Mapper，提供外设 Pofile 及解析机制，以及实现对不同南向外设的管理、控制、业务流的接入，可兼容 EdgeX Foundry 开源生态。
- 边缘数据服务：通过边缘数据服务实现消息、数据、媒体流的按需持久化，并具备数据分析和数据导出的能力。



Sedna 边云智能协同框架

边云智能协同功能如下：

- Sedna GM 及 LC：GM 全局任务协调与控制，LC 本地数据集管理与模型管理，状态同步，边缘自治，提供基础的边云协同推理、联邦学习、增量学习等能力，并实现了基础的模型管理、数据集管理等，提升用户边云 AI 特性的训练与部署效率。
- Sedna Lib：使能开发者快速开发边云 AI 协同特性，提升开发效率。

应用场景

可应用安平、能源、交通、制造、金融、医疗、园区、无人系统等广泛的边云协同场景。

仓库地址

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

基础能力创新

A-Tune 智能调优引擎

服务器

A-Tune SIG

A-Tune 是一款基于 AI 的操作系统性能调优引擎。A-Tune 利用 AI 技术，使操作系统“懂”业务，简化 IT 系统调优工作的同时，让应用程序发挥出色性能。

技术挑战

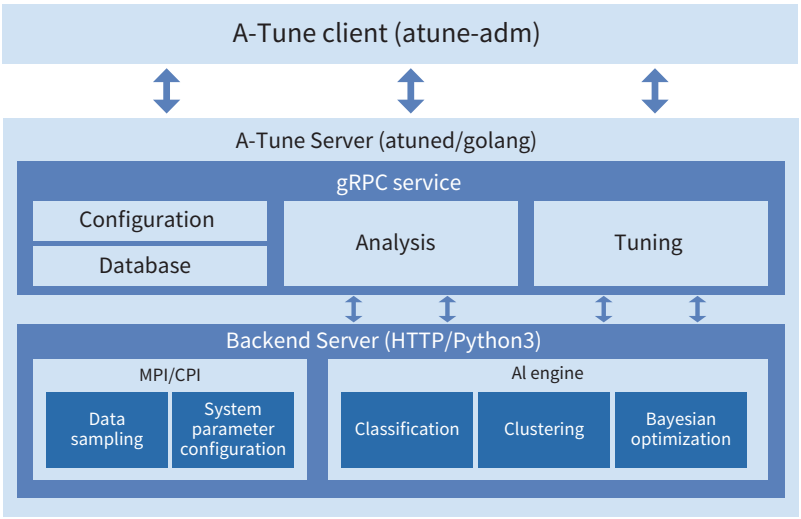
随着几十年来硬件和软件应用的不断发展，Linux 内核正变得越来越复杂，而整个操作系统也变得越来越庞大。在 openEuler 开源操作系统中，仅 sysctl 命令（用于运行时配置内核参数的命令）的参数（sysctl -a | wc -l）就超过 1000 个，而完整的 IT 系统从最底层的 CPU、加速器、网卡，到编译器、操作系统、中间件框架，再到上层应用，可调节参数超过 7000 个。而大部分使用者只使用了这些参数的默认配置，因此无法充分发挥系统最佳性能。然而，针对特定的应用场景进行调优存在以下几方面的难点：

- 参数数量多，且参数间存在依赖关系
- 上层应用系统种类多，不同应用系统的参数不同
- 每个应用的负载也复杂多样，不同负载对应的最优参数值也不同

项目介绍

功能描述

A-Tune 的整体架构如图所示，其整体上是一个 C/S 架构。客户端 atune-adm 是一个命令行工具，通过 gRPC 协议与服务端 atuned 进程进行通讯。服务端中 atuned 包含了一个前端 gRPC 服务层（采用 golang 实现）和一个后端服务层。gRPC 服务层负责优化配置数据库管理和对外提供调优服务，主要包括智能决策（analysis）和自动调优（tuning）。后端服务层是一个基于 Python 实现的 HTTP 服务层，包含了 MPI（Model Plugin Interface）/CPI（Configurator Plugin Interface） 和 AI 引擎。其中，MPI/CPI 负责与系统配置进行交互，而 AI 引擎负责对上层提供机器学习能力，主要包括用于模型识别的分类、聚类和用于参数搜索的贝叶斯优化。



A-Tune 软件架构

A-Tune 目前主要提供两个能力：智能决策和自动调优。

智能决策的基本原理是通过采集系统数据，并通过 AI 引擎中的聚类和分类算法对采集到的数据进行负载识别，得到系统中当前正在运行的业务负载类型，并从优化配置数据库中提取优化配置，最终选取适合当前系统业务负载的最优参数配置。具备以下功能：

- 重要特征分析：自动选择重要特征，剔除冗余特征，实现精准用户画像
- 两层分类模型：通过分类算法，准确识别当前负载
- 负载变化感知：主动识别应用负载的变化，实现自适应调优

自动调优的基本原理是基于系统或应用的配置参数及性能评价指标，利用 AI 引擎中的参数搜索算法，反复迭代，最终得到性能最优的参数配置。具备以下功能：

- 重要参数选择：自动选择重要的调优参数，减少搜索空间，提升训练效率
- 调优算法构建：用户可从适用场景、参数类型、性能要求等方面选择最优算法
- 知识库构建：将当前负载特征和最优参数增加到知识库，提升后续调优效率

应用场景

A-Tune 在 openEuler 等 Linux 环境里面应用比较广泛，应用场景涵盖大数据、数据库、中间件、高性能计算等场景。助力金融、电信等行业客户实现 MySQL、Redis、宝兰德中间件等应用性能提升 12%~140%。

仓库地址

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

BiSheng JDK 毕昇 JDK

服务器 云计算 边缘计算

Compiler SIG

毕昇 JDK 是基于 OpenJDK 定制的 Huawei JDK 的开源版本，是一款高性能、可用于生产环境的 OpenJDK 发行版，2022 年 05 月 26 日成功登陆了 Eclipse Adoptium 开源 JDK 发行版市场。毕昇 JDK 团队积累了丰富的开发经验，解决了许多实际业务中由原生 OpenJDK 缺陷引起的问题，毕昇 JDK 致力于为 JAVA 开发者提供一款稳定可靠、高性能、易调测的 JDK，也为用户在鲲鹏 AArch64 架构上提供一个更好的选择。

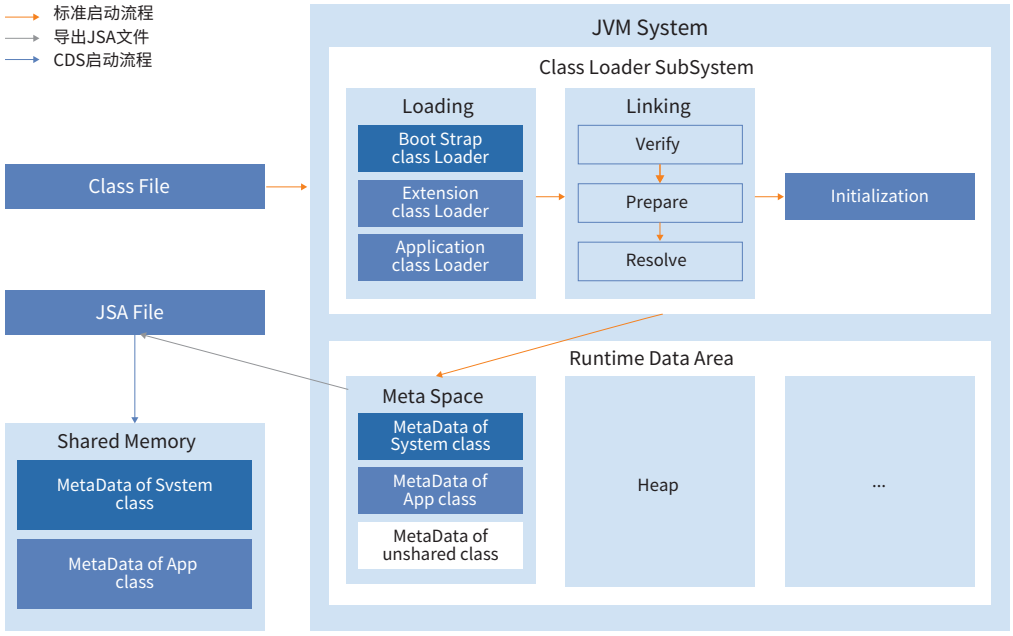
技术挑战

JDK 作为 java 运行的基础软件，性能和稳定性一直是 java 开发者和 java 产业关注的重点；众所周知，OpenJDK8 已进入维护期，引入大的特性比较困难，同时参与 OpenJDK 开发和维护工作的门槛相对较高。针对 java 启动时间慢、GC 吞吐量低、延时高、加解密性能弱及问题交流门槛高等问题，引入了新的优化特性。

项目介绍

功能描述 1：AppCDS 特性

Java 程序运行初始阶段，类的加载是一个比较耗时的过程，且在每次程序运行中均需要执行一遍。而 CDS（Class Data Sharing）技术，就是把类加载后的的数据保存到文件中，下次运行时，直接将加载后的类数据从文件中恢复到内存中，不需要再重新执行类的加载过程，从而提高性能。毕昇 JDK8 在 OpenJDK 提供的 CDS 特性基础上，扩展提供了 AppCDS 特性，增加了对应用类的支持。该特性在 Hive-Sql 场景平均性能提升 7%+；



AppCDS 业务流程图

功能描述 2：G1GC 堆内存伸缩特性

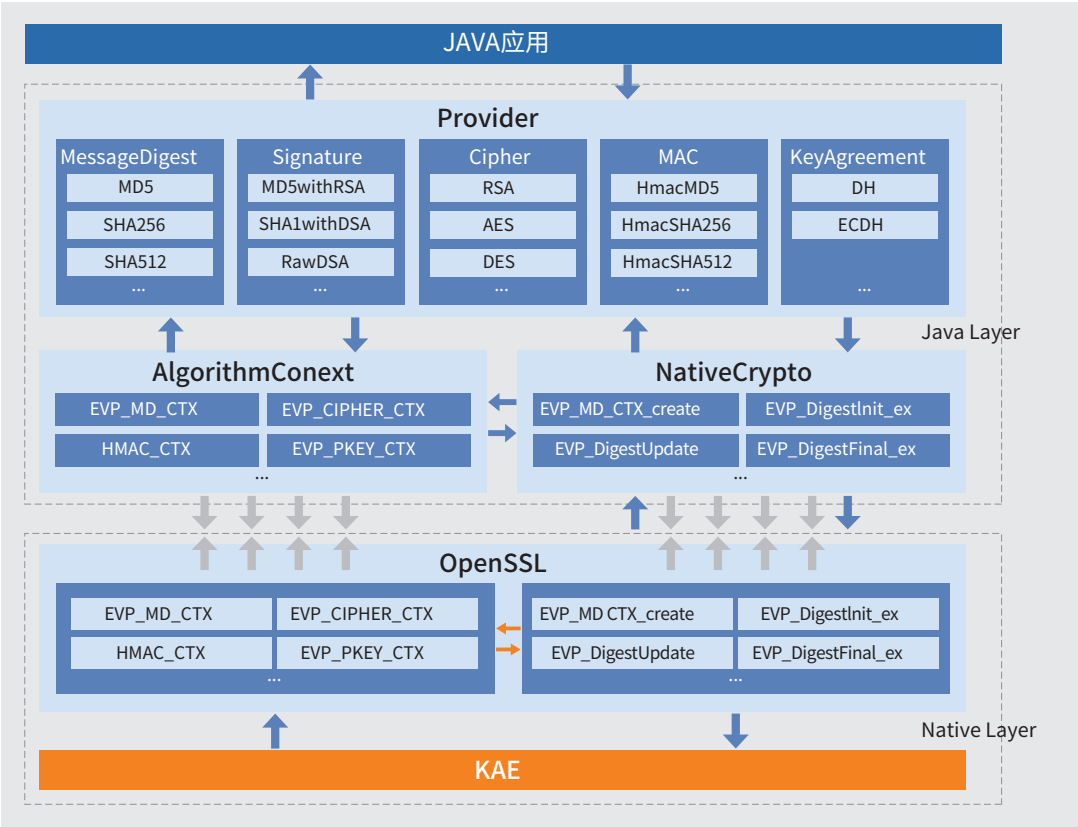
在 OpenJDK 8 中，G1GC 不会及时将空闲的 Java 堆内存释放给操作系统，其仅在 Full GC 时才会把空闲的 Java 堆内存进行释放。由于 G1 尽可能避免触发 Full GC，因此在许多情况下，除非强制从外部执行 Full GC，否则 G1 不会将空闲的 Java 堆内存释放给操作系统。

在按资源使用量付费的容器场景中，G1 不释放空闲的 Java 堆内存返回给操作系统的行为非常糟糕。即使在应用负载下降或不活跃时，G1 也会保留和占用所有 Java 堆。这会导致客户一直在为所有 JVM 占用资源付费，并且云提供商也无法充分利用其硬件资源。如果 JVM 能够检测到应用负载下降和 Java 堆有空闲内存的情况，并自动减少 JVM Java 堆占用情况，那么双方都会受益。在某 49 个微服务场景，开启 G1 堆内存回收特性比默认 G1GC 的实际物理内存减少 40%。

功能描述 3：KAE Provider 特性

KAE 加解密是鲲鹏加速引擎的加解密模块，鲲鹏硬加速模块实现了 RSA/SM3/SM4/DH/MD5/AES 算法，提供了高性能对称加解密、非对称加解密算法能力，兼容 openssl1.1.1a 及其之后版本，支持同步和异步机制。

毕昇 JDK 8 通过利用 Provider 机制，实现对鲲鹏服务器 KAE 加解密特性的支持，以帮助用户提升在鲲鹏 AArch64 服务器加解密业务的竞争力。在 Https 场景，性能提升 1 倍。



KAE Provider 业务架构图

KAE Provider 已支持算法列表：

算法	说明
摘要算法	包括 MD5、SHA256、SHA384、SM3
对称加密算法 AES	支持 ECB、CBC、CTR、GCM 模式
对称加密算法 SM4	包括 ECB、CBC、CTR、OFB 模式
HMac	包括 HmacMD5、HmacSHA1、HmacSHA224、HmacSHA256、HmacSHA384、HmacSHA512
非对称加密算法 RSA	支持 512、1024、2048、3072、4096 位秘钥大小
DH	包括 DHKeyPairGenerator 和 DHKeyAgreement，支持 512、1024、2048、3072、4096 位秘钥
ECDH	包括 ECKeyPairGenerator 和 ECDHKeyAgreement，支持曲线 secp224r1、prime256v1、secp384r1、secp521r1
RSA 签名	包括 RSASignature 和 RSAPSSSignature，私钥只支持 RSAPrivateCrtKey

应用场景

毕昇 JDK 是基于 OpenJDK 开发和发行的 java 基础软件，在 openEuler 等 Linux 环境里面应用比较广泛，应用场景涵盖大数据、中间件、加解密敏感等场景。助力金融、中间件、运营商、互联网等行业客户实现在大数据 spark 性能提升 10%，加解密场景性能提升 100%+。

仓库地址

毕昇 JDK 8、11 和 17 均已开源，且每隔 3 个月进行版本升级和新特性合入，java 开发者可以在毕昇 JDK 开源社区获取最新信息、开展相关交流。

软件产品类型	交付类型	链接
毕昇 JDK 8	开源代码仓	https://gitee.com/openeuler/bishengjdk-8
毕昇 JDK 11	开源代码仓	https://gitee.com/openeuler/bishengjdk-11
毕昇 JDK 17	开源代码仓	https://gitee.com/openeuler/bishengjdk-17/

etmem 内存分级扩展

服务器 云计算

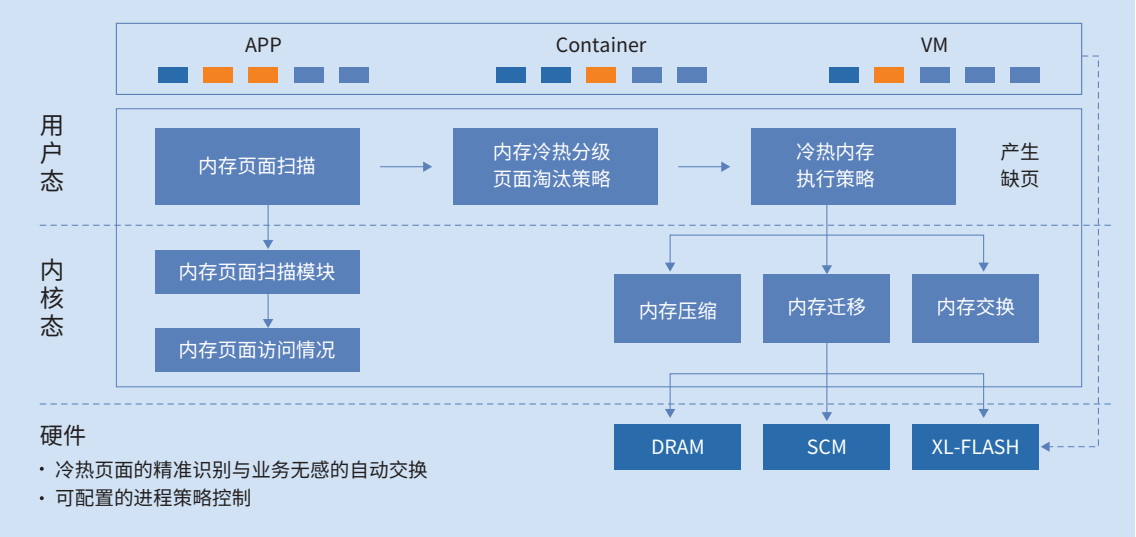
storage SIG

内存分级扩展利用 DRAM 和低速内存（如 SCM）或硬盘等不同的介质组成多级内存供应用进程使用。通过内存的自动调度，让热数据集中在 DRAM 高速内存区中运行，把冷数据放置到慢速介质中，从而达成内存可用量的扩大或业务性能的提升。

技术挑战

当前生态发展让每个 CPU 核的成本越来越低，但是内存制造工艺已经达到瓶颈，短期突破难度大，导致内存成本在整机成本中的占比越来越大。同时数据库、虚拟机、大数据、人工智能、深度学习等场景均需要算力和大内存的支持。节省内存成本和扩大内存容量成为迫切要解决的问题。

项目介绍



如上图所示，etmem 分为内核态和用户态两部分，通过这两个模块的协同，实现了如下功能：

- 进程级控制：etmem 支持通过配置文件来进行内存扩展的进程，相比于操作系统原生的基于 LRU 淘汰的 kswap 机制，更加灵活和精准。
- 冷热分级：用户态触发对指定进程进行内存访问扫描，根据分级策略配置文件，对内存访问结果进行分级，区分出热内存和冷内存。
- 淘汰策略：根据配置文件和系统环境配置，对冷内存进行淘汰，淘汰流程使用内核原生能力，安全可靠，用户无感知。

内核态模块

内核添加的模块为内存页面扫描模块和内存淘汰的压缩迁移交换模块。内存页面扫描模块负责识别页表特征：由用户态周期性触发，统计内存页面的访问情况，将统计结果上报回给用户态；内存淘汰模块实现的流程为：接收用户态传入的内存页面地址（经用户态决策后需淘汰的），复用开源原生能力将这些页面压缩、迁移或交换出去。

用户态模块

- 内存扫描：触发内存页面扫描动作并收集结果；
- 内存冷热分级：通过配置的页面淘汰策略对已统计的结果进行内存的冷热分级；
- 冷热内存执行策略模块：根据配置对冷热内存分别执行相应动作。

应用场景

etmem 针对内存使用较多，且访问相对不频繁的业务软件，扩展效果较好，比如 MySQL、Redis、Nginx 等。实测 MySQL TPCC 场景中，等成本条件下性能提升 40%。

etmem 当前仅支持节点内业务进程的内存分级扩展，不涉及跨节点远端操作。在用户态存储框架的场景中，可通过策略框架的用户态 userswap 功能，使用用户态存储设备作为内存交换设备。

仓库地址

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

EulerFS 新介质文件系统

服务器 云计算

Kernel SIG

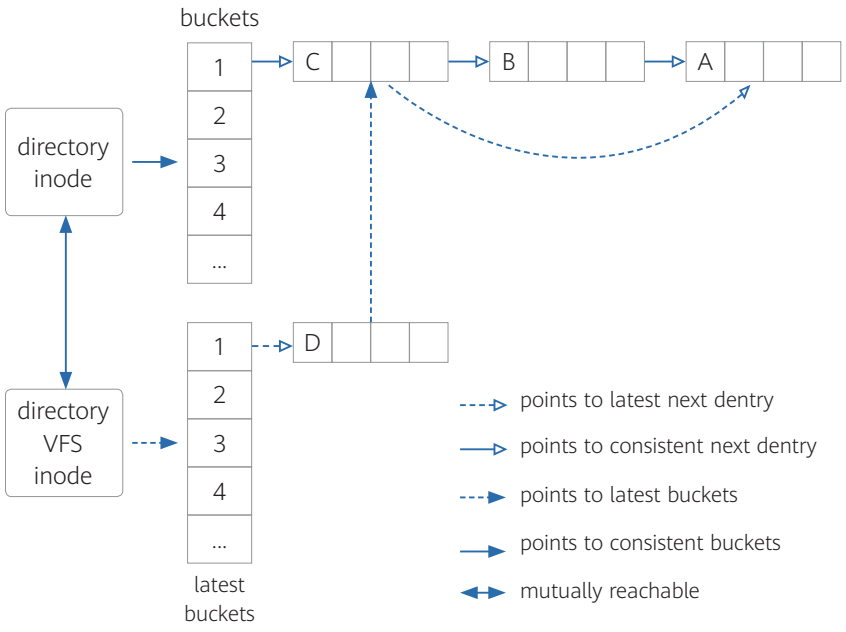
非易失性内存（NVDIMM，比如 Intel Optane）是一种提供字节访问粒度的新型高速存储介质，现有内核文件系统 EXT4，可以协同 DAX 特性改善 NVDIMM 新介质数据读写性能。但在元数据管理方面，基于现有 journal 同步机制，元数据管理开销大，且容易出现写放大问题，NVDIMM 优势无法充分发挥。

EulerFS 创新元数据软更新技术（Soft Update），基于指针的目录双视图计数机制，减少元数。

据同步开销，有效提升文件系统 create、unlink、mkdir、rmdir 系统调用性能，较 EXT4 DAX，元数据操作延时降低 1~4 倍，带宽提高 0.2~4 倍。

功能描述

- 哈希表目录：采用哈希表来管理目录项，提高线性查找效率，减少伪共享。
- 统一的分配器：数据结构使用统一的分配器，这样可以打破不同数据结构之间的界限，使得内存管理更加地简单与灵活。
- 软更新：一种轻量级的保证文件系统一致性的技术，简化了文件系统一致性的实现复杂度。
- 基于指针的目录双视图：一种减少元数据同步开销的机制，有效提升文件系统读写性能。
- 依赖跟踪：目录项的新建、删除等操作并不是立刻持久化的，在进行相应的操作后，只是在 inode 中跟踪依赖，后续通过异步的方式进行持久化，可以大幅提高性能。



应用场景

该技术适用于所有基于 NVDIMM 的新介质，可代替 EXT4，XFS 等文件系统，满足单机应用、云原生分布式应用高性能数据存储诉求。

仓库地址

https://gitee.com/openeuler/eulerfs

Gazelle 轻量级用户态协议栈

服务器 | 云计算 | 边缘计算 | 嵌入式

high-performance-network SIG

Gazelle 是基于 DPDK 和 LWIP 开发的轻量级用户态协议栈，在满足高性能的同时，具备良好的通用性和易用性。

技术挑战

网络协议栈作为现代应用的关键路径，一直是性能研究的热点。近年来，随着软硬件技术的发展，网络协议栈面临以下问题和诉求。

硬件视角：

- CPU 算力和网卡算力差距逐渐增大，单核 CPU 无法充分发挥网卡带宽的发展红利。
- 众核架构下，协议栈设计需避免 NUMA 内存访问陷阱。

软件视角：

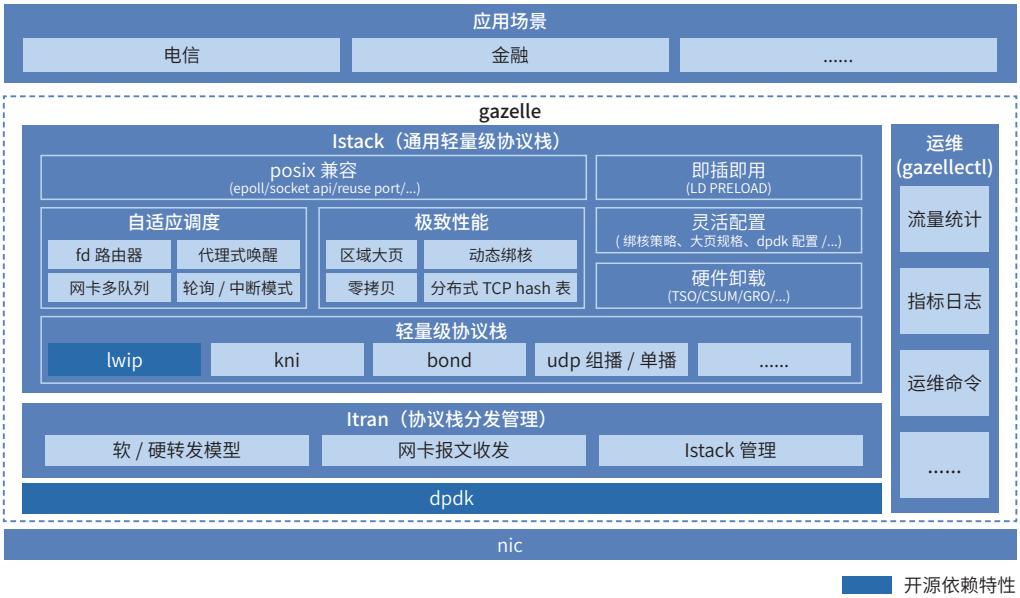
- 现代大型软件性能要求高，往往采用多线程架构充分利用 CPU、网卡等硬件资源，并期望随着线程数的增加软件性能线性增长。
- 应用网络模型多样，对协议栈通用性诉求高。
- 当前内核协议栈具备通用性高、分层解耦的特点，但性能不足；用户态协议栈一般面向特定场景，追求极致性能的同时，往往无法兼顾通用性。

协议栈设计挑战：

数据库等场景，应用网络模型多样，且对网络性能要求高，实现兼顾高性能和通用性的协议栈软件难度大。

项目介绍

Gazelle 的软件架构如下图所示：由 ltran、lstack、gazellectl 组成。ltran 负责协议栈分发管理，lstack 提供通用轻量的协议栈能力，gazellectl 作为运维工具完成流量统计、诊断日志等功能。



Gazelle 关键特性如下：

高性能

- 超轻量：基于 DPDK、LWIP 实现高性能轻量协议栈。
- 极致性能：基于区域大页划分、动态绑核、全路径零拷贝等技术，实现高线性度并发协议栈。
- 硬件加速：支持 TSO、CSUM、GRO 等硬件卸载，打通软硬件垂直加速路径。

通用性

- POSIX 兼容：接口完全兼容 POSIX 接口，应用零修改。
- 通用网络模型：基于 FD 路由器、代理式唤醒等机制实现自适应网络模型调度，满足任意网络应用场景。

易用性

即插即用：基于 LD_PRELOAD 实现业务部署免配套，安装 Gazelle 后协议栈加速效果立刻生效。

易运维

运维工具：提供流量统计、日志、命令行等运维手段。

应用场景

Gazelle 适用于数据库加速等场景。目前已应用于电信、金融等行业，实现 MySQL 事务处理测试性能提升 20% 以上，Redis 吞吐量提升 50% 以上。

仓库地址

https://gitee.com/openeuler/gazelle

GCC for openEuler

服务器 | 云计算 | 边缘计算 | 嵌入式

Compiler SIG

GCC for openEuler 编译器基于开源 GCC（GNU Compiler Collection，GNU 编译器套装）开发。开源 GCC 是一种支持多种编程语言的跨平台开源编译器，采用 GPLv3（GNU General Public License, version 3）协议，是 Linux 系统上目前应用最广泛的 C/C++ 编译器，基本被认为是跨平台编译器的事实标准。而 GCC for openEuler 在继承了开源 GCC 能力的基础上，聚焦于 C、C++、Fortran 语言的优化，增强自动反馈优化、软硬件协同、内存优化、自动向量化等特性，并适配国产硬件平台，如鲲鹏、飞腾、龙芯等，充分释放国产硬件算力。

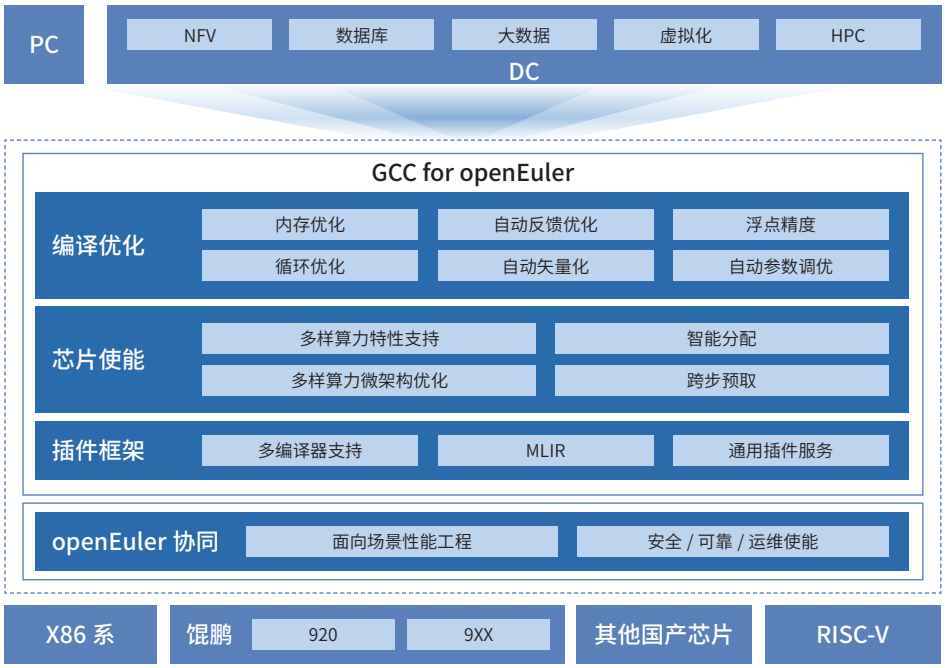
技术挑战

GCC 作为 Linux 内核的默认编译器，跨平台编译器的事实标准，是操作系统中至关重要的基础软件。GCC 的修改往往牵一发而动全身，对上层应用影响甚大，因此 GCC 开发者不仅要熟悉编译原理等基础知识，还要有充足的技术储备，加强特性的安全性、鲁棒性，在增强竞争力的同时，保证 GCC 本身的安全稳定。GCC for openEuler 致力于在开源 GCC 基础上，提供更多元化的竞争力特性，通过编译优化、反馈优化等手段，提升上层软件的性能表现。在 Compiler SIG 双周例会上有 GCC 的固定议题，欢迎各位社区开发者随时与会交流。

项目介绍

GCC for openEuler 支持鲲鹏、x86 等主流硬件平台，支持与 openEuler 性能 / 安全 / 可靠 / 运维工程进行协同，对接编译器插件框架，提供通用化插件功能，支持多样算力特性支持和微架构优化，实现内存智能分配、内存优化、自动矢量化等特性，并通过整合业界领先的反馈优化技术，实现自动反馈优化，提升数据库等场景应用性能。GCC for openEuler 在以下四个方向实现主要突破。

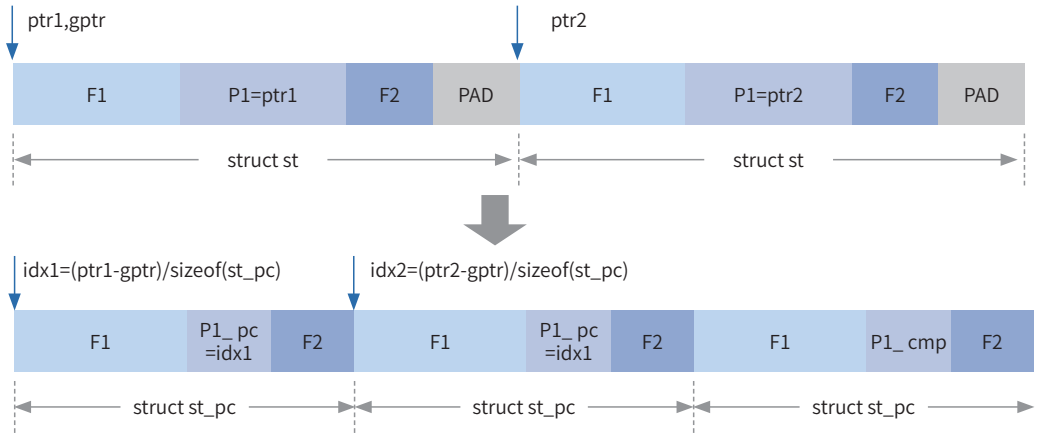
- 基础性能：基于 GCC 开源版本，提升通用场景性能，服务多样算力。
- 反馈优化：整合业界领先的反馈优化技术，实现程序全流程和多模态反馈优化，提升数据库等云原生场景重点应用性能。
- 芯片使能：使能多样算力指令集，围绕内存等硬件系统，发挥算力优势，提升 HPC 等场景化性能。
- 插件框架：使能多样算力差异化编译诉求，一套插件兼容不同编译框架，打通 GCC 和 LLVM 生态。



功能描述 1：指针压缩特性（基础性能）

当结构体域成员出现结构体指针时，8 bytes 的指针大小容易造成结构体成员对齐空隙，而且指针与小于 8 bytes 的基本类型混合使用时容易造成内存 padding，浪费内存空间，造成页表刷新频繁、内存访问延时，程序性能不佳。

结构体指针压缩（以下简称为指针压缩）优化适用于该场景。指针压缩将结构体域成员中的结构体指针由 64 bits 压缩至可选的 8、16 和 32 bits 整形，缩小结构体占用内存大小，降低从内存中读写数据时的带宽压力，从而提升性能。



功能描述 2：全流程反馈优化（反馈优化）

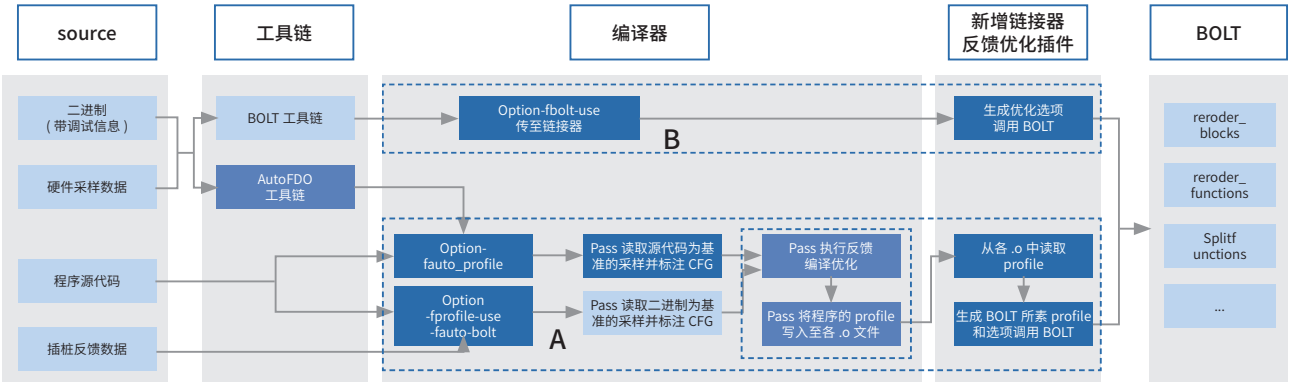
目前业界主流反馈优化技术分为 PGO、AutoFDO 编译阶段反馈优化，Bolt 二进制阶段反馈优化。

PGO 是一种编译器优化技术。通过收集程序运行时信息（profile）进行优化决策。编译器根据这些运行时信息指导各种编译优化技术进行更准确的优化决策，生成目标程序。

AutoFDO 主要是通过采样方式收集程序的运行信息，间接得到程序的执行情况，使用 perf 收集 profile，对程序性能影响较小，实现程序源码与 profile data 解耦，对程序代码变化相对不那么敏感，开发和测试阶段收集的 profile 可用于优化目标程序，复用性较好。

Bolt 通过在编译器在生成二进制时预置重定位信息，在二进制层面上进行 BB 块重排、函数重排、冷热分区等优化，补齐二进制全局的优化机会。

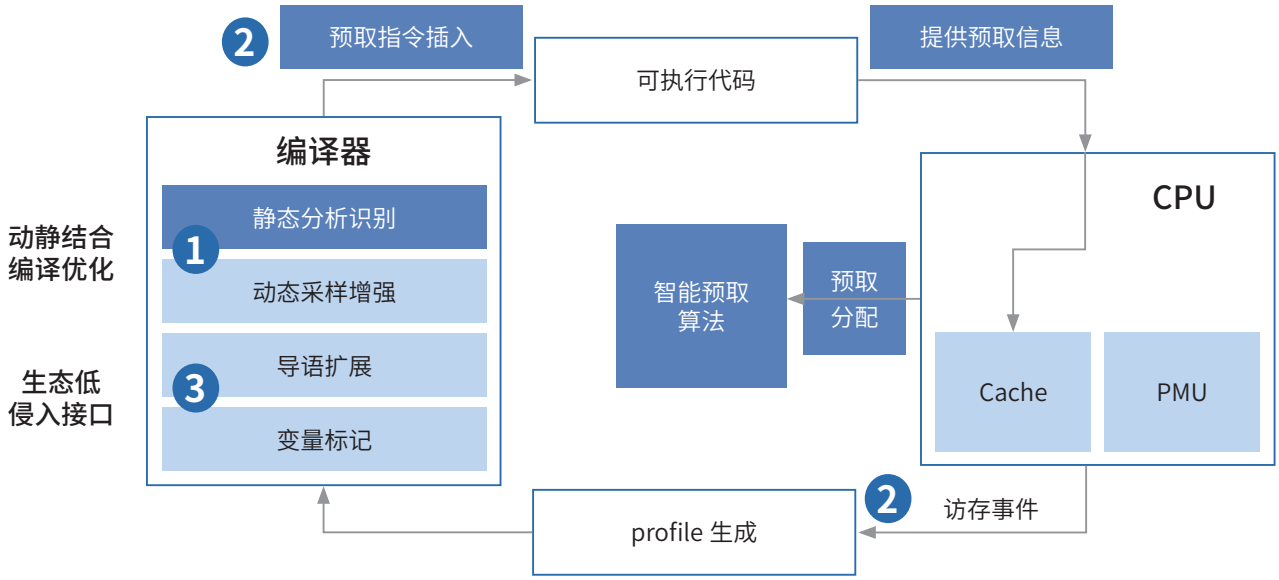
全流程反馈优化，整合反馈数据，打通传统自动反馈优化（AutoFDO）、反馈优化（PGO）与二进制反馈优化（BOLT）流程，极大提高反馈优化的易用性，并通过 MCF 算法修正、discriminator 支持优化，进一步提升优化效果。



功能描述 3：智能分配预取（芯片使能）

当前，智能分配预取支持 HPC 应用静态优化分析，通过访存数据复用分析和插入预取指令，openFOAM、SPMV 和 WRF 三个应用内核函数平均可提升 30%。

- 动静结合编译优化：编译器中增加静态分析和根据动态执行反馈修正的数据复用评分模型，对高并发的热点数据进行分析和识别。
- 内存智能分配预取：通过预取指令生成和插入，显式提供预取信息告知硬件，结合硬件的替换策略提高缓存的利用率和命中率。
- 生态低侵入式编程接口：未来将提供类 CUDA 的变量属性和 OpenMP 导语扩展，编译器自动生成代码，提高开发便利性和兼容性。



应用场景

GCC for openEuler 是基于开源 GCC 开发和发行的 GCC 基础软件，在 openEuler 等 Linux 环境里面应用比较广泛，应用场景涵盖数据库、虚拟化、HPC 等重要场景。实现 Arm 平台下，SPEC2017 基础性能相比开源 GCC 提升 20%，助力运营商、云厂商、安平等行业客户 MySQL 数据库性能提升 15%+。

仓库地址

GCC for openEuler 代码已开源，且随 openEuler 正式版本进行版本升级和新特性合入，GCC 开发者可以在 openEuler 开源社区获取最新信息、开展相关交流。

软件产品	交付类型	链接
GCC for openEuler	代码仓	https://gitee.com/openeuler/gcc
	软件包仓	https://gitee.com/src-openEuler/gcc

HSAK 混合存储加速套件

服务器 云计算

storage SIG

混合存储加速套件 HSAK (Hybrid Storage Acceleration Kit) 用于提升 NVMe 设备的 IO 性能，该软件库实现了高性能的 NVMe 设备的 IO 软件栈，其核心是用户态、异步、无锁、轮询方式。与传统 Linux 内核的 NVMe 设备 IO 软件栈相比，它可以大幅度降低 NVMe command 的延迟，同时提高单 CPU 的 IO 处理能力（IOPS），从而形成一套高性价比的解决方案。

技术挑战

随着 NVMe SSD、SCM 等存储介质性能不断提升，介质层在传统 IO 栈中的时延开销不断缩减，使得软件栈的开销逐渐成为瓶颈和核心痛点。据分析，在使用高性能存储介质时，传统内核 IO 软件栈的开销占 IO 总开销的 60%+，其原因如下：

- 用户态业务进程下发请求到磁盘过程中需要经过多次内存拷贝动作；
- 磁盘处理完成 IO 请求返回的过程中需要通过两次中断处理，导致进程调度、上下文切换等开销；

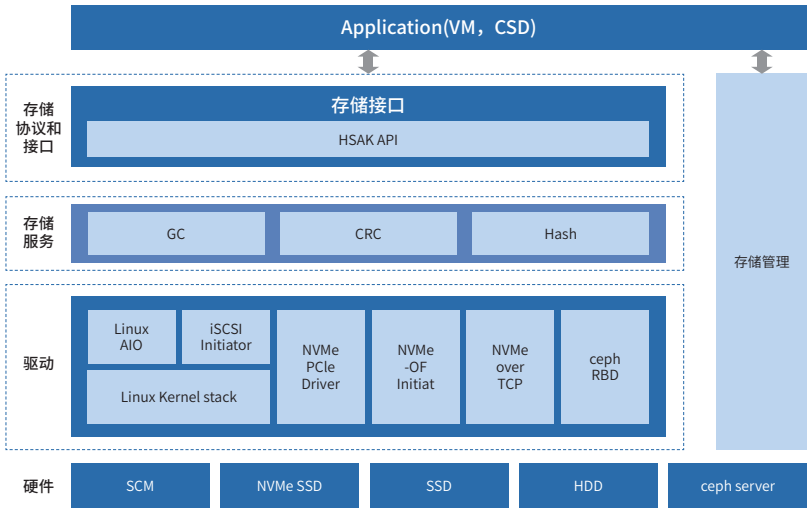
同时业界的各解决方案大多存在下述问题：

- 软件更新频繁，对外接口不稳定；
- IO 数据面功能单一简单，无法使用 NVMe 盘本身提供的一些其他硬件能力；
- 管理面能力不足，针对设备管理或 IO 监控等手段缺失。

项目介绍

如右图所示，HSAK 主要包含数据面的三层结构以及存储管理模块：

- 存储协议和接口层：北向提供稳定统一的存储接口，屏蔽存储协议差别；
- 存储服务层：提供垃圾回收 GC 功能、CRC 校验功能、Hash 等多样化存储介质服务；
- 驱动层：南向对接不同设备或逻辑卷，统一的设备驱动注册接口，对接多样化存储介质；
- 存储管理：提供设备管理、IO 监控、维测工具等功能来管理设备；



应用场景

HSAK 适用于使用 NVMe 盘的分布式存储业务或传统存储业务中，通过用户态 NVMe 驱动接管磁盘，裸盘读写性能相比内核 IO 栈管理磁盘时提升 10 倍，IO 开销降低 50% 以上。

仓库地址

https://gitee.com/openeuler/hsak

iSulad 轻量级容器引擎

服务器 | 云计算 | 边缘计算 | 嵌入式

iSulad

iSulad 是一个由 C/C++ 编写实现的轻量级容器引擎，具有轻、灵、巧、快的特点，不受硬件规格和架构限制，底噪开销更小，可应用的领域更为广泛。

技术挑战

容器是一种创建隔离环境，方便高效打包和分发应用的技术。由于其相比于虚拟化技术具备更高的分发效率，以及更小的运行开销，有效提升了开发和部署的效率，这使得越来越多的用户选择使用容器。随着 Docker 容器引擎、Kubernetes 容器编排调度，以及云原生概念的提出，容器生态越来越完善，容器技术也得以快速推广。

然而，随着容器技术的发展，用户对容器的需求场景越来越多样化。

- 用户对于容器的启动速度和部署速度要求越来越高。
- 用户对容器的资源开销要求越来越高。
- 物联网、边缘计算领域的蓬勃发展对容器技术提出了新要求。

正式基于这种背景，我们提出了 iSulad 容器解决方案，一种更加轻量、快速的容器引擎。

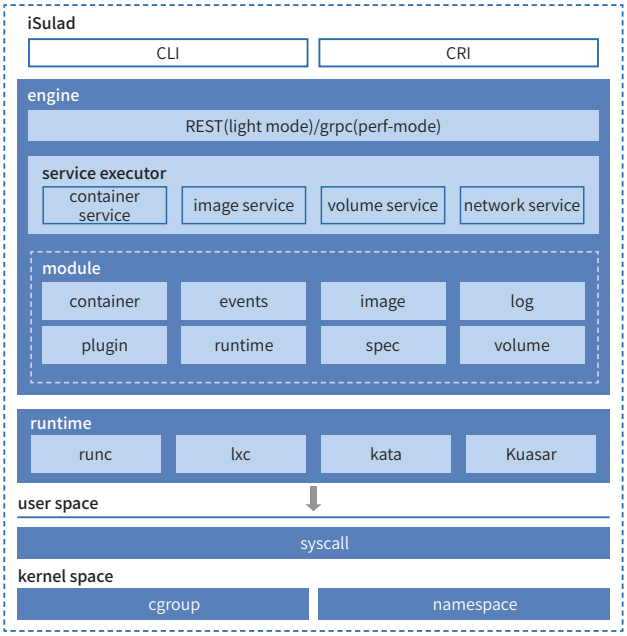
项目介绍

功能描述

iSulad 是 openEuler 提供的新的容器引擎，其统一的架构设计能够满足 CT 和 IT 领域的不同需求。相比 Golang 编写的 Docker，iSulad 资源占用更少，容器启动更快，可应用范围更广。

iSulad 的名字来自于南美的子弹蚁，其个头虽然小，但是力量巨大，被它咬一口，犹如被子弹打到那般疼痛，它是世界上最强大的昆虫之一。iSulad 也是如此，其虽然轻量，能力却不弱，可以为多种场景提供灵活、稳定、安全的底座支撑，与子弹蚁的形象不谋而合。

iSulad 容器引擎提供了与 Docker 类似的命令行，方便用户操作使用。其北向支持 CRI 接口，可以对接 Kubernetes，用户可以使用 iSulad 作为底座，通过 Kubernetes 进行容器的编排调度。iSulad 南向支持 OCI runtime 标准，能够灵活对接 runc、lxc、kata、kuasar 等多种容器运行时，兼容容器生态。



iSulad 软件架构

iSulad 核心能力，包括容器服务，镜像服务、卷服务以及网络服务。容器服务，用来负责容器生命周期的管理。镜像服务，负责提供对容器镜像的操作。iSulad 支持符合 OCI image 标准的镜像格式，保证 iSulad 能够支持业界主流镜像。此外，iSulad 还支持用于系统容器场景的 external rootfs 以及嵌入式场景的 embedded 镜像格式。卷服务，为用户提供容器数据卷管理的能力。网络服务，可以与符合 CNI 标准的网络插件一起，为容器提供网络能力。

iSulad 作为一款通用容器引擎，除了支持运行普通容器之外，还支持运行系统容器与安全容器。

- 普通容器：传统的应用容器
- 系统容器：在普通容器基础上的功能扩展，相比较普通容器，系统容器具备 systemd 管理服务的能力，支持在容器运行时动态添加 / 释放磁盘设备、网卡、路由以及卷。系统容器主要应用在重计算、高性能、大并发的场景下，可以解决重型应用和业务云化的问题。
- 安全容器：安全容器是虚拟化技术和容器技术的结合，相比于普通容器共用同一台宿主机内核存在的安全隐患，安全容器通过虚拟化层实现容器间的强隔离，每个安全容器都有一个自己单独的内核和轻量级虚拟机运行环境，保证同一个宿主机上不同安全容器的运行互相不受影响。

与 Docker 相比，iSulad 不仅在容器启动速度上更快，而且在资源开销方面也更低。这是因为 iSulad 是通过 C/C++ 实现的，相比于其他语言其运行开销更小。其次，iSulad 在代码层面对调用链路进行了优化，相较于 docker 多次 fork 及 exec 调用二进制的方式，iSulad 较少调用次数，直接通过链接库的方式进行函数调用，减少调用长度，从而使得其容器启动速度更快。此外，由于 C 语言是天然的系统级编程语言，使得在嵌入式、边缘测等终端设备上，Golang 实现的 Docker“望洋兴叹”，而 iSulad 却可以 " 大显身手 "。

经过实验测试，iSulad 底噪开销仅为 Docker 的 30%，在 Arm 及 x86 环境下，iSulad 并发启动 100 个容器时间相较于 Docker 提升了 50%+。这使得用户使用 iSulad 进行业务部署时，不仅能够更快的启动业务容器，同时可以减少容器引擎带来的额外资源开销，避免其影响业务的正常性能。

应用场景

作为一款轻量级容器引擎，iSulad 在云计算、CT、嵌入式、边缘侧等场景应用广泛，应用场景覆盖银行、金融、通信、云业务等，助力客户提升容器启动性能 50%+。此外，iSulad+Kuasar+StratoVirt 方案也正在推进，共同构建 openEuler 社区的全栈安全容器解决方案。

仓库地址

https://gitee.com/openeuler/iSulad

Kmesh 高性能服务治理框架

云计算 边缘计算

sig-ebpf

Kmesh 是一种高性能服务网格数据面软件，基于可编程内核，将流量治理逻辑从代理程序下沉到操作系统，实现流量路径多跳变为一跳，大幅提升服务网格下应用访问性能。

技术挑战

随着直播、人工智能等大应用的兴起，数据中心集群规模越来越大，数据规模呈爆炸式增长。如何高效地实现数据中心内微服务间的流量治理一直是大家关心的问题。

服务网格作为下一代微服务技术，将流量治理从服务中剥离出来，下沉到网格基础设施中，很好地实现了应用无感的流量编排；但其代理架构引入了额外的时延底噪开销（例如业界典型软件 istio，单跳服务访问时延增加 2~3ms），无法满足时延敏感应用的 SLA（Service Level Agreement）诉求。

如何实现应用无感的高性能流量治理，是当前面临的技术挑战。

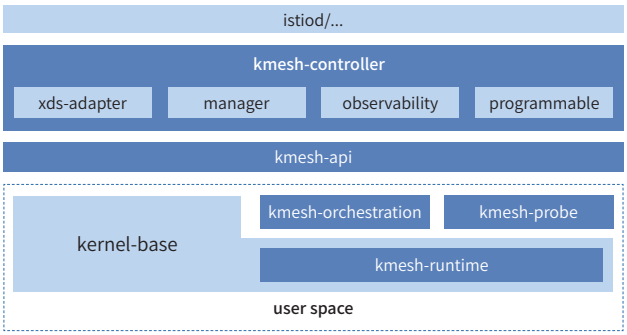
项目介绍

Kmesh 基于可编程内核，将流量治理下沉操作系统，实现高性能服务网格数据面；Kmesh 当前支持的主要特性包括：

1. 支持对接遵从 XDS 协议的网格控制面（如 istio）

2. 流量编排能力

- 负载均衡：支持轮询等负载均衡策略
- 路由：支持 L7 路由规则
- 灰度：支持按百分比灰度方式选择后端服务策略



如上 Kmesh 软件架构图所示，其主要部件包括：

- kmesh-controller：Kmesh 管理程序，负责 Kmesh 生命周期管理、XDS 协议对接、观测运维等。
- kmesh-api：Kmesh 对外提供的 API 接口层，主要包括 XDS 转换后的编排 API、观测运维通道等。
- kmesh-runtime：kernel 中实现的支持 L3~L7 流量编排的运行时。
- kmesh-orchestration：基于 eBPF 实现 L3~L7 流量编排，如路由、灰度、负载均衡等。
- kmesh-probe：观测运维探针，提供端到端观测能力。

应用场景

Kmesh 适用于电子商务、云游戏、在线会议、短视频等时延敏感应用。http 测试场景下对比业界方案（istio）转发性能提升 5 倍。

仓库地址

https://gitee.com/openeuler/Kmesh

LLVM for openEuler

服务器 云计算 边缘计算 嵌入式

Compiler SIG

LLVM 项目是一个开源的编译器基础设施项目，它提供了一套用于编译程序的工具链和库。近年来，LLVM 项目越来越得到开发者的关注，社区非常活跃，商业公司也纷纷基于 LLVM 项目推出商业编译器。LLVM for openEuler 致力于在开源 LLVM 基础上与 openEuler 协同创新，包括兼容性、性能和开发态安全编码特性，为 openEuler 上的编译器提供第二选择，并适配多种硬件平台，如鲲鹏、飞腾、龙芯等，充分释放多样性硬件算力。

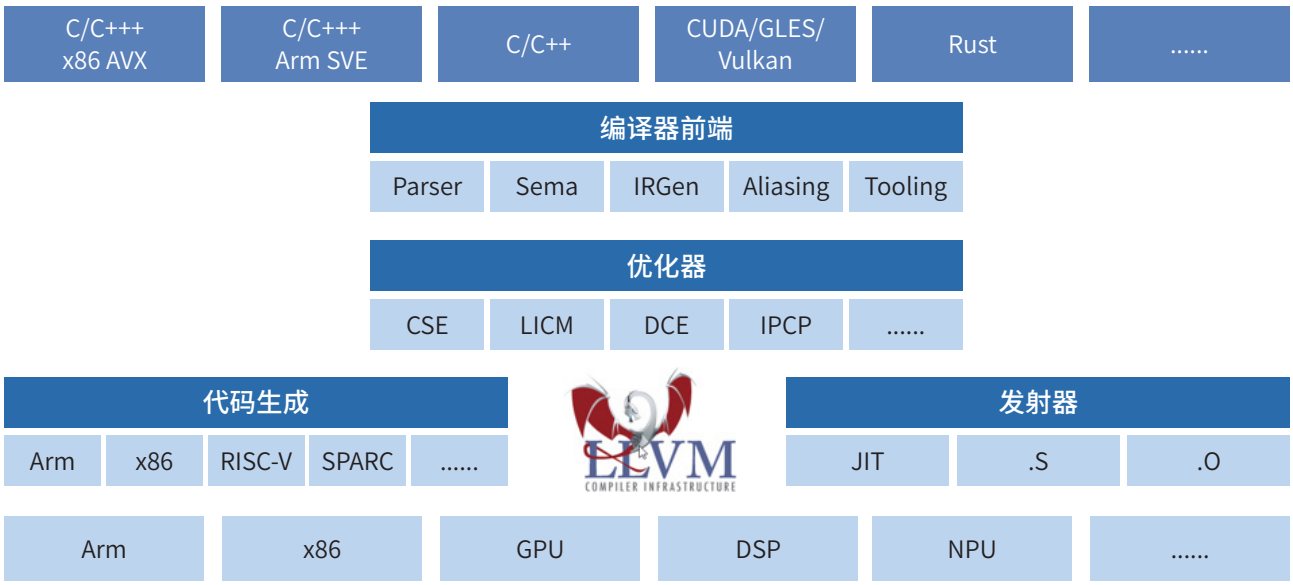
技术挑战

LLVM for openEuler 作为 openEuler 上编译器的第二选择，需要提供比 GCC 更多的竞争力，同时生态系统也需要进一步完善，这样才能通过 openEuler 为最终客户带来价值。从这两点出发，一方面 LLVM for openEuler 需要进一步提供强大而可扩展的优化能力，在计算主力场景如数据库、分布式存储、虚拟化上提供更多性能收益，另一方面需要不断壮大社区和生态，兼容现存软件包，为新开发的软件包提供更好的编译工作和服务。

项目介绍

LLVM 采用了模块化架构设计，将编译过程分为多个独立阶段，如前端、优化和后端。这种设计使得 LLVM 更加灵活和可扩展，有助于各阶段模块分别演进创新，而通过统一的 IR 表示又将不同的模块有机的结合起来。目前 LLVM 项目包含多个子项目，如 clang、flang、llvm、mlir、lld 等。LLVM 9.0 版本之后采取 Apache License，截至目前，LLVM 社区社区贡献者已经达到 2634 人，2022 年增加 340 人，涉及公司 150+，周平均 Commit 数量超 500+，社区比较活跃。

架构描述



模块化解耦架构，统一 IR 表示，助力架构级创新

功能描述 1：Sanitizer

LLVM 的 Sanitizer 是一组用于进行动态代码分析和检测的工具，皆在帮助开发人员发现和调试常见的内存错误和安全问题，这些工具被设计为与 LLVM 编译器和运行时库紧密集成，提供了一种便捷的方式来检测和诊断代码中的问题。

功能	使用方法	探测问题列表
快速内存错误检测	-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
数据竞争检测	-fsanitize=thread	• Data races
内存检测器	-fsanitize=memory	• uninitialized reads • use-after-destruction
未定义行为检测	-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
硬件辅助内存错误检测	-fsanitize=hwaddress	• Same and AddressSanitizer
堆栈缓冲区溢出	-fsanitize=safe-stack	• 保护程序免受基于堆栈缓冲区溢出的攻击

功能描述 2：clang extra tools

Clang Extra Tools 是一组由 LLVM 项目提供的额外工具，用于与 Clang C/C++ 编译器一起使用，皆在提供对代码静态分析、代码重构和代码风检查等功能的支持。

- Clang-Tidy：Clang-Tidy 是一个强大的静态代码分析工具，用于检查 C、C++ 和 Objective-C 代码中的常见错误、潜在问题和代码风格违规。它可以自动检测和修复代码中的问题，帮助开发人员编写更高质量、更规范的代码。
- Clang-Format：Clang-Format 是一个代码格式化工具，用于自动格式化 C、C++ 和 Objective-C 代码。它可以根据配置规则自动调整代码的缩进、换行、空格等，以保持一致的代码风格。Clang-Format 可以帮助团队在代码风格上达成一致，提高代码的可读性和维护性。
- Clang-Check: Clang-Check 是一个用于编写自定义静态分析检查器的工具。它允许开发人员编写自定义的静态分析规则，用于检测代码中的特定问题或潜在错误。Clang-Check 提供了强大的 API 和框架，使开发人员能够根据自己的需求创建定制化的代码检查工具。

功能描述 3：clang+llvm 构建更多软件包 --LLVM 平行宇宙计划

鉴于 LLVM 项目的发展趋势及开发者的需求，是否可以基于 LLVM 技术栈构建 openEuler 版本呢？ LLVM 平行宇宙计划是由 Compiler SIG 和 RISCV SIG 联合发起的，致力于使用 LLVM 项目技术栈构建 openEuler。

收益分析：

基础性能：LLVM 相对 GCC 更容易进行编译优化增强，具备更新的性能潜力；另外 LLVM 具有功能更强大的 LTO 能力。

软件包性能：软件包维护者可以选择 GCC 或 LLVM（性能更好者）作为构建工具链，可以释放更多精力在软件功能实现上。

代码安全：clang+llvm 通常对 C/C++ 语言标准遵从更严格，同时通过静态检查及 Sanitizer 动态检测可能发现软件包潜在缺陷。

应用场景

作为 C/C++/Rust 语言编译器可以用于编译构建服务器、云计算、边缘计算、嵌入式场景应用编译构建。23.03 已发布嵌入式场景下 image，编译时间下降 16%，codesize 下降 1.5%，Coremark 性能提升 6%。另外作为通用编译器，推荐用于通用计算主力场景，如数据库、分布式存储、虚拟化等场景。

仓库地址

https://gitee.com/openeuler/llvm-project 源码仓

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 可编程内核

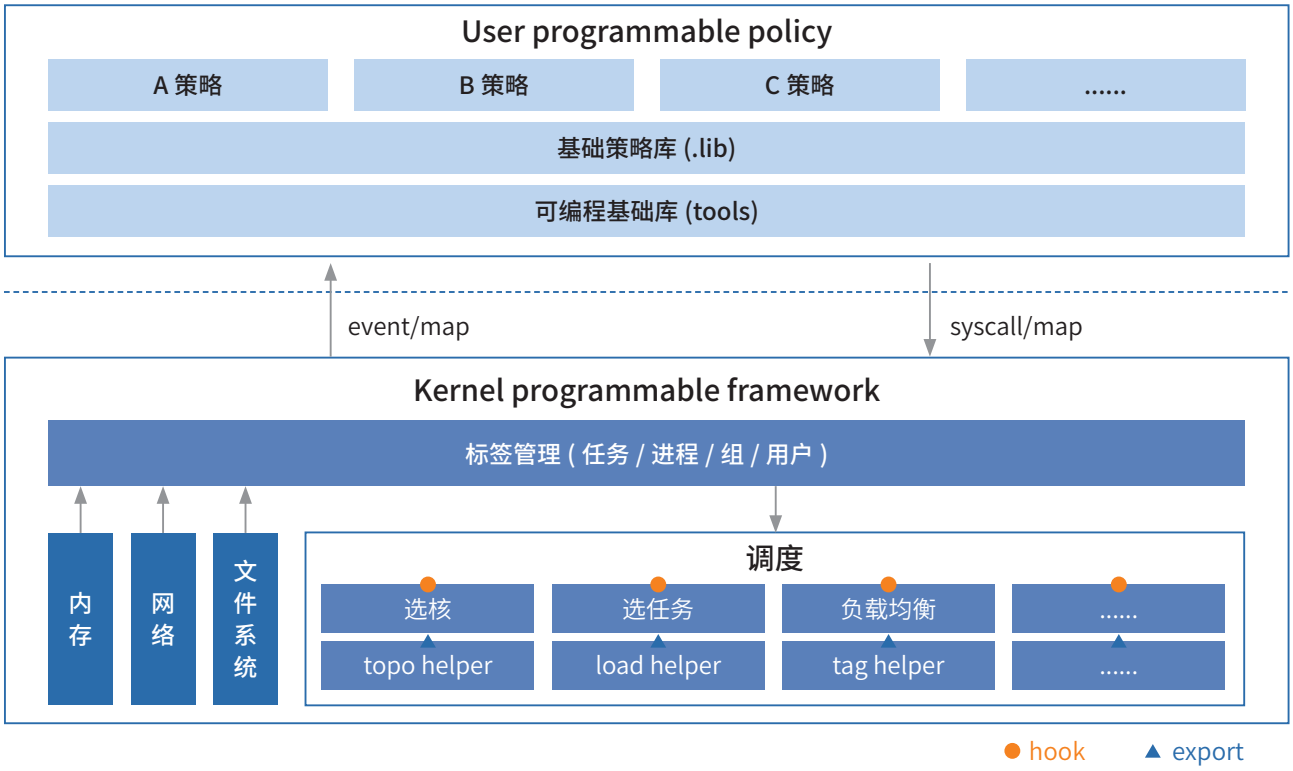
服务器 云计算

ebpf-sig

基于 eBPF 的可编程调度框架，支持内核调度器动态扩展调度策略，以满足不同负载的性能需求，具备以下特点：

- 标签管理机制：开放对任务和任务组进行标签标记的能力，用户和内核子系统可通过接口对特定工作负载进行标记，调度器通过标签可以感知特定工作负载的任务。
- 支持抢占、选核、选任务等功能点的策略扩展：可编程调度框架支持 CFS 调度类抢占，选核，选任务等功能的策略扩展，提供精心设计的扩展点和丰富的辅助方法，帮助用户简单，高效的扩展策略。

功能描述



- 基础库函数与策略库，提供编写用户态策略的基础库函数与可配置的调度策略模板，支持用户快速编排和扩展，对用户编程友好。
- 标签管理机制，支持对任务 / 进程 / 组 / 用户等对象的自定义扩展标签，承载用户态与内核态，内核态组件之间的协同调度语义。
- 调度组件 hook 点与 helper 函数，支持对 CFS 调度类的选核、选任务、抢占流程的自定义策略注入。

应用场景

开发人员、系统管理人员基于可编程内核框架针对不同应用场景，开发自定义策略，动态加载到内核执行。

StratoVirt 轻量虚机运行时

云计算

Virt SIG

StratoVirt 是计算产业中面向云数据中心的企业级虚拟化平台，实现了一套架构统一支持虚拟机、容器、Serverless 三种场景。StratoVirt 在轻量低噪、软硬协同、Rust 语言级安全等方面具备关键技术竞争优势。

技术挑战

随着近几十年 QEMU 虚拟化软件的发展，核心开源组件代码规模越来越大，其中包含大量陈旧的历史代码，同时近年来 CVE 安全漏洞频出，安全性差、代码冗余、效率低问题越来越明显。业界逐步演进出以内存安全语言 Rust 实现的 Rust-VMM 等架构。安全、轻量、高性能的全场景（数据中心、终端、边缘设备）通用的虚拟化技术是未来的趋势。StratoVirt 作为 openEuler 开源平台上实现的下一代虚拟化技术应运而生。

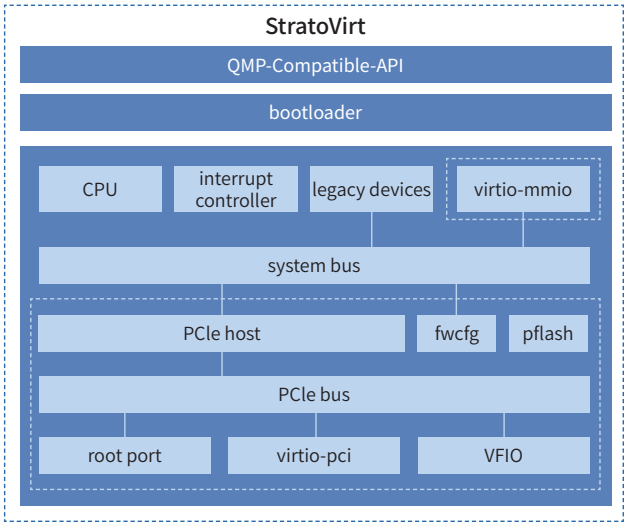
项目介绍

功能描述

StratoVirt 是一种基于 Linux 内核虚拟化（KVM）的开源轻量级虚拟化技术，在保持传统虚拟化的隔离能力和安全能力的同时，降低了内存资源消耗，提高了虚拟机启动速度。StratoVirt 可以应用于微服务或函数计算等 Serverless 场景，保留了相应接口和设计，用于快速导入更多特性，直至支持通用虚拟化。

StratoVirt 的核心架构如下图所示，从上到下分为三层：

- 外部 API: StratoVirt 使用 QMP 协议与外部系统通信，兼容 OCI，同时支持对接 libvirt；
- bootloader: 轻量化场景下使用简单的 bootloader 加载内核镜像，而不像传统的繁琐的 BIOS 和 Grub 引导方式，实现快速启动；通用虚拟化场景下，支持 UEFI 启动；
- 模拟主板 microvm：为了提高性能和减少攻击面，StratoVirt 最小化了用户态设备的模拟。模拟实现了 KVM 仿真设备和半虚拟化设备，如 GIC、串行、RTC 和 virtio-mmio 设备；
- 通用机型：提供 ACPI 表实现 UEFI 启动，支持添加 virtio-pci 以及 VFIO 直通设备等，极大提高虚拟机的 I/O 性能；



StratoVirt 软件架构

应用场景

StratoVirt 配合 iSula 容器引擎和 Kubernetes 编排引擎可形成完整的容器解决方案，支持 Serverless 负载高效运行。

仓库地址

https://gitee.com/openeuler/stratovirt

编译器插件框架

服务器 | 云计算 | 边缘计算 | 嵌入式

Compiler SIG

编译器插件框架（compiler plugin framework）提供面向 MLIR 的插件开发接口，以一次开发、多编译器落地为目标，避免编译工具的重复开发。编译器插件框架作为插件工具开发平台，提供对工具兼容性、完整性校验等公共能力的支持与维护，帮助用户以插件的形式提高优化特性的开发效率。

技术挑战

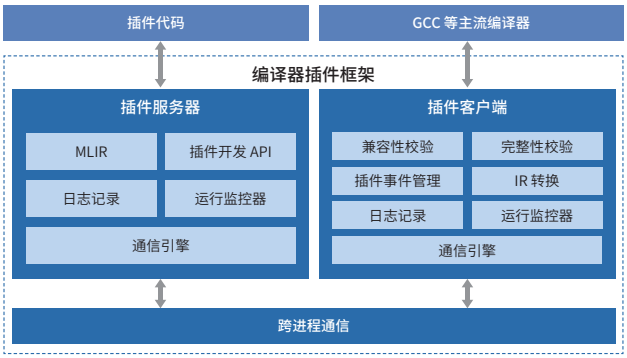
当前存在两款主流的编译器框架为 GCC 和 LLVM。市场上大量的编译工具和编译扩展能力，都是基于这两类编译器完成的。由于任何的编译工具都需要选择两种编译框架之一进行开发，而当其需要使能其他框架时，便会出现重复开发的问题。因为编译器框架的不同，导致即使是相同的工具设计逻辑，其代码也无法复用，需要在不同的编译器框架上做重复的代码开发并分别进行维护，这样直接拉高了工具的开发和维护成本。因此，当前编译工具的开发存在以下几方面的难点：

- 需要深入修改编译器，难度高、难维护。
- 编译工具需要同时使能两种编译框架时，存在重复开发的问题。
- 缺少兼容性等公共能力，拉高工具的开发和维护成本。

项目介绍

功能描述

- 提供基于 MLIR 的插件开发能力，支持与 GIMPLE 等编译器中间表示的转换。
- 插件框架提供对 19 类 GIMPLE 的支持。
- 支持兼容性检测、二进制完整性校验等公共能力。
- 支持安全编译选项校验、操作合法性校验等插件运行监控校验功能。
- 支持插件客户端作为 GCC 插件加载，可以在完全不需要修改 GCC 编译器代码的情况下实现插件功能。
- 支持为链接时优化（LTO，Link Time Optimization）使能插件框架。



应用场景

场景一：编译工具开发者构建工具，并需要完整性校验等公共能力。

有多编译器落地需求的工具开发者，可以使用编译器插件框架作为开发平台，基于 MLIR 进行一次工具开发，即可在 GCC 等主流编译器上使能工具。编译器插件框架统一提供对兼容性检测、二进制完整性校验等公共能力的支持和维护。

场景二：开发者以插件形式快速使能与验证编译相关工具。

编译器插件框架提供以插件的形式运行在 GCC 等主流编译器上。无需对编译器进行源码改动，提高开发效率。

仓库地址

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

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

IMA 内核完整性度量架构

服务器 云计算 边缘计算 嵌入式

security-facility

IMA 是内核中提供对文件完整性保护的一个强制访问控制（MAC）子系统，自 Linux 内核 2.6 版本引入。IMA 摘要列表是在此基础之上增加构建阶段文件度量摘要基线值的生成和保护，启动阶段验证导入度量基线值，在运行过程中对系统重要文件进行完整性保护。

技术挑战

现网运行环境复杂，在运行阶段可能暴露在各种形式的攻击下，攻击者篡改了系统原先携带的可执行文件，或植入未知的恶意程序，可能对系统造成不可预知的损害，IMA 完整性度量特性是对可信启动机制的延伸，在内核态可信的前提下，进一步提供对用户态文件的完整性保护。

IMA 能够基于用户自定义的策略对通过特定系统调用（例如 execve、mmap）访问的文件进行度量，度量结果可被用于两个目的：

- 度量（measure）：检测对系统的意外或恶意修改，支持本地或远程证明。
- 评估（appraise）：度量文件并与预先存储的参考值比较，以保护本地文件完整性。

Linux 内核原生 IMA 特性存在痛点问题如下：

- 部署复杂：原生 IMA 机制通过文件扩展属性存放完整性信息，需要先将系统设置为 fix 模式，为文件预生成并标记扩展属性后重启进入 enforce 模式，才能开启 IMA 校验保护；
- 性能下降：原生 IMA 度量模式下，每次触发文件度量，都会触发 TPM PCR 寄存器扩展，TPM 为低速芯片，扩展流程会导致大量时间消耗；原生 IMA 校验模式下，每次触发文件校验，都需要验证文件扩展属性中存放的签名或 HMAC，验证过程也会产生时间消耗，导致性能下降。

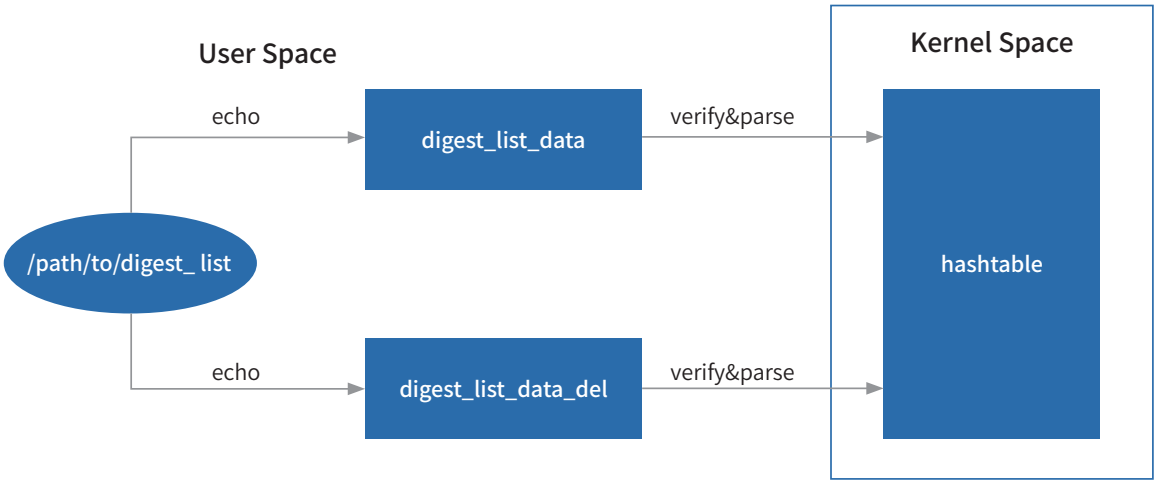
项目介绍

功能描述

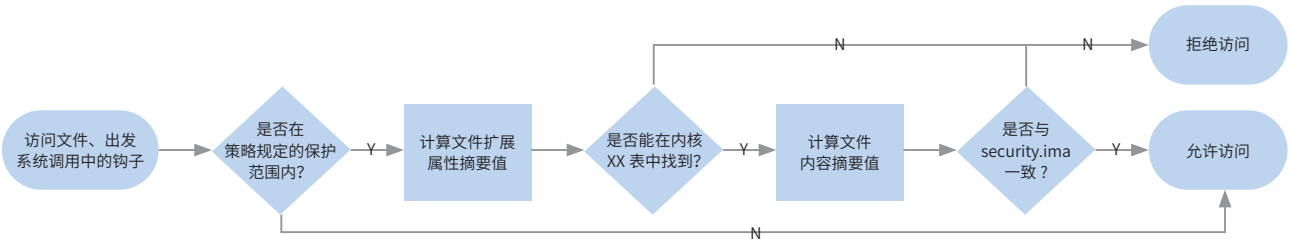
IMA 作为可信计算在 openEuler 中的实现之一，连接了信任链中的可信操作系统和可信应用。 IMA Digest Lists（IMA 摘要列表扩展）是 openEuler 对内核原生完整性保护机制的增强，它取代了原生 IMA 机制为文件完整性提供保护。

“摘要列表”（digest lists）是一种特殊格式的二进制数据文件，它与 rpm 包一一对应，记录了 rpm 包中受保护文件（即可执行文件和动态库文件）的哈希值。

当正确配置启动参数后，内核将维护一个哈希表（对外不可见），并通过 securityfs 对外提供更新哈希表的接口（digest_list_data 和 digest_list_data_del）。摘要列表在构建阶段经过私钥签名，通过接口上传到内核时，需经过内核中的公钥验证。



在开启 IMA 评估的情况下，每当访问一个可执行文件或动态库文件，就会调用内核中的钩子，计算文件内容和扩展属性的哈希值，并在内核哈希表中进行搜索，如果匹配就允许文件的执行，否则就拒绝访问。



应用场景

IMA 摘要列表主要在数据中心、云计算、边缘、嵌入式等场景中为保证系统运行完整性的场景下使用，也是可信计算应用场景里面的关键技术。通过摘要列表构造可信的本地环境，使得可信计算的信任链能够扩展到应用层，同时也支持通过度量日志对接远程证明，即可验证被测试平台加载的文件及系统运行状态是否可信。

仓库地址

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

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

https://gitee.com/openeuler/kernel

kunpengsecl 鲲鹏安全库

服务器 云计算 边缘计算 嵌入式

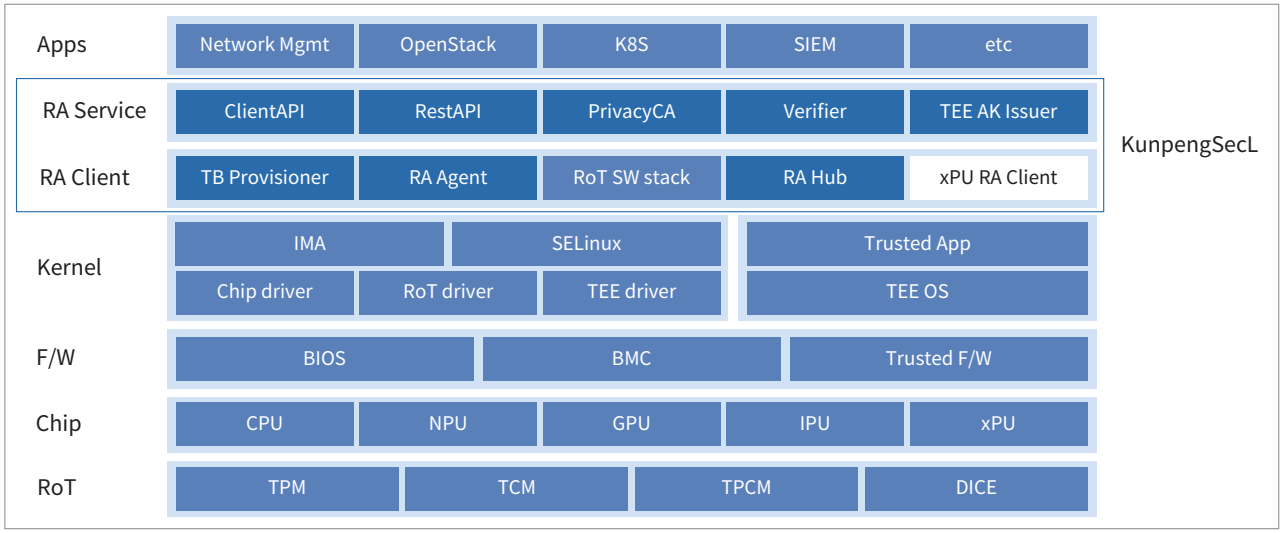
security-facility

鲲鹏安全库开发运行在鲲鹏处理器上的基础安全软件组件，先期主要聚焦在远程证明等可信计算相关领域，使能社区安全开发者。

技术挑战

在可信计算、机密计算和可信 AI 领域的服务器安全技术在不断与时俱进地迅速发展，服务器硬件所提供的安全特性往往与普通软件开发者和用户的需求之间存在易用性的鸿沟，需要通过安全软件中间件的方式来拉近软件开发者与硬件安全特性之间的距离。

项目介绍



鲲鹏安全库的每个特性都可以由两大部分组成：组件和服务。组件部署在提供资源（计算、存储、网络）为用户运行工作负载的工作服务器节点上，将平台安全可信能力转化为软件接口，并将其提供给服务。服务则部署在专门的管理服务器节点上，汇聚来自所有工作服务器节点的安全可信能力，并将其提供给用户及其指定的管理工具以达成用户的对系统安全可信设计的具体要求。

鲲鹏安全库的首个安全特性就是远程证明，目的就是帮助用户获取工作服务器节点的软硬件可信状态，支持端到端的可信计算远程证明解决方案，让各种资源管理工具可以根据可信报告制定策略，对各种服务器资源进行差异化的调度和使用。

鲲鹏安全库的远程证明特性目前支持：

- 基于 TPM 的通用平台远程证明。
- 对鲲鹏服务器 iTrustee TEE 的远程证明。

应用场景

应用场景 1：可信云主机

通过云物理服务器的可信启动与平台远程证明的结合，对虚机运行的主机环境进行可信验证，为云主机用户提供安全可信的底层支持，同时借助虚机 vtpm 的特性完成对虚机可信启动和虚机远程证明的支持，进而使能可信云主机用户对可信云主机自身的安全可信状态进行直接感知，增强用户对云主机安全可信的信心。

应用场景 2：密钥缓存管理

利用平台远程证明，TEE 远程证明以及 TEE 本地证明对可信应用（TA）需要从企业或云基础设施的密钥管理服务（KMS）获取并缓存密钥的场景进行安全加固，让密钥的传输、存贮和使用中的安全性得到更好的保障。

仓库地址

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

secCryto 全栈国密

服务器 云计算

security-facility

openEuler 操作系统国密旨在系统中提供国密算法库、证书、安全传输协议等密码服务支持，并对操作系统用户鉴别、磁盘加密、完整性保护等使用到密码算法的关键安全特性进行国密算法支持。

技术挑战

国产商用密码应用加速，OS 大量开源软件涉及密码算法，但国密算法普遍支持程度低，系统自身安全功能及上层应用无法使用 OS 原生的国密能力来保障业务安全。

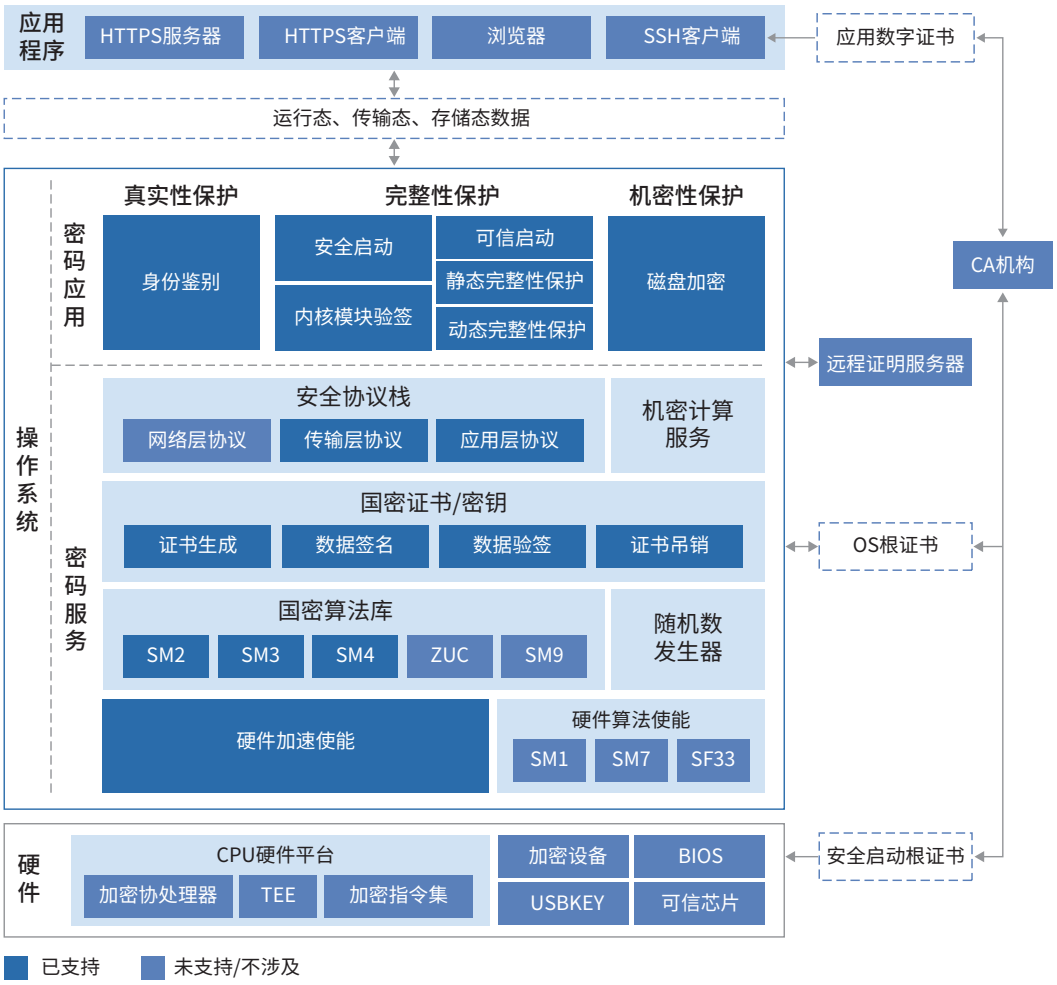
同时当前国密算法性能优化不足，国密算法的应用也带来新的性能损耗，需软硬协同进一步优化。

项目介绍

功能描述

openEuler 当前支持的国密特性包括：

- openSSL/libgcrypt 等用户态算法库支持 SM2/3/4 算法；
- openSSH 支持 SM2/3/4 国密算法套件；
- openSSL 支持国密 TLCP 协议栈；
- 磁盘加密（dm-crypt/cryptsetup）支持 SM3/SM4 算法；
- 用户身份鉴别（pam/libuser/shadow）支持 SM3 口令加密；
- 入侵检测（AIDE）支持 SM3 摘要算法；
- 内核加密框架（crypto）支持 SM2/3/4 算法，以及 AVX/CE/NEON 等指令集优化；
- 内核完整性度量架构（IMA/EVM）支持 SM3 摘要算法和 SM2 证书；
- 内核模块签名 / 验签支持 SM2 证书；
- 内核 KTLS 支持 SM4-CBC 和 SM4-GCM 算法；
- OS 安全启动（shim/grub）支持国密证书签名验签；
- 鲲鹏 KAE 加速引擎支持 SM3/4 算法加速。



OS 国密规划全景图

应用场景

OS 全栈国密主要应用于服务器、云计算等场景，通过在内核及用户态支持 SM2/3/4 算法，自身完成国密改造，作为信息系统底座支撑全行业国密使能，满足密评。

仓库地址

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 机密计算统一开发框架

服务器 云计算

confidential-computing SIG

secGear 是面向计算产业的机密计算安全应用开发套件，屏蔽不同的 TEE（Trusted Execution Environment）SDK 差异提供统一的开发框架，同时提供开发工具、通用安全组件等，帮助安全应用开发者聚焦业务，提升开发效率。

技术挑战

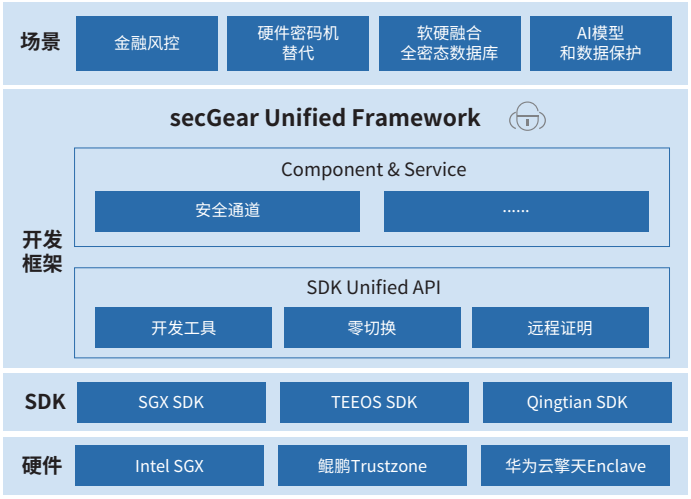
随着机密计算技术的快速发展，各芯片厂商纷纷推出自己的机密计算解决方案，安全应用开发者面临以下几个难点：

- 生态隔离：同一机密计算应用，想要部署到其他平台，需要基于不同平台 SDK 做二次开发。
- 难开发：部分平台仅提供底层接口，使用困难，学习开发成本较高。
- 低性能：机密计算应用需要拆分成 REE-TEE 两部分，频繁切换性能下降明显。
- 生态隔离：同一机密计算应用，想要部署到其他平台，需要基于不同平台 SDK 做二次开发。

项目介绍

secGear 的整体架构如图所示，主要提供三大能力：

- 跨架构：屏蔽不同 SDK 接口差异，提供统一开发接口，实现不同架构共源码。
- 易开发：提供开发工具、通用安全组件等，帮助用户聚焦业务，开发效率显著提升。
- 高性能：提供零切换特性，在 REE-TEE 频繁交互、大数据交互等典型场景下提升 REE-TEE 交互性能 10 倍 +。



secGear 架构图

应用场景

secGear 在数据库、硬件密码机替代、AI 模型和数据保护、大数据等场景应用广泛，助力金融、电信等行业客户业务快速迁移到机密计算环境，保护数据运行时安全。

仓库地址

https://gitee.com/openeuler/secGear

secPaver 应用程序安全策略工具

服务器 云计算

security-facility

secPaver 是一个 SELinux 安全策略开发工具，其核心理念是抽象封装出一组通用的策略描述方法和策略操作接口，在进行策略开发时，开发者无需详细了解安全机制细节，只需使用 secPaver 配置策略描述即可，具体的安全策略由 secPaver 自动生成。

技术挑战

尽管 Linux 提供了 SELinux、AppArmor 等强制访问控制机制使系统更加安全，但是至今为止，它们在实际场景下并未得到广泛的应用。主要的原因还是 SELinux、AppArmor 等安全策略配置复杂：

- SELinux 策略复杂，系统中存在数十万条规则，给单个应用配置安全策略至少需要数百条规则；
- SELinux 以白名单的形式组织策略，一旦策略配置疏漏，应用程序甚至无法正常运行。
- 应用开发者缺乏安全策略定义经验，大部分应用程序没有在开发过程中完成安全策略的定义，而系统管理员无法了解每一个应用程序的运行原理、资源访问等细节，无法详细定义应用程序的安全策略；

因此，需要提供简化的安全策略配置工具帮助开发者及系统管理员针对需要运行的应用快速定义自己的安全策略。

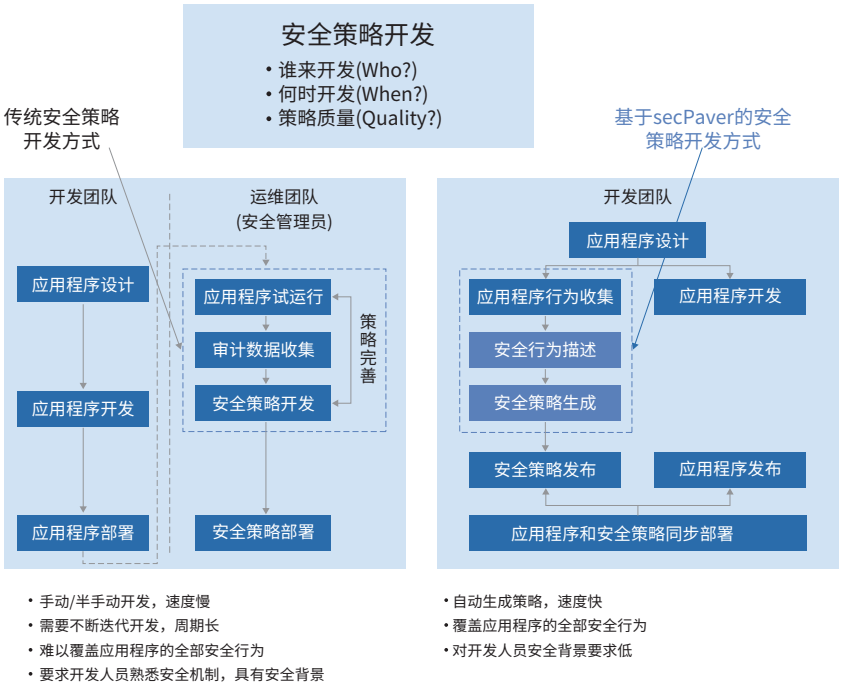
项目介绍

功能描述

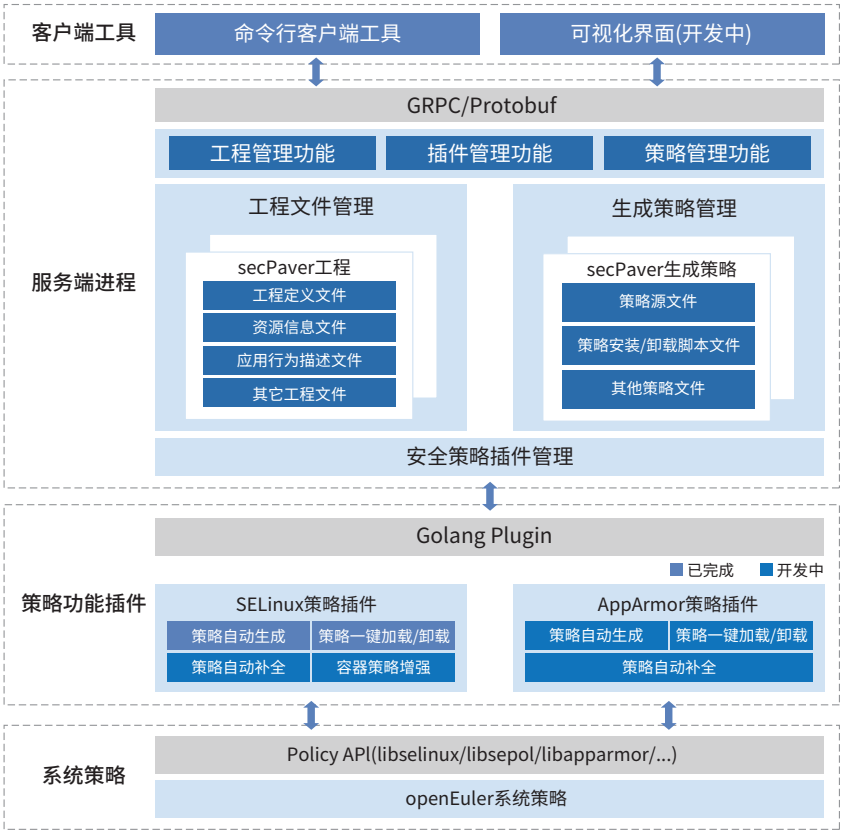
应用程序安全策略是指对一个程序的运行设定安全规则，符合规则的行为允许执行，否则拒绝执行。合理的安全策略可以有效提高系统的安全性和韧性，即使应用程序被攻击，也无法执行超出规则限制的动作。

安全策略的核心是访问控制，即对应用程序对资源的访问增加检查，如文件读写、socket 使用等，并由操作系统判断访问是否需要被拦截。我们提供此工具帮助用户针对应用程序生成对应的安全策略，简化系统管理员对复杂安全策略的管理。

相比传统的安全策略开发方式，使用 secPaver 可以降低开发者对安全机制的知识背景要求，简化策略开发流程，提高策略开发效率，实现软件和安全策略同步发布。

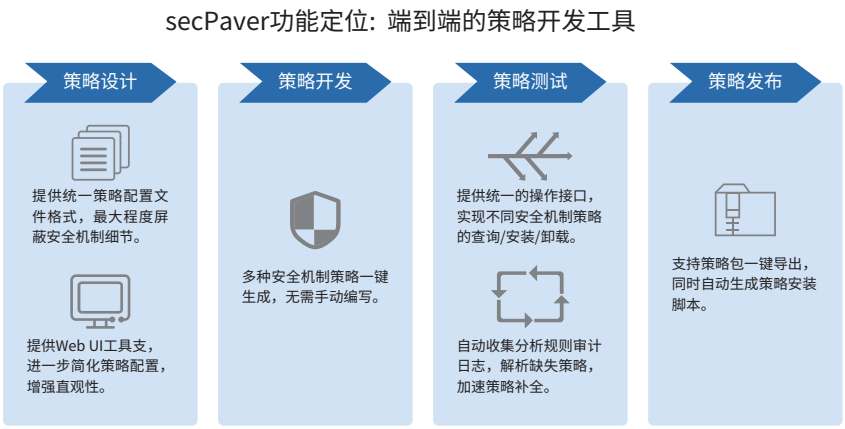


secPaver 使用 go 语言开发，为客户端 / 服务端架构，用户可以通过客户端工具与服务端进程交互，进行安全策略开发工作。服务端为一个 systemd 服务进程，提供策略开发所需要的具体功能。secPaver 采用 go-plugin 机制，对于不同安全机制的开发的差异化功能实现，封装在不同的插件，由服务端进程加载调用。secPaver 当前支持 SELinux 策略开发功能，未来将支持 AppArmor 以及更多的策略开发功能。



应用场景

secPaver 工具帮助用户在应用开发阶段辅助生成对应的安全策略。从安全策略的开发周期来看，secPaver 的功能涵盖策略设计、迭代开发、策略发布等一系列流程。



仓库地址

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

https://gitee.com/openeuler/secpaver

sysMaster 系统管理大师

服务器 | 云计算 | 边缘计算 | 嵌入式

dev-utils SIG

sysMaster 是一套超轻量、高可靠的服务管理程序集合，是对 1 号进程的全新实现，旨在改进传统的 init 守护进程。它使用 Rust 编写，具有故障监测、秒级自愈和快速启动等能力，从而提升操作系统可靠性和业务可用度。

技术挑战

在 Linux 操作系统中，1 号进程（通常为 init 进程）是所有用户态进程的祖先。init 进程是系统启动时第一个被创建的进程，负责启动和管理其他所有进程，并在系统关机时关闭这些程序。在现代 Linux 系统中，init 进程已经被 systemd 进程取代，但是 1 号进程（最小功能包括系统启动和僵尸进程回收）的概念仍然存在。

1 号进程处于系统关键位置，负责系统初始化和运行时服务管理，它面临如下挑战：

- 可靠性差：1 号进程的功能问题带来的影响会被放大，自身故障时，必须重启操作系统才能恢复。
- 复杂性高：systemd 成为 1 号进程事实上的标准。它引入了许多新的概念和工具，诸多扩展组件间依赖关系繁杂，难以针对实际使用场景进行裁剪。

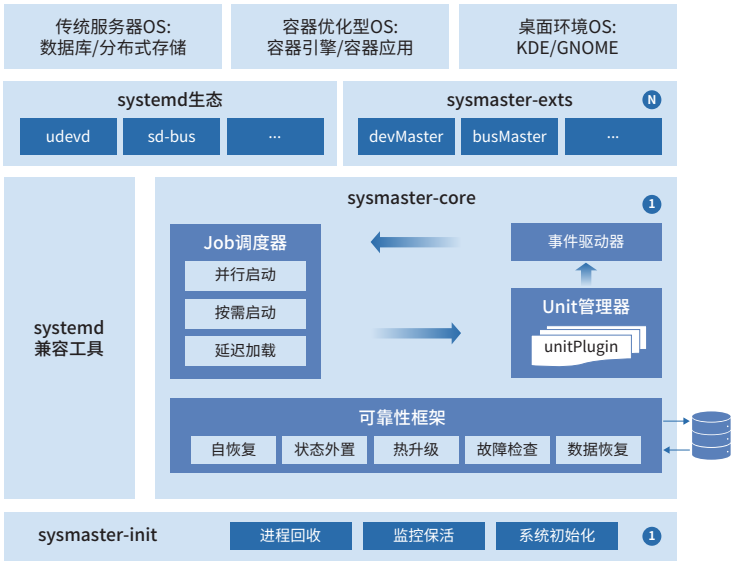
项目介绍

功能描述

sysMaster 支持进程、容器和虚拟机的统一管理，并引入了故障监测和自愈技术，从而解决 Linux 系统初始化和服务管理问题，其适用于服务器、云计算和嵌入式等多个场景。

sysMaster 实现思路是将传统 1 号进程的功能解耦分层，结合使用场景，拆分出 1+1+N 的架构。如下面 sysMaster 系统架构图所示，主要包含三个方面：

- sysmaster-init：新的 1 号进程，功能极简，代码千行，极致可靠，提供系统初始化 / 僵尸进程回收 / 监控保活等功能，可单独应用于嵌入式场景。
- sysmaster-core：承担原有服务管理的核心功能，引入可靠性框架，使其具备崩溃快速自愈、热升级等能力，保障业务全天在线。
- sysmaster-extends：使原本耦合的各组件功能独立，提供系统关键功能的组件集合（如设备管理 devMaster，总线通信 busMaster 等），各组件可单独使用，可根据不同场景灵活选用。



sysMaster 软件架构

sysMaster 组件架构简单，提升了系统整体架构的扩展性和适应性，从而降低开发和维护成本。其主要特点如下：

- 具有自身故障秒级自愈和版本热升级能力。
- 具备快速启动的能力，更快的启动速度和更低的运行底噪。
- 采用插件化机制，支持按需动态加载各种服务类型。
- 提供迁移工具，支持从 Systemd 快速无缝迁移到 sysMaster。
- 结合容器引擎 (iSulad) 和 Qemu，提供统一的容器实例和虚拟化实例的管理接口。

未来，sysMaster 将继续探索在多场景下的应用，并持续优化架构和性能以提高可扩展性和适应性。同时，我们还将开发新的功能和组件以满足容器化、虚拟化、边缘计算等场景的需求。让 sysMaster 成为一个强大的系统管理框架，为用户提供更好的使用体验和更高的效率。

应用场景

sysMaster 可应用于容器、虚拟化、服务器和边缘设备，带来极致可靠和极度轻量的体验。

仓库地址

https://gitee.com/openeuler/sysmaster

A-Ops 智能运维

服务器 云计算

Ops SIG

A-Ops 是一款基于操作系统维度的故障运维平台，提供从数据采集，健康巡检，故障诊断，故障修复的到智能运维解决方案。

技术挑战

云基础设施在近几年随着云原生、无服务化等技术的实施，其运维的复杂性变得越来越有挑战性，尤其是亚健康问题特点（间歇性出现、持续时间短、问题种类多、涉及范围广等）给云基础设施故障诊断带来重要挑战。亚健康故障诊断的挑战（包括可观测能力、海量数据管理能力、AI 算法的泛化能力等）在 Linux 场景中变的尤为突出。在 openEuler 开源操作系统中，现有的运维手段不足以及时发现、定位亚健康问题，存在包括：缺乏在线、持续性监控能力；缺乏应用视角精细化的观测能力；缺乏基于全栈观测数据的自动化、AI 分析能力等问题。然而，针对亚健康故障的诊断能力其难点包括：

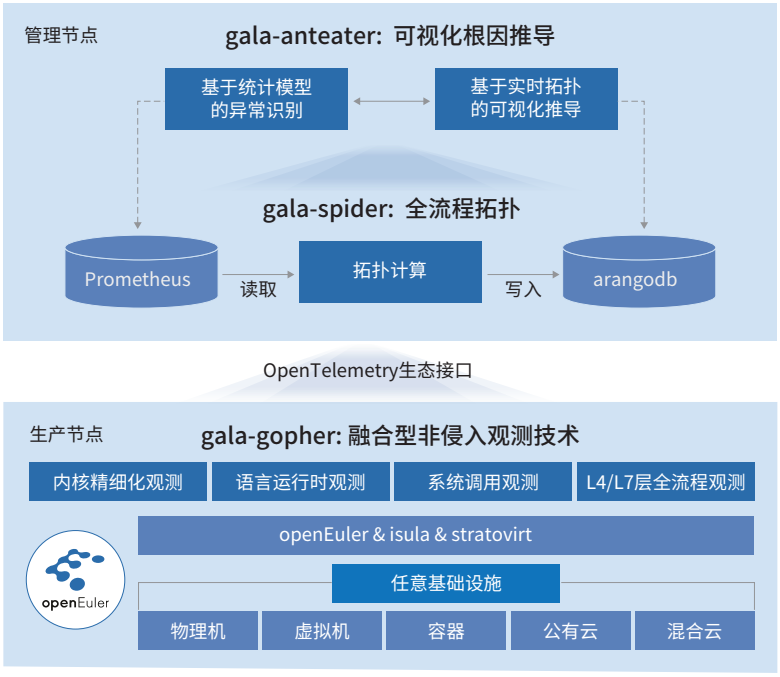
- 全栈的无侵入可观测观测能力。
- 持续、精细化、低负载的监控能力。
- 自适应不同应用场景的异常检测、可视化故障推导能力。
- 业务无感的补丁管理、修复。

项目介绍

功能描述

针对上述问题 A-Ops 项目包括了诺干子项目：覆盖故障发现（gala），故障定位支撑（X-diagnosis），缺陷修复（apollo）等。下面分别介绍下各个子项目：

gala 项目介绍：基于 eBPF + java agent 无侵入观测技术，并以智能化辅助，实现亚健康故障（比如性能抖动、错误率提升、系统卡顿等问题现象）诊断。其架构如图：



功能列表如下：

- 在线应用性能抖动诊断：提供数据库类应用性能在线诊断能力，包括网络类（丢包、重传、时延、TCP 零窗等）问题、I/O 类（磁盘慢盘、I/O 性能下降等）问题，调度类（包括 sysCPU 冲高、死锁等）问题、内存类（OOM、泄漏等）问题等。
- 系统性能诊断：提供通用场景的 TCP、I/O 性能抖动问题诊断能力。

- 系统隐患巡检：提供内核协议栈丢包、虚拟化网络丢包、TCP 异常、I/O 时延异常、系统调用异常、资源泄漏、JVM 异常、应用 RPC 异常（包括 8 种常见协议的错误率、时延等）硬件故障（UCE、磁盘介质错误等）等秒级巡检能力。
- 系统全栈 I/O 观测：提供面向分布式存储场景的 I/O 全栈观测能力，包括 GuestOS 进程级、Block 层的 I/O 观测能力，以及虚拟化层存储前端 I/O 观测能力，分布式存储后端 I/O 观测能力。
- 精细化性能 Profiling：提供多维度（包括系统、进程、容器、Pod 等多个维度）、高精度（10ms 采样周期）的性能（包括 CPU 性能、内存占用、资源占用、系统调用等类型）火焰图、时间线图，可实时在线持续性采集。
- K8S Pod 全栈可观测及诊断：提供 K8S 视角的 Pod 集群业务流实时拓扑能力，Pod 性能观测能力、DNS 观测能力、SQL 观测能力等。

X-diagnosis 项目介绍：X-diagnosis 是 Linux 操作系统运维套件，主要包含问题定位工具集、系统异常巡检、ftrace 增强等功能。

功能列表如下：

- 丰富的问题定位工具集：提供了网络、IO、CPU 调度、文件系统、内存等类型的问题定位工具，例如 ICMP、TCP、UDP 丢包及异常检测、系统文件只读或 IO 慢等场景的定位。
- 丰富的系统问题巡检项：支持网络、CPU 调度、磁盘、服务及配置、系统资源等类型的问题项巡检，巡检结果支持日志和接口方式输出告警。例如 DNS 配置异常、CPU 冲高、磁盘分区空间满、时间跳变、进程数超限制等异常项巡检。
- 提供系统调试和分析工具 eftrace 和 ntrace。eftrace 是 ftrace 工具的增强，支持自动计算结构体偏移量，降低了 ftrace 使用难度；ntrace 支持快速输出协议栈固定函数 kprobe 的关键参数，协助快速定位协议栈流程问题，同时支持 IP、端口、协议类型过滤。

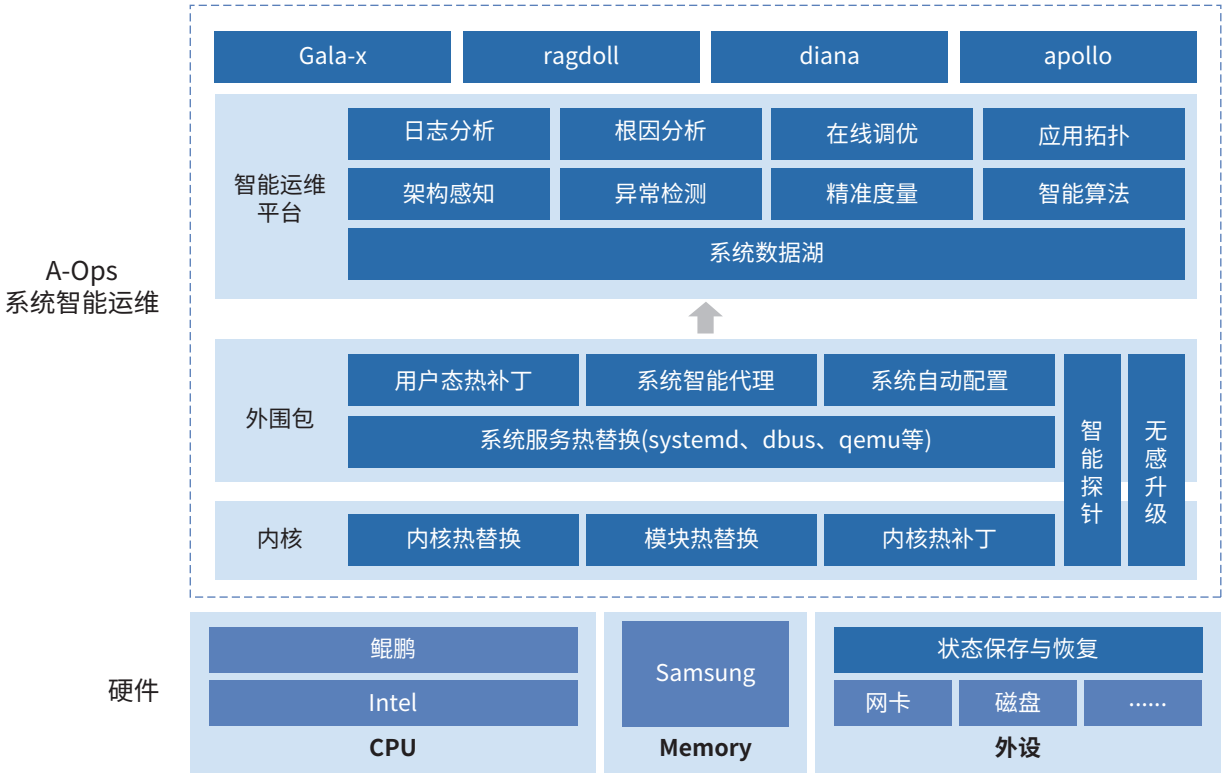
apollo 项目介绍：智能补丁管理框架，提供 CVE/Bug 实时巡检，冷热补丁修复，实现自动发现和零中断修复。

功能列表如下：

- 补丁服务：支持冷热补丁订阅，提供补丁在线获取能力
- 智能补丁巡检：支持基于单机和集群的 CVE/Bug 巡检和通知能力，具备冷热补丁混合管理，一键式修复和回退功能，极大减少补丁管理成本。

ragdoll：配置导致的故障比例占 OS 问题总数达 50% 以上，ragdoll 提供系统配置监控能力，实时发现系统配置变化，快速定位配置错误问题。

- 配置基线：支持按照集群基线已有配置文件，可以通过插件和方式扩展配置文件类型，支持客户自定义的配置文件。
- 配置溯源：支持后台自动巡检系统配置文件变更情况，并通过告警和邮件通知管理员。
- 配置定位：通过 ebpf 监控文件操作快速定位配置文件变更原因（开发中）。



A-ops 软件架构

应用场景

A-Ops 在 openEuler 等 Linux 环境主要面向场景包括数据库、分布式存储、虚拟化、云原生等场景。助力金融、电信、互联网等行业客户提供全栈可观测能力，能实现亚健康故障诊断；集群情况下人为导致的配置错误具备实时检查能力；冷热补丁混合管理能力，避免引入热补丁导致的补丁管理复杂。针对内核高分 CVE 直接提供热补丁，避免修复 kernel 紧急问题而需要重启系统。

仓库地址

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 容器故障检测系统

云计算 边缘计算

sig-CloudNative

容器集群故障检测系统，该软件系统实现了对容器 TOP 故障、亚健康检测的监测与识别。

技术挑战

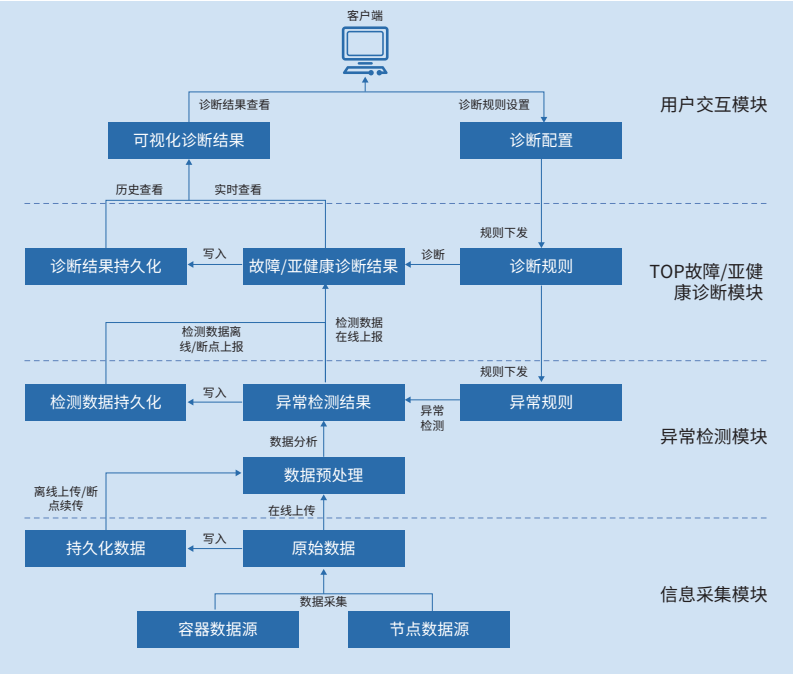
业务规模的增长，容器集群规模不断扩张，IT 运维压力也成比例增大。各种软、硬件故障而造成的业务中断，成为稳定性影响的重要因素之一。目前业内对容器集群故障的检测方案主要基于集群组件状态检测、服务入口监控、自定义接口检活等，具有一定的局限性，难以对服务的亚健康状态进行检测与识别。处理方式也缺乏故障的诊断与执行策略的制定，难以处理一些关键、核心故障。

项目介绍

功能描述

CPDS (Container Problem Detect System) 容器故障检测系统由 4 个组件组组成，整体采用微服务架构，组件之间通过 API 进行通信。

- 信息采集组件 (CPDS-Agent): 负责采集集群各节点的容器和系统原始数据
- 异常检测组件 (cpds-detector): 根据配置的异常规则对各节点原始数据进行分析，检测节点是否存在异常
- 故障 / 亚健康诊断组件 (cpds-analyzer): 根据配置的诊断规则，对异常节点进行健康分析，计算出节点当前健康状态
- 用户交互组件（cpds-dashboard）：提供 web 页面，对集群内节点健康情况进行展示，支持诊断规则配比下发。



采集集群信息

在宿主机上实现节点代理，采用 systemd、initv、ebpf 等技术，对容器关键服务进行监控；对节点网络、内核、磁盘 LVM 等相关信息进行采集；对容器内的应用状态、资源消耗情况、关键系统函数执行情况、io 执行状态等执行异常进行监控。

集群异常检测

采集各节点原始数据，基于异常规则对采集的原始数据进行异常检测，提取关键信息。同时基于异常规则对采集数据进行异常检测，后将检测结果数据和原始据进行在线上传，并同步进行持久化操作。

节点、业务容器故障 / 亚健康诊断

基于异常检测数据，对节点、业务容器进行故障 / 亚健康诊断，将分析检测结果进行持久化存储，并提供 UI 层进行实时、历史的诊断数据查看。

应用场景

CPDS 主要应用在云原生场景的基础设施中，提供容器集群故障检测。

仓库地址

https://gitee.com/openeuler/Cpds

CPM4OSSP 操作系统软件包集中管理平台

服务器

ops SIG

操作系统软件包集中管理平台（Centralized management platform for operating system software package）（以下简称：CPM4OSSP）是由北京凝思软件股份有限公司设计的针对操作系统软件包集中管理平台，该软件实现多个服务器的软件包安装、升级、卸载统一管理的功能。

技术挑战

目前大部分 Linux 操作系统，均采用单个主机执行命令或者使用 ansible 等远程工具从源服务器下载安装软件包的方式，无法实现多个主机软件包、软件源的统一批量管理。针对这种场景，CPM4OSSP 设计了一种多节点集中统一软件包管理解决方案。

项目介绍

CPM4OSSP 主要分为如下几个模块：

- 软件源管理模块：实现软件源模板的生成、查看、下发、编辑、删除等操作，方便运维人员根据软件源模板快速搭建依赖环境。
- 软件包管理模块：实现软件包的安装、卸载、升级、回退等功能，简化运维人员对软件包的维护工作，提升运维效率。
- 审计管理模块：实现对用户操作的全审计功能，方便事后定位问题。
- 用户管理模块：通过用户管理模块，遵循三权分立原则，系统用户、安全用户、审计用户各司其职，增强平台的安全性和稳定性。

应用场景

本项目将为用户提供以下功能：

- 提供易操作的图形化管理界面，支持用户分权管理。
- 提供节点主机软件源的更新，回退功能
- 提供节点软件包更新检测、一键升级、搜索功能
- 提供软件包分类管理功能
- 提供软件包卸载功能
- 提供操作审计功能

仓库地址

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

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

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

CTinspector

服务器 云计算

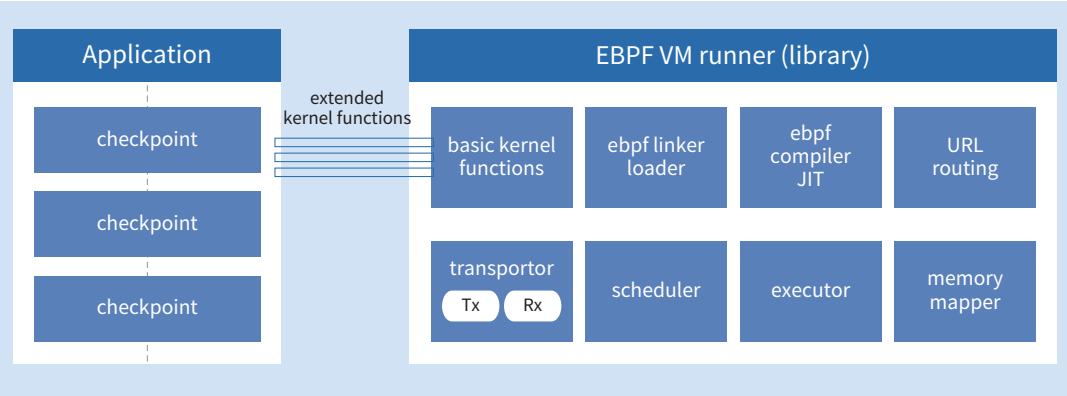
Ops SIG

CTinspector 通过 ebpf vm 技术进行多节点协同带状态的监测与分析网络流量，能够快速定位网络性能瓶颈。

技术挑战

传统性能检测工具无法多节点协同，无法进行有状态检测，无法实时编程应对需求快速变化的困难。

项目介绍



CTinspector 通过 ebpf vm 技术进行多节点协同带状态的监测与分析网络流量，能够快速定位网络性能瓶颈。 基于 ebpf vm 的 ACL 将传统 iptables 命令行配置替换为 ebpf 程序，简化 ACL 的功能开发、提高下发速度。基于 ebpf VM 的 RPC 能够节省网络带宽和缩小时延，以运维人员编好的脚步自适应的诊断网络问题 。

CTinspector 的服务端框架分为内核函数、加载器、编译器、路由器、转发器、调度器、执行器、内存映射几大模块。当外部任务下发时，脚本内容、上下文环境、运算结果等会被全部封装在 ebpf VM 中，从而可以实现非交互式链式传播，大大提高执行和结果运算的效率。

应用场景

运维人员在网络故障时经常需要 dump OVS 的流表查看问题在哪，但是流表太多不方便查看，另外 dump 大量的流表也耗时甚多，这就需要对流表进行过滤。传统的工具很难在线上临时新增过滤字段，也难以做有状态的过滤，比如查看转发报文最多的 top 3 flow。使用 CTinspector 运维人员可以在 dump 流表时指定一个 ebpf 程序，将每条流传入 ebpf 程序由其决定是否过滤，在其内部状态数据中记录报文统计数从而选出 top 3 flow。

仓库地址

https://gitee.com/openeuler/CTinspector

eggo K8s 集群部署解决方案

服务器 云计算

sig-CloudNative

eggo 项目旨在解决大规模生产环境 K8S 集群自动化部署问题、部署流程跟踪以及提供高度的灵活性。

技术挑战

集群部署一直是K8S集群使用的痛点，业界也给出了多种解决方案。但是openEuler云原生SIG组对集群部署存在很多诉求，而业界方案无法满足。主要挑战如下：

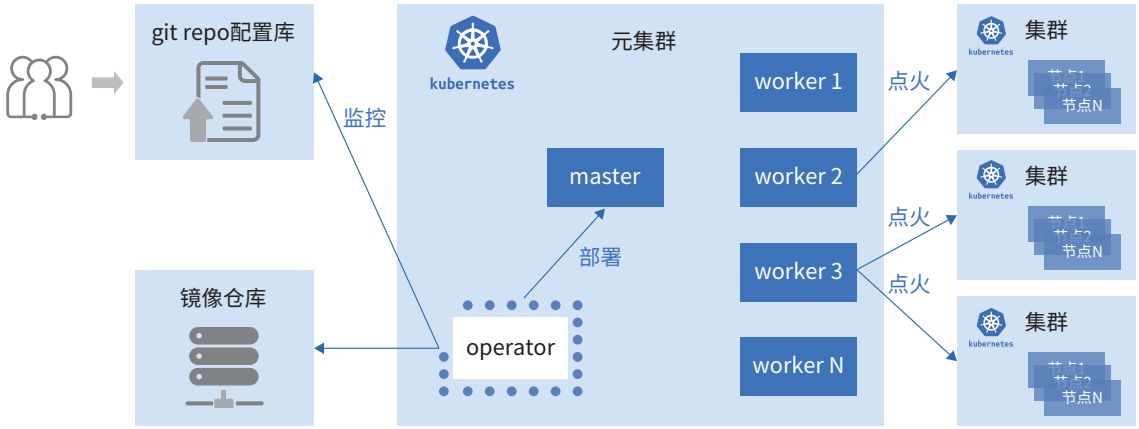
- 多种部署方式；
- 在线和离线部署模式；
- 集群内节点支持多种架构；
- 可跟踪、可溯源的集群配置管理。

项目介绍

功能描述

eggo 可以作为独立组件运行为个人提供集群部署和管理的能力；也可以与 GitOps 结合通过仓库管理集群部署配置以实现云原生的集群管理部署方式。eggo 当前支持的能力如下：

- 支持在多种常见的 Linux 发行版本上部署 k8s 集群：例如 openEuler/CentOS/Ubuntu；
- 支持多架构节点部署，一个集群支持多种架构 (AMD64/Arm64 等) 的节点；
- 支持 K8S 组件二进制部署方式；
- 支持在线部署、离线部署模式。



部署任务的调度，是由目标集群和元集群中worker节点的网络亲和性决定的。

关键组件说明：

- git repo 配置库：git 代码仓库，用于存放集群的部署配置信息，仓库配置信息变更云集群可以通过注册 webhook 感知，从而触发集群管理相关的操作；
- 镜像仓库：集群使用的容器镜像仓库；
- 元集群：部署 eggo 管理程序的 K8S 集群，感知用户设置的集群配置仓库变化、以及集群生命周期管理等；
- operator：元集群的 CRD，负责感知集群配置、以及负载集群的生命周期管理；
- worker 节点：元集群的负载节点，用于执行目标集群部署任务（可以根据目标集群的网络亲和性选择节点）；
- 负载集群：eggo 管理的集群，用于用户业务运行，根据用户需要随时可通过 eggo 进行管理（例如删除、创建、更新节点等）；

应用场景

大规模生产环境自动化部署 K8S 集群的场景，且能跟踪集群的生命周期、变更等。

仓库地址

https://gitee.com/openeuler/eggo

nvwa 内核热升级

服务器 云计算 边缘计算 嵌入式

ops SIG

内核热升级是指在业务服务不中断的情况下，升级系统内核，从而修复内核漏洞，保障系统稳定运行。

技术挑战

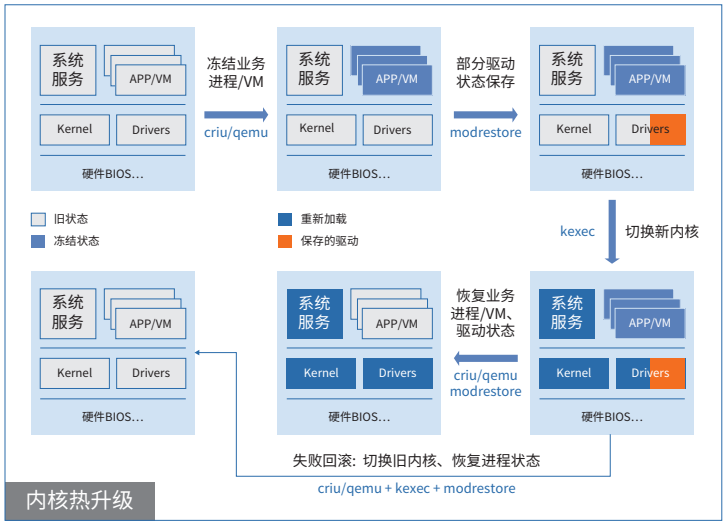
Linux 内核日趋复杂，新的 CVE 漏洞陆续被发现，而 60% 以上的漏洞无法通过热补丁修复，只能升级内核解决。修复内核漏洞面临以下困难：

- 大部分业务场景要保障业务连续，不允许系统重启
- 业务迁移的时间和空间成本无法接受
- 大量业务无法迁移

项目介绍

nvwa 项目提供了一条简洁的 nvwa update 命令，进行内核的热升级操作。

如技术原理图所示，nvwa 通过进程 Checkpoint/Restore 技术保持业务不中断，通过 kexec 及增强技术完成内核快速切换，通过一系列驱动、内核状态保存机制，保障系统恢复后业务可以恢复运行。其关键技术点如下：



- 进程 / 虚拟机冻结、恢复技术：增强 criu、改进 QEMU，将进程或虚拟机的完整状态冻结在内存中，内核升级后进行进程 / 虚拟机的恢复，用户态无感知。
- 内核快速重启技术：在 kexec 基础上增加大量优化，使得内核在 500ms 内完成重启。
- 硬件状态保持技术：硬件、BIOS 等不进行操作，保持状态不变，内核升级过程中 DMA 可以继续运行。
- 内核模块状态保持技术：提供 module_suspend/module_resume 接口，允许必要的驱动在内核升级前后进行状态保存和恢复。
- pin 内存技术：允许用户态进程、内核态模块将所使用内存冻结，系统升级后这部分内存恢复映射，可继续使用。

应用场景

nvwa 适用于 x86 和 Arm64 架构。在虚拟机运行常见服务（例如 MySQL、Redis、NGINX）场景可直接使用。在物理服务器场景，如有必要可使用框架进行驱动适配。

仓库地址

https://gitee.com/openeuler/nvwa

PilotGo 运维管理平台

服务器

Ops SIG

PilotGo 运维管理平台是一个在 openEuler 社区孵化的插件式的运维管理软件，提供服务器集群全生命周期的管理能力，提供插件式的功能扩展，致力于打破不同运维管理工具之间的障碍，实现完整的自动化运维管理流程。

技术挑战

运维技术覆盖的业务场景非常广泛，包括安全、日志采集、指标监控、故障定位、性能优化、配置管理、自动化流程等各个方面。特定的业务运维场景均有独立的运维平台，功能完整且相对复杂，学习成本较高。除此之外，各个业务场景之间往往没有相关性，在跨场景的运维操作当中，需要切换不同的运维工具，难以实现有效的全流程自动化。然而不同运维平台的功能不尽相同，且不一定提供方便的 API 或 CLI 接口，难以进行有效的封装。如何打通不同的运维工具，是一个非常困难的问题。

项目介绍

功能：

PilotGo 是一个插件化的运维管理平台，平台核心仅提供有限的必要的功能；业务场景相关的功能均由平台插件来提供。

核心功能：

PilotGo 的核心功能用于对平台的基本能力进行支撑，并对插件系统提供相关服务。具体包含如下：

- 用户管理：提供账号信息管理功能
- 主机管理：提供主机集群管理功能
- 权限管理：提供权限管理功能
- 审计日志：提供审计功能
- 批次管理：提供批量任务管理功能
- 配置管理：提供系统及软件配置管理功能
- 远程命令执行：提供远程控制功能
- 插件管理：提供插件管理及插件扩展功能

大部分核心功能均通过接口暴露给插件，以便插件实现各类业务逻辑。除此之外，PilotGo 平台还提供全局的事件通知机制，提升整个集群的感知能力。

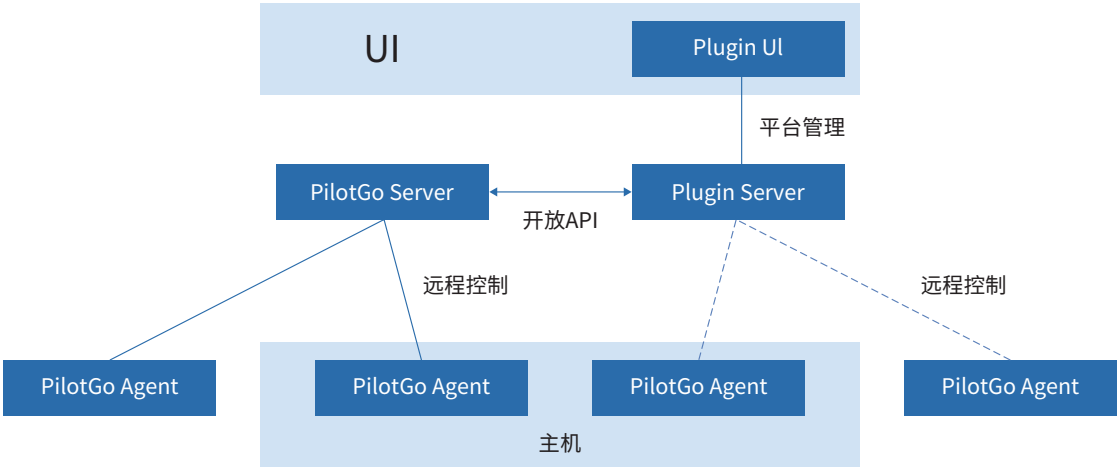
扩展功能

PilotGo 运维管理平台通过插件的方式扩展业务功能，这类扩展是用户最常使用的功能。 PilotGo 运维管理平台当前支持的扩展功能插件包括：

- prometheus：提供监控指标采集、展示及告警功能
- grafana 插件：提供美观的指标可视化功能
- gala 插件：提供 gala 智能运维软件的功能支持，如高级指标采集及展示、故障诊断、平台拓扑等

PilotGo 插件自身也可暴露功能接口，以便不同业务插件之间进行联动，提升整个平台的自动化能力。

架构



PilotGo 分为以下几个独立的组件：

- PilotGo server：PilotGo 平台核心逻辑组件
- PilotGo UI：提供微服务架构的前端框架，可扩展插件 UI 页面
- PilotGo agent：提供 PilotGo 平台的远程能力
- 插件 server：提供 PilotGo 平台扩展业务能力
- 插件 UI：提供插件业务功能相关的 UI 界面，嵌入到 PilotGo UI 当中
- 插件 agent（可选）：与插件 server 配合实现相关扩展业务能力

一般情况下，PilotGo server 和插件 server 各自部署在单独的服务器当中，通过网络协议进行通讯；PilotGo agent 和插件 Agent 部署在业务主机当中，仅与对应的 server 进行网络通讯。

应用场景

PilotGo 可用于裸金属、虚拟机、容器等各类 OS 及应用运行环境的监控及管理能力，以及容器集群、mysql 集群、nginx 集群等典型的应用场景的集群监控、运维能力。

仓库地址

主项目仓库： https://gitee.com/openeuler/PilotGo

插件项目仓库： https://gitee.com/openeuler/PilotGo-plugins

SysCare 系统热服务

服务器 | 云计算 | 边缘计算 | 嵌入式

ops SIG

SysCare 是一款操作系统运维工具，能够在线解决系统运行过程中的各类故障和风险，为 Linux 操作系统提供全方位热修复服务。

技术挑战

在 Linux 世界，有一个困扰大家已久的难题：如何在不影响业务的情况下，快速可靠地修复漏洞、解决故障。

当前常见的方法是采用热补丁技术：在业务运行过程中，对问题组件直接进行代码级修复，业务无感知。然而，当前热补丁制作方式复杂，补丁需要代码级匹配，且管理困难，特别是用户态组件面临文件形式、编程语言、编译方式、运行方式的多样性问题，当前缺少简便统一的补丁机制。

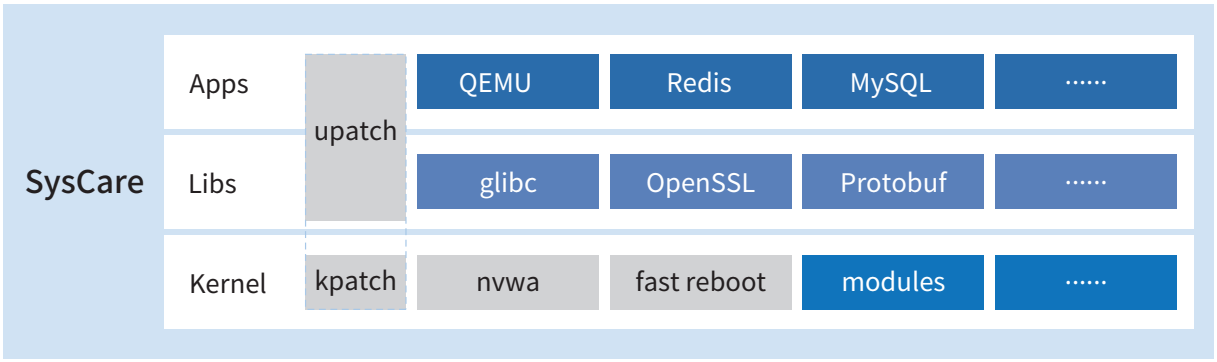
故障通过热补丁修复之后，系统仍需升级存在漏洞的软件，以便业务重启或系统重启后直接使用新版本软件。当前，软件升级和热补丁是两套独立的机制，互相不感知，升级操作复杂。

热补丁由于机制限制，只能修复约 40% 的问题，剩余部分问题只能采用热迁移结合冷升级的方式。冷升级方式，系统重启速度慢，会影响业务。因此，亟需一种快速且安全的系统重启方式。

为了解决以上问题，SysCare 应运而生。

项目介绍

SysCare 提供系统热服务解决方案，当前已经提供统一热补丁、快速重启等能力，包含应用热补丁（支持 C/C++ 常见应用，如 QEMU, Redis, MySQL 等）、动态库热补丁（支持 C/C++ 常用系统库，如 glibc, OpenSSL, Protobuf 等）、内核热补丁以及系统快速重启能力，集成了内核热升级能力（nvwa 项目），如下图所示：



SysCare 提供了如下关键功能：

用户态热补丁

针对内核热补丁，业界已有相对成熟的 kpatch、livepatch 等方案，并已有广泛应用。因此，SysCare 集成了 openEuler 已有的内核热补丁能力。针对用户态程序文件形式、编程语言、编译方式、运行方式多样等问题，SysCare 提出了新的热补丁解决方案，其大概思路如下：

- 对比代码修改前后编译生成的 .o 目标文件，将差异点提取并生成热补丁文件。
- 通过 uprobe 技术，向 GCC 等编译器注入代码，跟踪程序的整个编译过程，并获取制作热补丁所需的信息和 .o 目标文件。
- 使用 uprobe 技术，将热补丁与 ELF 文件绑定。在 ELF 文件运行时，通过 uprobe 触发补丁生效，这样就无需监控进程，无论进程是否已经运行，都可以在打补丁后或新进程运行时使补丁生效；同时该技术也可以给动态库打热补丁，解决了动态库热补丁的难题。

补丁管理

当前的热补丁管理方案比较复杂，SysCare 通过屏蔽内核热补丁和用户态热补丁的差异，简化了补丁管理。通过 syscare build 命令，即可制作指定组件的热补丁。另外，SysCare 也提供了 apply、active、deactive、remove、status、info、list 等补丁管理命令，用于应用、激活、去激活、移除、查询状态、查询信息、查询补丁列表等。

快速重启

SysCare 使用系统 kexec 功能，提供快速重启能力；增加了 CPU park、quick kexec 等技术，极大地提高了系统重启速度；同时融合 systemd 重启流程，保障业务安全退出和快速恢复。

应用场景

内核、动态库、用户态等需要通过热补丁解决 bug、CVE 等场景，软件升级需要系统重启的场景，均可使用 SysCare。

仓库地址

https://gitee.com/openeuler/syscare

05

开发者支持

Compass-CI

服务器 云计算 边缘计算 嵌入式

sig-CICD

Compass-CI 是一个可持续集成的开源软件平台。为开发者提供针对上游开源软件（来自 Github、Gitee、Gitlab 等托管平台）的测试服务、登录服务、故障辅助定界服务和基于历史数据的分析服务。Compass-CI 基于开源软件 PR 进行自动化测试（包括构建测试，软件包自带用例测试等），构建一个开放、完整的任务执行系统。

技术挑战

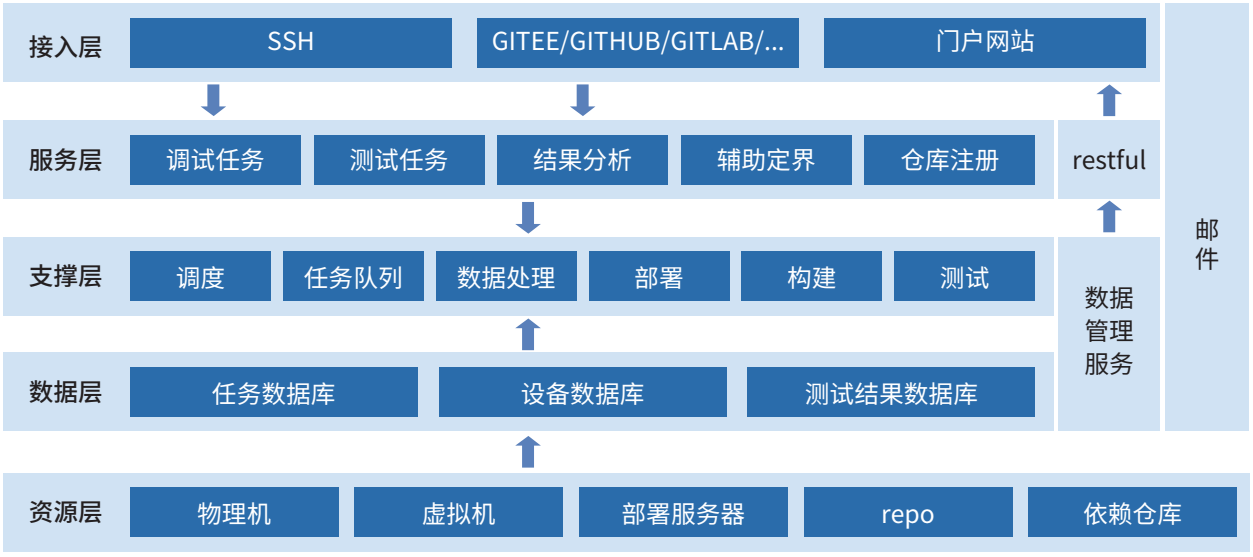
Linux 日趋复杂，开源软件开发者由于受到资源约束，往往只会对单一场景进行验证，面对众多的 Linux 发行版，开源软件如何快速引入及测试验证，是摆在开发者面前的首要问题。

资源单一，使用场景却是 {os, arch, machine, ..} 的矩阵组合。这导致大量的问题会在后续的使用过程中才会被发现，造成更大的修改成本。

问题复现困难，大量成本消耗在环境的准备中，无法进行快速的问题定位。

项目介绍

Compass-CI 是一个通用全栈软件测试平台，集构建 & 测试系统、登录调测、测试分析比较、辅助定位于一体，通过主动测试数以万计的开源软件，暴露这些软件在芯片和操作系统上的问题，在第一时间自动定位问题并以报告的形式反馈给第三方软件开发者，方便第三方开发者能及时处理问题，保障软件质量。给社区开发者提供友好的开发体验，与社区开发者一起繁荣开源软件生态及提升开源软件质量。



- 测试服务：支持开发者基于本地设备开发，往 github 提交代码，Compass-CI 自动获取代码开展测试，并向开发者反馈测试结果。
- 调测环境登录：Compass-CI 提供 SSH 登录能力，测试过程中如果测出问题，开发者可根据需要登录环境进行调测。
- 测试结果分析：Compass-CI 记录历史测试结果，对外提供 web 及命令行接口，支持开发者针对已有的测试结果进行分析，挖掘影响测试结果的因素。
- 辅助定位：Compass-CI 在测试过程中可以自动识别错误信息，触发基于 git tree 的测试，找出引入问题模块的变化点。

应用场景

聚合开发者测试用例：开发者向代码托管平台提交代码、测试用例、测试工具时，Compass-CI 自动获取提交的代码开展构建测试，同时获取开发者编写到开源软件包的测试用例进行自动化测试，并反馈测试结果。

登录环境随时调测：测试过程中，发现 Bug 时，可随时提供调测资源服务，登录到环境进行复现、调试。

快照数据分析对比：测试过程中，全面监控系统运行信息（CPU/MEM/IO/ 网络等），对测试过程中的数据做快照归档，提供多次测试之间快照数据分析对比能力，协助开发者对测试结果开展分析，找出影响测试结果的因素。

辅助定界：测试过程中，发现有 Bug 时，自动触发 Regression 机制，找出首次引入问题 Commit 信息。

仓库地址

https://gitee.com/openeuler/compass-ci

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

CVE Manager 漏洞管理

服务器 | 云计算 | 边缘计算 | 嵌入式

sig-infrastructure | security-committee

漏洞管理是 openEuler 社区对安全漏洞进行感知、收集、处理以及披露的流程、工具和机制的统称。

技术挑战

作为操作系统根社区，社区软件包及社区发行版的安全漏洞感知、处理及披露的及时性和有效性直接影响下游 OSV 及用户的系统安全。

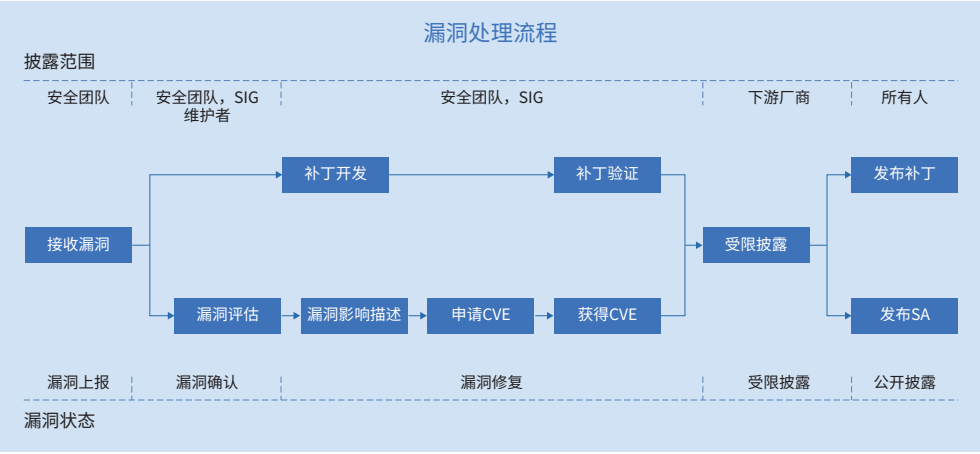
目标：

- 及时感知
- 高效分析
- 快速修复
- 受控披露

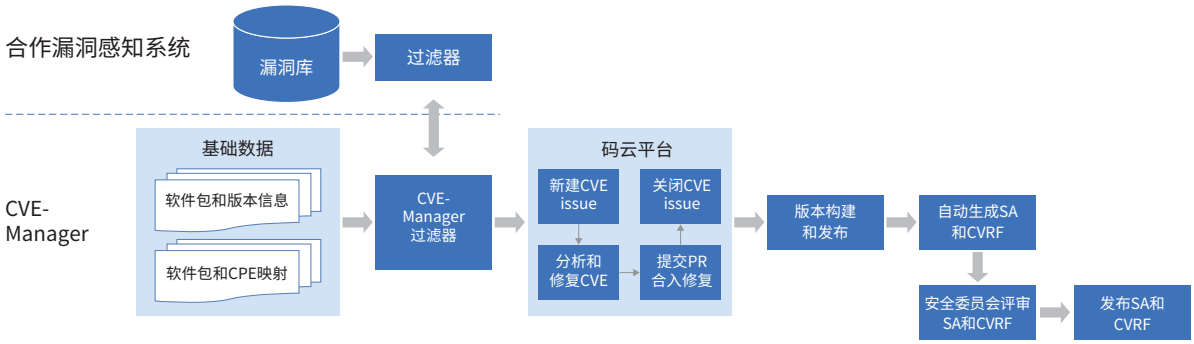
项目介绍

openEuler 社区非常重视社区版本的安全性，openEuler 安全委员会负责接收、调查和披露 openEuler 社区相关的安全漏洞。我们鼓励漏洞研究人员和行业组织主动将 openEuler 社区的疑似安全漏洞报告给 openEuler 社区安全委员会。我们会快速的响应、分析和解决上报的安全问题或安全漏洞。

漏洞响应流程主要支持 openEuler 社区的 LTS 发行版和其分支版本。漏洞端到端的处理流程如下图。



openEuler 安全团队希望上报者将 openEuler 产品疑似安全漏洞上报给 openEuler 社区，并相互配合以负责任的方式修复和披露该问题。漏洞上报方式为通过 email 将 openEuler 产品的潜在安全漏洞发送到 openEuler 安全团队邮箱（openEuler-security@openEuler.org）。安全团队将在 48 小时内响应通过邮箱上报的疑似安全漏洞，并向上报者反馈漏洞处理的进展。



openEuler 社区通过原创的 CVE-Manager 项目从合作漏洞感知系统获取公开漏洞感知信息，然后通过机器人在码云平台对应项目软件包仓创建并维护漏洞相关记录，漏洞修复后进入通用版本构建发布以及安全公告发布流程。

openEuler 使用 CVSSv3 进行漏洞评分。

为了保护 openEuler 用户的安全，在进行调查、修复和发布安全公告之前，openEuler 社区不会公开披露、讨论或确认 openEuler 产品的安全问题。安全漏洞修复后 openEuler 社区会发布安全公告，安全公告内容包括该漏洞的技术细节、CVE 编号、CVSS 安全评分、严重性等级以及受到该漏洞影响的版本和修复版本等信息。安全公告提供邮件订阅功能，同时社区也提供 CVRF 格式的安全公告。

应用场景

应用场景 1：对 openEuler 长期维护发行版进行公开漏洞的感知、分析、修复和披露。

应用场景 2：对 openEuler 长期维护发行版进行 0day 漏洞的收集、分析、修复和披露。

仓库地址

https://www.openEuler.org/zh/security/vulnerability-reporting/

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

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

EUR openEuler 用户软件仓库

openEuler 用户软件仓库，用于支撑社区最外围的第三方包生态，是社区针对开发者推出的个人软件包托管平台，开发者在 EUR 中拥有自己的工作空间，便于上传个人开发的软件包，社区还未引入的软件包或社区软件包的变体，基于 EUR 进行软件包的构建后，还可以便利的将软件包分发给其他用户使用。

技术挑战

围绕 openEuler 的第三方包，社区当前缺少一个比较好的构建，测试分发平台，能支持正在开发中的软件包、社区变体包或长尾包。例如用户期望为已发布的 20.03 LTS 版本提供最新的 nginx 版本，或开发者自己的软件期望尽早与社区版本适配，这样的现状一定程度上降低了开发者为社区适配软件包的热情。

项目介绍

EUR 支持开发者将多种源码格式的 spec 文件和源码包打包成任意 openEuler 版本可用的软件包，并自动完成软件包签名和软件仓库的生成，当前支持通过 git/svn 获取 spec 文件，也支持自动将 pypi，rubygem 上的软件包打包成 rpm 包。

应用场景

- openEuler 社区开发者为社区提供第三方长尾包支持（例如在社区已经下架的软件包）。
- openEuler 社区开发者为社区软件包提供第三方变体包支持（例如为老版本提供最新的 gcc，nginx，gnome 等组件）。
- 上游社区为 openEuler 社区提供快速迭代的第三方新版本软件包（例如 firefox，chrome 的 nightly 版本）。

仓库地址

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

OEPKGS openEuler 软件扩展仓库

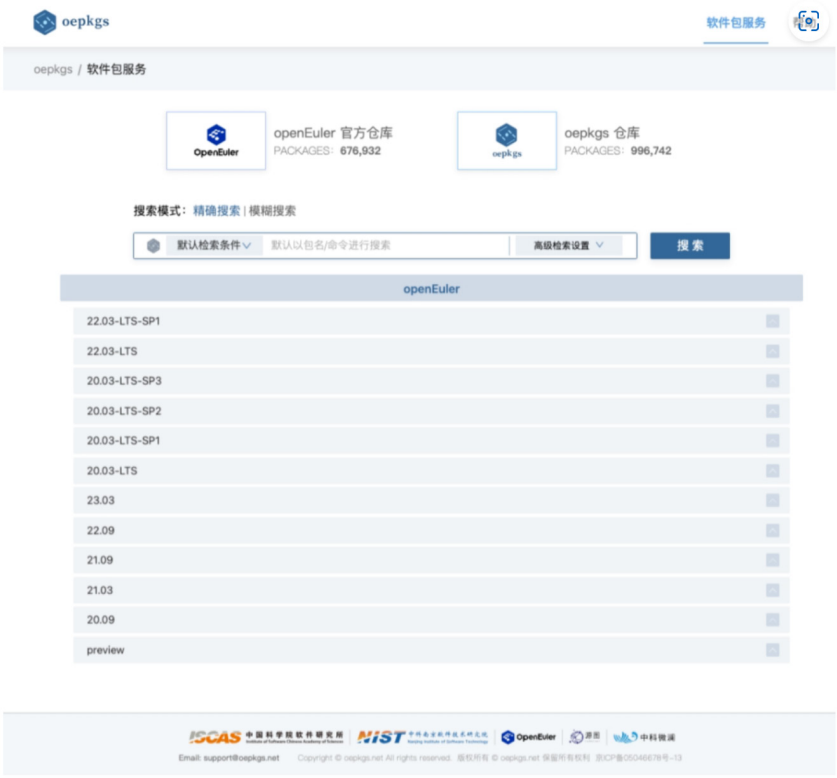
OEPKGS 开放软件包服务（Open External Packages Service）正式上线，为 openEuler 生态提供超过 3 万 + 源码包、百万级二进制软件包，可为 CentOS、Fedora 等系统向 openEuler 迁移的开发者、OSV、企业等用户提供一站式兼容性软件包、文件查询、下载、开源软件包的使用风险感知服务。

技术挑战

- 文件、软件包软件包精准及模糊检索。
- 软件包 CICD 门禁管理，依赖元数据分析及管理，软件包自由组合验证。
- 开源软件风险感知，安全性及合规性风险分析。

项目介绍

OEPKGS 由中国科学院软件研究所、中科南京软件技术研究院、openEuler 社区共同发起并提供支持的项目。旨在作为 openEuler 社区的扩展仓库，为开发者及广大用户提供丰富的软件包，同时提供一个成熟的 CICD 体系，支撑软件包引入溯源分析，源码构建，二进制扫描，基本功能验证，漏洞、合规风险感知，补丁、版本更新感知，保障软件仓库质量可靠及持续演进；结合 RPM 软件包检索、元数据分析、SBOM 和供应链分析、安全性及合规性风险分析等多项能力，同时提供一站式文件、软件包查询、风险感知查询、下载等服务，提升开发者及用户的使用体验。



应用场景

- 面向创新场景开发者提供快速的软件包引入平台，快速将上游项目形成 RPM 包。
- 面向迁移场景开发者提供多版本软件包引入能力，支持传统其他 Linux OS 快速替代。
- 面向广大用户，提供 openEuler 已兼容的软件包查询，文件所在软件包的查询平台。
- 面向企业用户，提供开源软件风险一站式查询平台，感知软件的风险。
- 面向企业，提供闭源软件分发渠道，支持企业软件直达用户。

使用指导

https://www.openeuler.org/zh/blog/liping/2022-11-10-OEPKGS-introduction.html

仓库地址

https://search.oepkgs.net/

https://gitee.com/src-oepkgs

openEuler 软件包贡献平台

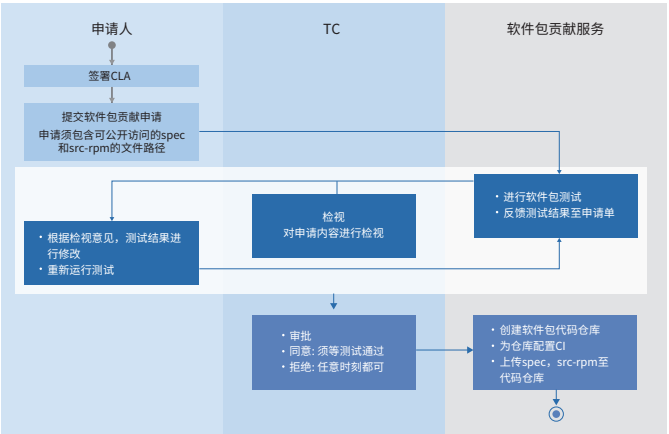
软件包贡献平台旨在为全球的贡献者提供一个统一的、可视化的、便捷的服务，用于将软件包贡献到 openEuler 社区。该平台实现了软件包贡献流水线，覆盖申请测、试、审批、发布等流程；并通过流水线视图，做到贡献过程透明化。平台支持将软件包代码发布到 Gitee 或 Github，海外贡献者可在 Github 上进行后续的版本开发，保证软件包贡献的连续性。该平台必将促进更多的软件包贡献到 openEuler 社区。

技术挑战

openEuler 社区的软件包共分为 3 类，按照核心层度从内到外分别是核心包，扩展包以及第三方包。当前按照统一的流程进行软件包贡献。第一步，到 openEuler/community 仓库提交申请；第二步，等待软件仓库创建完成后，提交软件包的代码并等待合入。整个流程是同步进行的，耗时比较长。另外当前贡献者只能在 Gitee 上进行贡献，海外用户在 Gitee 的使用上存在挑战。

项目介绍

该项目实现了针对 openEuler 软件包的一站式贡献服务，优化了当前的贡献流程，降低贡献者的学习成本，提高贡献效率。优化后的贡献流程如下图所示，将测试和审批流程提前；完成审批后，后续的创建仓库、代码提交等全部由平台完成，从而将贡献流程从同步变成了异步的过程。



应用场景

为社区开发者提供一站式软件包贡献到社区的服务。

仓库地址

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

Signatrust openEuler 软件包签名服务

openEuler Signatrust 是社区基础设施 SIG 组针对操作系统常见的签名场景推出的高效、便捷、安全的签名服务，可支持 openPGP 及 X509 体系的秘钥管理，同时对接了 EFI、RPM、KO、ISO 等多种软件包，支持大批量的软件包签名，可极大提升社区秘钥管理及软件包签名效率。

技术挑战

当前社区现有的 rpm 签名工具，在大批量签名的情况下，存在性能瓶颈，影响整体的软件包构建效率，同时基于 pgp-agent 的本地秘钥存储模式，存在秘钥泄露及管理易用性不够的问题。另外，随着社区发展，需要支持签名的场景及文件越来越多，包括不限于 Kernel Module、EFI、镜像等，需要有服务能在支持多种体系多种文件格式的同时兼顾秘钥存储的安全、签名的高效及管理的易用性。

项目介绍

Signatrust 项目基于 KMS 等体系对密钥对进行加密存储，支持在 TEE 或者内存中对文件进行签名，能有效保证密钥的安全性，同时整个项目基于异步框架，能有效提升高并发场景下的签名效率，最后项目引入了独立的 UI 界面能最大程度上方便管理员对密钥进行管理。

应用场景

- 支持社区版本的 openPGP 秘钥对管理，并对 RPM/SRPM、ISO、Repodata 等文件签名。
- 支持社区版本中的 X509 秘钥对管理，并对 Kernel Module 及 EFI 文件签名。
- 支持社区开发者在用户软件仓库中生成个人 openPGP 秘钥对，并用个人密钥对签名 RPM 包并验签。

仓库地址

https://gitee.com/openeuler/signatrust

EulerLauncher 跨平台 openEuler 开发者工具

EulerLauncher 是由 openEuler 社区技术运营团队及基础设施团队孵化的开发者工具集，通过对主流桌面操作系统中的虚拟化技术 (LXD、HyperV、Virtualization Framework) 等技术进行有机整合，使用 openEuler 社区官方发布的虚拟机、容器镜像，为开发者在 Windows、MacOS、Linux 上提供统一的开发资源 (虚拟机、容器) 发放和管理体验，提升主流桌面操作系统上搭建 openEuler 开发环境的便利性，有效提升开发者体验。

技术挑战

主流桌面操作系统上提供相关开发资源 (虚拟机、容器等) 的便利性和稳定性是影响 openEuler 开发者体验的重要因素，尤其是对于对开发资源受限的个人及高校开发者 openEuler 开发体验影响更为明显。当前常见的虚拟机管理平台有诸多局限性，如 VirtualBox 需要下载体积庞大的 ISO 镜像，同时需要进行操作系统安装等相关操作，WSL 无法提供真实的 openEuler 内核，绝大多数虚拟机管理软件目前对 Apple Sillicon 芯片支持尚不完善且众多软件需要付费等，这些简化了在已有桌面 OS 环境搭建 openEuler 开发环境的步骤，加强使用体验。

项目介绍

功能描述

EulerLauncher 支持在 Windows、MacOS 及 Linux 等主流桌面操作系统上提供方便、易用、统一体验的开发者工具集，硬件架构支持 x86_64 及 Aarch64(包含 Apple Sillicon 系列芯片)；并支持各平台对应的虚拟化硬件加速能力，为开发者提供高性能的开发资源。EulerLauncher 支持使用 openEuler 社区发布的虚拟机、容器 (规划中) 镜像、openEuler 社区提供的 Daily Build 镜像以及其他符合要求的自定义镜像，为开发者提供多种选择。

应用场景

- Windows/MacOS/Linux 桌面环境搭建 openEuler 开发环境。
- 快速获取 openEuler 开发工具及简化在社区贡献依赖的配置。

仓库地址

https://gitee.com/openeuler/eulerlauncher

EulerTest 测试管理平台

服务器 云计算 边缘计算 嵌入式

sig-QA

EulerTest 是一个 openEuler 社区孵化的用以承载社区全流程测试活动的管理平台。该平台核心为 web 端数据中台，帮助社区版本测试高效运作，使能社区版本测试可跟踪可追溯。具备支撑资源管理以及自动化测试功能的插件化服务，支持对接多元测试引擎。

技术挑战

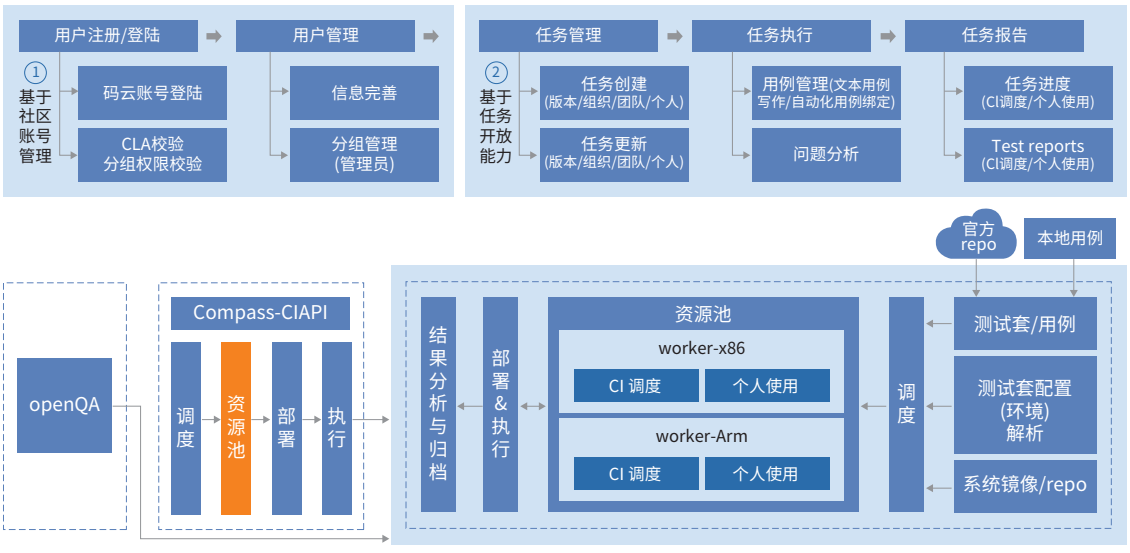
当前社区具备版本集成测试开源测试框架，但缺乏对开发者如何贡献该框架下的文本用例已经自动化脚本的足够牵引，以实现社区测试能力共建与用例上量。

且当前缺乏一个承载社区所有测试活动资产和流程的基础设施，社区各个团队的测试活动依赖于各自离散的本地化管理方式，也因此社区的版本测试活动无法达到完全的质量可信。同时，社区缺乏一个测试维度的枢纽对社区其他工程能力进行串联，从而提升流程效率。

项目介绍

功能描述

- 支持对物理机进行静态资源管理，包括资源的密钥修改、占用释放和系统重装。
- 支持对虚拟机进行动态资源管理，提供网卡磁盘配置热修改以及 web 控制台等功能。
- 支持文本用例的数据管理与和版本用例基线的制定，具备用例评审功能。
- 支持对产品和里程碑进行数据管理，支持和码云企业仓进行数据同步管理，提供版本质量看板支撑质量看护，使能测试可信。
- 支持管理测试任务数据，支持自动化测试触发执行，以及支持手工测试执行 IT 化管理。具备日志按测试步骤分割和分析标注的功能。
- 支持自动化从 openQA、Compass-CI 等平台读取执行结果矩阵，并基于社区既定模板整合为版本测试报告。



应用场景

开发者测试

面向生态伙伴与社区开发者开放测试环境与用例，社区测试环境统一管理，含镜像、虚机、容器，实现并发测试调度与执行。对接多元测试服务，支持伙伴自有环境接入、自定义任务与自助任务执行。

版本质量看护

面向社区正式 release 版本，每日构建、AT 执行、软件范围变更、测试执行、问题闭环、需求进展等多项过程质量，将质量标准 IT 化看护。

仓库地址

https://gitee.com/openeuler/radiaTest

pkgship

pkghsip 是由 openEuler 社区孵化的用于查询 openEuler 发布版本或其他 Linux 版本 rpm 包基本信息及依赖树的可视化工具，使用 pkgship 能大大缩短分析软件包依赖关系的时间，简化软件包引入或升级时分析步骤，有效提升在 openEuler 社区参与贡献和软件包运维的效率。

技术挑战

openEuler 操作系统包含大量的 rpm 包，这些 rpm 包的依赖关系错综复杂，当需要增加、删除或者升级某个包时，需要分析这个包影响或依赖哪些其他的 rpm 包，当前主流的查询方式是使用 dnf/yum 命令进行查询，参数复杂且功能单一，并需要提前加载对应 repo 文件，严重影响开发者使用体验和维护成本。

项目介绍

pkgship 通过分析用户配置的各发布版本的 repo 文件或 url，获取软件包的基本信息和依赖信息，重新组合后存入 Elasticsearch 数据库，并提供命令行和 Restful 接口进行快速查询。可查询内容包括软件包名称、版本、描述、LICENSE 等基本信息，以及一层及多层编译依赖、安装依赖、被依赖信息，并将这些依赖关系以图谱的形式进行展示。使用简单，且多数接口可在 1 秒内返回结果，查询速度快，并支持 repo 配置动态扩展。极大的提高了开发者查询和分析软件包信息的效率。

官方网站：https://pkgmanage.openEuler.org/

应用场景

构建 RPM 软件包自依赖信息查询。

仓库地址

https://gitee.com/openeuler/pkgship

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

QuickIssue 快捷的社区 issue 分类提交工具

QuickIssue 是 openEuler 基础设施团队开发的一个能够满足社区发展的问题跟踪系统。它也是一个更快捷的 issue 分类提交工具，在提交 issue 上具备独特的优势。

项目介绍

QuickIssue 是一个更快捷的 issue 分类提交工具，在提交 issue 上有一些独特的优势：

- QuickIssue 在 openEuler 官网提供统一的 issue 提交入口，开发者在提交 issue 时更便于查找对应仓库。
- QuickIssue 提供了两种提交 issue 的方式，无论开发者是否有 Gitee 账号，都可以提交 issue。
- QuickIssue 可以指导用户或开发者将 issue 提交到某个仓库中，也有默认的仓库可供开发者提交 issue。
- QuickIssue 只为 openEuler 服务，保证查询、搜索、筛选等操作足够顺滑。
- 可以和社区已有的 SIG 管理、贡献统计等服务互通信息。

QuickIssue 提供三个主要功能：新建 issue、查询 issue、查询 PR，可以为开发者提供更便捷的 issue 提交服务。

新建 issue

- 统一了 issue 提交入口，openEuler 社区的所有 issue 都可以通过这个入口提交。
- 解决了 issue 提交人没有代码托管平台账号的问题，可以直接使用邮件和验证码完成 issue 提交。
- 优化了 issue 提交人查找 issue 归属仓库的途径，可以按图索骥找仓库，也可以直接提交默认仓库。

查询 issue

QuickIssue 服务会展示 openEuler 社区所有的 issue 信息。

针对不同的使用场景，QuickIssue 提供了较为便捷的筛选功能，开发者可以按照自己需求定向查找。如果需要查找通过邮件提交的 issue，可以在提交人一栏输入邮箱前半段筛选。

查询 PR

QuickIssue 会展示 openEuler 社区所有的 PR 信息。开发者通过 PR 状态、提交人、标签等信息可以筛选出满足场景的 PR。QuickIssue 中只包含 openEuler 社区的全量 PR，比代码托管平台自身所管理的 PR 信息量少很多，且系统采用了缓存数据，查询操作响应速度有保障。

应用场景

社区开发者提交 issue 入口，快速搜索查看社区 issue、PR。

仓库地址

https://github.com/opensourceways/issue_pr_board

OSV 技术评测

openEuler OSV 技术测评是在开放原子基金会指导下成立的对 OSV 进行技术测评的技术规范，目前通过 openEuler 创新中心具体实施。

项目介绍

OSV 技术测评清单主要存放通过社区 OSV 基础测评的 OS 厂家、OS 版本信息。

OSV 技术测评主要通过对操作系统内核版本，KABI、内核配置，软件包范围、版本、配置、服务、命令、文件一致性检测，通过对 epol/OEPKGS 等仓库的复用度，运行时一致性检测，保持对 openEuler 系生态软件的可用性。

应用场景

通过技术测评，主要测评 OSV 对 openEuler 技术路线的一致性，保持与社区广泛兼容性生态复用，降低重复迁移适配工作量。

面向 OSV 版本开发过程，看护 OSV 版本前向兼容性。

面向迁移场景，制作差异数据库，与 x2openEuler 迁移工具结合，加速 OS 迁移及升级替代。

仓库地址

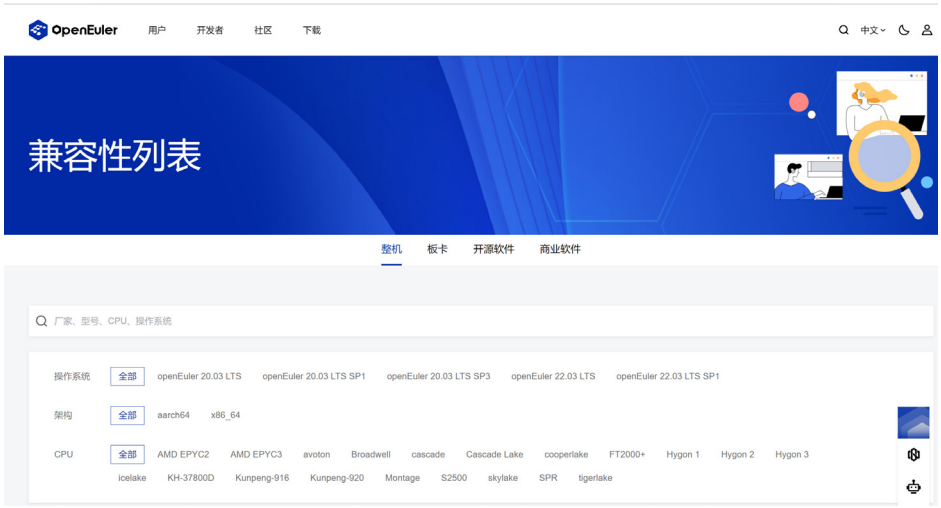
https://gitee.com/openeuler/oecp

openEuler 兼容性全景清单

openEuler 兼容性列表为广大用户提供整机、板卡、开源软件、商业软件查询平台。

项目介绍

openEuler 兼容性清单，分为整机、板卡、开源软件、商业软件等信息，链接参考：https://www.openEuler.org/zh/compatibility/ 。同时开源软件兼容性查询方式通过第三方进行补充，链接参考：https://search.OEPKGS.net/。



硬件兼容性：主要面向南向 CPU 架构、CPU 厂家、整机厂家、整机型号、部件芯片型号、部件芯片厂家等硬件，在社区建立了一套规范、一套流程、一套 CICD 流程，用于指导芯片及板卡厂家在社区建立独立仓库，并基于社区基础设施，保持持续演进；与对应厂家建立芯片功能、版本使能、生态适配三层能力看护，共同保障芯片功能、性能、兼容性、规范性、稳定性。当前已有 marvell、星云智联、云芯智联等公司在社区建仓，并跟随版本例行化运作，相关流程参考：

https://www.openEuler.org/zh/compatibility/hardware/

开源软件兼容性：主要基于业界主流上游活跃项目，按照 openEuler 社区包管理规范集成到 openEuler 社区及扩展社区，保持 openEuler 对主流软件的兼容性，同时建立一套软件包引入平台及机制，便于在 OS 迁移、升级场景快速引入及获取对应版本的软件包，相关流程参考：

https://www.openEuler.org/zh/compatibility/software/

商业软件兼容性：主要面向千行百业 ISV，提供一套对商业软件的测评体系，包括测评规范、流程、方案、工具链，支持 ISV 到创新中心进行测评，测评完成后，提供社区认证的证书，相关流程参考：

https://certification.openEuler.org/

应用场景

通过 openEuler 兼容性清单，用户可以查询及检索 openEuler 版本兼容的 CPU 架构、CPU 厂家、整机厂家、整机型号、部件芯片型号、部件芯片厂家、开源软件、商业软件的兼容性信息。

通过兼容性清单，可以获取 openEuler 兼容性的测试规范、测试方案、流程、相关工具。

通过兼容性清单，获取加入到兼容性的具体操作过程，帮助 IHV/ISV/ 开发者将关注的软硬件加入到兼容性清单。

仓库地址

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

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

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

openEuler 技术测评

openEuler 技术测评是在开放原子基金会指导下成立的对商业软件、硬件、OSV 进行技术测评的技术规范，目前通过 openEuler 创新中心具体实施。

项目介绍

openEuler 技术测评是基于 openEuler 操作系统，在多样性算力平台上，基于社区制定的统一规范，通过统一的设备测评工具、统一的检测标准，构建统一的 openEuler 生态体系。openEuler 技术测评对象主要包括三类：软件、硬件及操作系统。针对软 / 硬件产品、主要测评其对 openEuler 操作系统的兼容性，针对操作系统产品，主要测评其对 openEuler 技术路线的一致性。

当前广大的软件和硬件伙伴，面向操作系统做兼容性验证时，由于缺少平台化、工具化和自动化能力，面临测试效率低、算力成本高和反复测试等问题。

为此，我们基于 openEuler 社区，开发了 openEuler 生态服务平台，该平台可拉通多算力资源，通过提供资源的统一纳管、测试用例的自动执行、测试报告的自动生成等功能，形成一站式的生态服务，大幅提升伙伴生态服务体验。

- 测试更便捷：如前所述，通过统一的资源调度，以及环境安装 / 用例执行 / 报告生成等全流程自动化平台。
- 算力更丰富：社区协同 openEuler 生态创新中心，共建全算力硬件资源平台，当前已支持鲲鹏、x86 等主流算力能力，可为伙伴在国产操作系统的迁移、适配、测评中大幅降低算力成本。
- 服务更高效：通过与基础软件、整机厂商联合互认证，已可为伙伴实现一次 openEuler 测试、多方获取认证证书的能力，大幅提高软件伙伴生态构建效率。

应用场景

通过兼容性技术测评，可以帮助客户从纷杂的产品和解决方案中，快速识别经过严格验证的解决方案；帮助社区通过统一的技术测评体系，繁荣满足操作系统的产业技术生态。

仓库地址

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

感 谢

开发者贡献的每一行代码就像水滴，汇成长江、黄河，汇入 openEuler 社区，形成海纳百川之势，
才有 openEuler 社区三年跨越式的发展，感谢以下在 openEuler 社区贡献的企业：

华为技术有限公司
麒麟软件有限公司
统信软件技术有限公司
江苏润和软件股份有限公司
超聚变数字技术有限公司
中软国际科技服务有限公司
龙芯中科技术股份有限公司
湖南麒麟信安科技股份有限公司
北京拓林思软件有限公司
中国科学院软件研究所
粤港澳大湾区（广东）国创中心
SUSE
中国联通
软通动力信息技术（集团）股份有限公司
英特尔亚太研发有限公司
普华基础软件股份有限公司
中国电信股份有限公司云计算分公司
星辰天合（北京）数据科技有限公司
华中科技大学网络空间安全学院
Linaro Limited
中移（苏州）软件技术有限公司
北京网迅科技有限公司
深圳易宝软件有限公司
北京汇钧科技有限公司
北京青云科技股份有限公司
深信服科技股份有限公司
航天科工网络信息发展有限公司
陕西弓今网络科技有限公司
新华三技术有限公司
广西云旸软件有限公司
无锡先进技术研究院
合芯科技有限公司
网宿科技股份有限公司
北京凝思软件股份有限公司

（排名不分先后）

商标声明

在本手册中以及本手册描述的产品中，出现的商标，产品名称，服务名称以及公司名称，由其各自的所有人拥有。

免责声明

本文档可能含有预测信息，包括但不限于有关未来的财务、运营、产品系列、新技术等信息。由于实践中存在很多不确定因素，可能导致实际结果与预测信息有很大的差别。因此，本文档信息仅供参考，不构成任何要约或承诺，不对您在本文档基础上做出的任何行为承担责任。可能不经通知修改上述信息，恕不另行通知。

未经书面同意，任何单位和个人不得擅自摘抄、复制本手册内容的部分或全部，并不得以任何形式传播。