KYLINSOFT
麒麟软件

银河麒麟云底座操作系统 V10 2406

技术白皮书

麒麟软件有限公司

2024 年 7 月

商标声明

KYLINSOFT 麒麟软件和其他麒麟商标均为麒麟软件有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受麒麟软件有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，麒麟软件有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容有可能变更，麒麟软件有限公司保留在没有任何通知或提示的情况下对内容进行修改的权利。除非另有约定，本文档仅作为使用指导，并不确保手册内容完全没有错误。本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目录

1. 概述

银河麒麟云底座操作系统 V10（Nitrogen）是麒麟软件面向中小云厂商和央国企等云计算场景，针对云宿主机关键诉求，依托服务器操作系统产品重要成果，整合云业务专业能力，将计算底座与上层业务解耦，融合最新的云和容器的开源技术，打造的开放创新、能力先进、自主合规、安全无忧、技术兜底的云底座操作系统。

银河麒麟云底座操作系统 V10（Nitrogen）针对云 IaaS 场景，在 V10 2309 基础上进一步增强和发展，包括增强资源混部优化，高性能网络，优化虚拟机热迁移，增强虚拟机内存可用性，虚拟机跨宿主迁移技术，iSCSI 存储和 IO 优化，安全启动，可信完整性度量，不可变 OS 镜像，故障诊断和性能分析工具等重要功能。此外保持对云底座 2309 平滑升级、兼容性和南北生态。全方位满足用户对云场景底座操作系统的功能性、稳定性、安全性、高性能、可维护性等方面的需求，并提供全生命周期的系统管理和 CVE 维护能力。

2. 系统架构

2.1. 源自社区，回馈社区

银河麒麟云底座操作系统 V10（Nitrogen）源自 Openeuler 社区 NestOS 2203-SP2，并吸收来自多个上游包括 openeuler 在内的优秀技术，联合厂商合作和社区伙伴，共同推出的面向云计算数据中心核心底座操作系统场景的技术产品。并将在未来致力于加强和社区技术联动，形成社区孵化，商业落地。保持开放，创新，稳定，可靠。

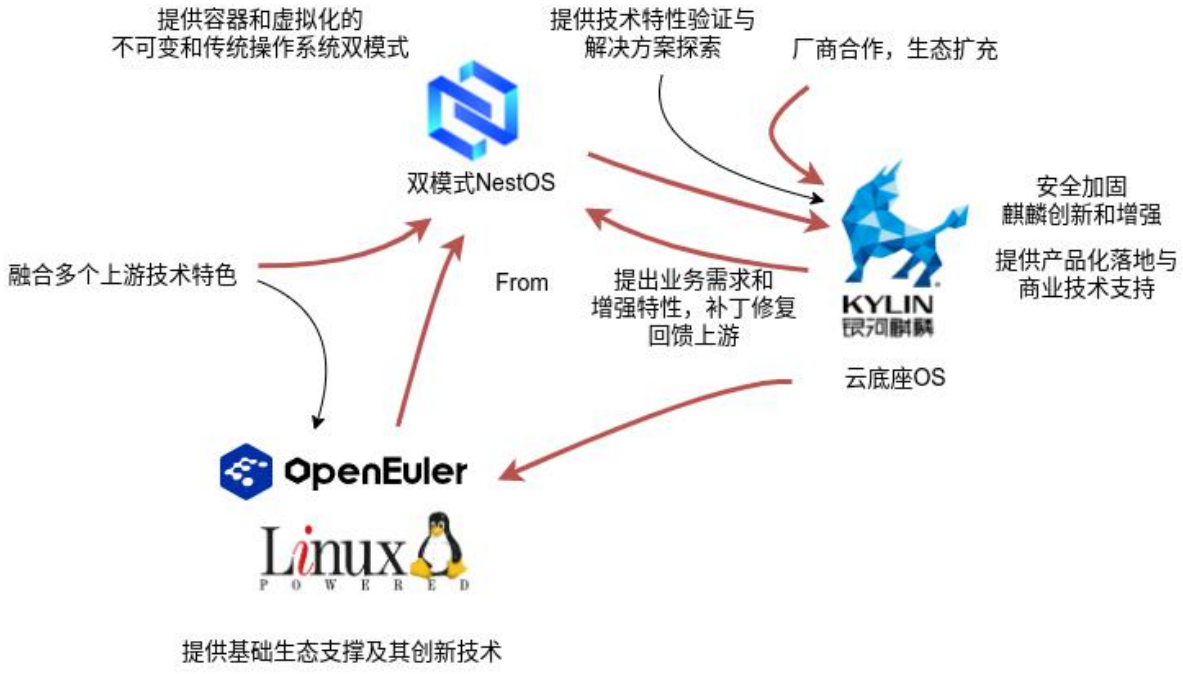

图 1 银河麒麟云底座 OS 与社区关系

2.2. 系统架构图

银河麒麟云底座操作系统 V10（Nitrogen）在设计架构相较于 2309 版本没有大的变化。主要架构新增和改动特色包括：

➢ 扩展硬件生态：支持 CPU 类型包括 Intel GEN4/5/6 系列，Hygon 3/4，Kunpeng 920/920+，飞腾 S2500，2000+。

➢ 云底座内核：本次云底座内核更新显著提升了特定场景系统性能、故障切换和资源利用率。通过并行化 iSCSI 链路切换，大幅缩短故障切换时间，减少业务中断。内核大块 IO 限速优化实现更精细的 I/O 速率控制。内存动态隔离与释放技术提供灵活管理策略，支持安全迁移内存，增强系统稳定性。核隔离增强减少噪声干扰，提升业务性能。这些优化不仅提高系统性能，还增强稳定性和安全性，为开发者提供高效开发环境。

➢ 故障和性能分析增强：本次在操作系统功能设计上，增强了故障和性能分析能力。加入麒麟自研故障分析助手和一键式性能分析调优工具。旨在帮助用户简化对操作系统运维排障和性能瓶颈分析当中的复杂度。优化工作效率，提升系统稳定性和性

能。

➢ 虚拟化特性增强：热迁移是云场景主要用来故障转移的重要手段之一。此次 2406 新增对大内存虚拟机热迁移性能优化技术。同时为了精准辨识内存故障所影响的虚拟机，云底座虚拟化特别基于 MCE 技术实现当内存 UCE 故障时告知用户准确的受影响虚拟机范围。

➢ 资源在离线混部调度：该技术一直是云底座中的核心技术。为满足统一调度下多重负载混合部署时的资源利用效率问题。在 2406 中混部技术再一次增强：混部技术新增基于 ISST 的动态调频功耗节约技术、PSI 反馈内存超分技术和 PSI 监控信息采集，实现高效资源调度与利用，进一步保证高优先级任务性能，同时提升内存使用效率。

➢ 高性能用户态网络协议栈：Gazelle 是一款高性能用户态协议栈，基于 dpdk 和 lwip 提供一套兼顾高性能与通用性的 4 层网络协议栈运行库，提供兼容 POSIX API 的接口，应用实现零修改可享受更高网络性能体验。

➢ NestOS 不可变操作系统：云底座在软件架构上，新增支持云原生场景下不可变 OS 形态。首先，基于 rpm-ostree 的封装技术，确保系统关键部分只读，增强系统稳定性；其次，优化构建工具 nestos-assembler 和配置文件 nestos-config 提供云底座专属镜像构建和配置；最后，支持通过 OCI 格式镜像更新，实现快速部署和升级，适应云计算的高效要求。

➢ 安全启动：新增对国密算法的支持，shim、grub 和内核文件使用 RSA 和国密算法证书双签名，当开启安全启动时，固件中导入 RSA 证书或者国密证书都能正常验签通过，系统正常启动。

➢ 虚拟机安全启动：虚拟机固件新增对国密算法的支持，默认使用麒麟证书作为虚拟固件的 PK、KEK，DB 中添加麒麟国密证书与非国密证书。虚拟机支持国密算法和非国密算法安全启动。

➢ 动态度量：是一款基于完整性进行周期性校验系统重要程序运行态资源的监控

管理组件。动态度量可根据实际策略配置对关键程序进程、内核代码段等系统关键运行资源进行周期度量，当发现异常时将记录日志并根据策略配置决策是否杀死异常进程。以此提高操作系统环境的可信任程度。

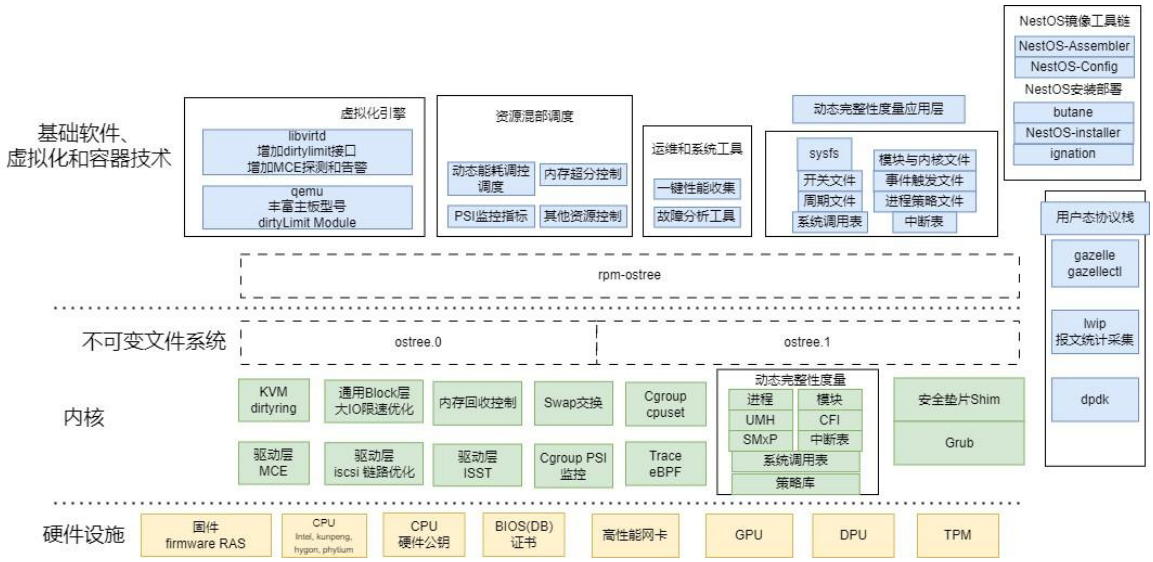

图 2 银河麒麟云底座 OS 技术架构图

3. 云内核增强

3.1. iSCSI 链路切换优化

在存储+iSCSI+多路径场景下，如果主机侧发生链路故障，多路径切换到其他有效路径耗时较高，导致了 IO 数据传输跌为 0 的时间过长，阻塞业务正常运行。耗时高是因为对不同的 session 在进行 block 处理时，采用了串行的方式。针对上述处理方式，进行颗粒的细化，当链路故障时，对 session 进行 block 处理由串行转化成并行方式，加速链路的切换，减小对业务的影响时间。

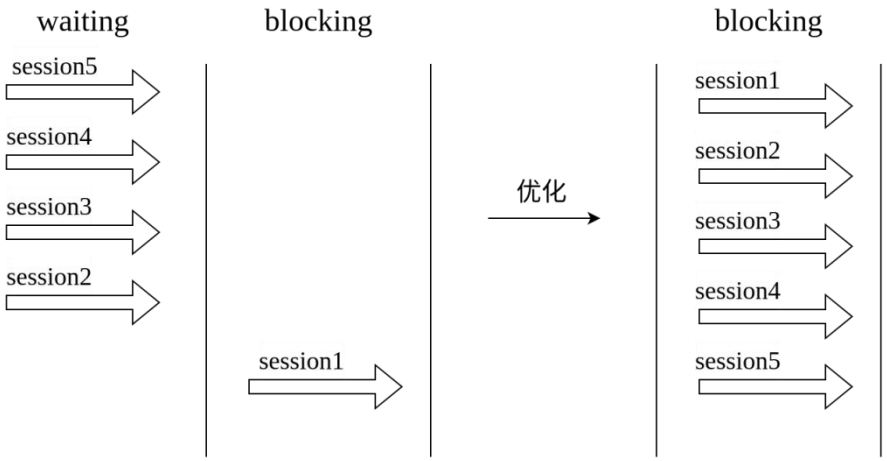

图 3 iSCSI 链路切换时间优化原理

链路切换时间同时还受每个 session 内关联的 lun 数量影响。在 40 个 session，且每个 session 创建 40 个 lun 时，优化前后的测试数据对比如下：

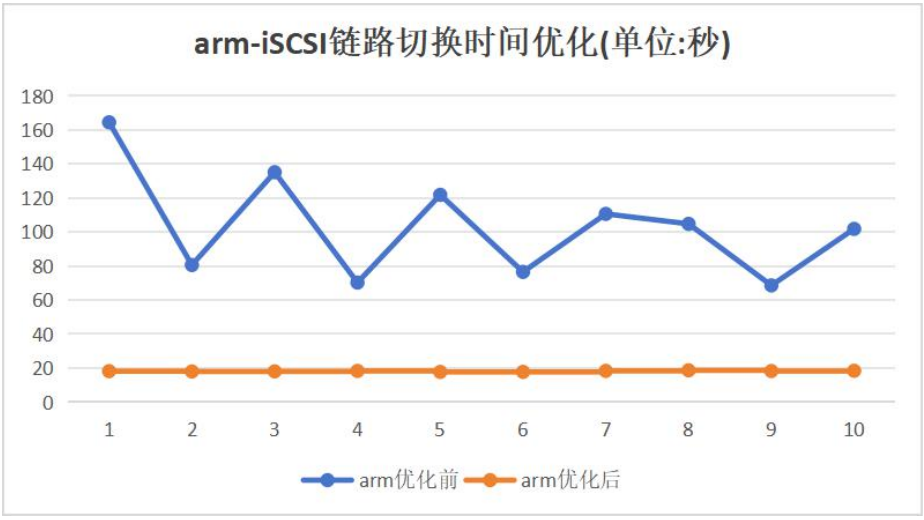

图 4 arm 架构 iSCSI 链路切换时间优化

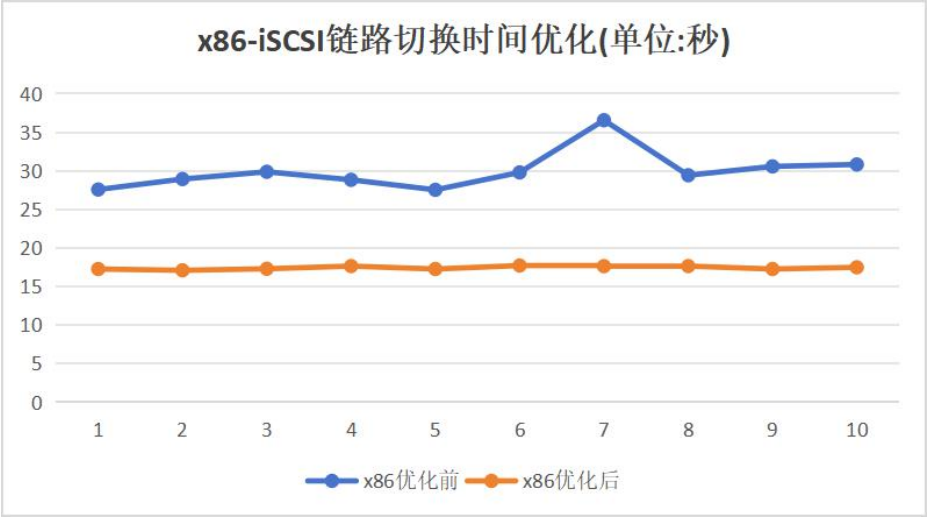

图 5 x86 架构 iSCSI 链路切换时间优化

可以看到，arm 相对 x86 提升较为明显。arm 架构下平均切换时间缩短 85s 左右，提升 82%；x86 架构下的平均切换时间缩短 12s 左右，提升 41%。

3.2. 内核大块 IO 限速优化

Linux Cgroup（Control Groups）是 Linux 内核提供的用于限制、记录、隔离进程组可以使用的资源（cpu、memory、IO 等）的一种机制。Cgroup 里每个子系统（SubSystem）对应一种资源，Cgroup blkio 子系统用于限制块设备 I/O 速率。本特性优化重点是，对比仅在_blk_queue_split()中进行限速，考虑了更多的 IO split（切分）场景，如 bio_split()和 submit_bio_noacct()等，完善了对大块 IO 的 IOPS 限速。

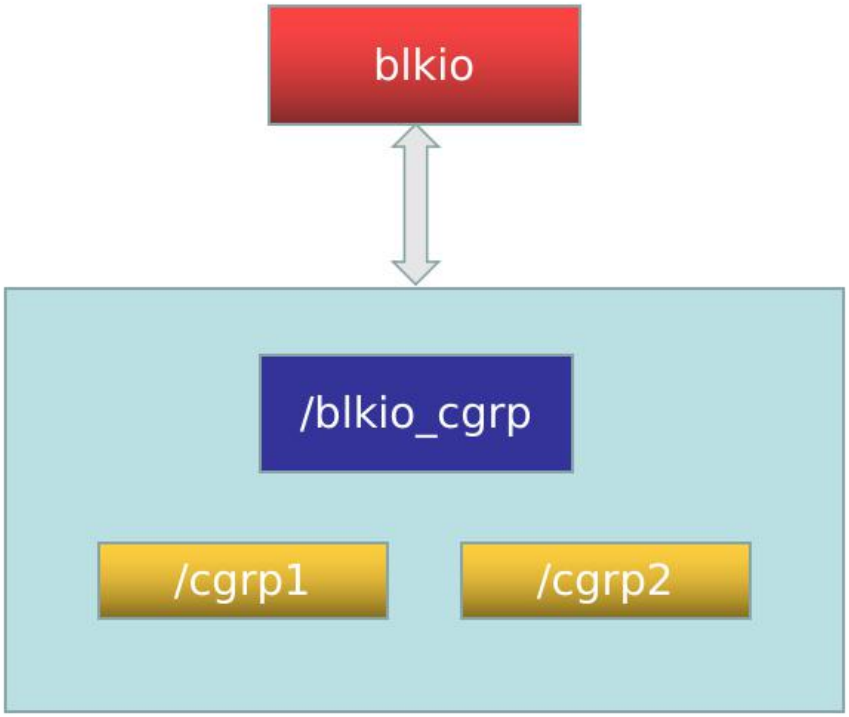

图 6 cgroup 结构图

3.3. 内存动态隔离和释放

内存动态隔离和释放是一种旨在实现安全、稳定的内存页面动态隔离和解除隔离的技术。它广泛应用于虚拟化环境、容器化平台和操作系统级别的隔离机制。其核心

目标是移动内存页面从一个实体到另一个实体，以实现页面隔离或解除隔离的效果，通过这种方式可以对内存资源进行更加灵活的管理和控制，提供安全、稳定的内存页面动态隔离与解除隔离的功能，支持隔离时安全迁移原内存。隔离过程中对待隔离、取消隔离页面进行额外的记录与检查，确保对页面的来源可靠，降低内核风险。

在实现内存动态隔离和释放时，做法是维护一个链表 eject_page_list，用于记录被驱逐或隔离的页面的页框号。只有在该链表中存在的页框号才允许再次上线，即被重新分配给其他实体使用。

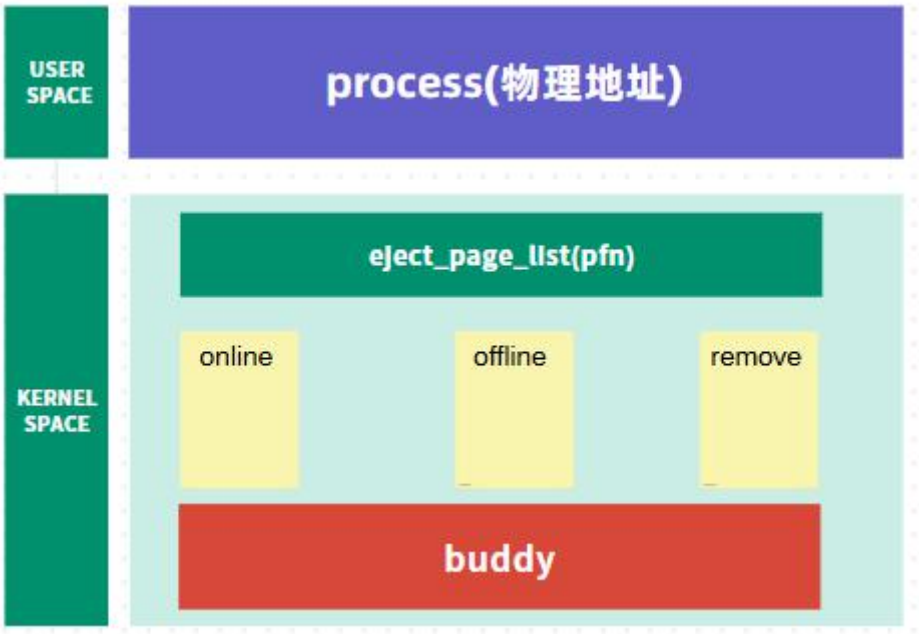

图 7 页面隔离/移除示意图

为了方便用户自主对页面执行上线和离线操作，向用户态提供 sysfs 接口，可以管理内部链表，实现对离线页面的移除操作。

1）/sys/kernel/page_eject/online_page：传入 pfn 将选择页面纳入链表进行隔离管理，方便后续对页面离线和移除操作。

2）/sys/kernel/page_eject/offline_page：页面发生迁移操作，并最终从伙伴系统中移除该页面，同时将该页面纳入链表进行管理。

3）/sys/kernel/page_eject/remove_page：从链表中移除指定的 pfn，并释放相关资源。

这样的内存动态隔离和释放特性提供了更高的灵活性和资源利用率，使系统能够

根据实际需求动态调整内存分配和隔离策略，有助于提高系统的性能、安全性和可靠性。

3.4. 核隔离增强

在 HPC 场景下，一般建议开启核隔离特性，此时 linux 内核会将系统 CPU 分成 housekeeping 和 non-housekeeping 两部分，non-housekeeping CPU 通常被用户态进程独占，主要用于执行业务，如 HPC 应用进程；housekeeping CPU 主要运行 OS 周期性的时钟维护等背景进程等噪声。在实际使用中发现，在开启了核隔离特性后，依然有部分中断出现在 non-housekeeping CPU 中，例如 managed_irq 磁盘中断。本特性对 managed_irq 磁盘中断的选核流程进行识别，将其集中到 housekeeping CPU 上，从而降低此类噪声对业务进程的干扰，达到某些场景提升业务性能的目的。

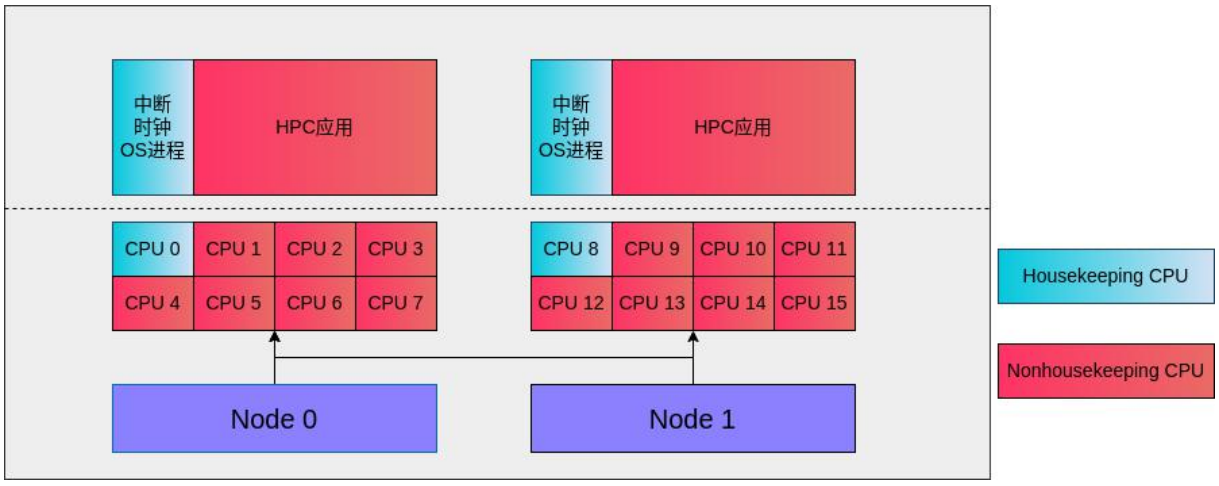

图 8 核隔离示意图

4. 资源调配和优化

4.1. 混部动态频率控制

基于 intel speed select technology 技术，对 k8s pod 根据优先级分配不同的 cpu 工作频率。在保证高优先级 pod 始终运行在高频 cpu 上的情况下，通过降低低优先级 pod cpu 工作频率，来降低整机功耗。

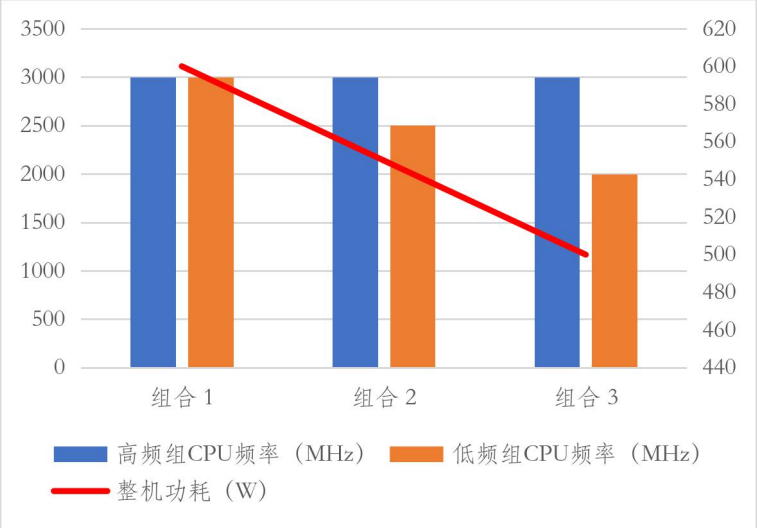

图 9 降低低频组 cpu 频率对整机功耗的影响示意图

（条形图对应左侧坐标轴，折线图对应右侧坐标轴）

同时，监控高优先级 pod 的 CPU 使用率，并根据监控结果进行高频 cpu 数量的动态扩缩，以保证高优先级 pod 拥有足够的计算资源。

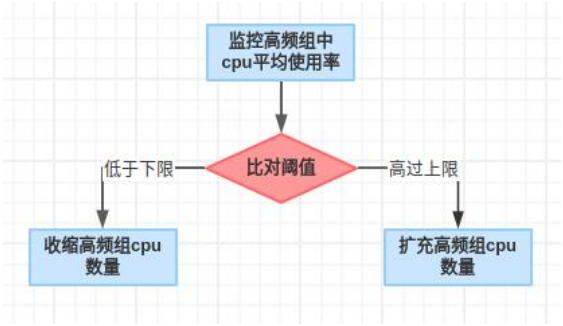

图 10 监控高频 CPU 使用率进行动态扩缩

4.2. 基于 psi 感知的内存超分技术

在离线混部的场景下，由于离线任务对内存的需求量相对较低，可以适当回收离线任务的内存来省出更多的内存空间部署更多的 pod，提高内存使用率。

该特性支持内存超分，用户指定内存超分比和离线任务专用的 swap 设备。rubik 维护一个 psi 回收水位线阈值。并定期检测当前离线内存使用量是否达到超分比。如果未达到超分比，则会收集每个 pod 的 swap psi 指标，将 swap psi 小于 psi 阈值的离线 pod 内存进行回收到用户指定的 swap 设备，每次回收使用量的 n%。在降低回收对 pod 的性能影响的同时达到内存超分的目的。

4.3. psi 信息采集

在离线混部的场景下，需要时刻感知每个业务 pod 的压力，方便用户及时响应对 pod 的维护操作。

该特性支持收集每个业务 pod 的 psi 压力，通过 prometheus exporter 的形式进行观测。rubik 会创建一个 prometheus exporter，并定期收集每个业务 pod cgroup 中的 pressure.stat 里的 psi 信息，包括由于 cgroup_memory_reclaim，global_memory_reclaim，compact，cgroup_async_memory_reclaim，swap，cpu_cfs_bandwidth，cpu_qos 因素导致的 psi 压力,，并通过 prometheus exporter 展示。

5. 虚拟化技术

5.1. 虚拟机热迁移效率优化

热迁移实现逻辑中，如果虚机内存负载高，源端不断产生新的内存脏页，热迁移由于不断传输源端新产生的脏页，导致剩余脏页量一直无法达到阈值，迟迟无法收敛。因此热迁移实现中，一个重要的逻辑就是实现脏页收敛算法，使得源端脏页能够尽快达到阈值。常用的收敛算法是 auto-converge。

auto-converge 是有效的收敛算法，其核心思想是降低脏页产生的速率。通过减少虚拟机 vCPU 运行时间来降低虚机脏页产生速率，使其小于迁移拷贝速率，以满足迁移收敛条件，该算法优点是任何虚拟化场景都适用且有效。缺点是在限制虚机脏页产生的同时，也限制了虚机 vCPU 运行时间，虚机的计算性能在迁移过程中也随之下降。

Dirty Limit 给每个 vCPU 设置一个速率上限，当 vCPU 超过上限时通过 throttle 的方式让其睡眠以达到降低脏页速率的目的。最后周期性计算 vCPU 脏页速率并对比设置的速率上限，当某个 vCPU 超过该上限时，就通过睡眠“惩罚”它。这样可以实现将所有 vCPU 都控制在一个设置的速率上限内。

下图是在一个场景下 dirty-limit 节流算法与 auto-converge 节流算法的迁移时长对比图。

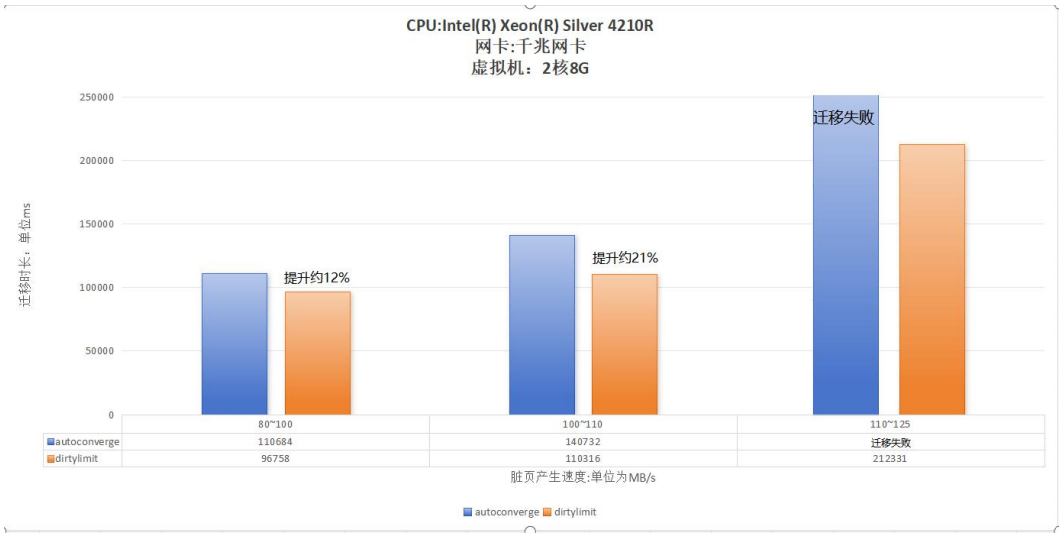

图 11 两种热迁移节流算法迁移时长对比

图中可以看出，随着脏页产生速率增长，dirty-limit 节流算法的迁移时长更短，甚至在 auto-converge 节流失败场景下，dirty-limit 仍然可以完成迁移。

dirty-limit 算法框架示意图见下图：

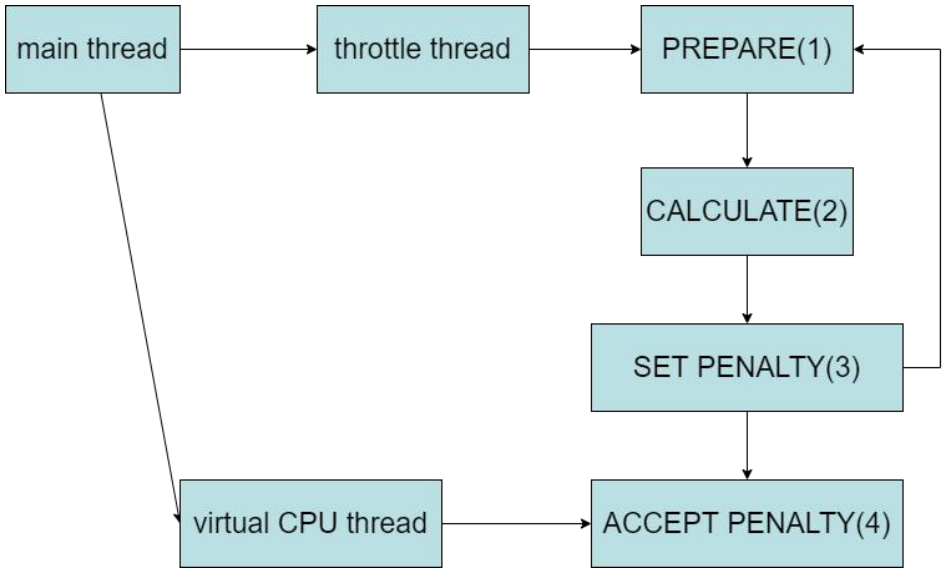

图 12 dirtylimit 算法框架

当 qmp_set_vcpu_dirty_limit 命令被调用时，QEMU 主线程启动了 throttle 线程，其作用是实现 DirtyLimit 主要逻辑，主要包含以下阶段：

1. PREPARE (1) - 准备阶段

这个阶段主要为二阶段- CALCULATE(2)的计算准备输入，主要准备两个值：虚机当前 vCPU 脏页速率（dirty page rate），用户配置的虚机 vCPU 脏页速率上限

（dirty page rate limit），其脏页速率通过调用现有的脏页速率计算接口得到，参考 QEMU 脏页速率计算，脏页速率上限通过用户配置得到。

2. CALCULATE (2) - 计算阶段

这个阶段主要判断 vCPU 的脏页速率是否低于用户配置的脏页速率上限，如果不满足，计算在三阶段-SET PENALTY (3)中用于惩罚虚机 vCPU 的睡眠时间。

3. SET PENALTY (3) - 惩罚阶段

这个阶段是具体让 vCPU 睡眠的阶段，其主要逻辑是在 vCPU 因为 KVM_EXIT_DIRTY_RING_FULL 而异常退出的代码路径上让 vCPU 睡眠。

通过上面三个阶段的操作。QEMU 期望让将 vCPU 的脏页速率限制在用户配置的阈值范围内。

5.2. 基于 Intel MCE 的虚拟机故障定位

MCE 是 Intel 实现的一种机器检查架构，提供能够检测和报告硬件（机器）错误的机制。libvirt 支持获取系统 MCE 报告的内存错误故障点，然后通过对所有运行中的虚拟机进行内存映射检索，诊断受影响的具体虚拟机。该功能能够更精准地诊断与定位硬件故障对虚拟化业务的影响，便于进行故障处理，减少对虚拟化业务的影响和故障处理的开销。

在代码实现上，libvirt 通过内置的事件机制，轮询监听 mcelog 软件的输出信息，当读取到系统报告的 MCE 内存错误地址时，将对所有正在运行的虚拟机逐个遍历其内存地址映射，判断是否有受内存错误影响的虚拟机。然后通过 libvirt 通知接口，将判断结果输出到 libvirtd 服务信息中。用户或其他平台可以通过 libvirt 通知接口获取受内存故障影响的具体虚拟机，从而能够精准地进行故障处理。

其处理流程如下：

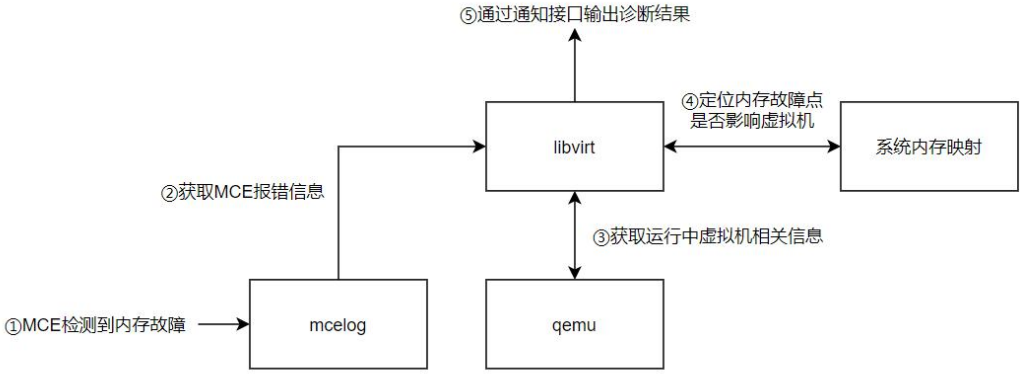

图 13 MCE 故障相应流程

结构图如下：

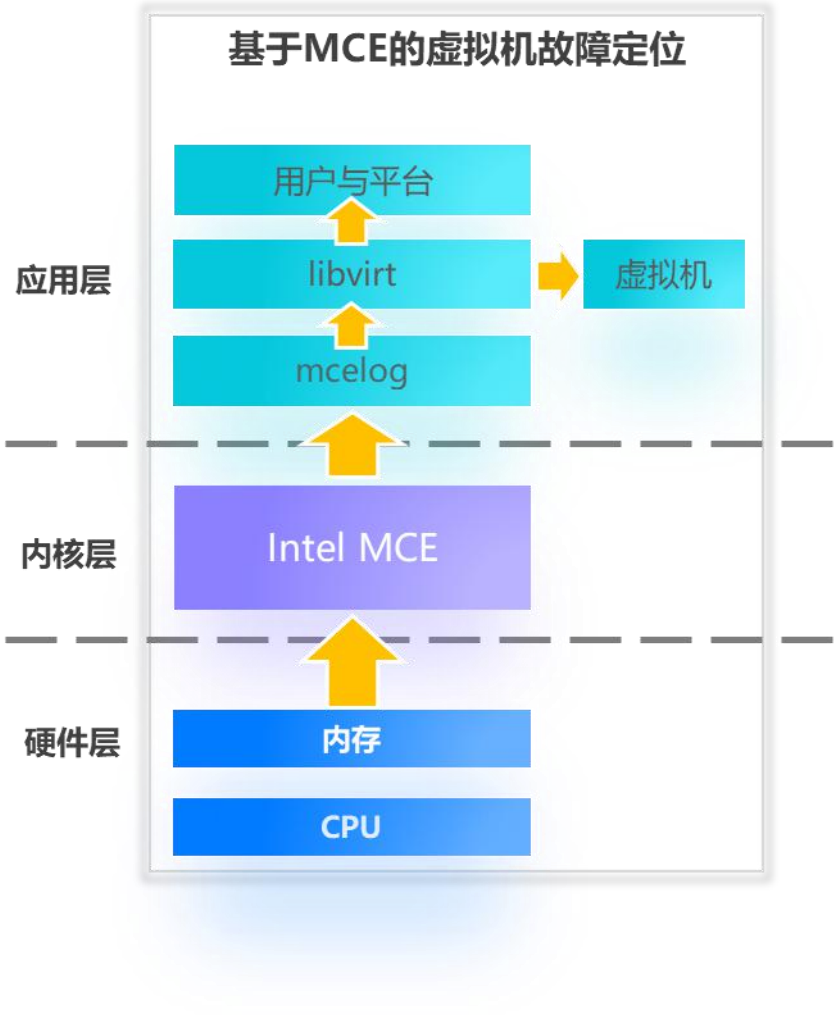

图 14 MCE 故障相应层次结构

注：MCE 只在 Intel 架构 CPU 上支持，其他架构的机器上不支持 MCE 机制，也就无法支持该功能。

5.3. 支持虚拟机跨系统热迁移

支持将 openEuler 22.03 系列及 RHEL8 系列系统上运行的虚拟机热迁移到银河麒麟云底座操作系统上。通过在银河麒麟云底座操作系统上的 Qemu 软件中添加对于 openEuler 22.03 系列及 RHEL8 系列 Linux 发行版上 Qemu 软件机器类型的支持，使虚拟机能够初步支持从 openEuler 22.03 系列及 RHEL8 系列系统热迁移到银河麒麟云底座操作系统。与此同时，对于跨系统热迁移过程中由于 ARM 架构控制寄存器值、64K 页内核与 4K 页内核兼容性、虚拟机固件存放路径变化等问题在 Qemu 软件及内核中进行了针对性的修复，从而完全支持虚拟机从 openEuler 22.03 系列及 RHEL8 系列系统热迁移到银河麒麟云底座操作系统。

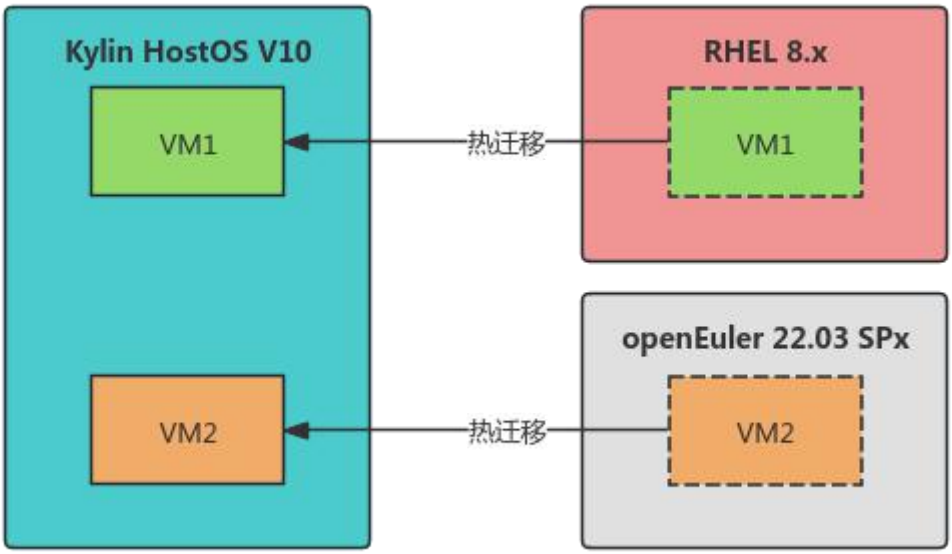

图 15 跨宿主热迁移示意图

6. 故障和性能分析工具

6.1. 系统级故障分析工具

随着系统设计越来越复杂，故障调试成为复杂挑战。已知问题、配置错误、日志不足、监控缺失等问题频发，对技术人员能力要求高。为满足市场高要求及内部 SLA/OLA 标准，我们提出引入“系统体检”机制，主动扫描和检查问题，同时加强主动监控，提前告警并收集关键信息，以减少故障冲击，提高解决效率。整体架构如下图所示：

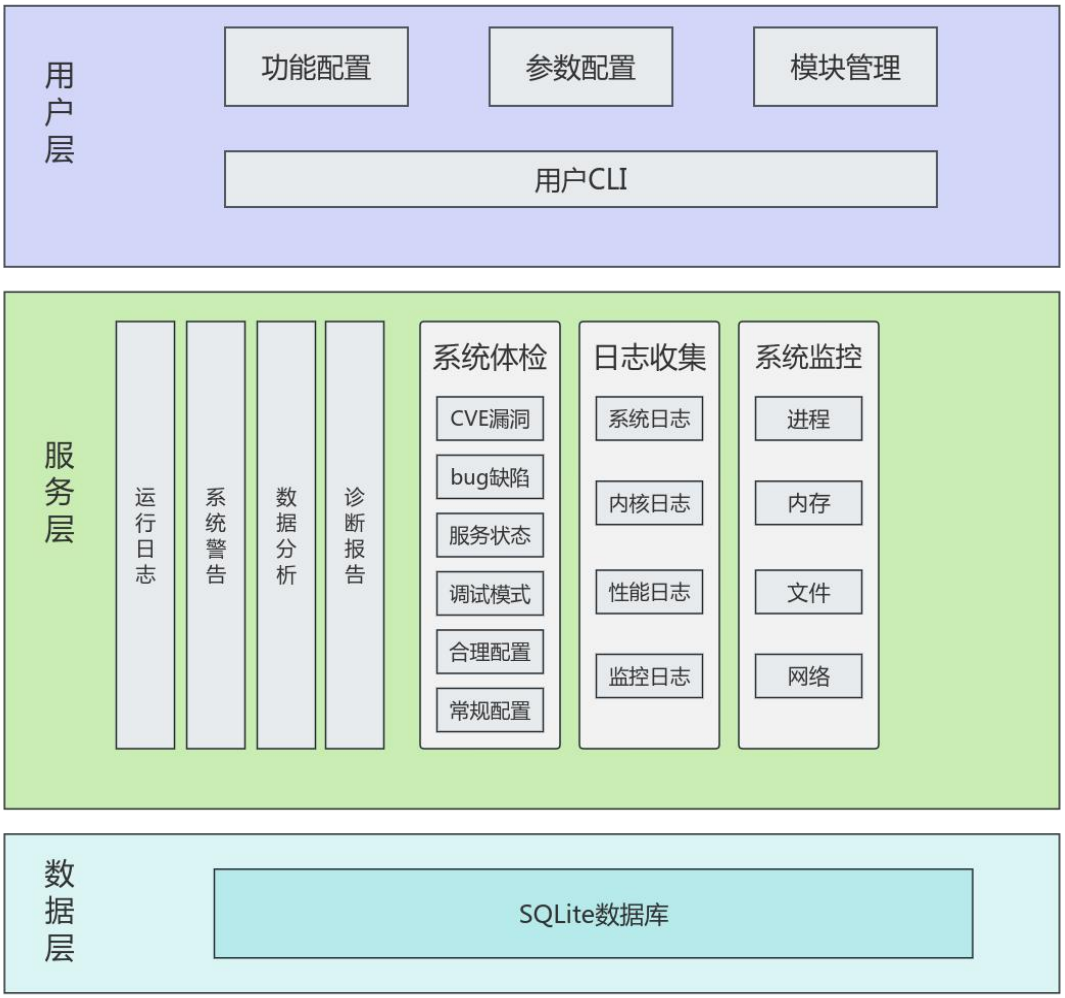

图 16 系统级故障分析工具技术架构图

系统体检功能包含系统上安全漏洞的检测、软件包以及内核缺陷的检查、系统组件服务状态检查、调试项的状态检查、对系统上的特定配置进行合理性检查以及对系

统的必要性配置进行状态检查。

日志收集包含一键收集全量日志的功能、选择性的收集和问题相关的日志以及在监控到系统异常时收集异常点相关的日志信息。

系统监控包含指定目录或文件的元数据和数据操作监控、对系统进程和系统负载的监控、对内存回收状态、内存泄露的监控以及对网络包在各协议栈流转的监控。

6.2. 一键式性能收集工具

一键式性能调优工具是一款旨在帮助用户优化其应用性能的工具，它通过分析当前系统性能指标存在的瓶颈，并以 html 报告的形式给出相应调优建议，进而优化应用性能。工具主要分为三个主要功能模块：采集、分析、静态调优。采集功能主要包括收集 CPU、内存、网络、IO、JVM、系统日志、perf 热点数据信息；分析功能主要包括高亮显示异常指标、识别系统瓶 颈信息并给出调优建议；静态调优功能主要是针对典型应用场景下发优化配置参数。

图 17 性能收集工具架构图

用户可根据需求进行功能参数配置，例如，针对读写 I/O 较多的场景，适当增加收集磁盘信息的时间，使得工具收集更多相关性能数据，分析更加准确，进而给出更合理的调优建议，提高性能调优的效率。

7. 低延迟高性能网络协议支持

Gazelle 是一款高性能用户态协议栈，兼顾高性能与通用性。它提供标准 POSIX API，应用程序可通过 LD_RELOAD 方式加载 gazelle，实现零修改。Gazelle 基于 DPDK 在用户态直接读写网卡报文，共享大页内存传递报文，并轮询模式收报，避免

中断和调度开销。Gazelle 使用轻量级 LwIP 协议栈，将从 DPDK 接收到的报文直接在用户态进行拆解组装转发，以配合 DPDK 将报文收发处理全部在用户态完成，以实现应用网络 I/O 吞吐能力的大幅度提升。

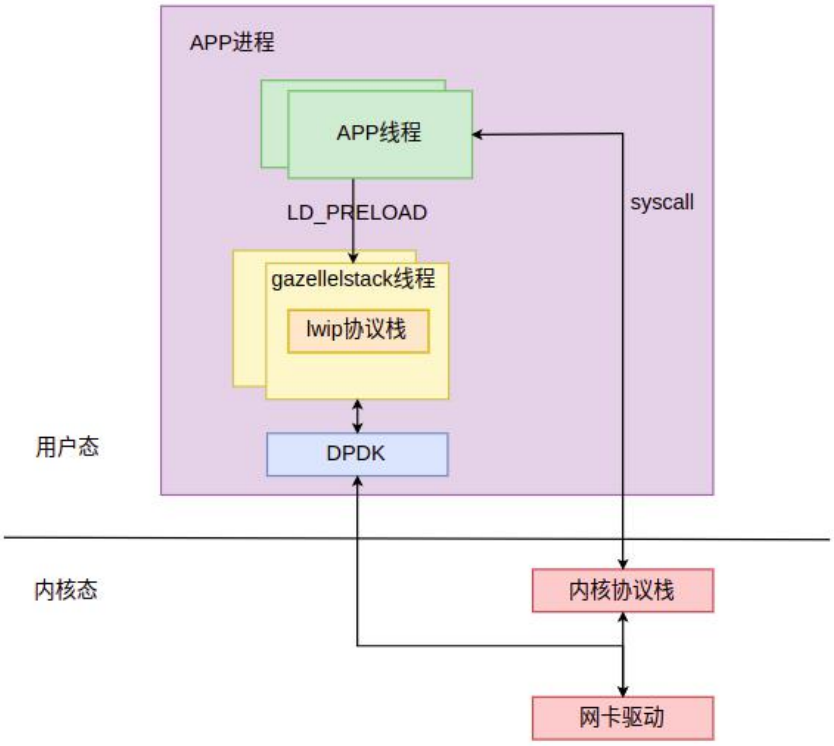

图 18 gazelle 技术架构图

8. 生成不可变操作系统镜像

随着云计算技术的不断发展，操作系统作为云计算底座的重要组成部分，其稳定性和安全性日益受到重视。NestOS 作为一款基于不可变/原子化原理设计的操作系统，以其独特的架构和优势，成为容器云场景下的理想选择。

本产品提供基于 NestOS 技术路线的不可变操作系统镜像定制能力。用户可根据本产品软件源中的构建工具、配置工具和安装部署工具软件包，进行自定义的 NestOS 镜像构建。同时，为简化用户使用操作，本产品将提供构建容器镜像和使用脚本的封装，以及 NestOS 配置规范的校验和转换为点火文件的能力。NestOS 整体架构如下：

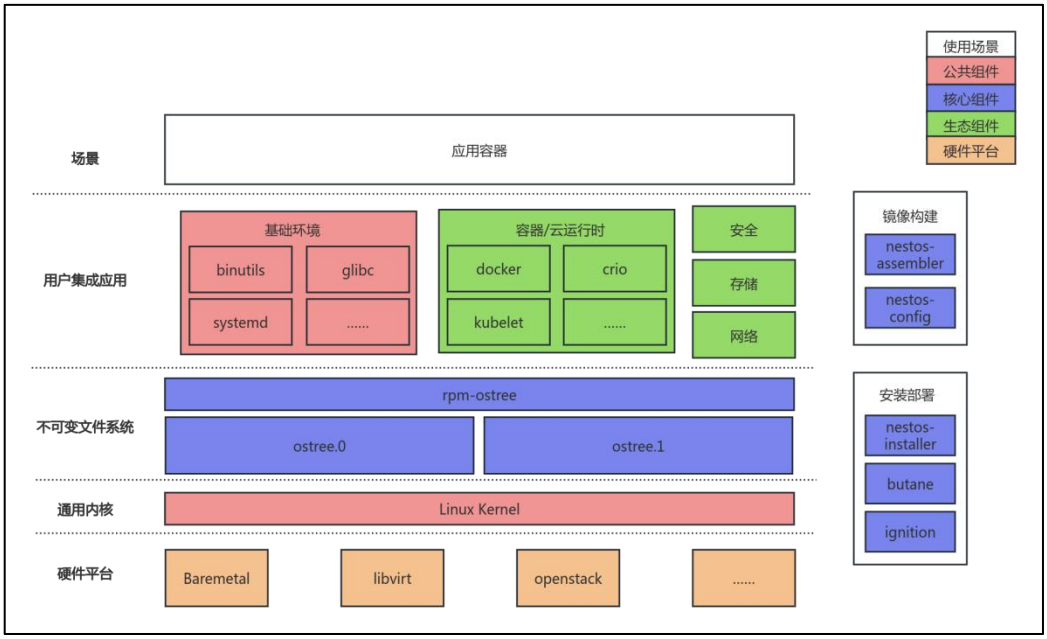

图 19 NestOS 技术架构图

其能力主要涉及以下几方面：

- NestOS 采用基于 ostree 和 rpm-ostree 技术的操作系统封装方案，将关键目录设置为只读状态，保障核心系统文件和配置不被恶意修改。

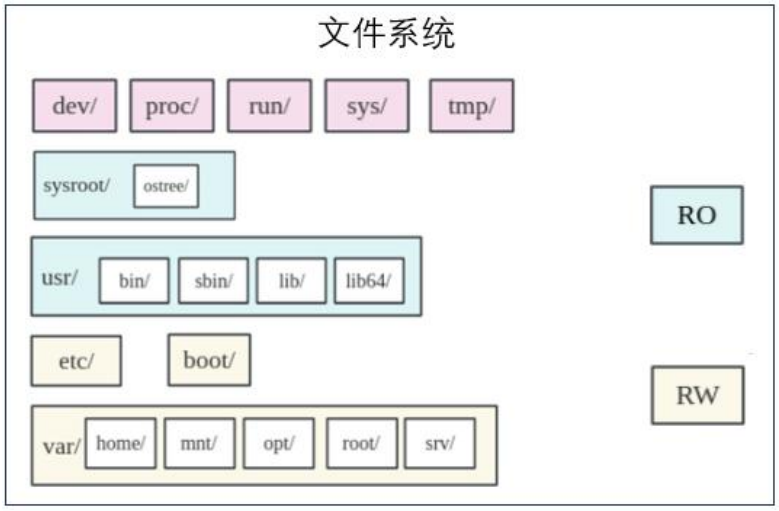

图 20 NestOS 目录结构

- 支持通过 RPM 包的形式分发构建工具 nestos-assembler 和配置文件 nestos-config，帮助用户便捷搭建构建环境，快速实现镜像构建；
- 支持配置规范版本 V1.0.0 与变体 nestos 设置，与 ignition 配置规范 v3.3.0 对应，支持灵活多样的用户自定义配置。
- NestOS 支持通过 OCI 格式镜像更新，实现以镜像为最小粒度进行操作系统版本

的切换，满足快速部署和升级的需求。

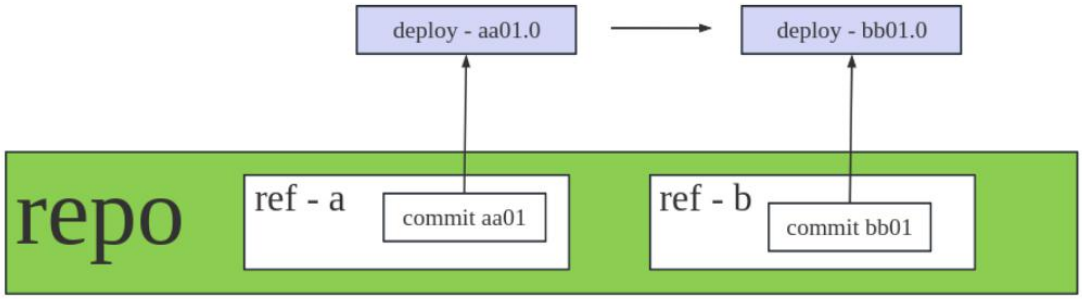

图 21 Ostree 部署切换示意图

9. 安全启动

安全启动（Secure Boot）就是利用公私钥对启动部件进行签名和验证。在启动过程中，前一个部件验证后一个部件的数字签名，如果能验证通过，则运行后一个部件；如果验证不通过，则暂停启动。通过安全启动可以保证系统启动过程中各个部件的完整性，防止没有经过认证的部件被加载运行，从而防止对系统及用户数据产生安全威胁。

安全启动涉及的验证组件： BIOS->shim->grub->vmlinuz（依次验签通过并加载），其中 vmlinuz 是内核镜像。

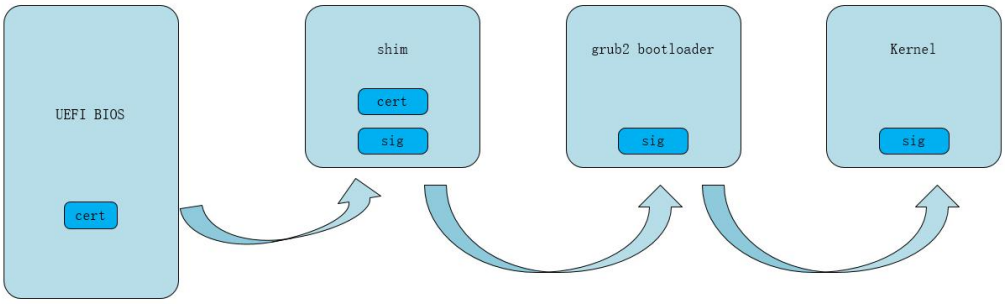

图 22 安全启动验签流程图

10. 动态度量

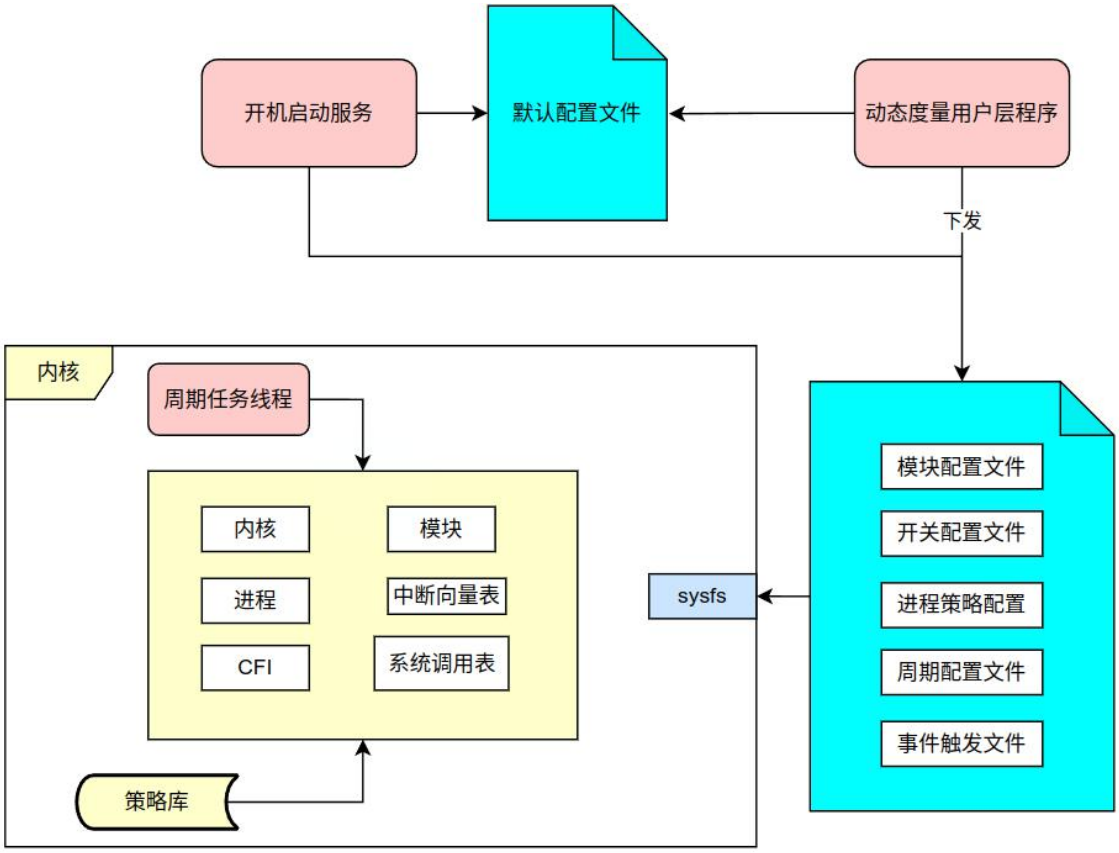

图 23 动态度量结构框图

动态度量是针对特定运行时程序的周期性完整性检查功能，目的是监视被保护集合是否发生恶意篡改，当篡改发生时记录篡改告警信息到审计日志中。被检查对象可根据策略进行配置，其中被度量目标包括进程代码段、内核代码段和只读数据段等。度量周期频度可根据实际情况进行配置以均衡性能占用与时效性。针对用户层软件主体除可监测进程是否发生篡改外，还可以通过策略配置实现杀死被篡改进程的能力。针对策略配置及查询动态度量为用户提供了命令行策略查询配置功能。