OpenAnolis
龙蜥社区

龙蜥操作系统开源社区白皮书

2025 OPENANOLIS WHITEPAPER

CONTENTS

目录

04 社区伙伴风采展

05 社区技术演进

06 原生技术概览

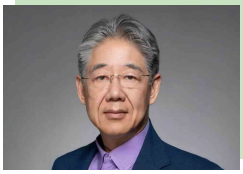
07 社区产品与生态

08 社区风采

09 社区年鉴

10 他们说

1. 寄语



丁津泰

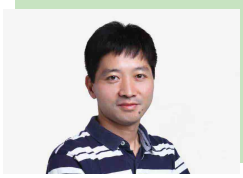
西交利物浦大学数学物理学院院长、PQC-X 实验室主任、龙蜥社区高级顾问

我始终坚信，开源不仅是技术的共享，更是人类智慧跨越边界的协作。龙蜥社区自成立以来，秉持“平等、开放、协作、创新”的愿景，已迅速成长为全球开源版图中的重要力量，这令我倍感欣慰。过去的一年，我们见证了龙蜥作为“开源根社区”的深厚底蕴。作为社区的高级顾问，我正带领西浦 PQC-X 实验室与龙蜥深度对接。面对量子计算时代的安全性挑战，我们正将世界前沿的后量子密码学（PQC）理论转化为操作系统的安全底座，共同定义下一代抗量子攻击的技术标准。《2025 龙蜥社区白皮书》不仅是创新成果的集锦，更是对开源生态路径的深度探索。未来，我们将继续以全球视野赋能本土生态，与龙蜥社区并肩前行，让开源技术在安全与创新的双重驱动下，为信息时代筑起坚不可摧的数字基石。

包云岗

中国科学院计算技术研究所研究员、副所长，龙蜥社区高级顾问

龙蜥社区作为我国操作系统开源根社区的重要力量，在推动国产操作系统与 RISC-V 架构深度融合方面走在了行业前列。过去一年，龙蜥社区完成了 Anolis OS 对 RISC-V 架构的全栈适配，从内核优化、工具链完善到应用生态迁移，实现了性能与兼容性的双重突破。这一成果不仅为 RISC-V 生态提供了坚实的操作系统底座，更为我国"芯片 - 系统"协同创新探索了可行路径。《2025 龙蜥社区白皮书》系统梳理了龙蜥在 RISC-V 领域的技术积累与生态实践，期待未来龙蜥社区继续携手产业伙伴，深化开源协作，共同推动 RISC-V 生态繁荣发展，为我国信息技术自主可控贡献更多力量。



章文嵩博士

LVS 开源项目创始人、CCF 开源发展技术委员会副主任、龙蜥社区高级顾问

龙蜥社区自成立以来，汇聚了产业与开发者的智慧，在技术探索、生态建设上取得了令人瞩目的成绩，成为开源服务器操作系统的几大选择之一。希望龙蜥社区始终坚持以用户需求为导向，不断提升操作系统的使用体验。期待龙蜥社区治理体系在开放、透明、共建、共治的原则下日臻完善，平台能开发、透明地看到各方的贡献，也能吸引更多开发者积极参与建设，相互激发创新，成为开源生态共建共治的典范。在智能化时代，也期待龙蜥社区能勇立潮头，围绕 AI 场景深化软硬协同创新，共同打造更智能、更高效、更可靠的基础软件栈。

刘澎

中国开源软件推进联盟副主席兼秘书长、中国科学院软件所研究员
龙蜥社区特约顾问

自成立以来，龙蜥社区始终坚持“平等、开放、协作、创新”的原则，在众多开发者与合作伙伴的共同努力下取得了令人瞩目的成就。过去的一年里，我们见证了一个充满活力与创新精神的开源社区迅猛发展，成为推动信息技术领域向前发展的重要力量之一，尤其是在基础软件促进人工智能，人工智能融入基础软件方面发挥了领导作用。作为国内领先的操作系统开源根社区，《2025 龙蜥社区白皮书》聚焦开源生态发展路径，汇集诸多龙蜥生态的创新成果，为我国操作系统发展提供了更多龙蜥方案。未来，中国开源软件推进联盟将继续和龙蜥社区一起，积极推进开源事业发展，共同为我国开源生态建设贡献力量。

潘爱民

杭州指令集创始人、首席专家，龙蜥社区特约顾问

龙蜥社区是中国软件开源和生态发展的典范，五年多发展硕果累累。尤其是，在人工智能迅速发展并且深刻影响软件工程技术发展的2025年，龙蜥操作系统作为通用基础软件，一方面成为人工智能整体技术栈的重要基础，另一方面也是人工智能转变成生产力的重要发挥场景。《2025 龙蜥社区白皮书》呈现了龙蜥操作系统社区的一年进展，从技术演进、社区合作、场景融合等方面阐述了所取得的成绩。期待龙蜥社区再接再厉、坚持创新、勇于突破，发展成一个“人工智能原生社区”。

黄晓庆

达闼机器人创始人、龙蜥社区特约顾问

感谢龙蜥社区为中国的 Linux 开源社区所做的贡献，特别是在用我们国产的 Linux 系统代替传统的欧美主导的开源体系，比如说 CentOS 等 Linux 操作系统。希望龙蜥社区在云计算、人工智能、运营商后台系统、金融业和各个行业推动国产软件替代海外的开源体系，同时也感谢大量的开源社区的参与者和贡献者对龙蜥社区和核心开源软件的贡献。希望在 2026 年，龙蜥社区能够更加关注 AI 领域的发展，特别是为 Agentic 系统和应用领域提供更智能化的解答方案和更优秀的操作系统环境。

章明星

清华大学计算机科学与技术系副教授、KVCache.AI 团队负责人
龙蜥社区特约顾问

在当下的技术变革中，开源已然成为推动前沿创新与产业繁荣的核心力量。过去一年，我们欣喜地见证了开源生态的巨大潜能，特别是 Mooncake 与龙蜥社区的深度合作取得了极大的行业影响力，生动诠释了顶层 AI 基础设施与底层操作系统协同创新的巨大价值。当前，我们正大步迈入全新的 Agent 时代。面对更加复杂的系统架构与计算范式，应用场景对强大、灵活的操作系统的需求比以往任何时候都更为迫切。这是前所未有的挑战，更是属于开源操作系统的巨大机会。展望新的一年，期待龙蜥社区在新的技术浪潮中乘风破浪，凝聚更多开发者的智慧，取得更大的成功！

2. 序

在这个充满挑战与机遇的智算新时代，非常荣幸我能和大家一起再次共同见证《2025 龙蜥操作系统开源社区白皮书》的全新发布。自 2021 年首次发布以来，这份白皮书已成为龙蜥社区技术创新与生态演进的年度里程碑。值此辞旧迎新之际，带着满心的感激与自豪，请允许我向您呈现 2025 年的龙蜥社区白皮书，一同回顾过去一年的非凡成就，并展望 “AI for System” 与 “System for AI” 双向融合下的宏伟蓝图。本白皮书部分数据更新至 2026 年 4 月。

过去的一年，龙蜥社区在 “智算” 浪潮的推动下发展势头更加迅猛。截至目前，我们的大家庭已汇聚超过千家合作伙伴，涵盖了从底层芯片、整机硬件到云计算平台及上层应用的全产业链佼佼者。龙蜥操作系统装机量突破 1000 万，秉承 “开放、平等、协作、创新” 的社区精神，我们共同推动了面向人工智能时代的开源新基建，致力于将龙蜥社区打造成为全球智能计算的坚实基石。在此，我们要特别感谢中兴通讯、浪潮信息、海光信息、AMD、英特尔、统信软件和阿里云等核心伙伴单位的卓越贡献。正是这些伙伴的多元投入，让龙蜥社区的技术生态更加开放、繁荣，他们的精彩案例与卓越成就也在本书中得到了重点展示。

2025 年，我们积极响应全球智算转型的浪潮，聚焦于 RISC-V 架构的创新突破与 AI 全栈能力的深度融合，取得了不菲的成果。在 RISC-V 领域，社区携手中兴、达摩院、浪潮信息、中科院软件所、在 RISC-V 领域，社区携手阿里云、中兴通讯、浪潮信息等伙伴发布支持 RVA23 高性能扩展的 Anolis 23 RISC-V 正式版，并制定了 RISC-V SIG 2.0 规划，共同推动 RISC-V 向数据中心级芯片技术架构演进。此外，社区专家在 RISC-V 国际基金会担任主席/副主席要职，主导 Data Center SIG 运作，积极参与全球标准建设。2025 年 8 月，龙蜥社区智算联盟成立，秉承 “开源开放 求同存异” 理念，旨在满足人工智能大模型落地需求，推动操作系统和 AI 融合发展，促进 AI 技术在各行业的落地应用。

从生态共建到国际标准主导，龙蜥社区的发展，离不开每一位成员的辛勤付出和智慧贡献。我们深知，面对 AI 与芯片架构变革的双重机遇，技术的创新和社区的壮大需要每一位成员的共同努力。因此，我们始终坚持开放、平等、协作、创新的原则，鼓励每一位成员积极参与到社区建设中来。我们相信，通过大家的共同努力，龙蜥社区将能够在全球智能创新的舞台上发挥更加核心的作用。

在这一年中，我们也获得了业界的广泛认可，荣获 2025 年度 AI 工程与部署卓越奖、“人工智能+” 行业最佳解决/落地方案” 奖、OS2ATC 2025 最具影响力开源创新贡献奖等多项殊荣。这些荣誉不仅是对龙蜥社区过去一年在业界深耕的肯定，更是对我们未来引领技术变革的鼓励和鞭策。

展望未来，我们坚信龙蜥社区在 “AI for System” 与 “System for AI” 双引擎驱动下的发展前景是光明的。我们将继续秉承开源精神，推动架构创新与智能融合，加强产业合作，扩大社区影响力。我们期待与更多的伙伴携手，共同打造一个更加开放、更加智能、更加繁荣的龙蜥社区，共创智算新纪元。

龙蜥社区理事长、阿里云智能集团研发副总裁 马涛

3. 关于龙蜥

龙蜥社区（OpenAnolis）是立足中国、面向全球的 Linux 服务器操作系统开源根社区，致力于引领云智融合时代的国产操作系统创新。截至 2026 年 4 月，社区理事会已汇聚阿里云、浪潮信息、统信软件、中兴通讯等 25 家国内外领军企业，海光信息与 AMD 亦分别于 2025 年晋升为副理事长及理事单位行列。目前，社区已吸引超 2 万名开发者及 1000 余家全产业链生态伙伴共建，覆盖芯片、软件、整机及操作系统厂商。龙蜥操作系统装机量突破 1000 万，服务了金融、通信、政务、能源、交通、互联网、AI 模型及应用等众多行业超过 200 多万用户。

龙蜥操作系统（Anolis OS）性能和稳定性经过历年“双 11”历练，能为云上典型用户场景带来 40% 的综合性能提升，故障率降低 50%，兼容 Linux 生态，支持一键迁移，并提供全栈国密能力。截至目前，龙蜥操作系统已发布 Anolis OS 23 系列版本和 RISC-V 正式版、Anolis OS 8 等多个社区版本。其中，Anolis OS 23.4 版本支持开源发展合作倡议针对 RISC-V 新一代芯片选型规范，同时在核心工具链性能和安全、软件包生态以及 AI 生态扩展上持续增强优化。当前，阿里云、浪潮信息、中兴通讯 | 新支点、统信软件等超过 14 家合作伙伴也已基于龙蜥操作系统发布商业版本，为用户提供高效、专业的技术支持和服务。

龙蜥社区组织架构

4. 社区伙伴风采展

4.1 “海”“蜥”融智 共铸芯魂

海光信息技术股份有限公司成立于 2014 年，主要从事高端处理器、加速器等计算芯片产品和系统的研究、开发，目标成为世界一流的芯片企业，为数字中国提供核心计算引擎。作为国产先进微处理器产业的推动者，海光信息以务实的态度、创新的理念、先进的技术和可靠的产品，致力于促进我国信息产业核心竞争力的提升。面向企业计算、云计算数据中心、大数据分析、人工智能、边缘计算等众多领域，海光信息提供了多种形态的海光处理器芯片，满足互联网、电信、金融、交通、能源、中小企业等行业的广泛应用需求。

海光信息作为龙蜥社区的副理事长单位，始终深度且积极地投身于龙蜥社区的生态建设工作。在社区发展进程中，海光信息先后创建了 Hygon Arch SIG 组和编译 SIG 组，其中，Hygon Arch SIG 组凭借活跃的交流氛围与丰富的成果产出，成为当下社区中活跃度最高的 SIG 组之一。依托海光先进的安全可信技术，海光信息与龙蜥社区联合发布了基于海光平台的机密计算方案，并携手安全厂商对该方案进行拓展完善。此外，海光信息还深度参与龙蜥社区安全联盟、龙蜥社区智算基础设施联盟建设。在云栖大会上，海光信息与龙蜥重要理事共同推进 OASA 硬件安全合作计划，为龙蜥社区的硬件与软件安全筑牢根基。凭借在社区的卓越贡献，2025 年，海光信息荣获龙蜥社区最佳合作伙伴奖、优秀贡献者奖、最佳联合解决方案奖等多项荣誉。

4.2 “云”“蜥”共舞 智领未来

阿里云计算有限公司是全球领先的云计算及人工智能科技公司，提供云服务器、云数据库、云安全、云存储、企业应用及行业解决方案服务。其研发的稳定、安全、高性能的阿里云服务器操作系统 Alibaba Cloud Linux（以下简称 Alinux）为用户提供最长13年的企业级支持和维护，满足合规要求，已累计服务超过 60 万+ 用户。其中，Alinux 3 是云上部署规模最大，CentOS 停服替代首选，符合高标准合规性要求，为千行百业客户提供稳定、安全、高性能的业务连续性保障。Alinux 4 是一款 AI 驱动，原生安全，面向云+AI 场景协同优化的全新一代自研操作系统产品，AI 时代开箱即用的软件运行环境基座。

2025 年，阿里云作为龙蜥社区理事长单位，在“云+AI”战略驱动下和龙蜥社区一同在技术、生态与商业维度实现全方位跃迁。技术上，阿里云联合龙蜥社区提出 System for AI 与 AI for System的“双循环”技术路线，提升上层 AI 的训练推理性能；同时通过三大合作计划，进一步强化了Alinux AI运行环境的开箱即用能力，并在安全供应链建设、RISC-V 适配等领域有关键突破。生态上，阿里云作为核心理事单位参与承办2025 龙蜥操作系统大会，并带头启动龙腾计划 3.0–– “AI 引擎生态加速合作计划”，共同牵头成立“龙蜥社区智算联盟”，推动龙蜥系操作系统装机量突破 1000 万、市场份额近 50%。商业上，推动龙蜥社区衍生版 Alibaba Cloud Linux 应用于交通、金融等重点行业场景，2025 年已在中华财险、极氪汽车、小鹏汽车、地平线等行业领军企业的实际业务中落地。

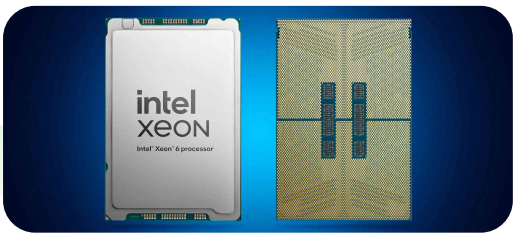
4.3 “潮”“蜥”共振 筑基智算

浪潮信息是全球领先的 IT 基础设施产品、方案和服务提供商，秉承“计算力就是生产力，智算力就是创新力”的理念，致力于推动智慧计算技术创新和应用。

浪潮信息作为龙蜥社区副理事长单位，依托浪潮信息-龙蜥联合实验室，从技术创新、生态共建、运营推广、人才培养等多个维度推动操作系统技术生态的发展，获颁龙蜥社区 “最佳合作伙伴奖”、“最佳联合解决方案奖”、开放原子校源行“优秀合作伙伴奖”等多项荣誉。

技术上，浪潮信息深度参与社区版本的迭代发布，贡献 Mooncake、KSecure 等多个开源项目，牵头编写和制定多项团体标准及社区规范，持续引领行业技术实践与标准化落地。基于龙蜥路线升级的浪潮信息云峦服务器操作系统 KeyarchOS，实现大模型运行吞吐量、GPU 带宽使用率大幅提升，且成为国内首个通过 EAL5+认证的服务器操作系统产品。生态上，浪潮信息牵头成立龙蜥智算联盟，联合十余家国内外主流 AI 芯片厂商打造智算基础设施生态；推动“一测多证”标准体系建设，其 KOS 5.8 SP2 U1 版本率先通过认证。人才培养上，浪潮信息结合自身软硬优势及十余年操作系统开发服务经验，推出初级、中级认证课程，并联合大连理工开发了面向下一代操作系统的培训课程，覆盖全国约 150 家企事业单位及高校。

4.4 “芯”有灵“蜥”软硬合璧



2025 年，Intel 持续深度参与龙蜥社区的核心研发与生态建设工作。面对数据中心与云基础设施场景的飞速发展，Intel 紧密围绕新一代至强平台使能、操作系统稳定性优化以及编译工具链支持三大核心维度，大力推动国产开源操作系统在千行百业的落地与规模化应用。

过去一年，Intel 团队在龙蜥社区的贡献结出了丰硕成果。我们全面支持了两大新一代至强服务器平台：基于 P-core 的 Granite Rapids（GNR）以及基于 E-core 的 Clearwater Forest（CWF）。在系统版本迭代上，Intel深度参与并支持了 3 个重要操作系统版本的发布，即 Anolis OS 8.10 官宣支持 Intel GNR 平台；Anolis OS 23.3 官宣新增支持 Intel GNR-SP 平台；Anolis OS 23.4 官宣支持了 Intel CWF 平台的核心特性。

在底层代码层面，Intel 参与并支持了 2 个内核版本及 2 个 GCC 工具链版本的开发，全年完成 40 余次代码提交，向社区成功合入 300 多个涵盖 Kernel、QEMU 及 GCC 的关键补丁，技术贡献广泛覆盖内核平台使能、虚拟化能力增强、性能与功耗管理、RAS（可靠性/可用性/可维护性）机制、性能可观测性以及工具链编译支持等核心领域。

通过长期的技术投入，Intel 在龙蜥社区成功打通了从“CPU 到内核再到工具链最后再到操作系统”的完整生态支持链路。未来，Intel 将继续携手龙蜥，在 AI for OS 与 OS for AI 的双轮驱动下深化协同创新，以领先算力与繁荣开源共绘智算时代的产业新蓝图。

4.5 聚力同行 智启未来

中兴通讯是全球领先的综合通信信息解决方案提供商，为全球电信运营商、政企客户及个人消费者提供创新的技术与产品解决方案。在操作系统领域，中兴通讯于 2002 年启动新支点操作系统研发，产品已覆盖服务器、嵌入式、机载、车用、桌面等场景，广泛应用通信、电力、汽车、金融、电子政务等领域。

在 2025 年，中兴通讯基于龙蜥社区正式发行新支点服务器操作系统 V7 全新版本，该版本在技术和应用适配方面实现重大突破，并全面提升系统对 AI 应用的支撑能力，为企业从数字化到智能化的跃进提供坚实基础和强有力支持。新支点服务器操作系统凭借 80% 人力成本降低、95% 业务中断时间缩短、100% 全面迁移成功在运营商实现数万套 HostOS 迁移，凭借该优异的表现，新支点操作系统迁移方案成功入选第四届“鼎新杯”数字化转型应用典型案例。

在过去的一年，新支点服务器操作系统已通过亚马逊云服务认证，并正式上线亚马逊云平台，开启国产操作系统迈向国际云计算版图，为全球用户提供更加多样化、高性能的操作系统选择，也显著提升国产操作系统在国际云计算市场的影响力与服务覆盖能力。

4.6 芯耀龙蜥 潜心砺芯

AMD 是高性能与自适应计算领域的领先企业，致力于提供优质的产品和服务，助力客户解决各种重大的挑战。我们的技术推动着数据中心、嵌入式系统、游戏和 PC 市场迈向未来。AMD 于 1969 年在硅谷创立，最初只有几十名员工，从那时起 AMD 便踏上创新之路，致力于引领半导体产品领域的最前沿。如今，AMD 已经成长为一家现代化的全球性企业，凭借先进技术和诸多突破性行业创新，树立现代计算新标杆。

2025 年，AMD 与龙蜥社区保持了持续而深入的合作，在内核演进、平台使能和生态共建等方面取得了扎实进展。围绕龙蜥 6.6 内核，AMD 系统性引入并完善了 Bus Lock Trap、AMD Pstate、INVLPGB、perf IBS、UMC 以及 Prefetch 等一系列关键补丁，全面使能并增强了第四代与第五代 AMD EPYC 处理器的多项高级硬件特性，为新一代 EPYC 计算平台提供了稳定、可靠的软件基础。在此过程中，AMD 还对 SEV-SNP 相关补丁进行了多轮迭代与重构，并补充引入多项最新缺陷修复，显著提升了平台在虚拟化与安全方向的能力，进一步增强了龙蜥操作系统在云与企业级场景下的可用性与可信度。

与龙蜥操作系统深度适配的第五代 AMD EPYC（霄龙）处理器已在阿里云正式发布，用户可直接在阿里云平台上开启体验。该 CPU 平台在龙蜥内核支持下，充分释放 AMD EPYC CPU 在性能、能效以及安全特性方面的优势，为云计算与企业级应用提供了成熟、可靠的算力选择。

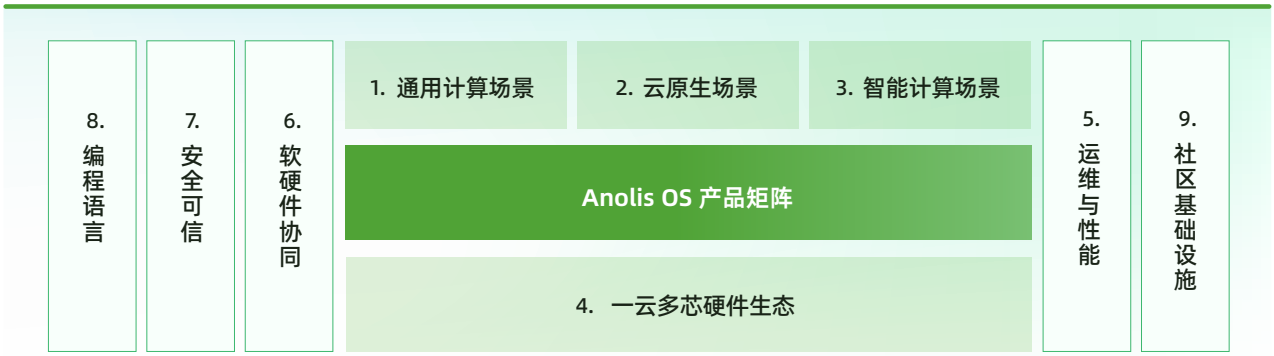
5. 社区技术演进

服务器操作系统作为计算产业生态承上启下的关键环节，正处于从"云原生"向"云原生+AI 原生"深度融合的历史性跨越。在 AI 技术爆发式发展的推动下，操作系统正从传统资源管理者演进为具备智能感知、自适应调度与安全可信能力的算力中枢。龙蜥社区坚持"开源、开放、共建、共享"理念，通过技术创新与生态协同，构建安全、稳定、高效的操作系统底座，并系统推进 AI 原生操作系统建设。截至 2025 年 11 月，龙蜥操作系统累计装机量突破 1000 万套。据**《国产服务器操作系统发展报告（2025）》**显示，龙蜥生态相关社区版和商业版的市场占比近 50%，用户愿意迁移到 OpenAnolis 生态社区版和商业版的比例达 54.25%，位居行业首位。

在通用算力领域，社区持续完善内核统一规范与跨架构适配能力，为 AI 基础设施提供坚实底座。在智能算力领域，龙蜥社区全面践行"System for AI"与"AI for System"双轮驱动战略：一方面，通过 OpenAnolis Confidential AI 机密计算解决方案、Confidential MaaS 可验证推理环境技术指南，构建硬件级 TEE 保护、远程证明与端到端加密的可信 AI 推理框架。**在 AI 多算力支持方面**，已开展适配国际、国内主流 GPU、NPU、AI 加速芯片生态，推动"一云多芯"异构算力统一调度规划建设；**在模型训练基础设施方面**，社区优化分布式训练的网络通信与存储性能，并持续完善 AI 容器商业化参考实现；**在推理优化方面**，提供多元异构算力商业参考实现，显著提升推理效率与资源利用率；**在 Agent 基础设施方面**，构建涵盖资源约束、强隔离执行、最小权限控制、可观测性与可恢复性的受控工作负载能力体系，为智能体应用提供稳定性、安全性与可运维性保障。另一方面，将 AI 能力深度内生于操作系统，推出或增强 OS Copilot 智能运维助手、SysAK AI 负载性能画像、Coolbpf 智能诊断等创新工具，实现系统自我感知、自我优化与风险预判。2025 年，社区正式启动"龙腾计划 3.0——AI 引擎生态加速合作计划"，联合 1000 余家合作伙伴共建开放创新的 AI 引擎生态，系统性推进操作系统与 AI 技术的融合演进。

5.1 社区技术布局

2025年，龙蜥社区围绕技术委员会（TC）确立的技术路线，延续九大技术方向的整体布局，深化九大技术方向的协同演进。Anolis OS 23.4 版本发布，完成对 RISC-V 架构的正式支持；Anolis OS 8.10 版本聚焦企业级稳定性和行业适配，实现软硬协同深度优化。这两大版本协同满足从传统企业到云原生、AI 场景的多元化需求。社区通过统一内核接口规范（KABI/KCONFIG）与 300+ 项驱动兼容性基线，保障各商业发行版的技术一致性与生态兼容性。九大技术方向由四个横向的技术场景方向和五个纵向的技术领域方向组成，如下图所示：



图/龙蜥社区技术布局：九大技术方向

5.1.1 一云多芯异构算力支持

- 国产芯片：Anolis OS 23.4 完成基础运行环境建立、IOMMU 初步适配、中断控制器适配、ACPI 底层兼容及 PCIe 总线接口适配；全面支持龙芯 3E6000 平台，虚拟化环境下的 AVEC 中断控制器支持，实现中断透传与高效虚拟化；深度优化海光四号平台，完成 CSV3 安全虚拟化技术中的共享内存页面管理机制增强，提升多租户场景下的内存隔离安全性。更进一步地，还全面支持了兆芯 KH-50000、鲲鹏 920 v200、飞腾 S5000C、申威 WX-H8000 等最新国产芯片平台。
- 国际芯片：Anolis OS 23.4 全面支持 Intel Granite Rapids-SP 平台与 Clearwater Forest E-core平台，以及 AMD 第五代 EPYC 9005（Turin）平台。Intel GNR/CWF 平台方面，完成 TDX Gen 2.0 安全虚拟化增强（支持 2048 独立加密密钥、AES-256 加密强度），实现 DSA 2.0/IAA 2.0/QAT 加速器全面适配，支持 Intel SST 电源与性能精细化管理，并通过 AMX FP16 指令提升 AI 推理性能。AMD Turin 平台方面，完成 Zen5 架构原生 AVX-512 指令集适配（512 位 FPU 单元直接执行），首次支持 SDCI 智能数据缓存注入技术显著提升 I/O 性能，实现 SEV/SEV-ES 机密计算能力，完善 PQoS 资源隔离与 BMEC/ABMC 带宽监控能力。
- RISC-V 突破：发布 Anolis OS RISC-V 正式版，基于 ANCK 6.6-004.1.1 内核，完成对阿里达摩院 C910 架构与算能 SG2042 服务器芯片的适配，支持 RISC-V AIA 扩展、SG2042 多核架构及核心驱动（MSI、MMC 等），同时支持 UEFI/ACPI/GRUB 启动机制。此外，社区有 20+ 位 RISC-V 国际基金会核心成员，其中多位专家担任 Chair/Vice-Chair 角色，共同推进 RVA23 Server SoC 标准建设。

5.1.2 软硬件协同

服务器架构在云计算时代从以 CPU 为中心走向 DSA、XPU 等异构架构。云计算在实现业务与基础设施解耦的同时，也为软硬件全栈协同提供了空间。社区在此领域推动了 CIPU、DPU 等相关软硬件协同标准的制定，实现了 VIRTIO、XQUIC、SMC-R 等接口或协议技术在社区伙伴业务上的规模化部署使用，并积极布局全链路的系统软件栈编译优化技术，在提升软硬件协同技术竞争力方面做出了有益探索。

- 内核优化：ANCK 6.6-003 版本完成了 EEVDF 特性调度特性的 backport，支持 task 和 cgroup 自定义 slice；内存子系统增强包括页表页绑核分配、内存隔离、THP 优化、代码段锁定等特性；网络栈优化 virtio-net 控制队列超时机制，提升 UDP 有连接接收性能；存储方面优化 cgroup writeback，支持 uncached buffer IO，显著提升大压力下 buffer IO 性能。ANCK 6.6-004 内核版本增强内存子系统，支持快速 OOM、页表页回收、slab 无锁收缩、异步 fork 等特性；ANCK 5.10-019 内核优化调度器，新增集群调度特性。上述两大版本均增强网络栈，集成社区 SMC 自研补丁（含 eRDMA 支持、SMCv2 支持、CQ 优化等）。
- 虚拟化增强：virtio-blk 驱动引入 io_uring passthrough 特性，通过绕过通用块层减少 I/O 路径开销；新增 ring_pair 特性，利用两个 virtio_ring 队列实现请求通道，提升高后端延迟与突发读写场景的吞吐能力。
- 存储优化：ext4 文件系统支持 large folio 实验性特性，通过 iomap 方案显著提升 buffered IO 性能；完成 EROFS、ext4、nfs 等社区稳定补丁同步。
- CIPU 协同：推动 VIRTIO-FS、XQUIC 3.0、SMC-R 2.0 等协议在云环境中的标准化部署，显著降低网络延迟并提升吞吐量。
- 驱动生态：ANCK 6.6-003 版本 OOT 驱动基线新增天翼存储 sxe/ps3stor、云脉芯联 xscale、海思 hns3/hinic、网迅 txgbe/ngbe 等多款网卡驱动，扩展硬件兼容性。

5.1.3 业务平滑迁移

用户在升级底层操作系统、更新基础设施过程中，会涉及自身应用的迁移。提供低门槛的迁移能力是 Anolis OS 在建设之初就确定的基本原则。为此社区专门成立了迁移工具 SIG 组来负责与用户业务平滑迁移相关的开源项目。当前，龙蜥社区已经发布了业务迁移相关的手册与工具 Anolis OS Migration Solutions (AOMS)。用户按照迁移手册的步骤或通过 AMOS 工具即可平滑完成业务迁移的相关工作。

- CentOS 替代支持：持续运营 Anolis OS 7 延保（ELS）服务，为存量用户提供安全更新；Anolis OS 8.10 提供生态兼容的环境，无缝接管CentOS 8停服后的业务需求。
- 迁移工具：Anolis OS Migration Solutions（AOMS）工具链完成 2.0 版本升级，支持应用依赖智能分析与配置自动迁移，在金融、政务等领域实现规模化应用，将迁移周期从周级缩短至小时级。
- 认证体系创新：“一测多证”认证体系全面落地，通过硬件兼容性互操作认证机制，使通过 Ancert 测试的衍生 OS 可直接进入龙蜥硬件兼容性列表，大幅提升生态认证效率。浪潮 KOS 5.8 SP2 已作为标杆案例成功完成认证。

5.1.4 保障业务稳定可靠运行

用户业务的稳定可靠运行是基础设施的核心使命。龙蜥社区持续沉淀百万级服务器生产环境的运维实践，构建了体系化的系统稳定性保障技术栈。在社区伙伴的共同贡献下，系统运维 SIG 与 eBPF SIG 持续推动运维技术创新，发展出 SysOM、SysAK、Coolbpf 等关键运维项目。SysOM 2.0 作为龙蜥社区一站式智能运维平台，集成 OS Copilot 智能助手，通过统一 Web 界面提供主机管理、系统监控、异常诊断、日志审计与安全管控等全栈能力；其深度诊断引擎 SysAK 源自大规模生产实践，新增 AI 负载性能画像功能，可精准识别业务瓶颈与资源热点。Coolbpf 创新提出远程编译架构，显著降低 eBPF 应用开发门槛，实现同一应用在 3.x/4.x/5.x/6.x 跨内核版本的无缝安全运行。

当前，龙蜥社区已形成以 Anolis OS 为核心、覆盖传统+云原生 +AI 原生场景的操作系统发行版生态矩阵。基于 Anolis OS，社区伙伴发布了包括 Alibaba Cloud Linux 4、统信服务器操作系统、银河麒麟高级服务器操作系统、中科方德服务器操作系统、浪潮信息 KeyarchOS V5.8 SP2、中兴新支点服务器操作系统、中国移动 BC-Linux、凝思安全操作系统、新华三磐宁服务器操作系统、长擎安全操作系统、LifseaOS 等在内的数十款商业发行版和社区版，广泛应用于政务、金融、电信、能源等关键行业。Anolis OS 23.3 通过严格安全供应链选型管理，结合精细化的维护策略与全链路质量保障机制，为用户提供安全、稳定、可靠的操作系统底座，持续筑牢业务连续性基石。

- 安全增强：OpenAnolis Confidential AI 1.0 正式发布，提供基于 Intel TDX、海光 CSV 等可信硬件的机密计算 AI 解决方案，实现模型与数据的全生命周期保护；发布 Confidential MaaS 技术指南，构建可验证的 AI 推理环境。
- 语言生态：Anolis OS 8.10 集成 Dragonwell 8/11/17/21 Java 组件，可替代开源 OpenJDK 系列；引入 python3.11 兼容包，在保持系统 3.6 主版本稳定性的同时，扩展 python 生态兼容。
- 运维工具：SysAK 在百万级服务器生产环境中沉淀经验，新增 AI 负载性能画像功能；Coolbpf 完成对 eBPF v2 特性的全面支持，实现跨内核版本（3.x/4.x/5.x6.x）的无缝运行。
- 桌面体验：Anolis OS 8.10 新增基于 QT 的超融合轻量级桌面环境——新支点（NewStart）桌面，提升终端用户体验。

5.2 技术生态与产业协作

龙蜥社区致力于构建开放协同、标准互认的技术生态与产业协作体系。在软件生态层面，Anolis OS 23.3 提供 2500 余款基础软件包，Anolis OS 8.10 新增超 200 个软件包，全面支撑云原生、大数据与AI原生应用场景；在“开源生态发展合作倡议”框架下推动国内外开源社区深度协作，在 Linux 内核规范、GLIBC 、GCC 、LLVM 等核心组件、500+ 核心软件包选型上达成技术对齐，有效降低生态碎片化与重复建设。安全方面，依托龙蜥社区安全联盟（OASA）建立 CVECenter 漏洞管理系统，构建从检测、分析到修复的全链路安全响应机制。同时，社区专家积极参与RISC-V国际基金会数据中心 SIG、云原生计算基金会（CNCF）、Linux 基金会等国际组织的标准制定工作，主导 RISC-V RVA23 Server SoC 规范建设，推动中国技术方案融入全球开源生态，实现“上游共建、下游繁荣”的可持续发展新格局。

- 软件生态：Anolis OS 23.3 提供 2500+ 基础软件包，集成 QEMU、libvirt 等虚拟化工具及 Java 17 企业级开发环境；Anolis OS 8.10 新增超 200 个软件包，强化大数据和 AI 应用场景支持。
- 跨社区协作：协同五家社区一起在 Linux 6.6 LTS 内核规范、GLIBC 2.38、GCC 13、LLVM 18 等核心组件上达成技术对齐，推动超过 500 个核心软件包选型共识，在保证差异化创新同时，有效降低生态碎片化与上中下游兼容适配成本。
- 安全生态：依托龙蜥社区安全联盟（OASA），增强 CVECenter 漏洞管理系统，建立从检测、分析到修复的全链路安全响应机制。
- 国际标准参与：社区专家主导 RISC-V 国际基金会数据中心 SIG 工作，推进 RAS/PMU 云方案增强、参与 AIOE 扩展及虚拟化标准制定；在云原生计算基金会（CNCF）、Linux 基金会等国际组织中积极贡献。

5.3 社区重点技术方向运营

- 两大产品线协同演进
  - Anolis OS 23.3：面向下一代云原生与 AI 场景，完成 RISC-V 预览支持，集成 ANCK 6.6-004 内核，优化内存子系统与调度器。
  - Anolis OS 8.10：面向企业级稳定需求，集成 ANCK 5.10-019 内核，提供 RHCK/ANCK 双内核选择，完成软硬协同深度优化。
  - 商业发行版矩阵：统信服务器操作系统、中科方德服务器操作系统、浪潮（云峦 V5）、中兴（新支点）、阿里云（Alibaba Cloud Linux 4）等基于两大产品线发布商业版本，覆盖政务、金融、电信、能源等核心行业。
- CentOS 替代与存量治理
  - ELS 延保服务：持续为 Anolis OS 7 提供延保服务，确保 CentOS 7 停服后业务连续性。
  - 认证创新："一测多证"认证流程已建立，浪潮 KOS 5.8 SP2 成功完成标杆认证，降低硬件兼容性认证门槛。
  - 规模迁移：2025 年，龙蜥社区持续深化 CentOS 替代工作，通过 AOMS 工具链与商业迁移服务双轨推进，在金融、电信等关键行业实现规模化迁移部署。依托累计超 1000 万套的装机规模与“一测多证”认证体系，有效支撑用户业务平稳过渡，显著降低 CentOS 停服带来的安全风险。

- AI 原生技术战略
  - Confidential AI：OpenAnolis Confidential AI 1.0 提供完整的机密计算 AI 解决方案，支持 Intel TDX、海光 CSV 保护的 Qwen 等大模型部署。
  - Confidential MaaS：发布技术指南，构建从 0 到 1 的可验证 AI 推理环境，通过硬件级 TEE、远程证明、端到端加密等技术，重塑 AI 服务信任逻辑。
  - 龙腾计划 3.0：启动"龙腾计划 3.0——AI 引擎生态加速合作计划"，构建开放创新的 AI 引擎生态。
  - OS Copilot：构建操作系统智能体，已在阿里云智能基础设施中实现试点部署，探索自然语言运维新范式。

6. 原生技术概览

6.1 通用计算场景

6.1.1 可预期的发行版路线图和基础能力

6.1.1.1 概述

关于 Anolis OS

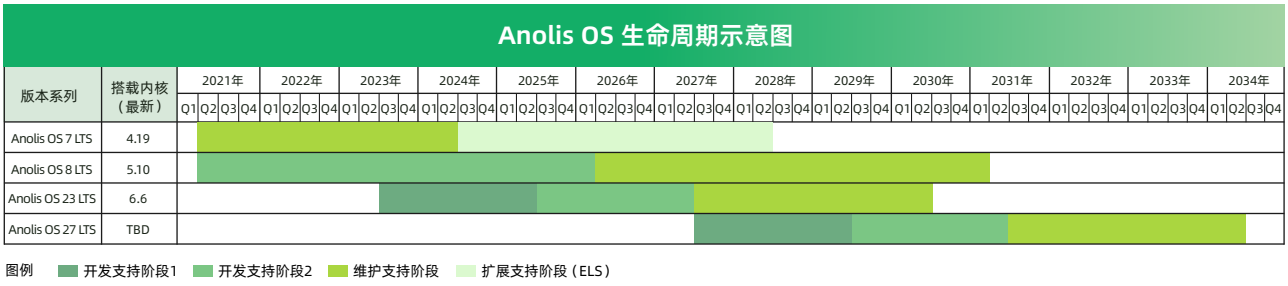
龙蜥操作系统 Anolis OS 是 OpenAnolis 龙蜥社区发行的开源 Linux 发行版，兼容主流 Linux 发行版操作习惯，支持多计算架构，提供稳定、高性能、安全、可靠的操作系统服务。当前 Anolis OS 提供 Anolis OS 7、Anolis OS 8 和 Anolis OS 23 三个主要版本。三个主要版本均集成 ANCK 内核版本（Anolis Cloud Kernel，Anolis 云内核），ANCK 是由社区 Cloud Kernel SIG 组基于上游 Kernel 研发，一款定制优化版的内核产品，在 Cloud Kernel 中实现了若干针对云基础设施和产品而优化的特性和改进功能，旨在提高云端和云下客户的使用体验。与其他 Linux 内核产品类似，Cloud Kernel 理论上可以运行于几乎所有常见的 Linux 发行版中。

关于 Anolis OS 生命周期

版本维护能力是操作系统提供服务的重要能力之一，龙蜥社区提供了 Anolis OS 各版本的维护生命周期，以及生命周期各个维护阶段所提供的服务支持能力。更多关于龙蜥社区生命周期涉及术语解释可通过官网了解：

https://docs.openanolis.cn/document/detail/sywixmzz

6.1.1.2 全版本生命周期一览



- Anolis OS 7，结束日期是 2028 年 6 月 30 日；
- Anolis OS 8，LTS 版本， 整体支持年限 10 年 ，分别是为期 5 年的 “开发支持” 阶段和为期 5 年的 “维护支持” 阶段。整体结束的支持日期为 2031 年 4 月 30 日 。

当前未正式发布的版本生命周期支持计划如下：

- Anolis OS 23，LTS 版本， 整体支持年限 7 年，分别是为期 4 年的“开发支持”阶段，和为期 3 年的“维护支持”阶段。已于 2023 年 7 月发布首个版本，并预计将于 2030 年 2 季度结束支持。
- Anolis OS 27，LTS 版本， 整体支持年限 7 年，分别是为期 4 年的“开发支持”阶段，和为期 3 年的“维护支持”阶段。预计 2027 年 2 季度发布，将于 2034 年 2 季度结束支持。

6.1.1.3 小版本生命周期

- Anolis OS 8 从 8.9 小版本起发布间隔将延长为一年。Anolis OS 23 及以后的版本的小版本发布间隔计划为半年。
- Anolis OS 8.8 之前发布小版本，只发布偶数版本（仅偶数小版本发布镜像）。Anolis OS 8.8 及以后版本，小版本号恢复依次递增。

Anolis OS 8 版本排期：

Anolis OS 8 小版本路线图

小版本号	维护起止时间	2021年				2022年				2023年				2024年				2025年				2026年				2027年				2028年				2029年				2030年				2031年			
		Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Anolis OS 8.2	2021.05.10-2022.03.31																																											
Anolis OS 8.4	2021.07.01-2022.06.30																																											
Anolis OS 8.6	2022.07.01-2023.03.30																																											
Anolis OS 8.8	2023.03.31-2024.04.18																																											
Anolis OS 8.9	2024.04.19-2025.04.30																																											
Anolis OS 8.10	2025.04.30-2031.03.30																																											

图例 开发支持 维护支持

Anolis OS 23 版本排期：

Anolis OS 23 小版本路线图

小版本号	维护起止时间	2023年				2024年				2025年				2026年				2027年				2028年				2029年				2030年				2031年			
		Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Anolis OS 23.0	2023.07.01-2024.06.05																																				
Anolis OS 23.1	2024.06.05-2024.12.31																																				
Anolis OS 23.2	2025.01.01-2025.06.30																																				
Anolis OS 23.3 (原25.0)	2025.07.01-2025.12.31																																				
Anolis OS 23.4 (原25.1)	2026.01.01-2026.06.30																																				
Anolis OS 23.5(原25.2)	2026.07.01-2026.12.31																																				
Anolis OS 23.6(原25.3)	2027.01.01-2030.06.30																																				

图例 开发支持1 开发支持2 维护支持

说明：开发支持1阶段，不承诺 ABI 兼容，开发支持2阶段，会有相应的 API/ABI 兼容规范，为生产级支持。

6.1.1.4 下一个版本特性概述

Anolis OS 23 后续预计发布的大颗粒特性

- 海光新平台支持。
- 鲲鹏 950 系列平台支持。
- 兆芯新平台支持。

- AMD firerange 平台支持。
- 飞腾新平台适配。
- RISC-V 架构引入及同源异构。
- 海光 CSV 及机密容器安全特性增强。
- 国产 OS 组件能力（基础 OS 能力、桌面、办公）增强。
- 智算 modelscope 生态支持。
- 智算 AMD 生态方案支持。

6.1.2 基于上游独立演进的内核基础能力

6.1.2.1 架构选型原则

在内核选型策略上，龙蜥社区充分考量了多元化的算力需求：一方面全面支持 Intel GNR、AMD Turin 等国际下一代主力机型；另一方面深度适配飞腾、海光、鲲鹏、龙芯、申威等国产主流芯片，并积极拥抱基于 RISC-V 架构的国产处理器生态。

社区通过与国内外主流厂商及合作伙伴（包括但不限于芯片、操作系统、云计算、服务器等厂商）紧密协同，最终确立了 ANCK 的演进路线：选定上游 Linux LTS 稳定分支作为基线内核，在此基础上，通过定期回合上游主线更新、引入自研创新特性以及修复关键缺陷等方式独立演进。

6.1.2.2 ANCK 6.6 重要自研/回合特性如下：

调度

- HT aware quota 技术：在任务使用 core sched 特性的情况下，通过感知 HT 对端是否空闲来控制消耗 cfs quota 的速率，使任务在每个调度周期执行的指令数量大致相同，从而达到可预期算力交付的效果，适用于计算型任务。
- CPU 动态隔离技术：将不同 CPU 核心或 CPU 集群分配给不同任务，可避免任务间对 CPU 资源的相互竞争，提高系统性能和稳定性。该功能支持系统运行时动态更改 CPU 隔离配置，适应运行时关键任务变化场景，提升 CPU 资源利用率。
- CPU burst 功能增强：解除可配置的 burst 池上限限制，允许对应 cgroup 累积更多历史资源，灵活度更大。
- sched sli、富容器：采集并统计容器级别的 CPU 使用率，loadavg，调度延迟等指标，提供了容器资源视图功能的相关接口，实现对容器资源的可见性增强，可以用于评估容器的负载及资源使用情况。
- 细粒度优先级：提供更细粒度的抢占优先级，当唤醒优先级高的任务时，可以更容易抢占优先级低的任务，在混部场景下能降低高优先级任务的长尾延迟。
- EEVDF：基于“虚拟截止时间”（virtual deadline）进行调度，结合了公平性和响应性，相比于原来的 cfs，在保证吞吐不变的同时，让延迟敏感的任务能更快地响应。
- sched ext：新增 ext 调度类，优先级介于 fair 和 idle 之间，通过 ebpf 程序实现调度策略，为不同的应用和业务定制最优的调度器。

- psi 增强
  - 支持 per cgroup psi 统计开关。
  - 支持 irq/softirq 负载统计。
  - 支持非特权 poll psi event。
- load balance 优化
  - 引入 SIS_UTIL 机制，基于 LLC 域内 CPU 的 util_avg 总和 动态限制 select_idle_cpu() 的扫描范围，避免在系统高负载时过度扫描，显著降低调度开销与 rq 锁竞争，在高并发场景（如 224 线程 netperf）下提升性能。
  - 修复 NUMA 负载均衡 中关于允许不平衡判断的不一致性问题，统一 find_idlest_group、find_busiest_group 与 task_numa_find_cpu 的计算逻辑，提升 NUMA 迁移决策的准确性。
- cgroup v2 cpuset 隔离完善
  - 新增 isolated 模式，相比于 root 模式，isolated 在 cpuset 范围内不会负载均衡，更适合 dpdk 等需要独占隔离核的负载。
  - 新增cpuset.cpus.exclusive 接口，确保某些 cpu 不会同时分配给多个子 cgroup。
- group identity 2.0：基于社区标准的 core scheduling 和 cgroup idle 特性，演进出 Group Identity 2.0 的全新设计。对负载均衡策略进行优化，在混部场景中，可提高在线任务的性能，避免离线任务饿死。
- Group Balancer 2.0：通过对拓扑结构的感知将整机 cpu 划分为若干分区，通过对 cgroup 规格和负载的感知将 cgroup 分配到一个合适的分区，减少跨 numa 访存以及 cache miss，提高任务的亲和性，从而提高任务性能。
- jbd2 代理执行：解决 jbd2 锁导致的优先级反转问题。

内存

- mTHP 支持：随着业务内存使用越来越大的趋势，使用 mTHP 来管理内存，可以在兼顾更好的业务生态的基础上，让应用更好的享受大内存带来的性能优势。
  - vmalloc 支持 huge mapping：允许 vmalloc() 内存分配支持大页分配和建立大页映射，降低 TLB miss，提升性能。
  - PCP 支持 large order：允许 PCP 缓存 large order，提升 large order 内存分配和释放性能。
  - madvise() 支持 MADV_COLLAPSE：扩展 madvise() 支持新的特性 MADV_COLLAPSE，允许用户空间在进程上下文中将符合条件的内存范围合并为 THP ，提升业务性能。
  - 文件系统支持 large folio：文件页的分配支持不同 size 的 large folio（不仅仅是 2M THP），更好的提升文件系统性能。
  - 匿名页支持 mTHP：将匿名页的分配支持 mTHP 分配，即不同 size 的 large folio，而不仅仅局限在 PMD-sized THP。这样匿名页的使用也更有弹性，并且可以利用一些硬件特性（比如 Arm contiguous PTE）来做一些软硬协同的优化。
  - shmem 支持 mTHP：将 shmem 内存分配和管理也扩展到 large folio，让 shmem 内存分配跟匿名页 large folio 分配保持了一致。

- tmpfs 支持 mTHP：扩展 tmpfs 文件系统支持不同 size 的 large folio 分配和管理，提升大页分配率并有效利用硬件特性优化性能。
- Arm 架构 large folio 软硬协同：对 ARM64 软硬协同的优化，在分配 large folio 时，建立映射时可以自动设置 contiguous PTE 标志来优化 TLB miss（TLB coalescence），当 PTE 有变化（比如 unmmap，zap，migration 等）会自动清除 contiguous PTE 标志。这让 Arm 的 contiguous PTE 特性可以透明的为内存管理使用，让用户无需关心底层硬件的细节。
- 内存规整支持 mTHP：通过该特性可以更好的进行 large order 的内存规整，避免系统碎片化。
- fork 支持 batch PTE：同样为了提升 large folio 解除映射时通过 PTE batch 进行性能优化。
- mlock 支持 mTHP：使能 mlock() 系统调用对 mTHP 的支持。
- brk() 支持 THP 对齐： 通过 brk() 分配堆内存时，可以提升 large folio 的分配率，提升业务性能。
- mTHP collapse 支持：对于一些场景无法分配出 large folio（比如：COW），该特性支持后台主动按需合并 large folio，提升 large folio 的使用率和业务性能。

- Lock Scalability 优化：
  - per-memcg LRU lock：将全局的 LRU 锁拆分到每个 memcg，从而提高内存回收的并发性。
  - per-VMA lock：细化 mmap_lock 粒度，按 VMA 粒度进行加锁。细化了 mmap_lock 的粒度，通过对每个 VMA 进行单独加锁，提升了并发访问的性能。
  - slab lockless shrink：使用 RCU 和 refcount 机制实现无锁的 slab 回收，提升slab回收并发性能。
  - Auto PCP：自动调节 per-CPU 页面缓存的容量，以适应不同的负载和工作负载，避免 zone lock 竞争，提升内存分配和释放并发性能。
- 内存成本优化：
  - HVO（HugeTLB vmemmap optimization）：通过释放 hugetlb 大页中不使用的 tail page struct，释放出 hugetlb 不用的元数据，可以节省内存（1T 大小的 1G hugetlb，可以节省 16G 内存）。
  - DAMON 机制（Data Access Monitor）：DAMON 的核心机制是“基于区间的采样”（region-based sampling）和“自适应区间调整”（adaptive regions adjustment），针对实际数据访问模式优化内存管理机制，比如主动回收冷内存等。
  - MGLRU 与 MGLRU 冷内存回收：MGLRU 通过实现多级 LRU，根据访问频率和活跃度进行管理，从而更准确地识别出不常用的页面，优化页面回收，并能在内存压力下提高性能。基于 MGLRU 对内存主动进行 aging（老化），并进行冷内存主动回收，更加高效和轻量级。
  - 页表页回收：会尝试在 MADV_DONTNEED 路径上回收页表页，以达到节省内存，避免提前进入 OOM 等问题。
  - Idlemd 冷内存回收：基于内核 kidled 内存扫描获取内存冷热信息，配合内核回收模块进行主动冷内存回收。
  - 低功耗容器内存回收：基于冷内存回收，扩展了强制内存回收模式，允许业务在不扫描 page age 的情况下强行回收匿名页和文件页，从而在潮汐混部场景业务容器不活跃时强行回收容器内存。

- tmpfs 零页填充：通过在 tmpfs 稀疏文件读/写错误分配页时将实际的物理页替换为系统零页，系统零页被复用到所有的文件空洞上，不占用额外内存，避免浪费内存。

- 内存软硬协同优化：
  - 内存分层（Memory tiering）：支持异构系统中存在 CXL 内存或者 PMEM 内存时，通过冷热分层机制对进程的冷热内存自动管理，提升性能。
  - Numa balancing 支持大页：NUMA balancing 支持 large folio，让业务尽量访问本地的 large folio，提升业务性能。
  - 代码段多副本（duptext）：通过代码多副本功能，可以将远程节点的代码段复制到本地节点，避免了跨节点访问，从而解决 NUMA 架构中因跨节点访问带来的性能延迟问题。
  - 页表页分配 NUMA aware：在页表页分配时，让页表页尽量分配在进程所在的 NUMA node 上，避免跨 NUMA 访问页表页带来性能问题。
- 内存 Debug 增强：
  - Kfence 增强：Kfence 功能进行了增强，支持灵活的动态开关 kfence 以及全面捕获内存污染问题，从而兼顾了线上探测与线下调试的需求。
  - OOT内存隔离：OOT 模块内存隔离主要面向使用未经充分测试和验证的 OOT 模块的业务场景，在 OOT 模块因为缺陷发生内存越界、UAF 等故障时，可以快速对问题进行定界，减少排查和修复所需要的时间，降低业务损失。
- 内存 QoS 增强：
  - 引入 per-memcg proactive reclaim 接口：为 memcg 提供用户主动内存回收的接口，这种方式能够更准确实时地估算工作集，因为 LRU列表会持续进行排序，从而可能提供更确定性的内存过度分配行为。
  - Memcg SLI 增强：为 memcg 增加更多的时延统计帮助分析和定位问题。
  - Memcg 全局最低水位线分级：当时延敏感型业务和资源消耗型任务共同部署时，允许不同的 memcg 设置不同的全局最低水位线，避免引起时延敏感型业务的性能抖动。
  - Memcg 后台异步回收：memcg 后台异步回收特性通过后台运行，避免业务上下文陷入直接内存回收，避免影响时延敏感型业务。
  - Memcg OOM 优先级：Memcg OOM 优先级策略可允许用户通过优先级配置来区分不同 cgroup 的 OOM 顺序，避免高优先级在线业务被杀死。
  - 僵尸 memcg 回收：zombie memcg reaper 提供了同步回收和后台回收 zombie memcg 的功能，避免僵尸 memcg 堆积对系统性能产生影响。
  - Memcg pagecache 限制和预留：Page cache 预留和限制特性主要面向内存使用量大，频繁触发 pagecache 回收的场景，在保证性能和稳定性的情况下提升内存等资源的利用率。
  - 快速 OOM：解决 OOM 时内存回收时间过长导致的整机夯死，进而造成对其它高优先级 memcg 的影响。

其他：

- Async fork：通过异步拷贝页表，提升 fork() 系统调用性能。
- 代码段锁定：锁定重要的代码段内存，避免内存颠簸带来的性能抖动问题。

存储

- io_uring
  - io_uring percpu sqthread polling：支持多个 io_uring 实例共享同一个 sqthread，优化 cpu 开销并降低上下文切换延时。
  - io-wq 优先级优化（Prioritized work completions）：优化 io_uring io-wq 机制，提升性能并优化时延。
  - 支持注册 ring fd（IORING_REGISTER_RING_FD）：多线程共享 file table 场景下，避免每次调用 io_uring_enter(2) 时获取和释放 ring fd 带来的原子引用计数开销。
  - Fast poll multishot 模式：支持 accept（IORING_ACCEPT_MULTISHOT）、recvmsg、timeout 等操作的 multishot 模式，用户只需提交一次请求，每当有数据到达时就会收到相应的完成事件，multishot 批量完成优化后 RPS 提升 2.3x。
  - 异步缓冲写（Async buffered writes）：在 fast path 增加 buffered 写支持，初始支持 xfs 文件系统，测试数据显示可带来约 3x 性能提升。
  - 网络零拷贝发送（Zerocopy send/sendmsg）：支持 registered buffers 和 normal buffers 的零拷贝发送，通过 io_uring 完成队列通知用户态 buffers 何时可重用。
  - Task work 锁优化：将 task work 的 spin lock 改为 lockless list，多线程环境下 task_work_add 性能提升约 20%。
  - SQ/CQ 大页支持：支持将 io_uring ring 放置于应用程序预分配的大页中，降低对连续普通页的依赖，减少 TLB 压力，提升大 ring 结构的性能。
  - 异步取消增强（IORING_ASYNC_CANCEL_OP）：支持基于原始请求的操作码进行匹配取消，增强取消操作的灵活性和精准度。
  - SQ/CQ 优化：使用 io_uring nop 测试能带来约 9% 的性能优化。
- ublk 用户态块驱动：基于 io_uring passthrough 机制的高性能用户态块设备，驱动程序负责将来自 ublk 块设备的 I/O 请求传递到用户态 ublk server，所有 I/O 逻辑由 ublk server 完成，内核驱动保持简单。
- erofs
  - 支持 subpage block，可应用于容器镜像场景中 erofs 直接索引容器镜像的 tar 包，省去容器镜像生命周期管理中 tar 包的 untar 以及清理流程，从而提升性能和稳定性，并提升容器镜像作为 golden image 的安全性。
- xfs
  - 支持 dax reflink。
- ext4
  - Ext4 large folio 支持：基于 buffer head 方案实现 Ext4 large folio 支持，通过 large folio 提升 LRU 管理效率，减少 page fault 次数，以 large folio 粒度建立大页映射可降低 page table walk 开销和 TLB miss，显著提升 buffered IO 性能。同时保持 4K 基础页的南北向生态兼容，应用可无感使用。
  - 批量按需预留：通过 Ext4 文件系统支持批量按需预留，提升前台 IO 读取/写入 page cache 的性能。
  - large folio 批量处理：在读写流程中支持 large folio 批量处理，预读和回写 IO 按 large folio 组装和迭代，大幅提升 IO 读写性能。fio 测试显示 64k 顺序读/写性能、1M 顺序读/写性能均有较大提升。

- fuse
  - 支持跨 namespace 传递 fuse 挂载点，实现 fuse 挂载点在非特权容器间的传播，提供一种基于 fuse 的远端存储在云原生场景下的解决方案。
  - 支持 fuse daemon 在故障恢复之后，可以请求对故障期间的请求进行重发。
- block
  - cgroup v1 支持 block throttle 限流和 iocost 权重控制。
  - tcmu 支持 bypass tcmu data area 和 zero copy 特性。
  - tcmu_loop 设备支持参数可配置，如 can_queue，nr_hw_queues，cmd_per_lun，sg_tablesize 等，当后端设备能力足够大的时候，调大这些参数能明显提升性能。
- 稳定性
  - 支持 IO hang 检测，通过扩展核心数据结构，提供相关接口导出 IO hang 信息，辅助快速分析和定位 IO hang 原因。

网络

- NAPI kthread 化：支持 NAPI 在 kthread 上 poll，而不是软中断上下文，避免大量抢占进程执行，支持让调度器来调度相应的软中断 kthread，性能无损的同时能实现 NAPI 的灵活调度。
- Skb drop reasons：可以更加清晰地知道 skb 为什么丢包，社区正在为整个网络栈增加越来越多的 drop reason，可以简化排查丢包问题的难度。
- 支持 multi-buffer XDP：通过支持 multi-buffer，让 XDP 能支持 jumbo frame 和 LRO，是使用 XDP 又想要使用 jumbo 的场景（如数据库 ipflt-xdp）的基础能力。
- 支持 tcp_shrink_window：允许 TCP 接收窗口动态缩小，以更严格地遵守内存限制，解决进程不及时收走 socket 上的数据时占用大量内存的问题。
- SO_PREFER_BUSY_POLL：busypoll 对于高性能高吞吐的场景有较为显著的性能提升。
- io_uring tx zero_copy：io_uring 实现发送方向零拷贝，使用方式清晰简单，相对于拷贝方式 4000 字节 UDP 性能和 MSG_ZEROCOPY 8192 字节 UDP 性能大幅提升，大包零拷贝场景较其他方案有优势。
- SO_REUSEPORT 连接处理增强：在进程退出时，将 accept queue 中没有被处理的请求重新分配给其他进程，避免进程退出时未处理的请求被丢弃。
- BPF_MAP_TYPE_USER_RINGBUF：高性能的 user -> kernel 通道，支持从用户态往内核态 BPF 程序吐数据。
- TCP 支持 SYN ACK RTO tunable by BPF：支持通过 eBPF 修改 SYN ACK 的重传超时。
- eBPF 支持触发 panic：某些情况下帮助生成 vmcore 分析异常故障，提升了 eBPF 应用范围。
- 性能优化：
  - IPv4/IPv6 BIG TCP：内核支持更大的 IP jumbo packets，可降低时延和提高吞吐（约 50%），cilium 中已支持并可使用加速单机网络性能。

  - pfifo_fast 性能优化：对于 pktgen 能提升 pfifo_fast 下性能，对用户无感提升，使用 pfifo_fast 且 PPS 较高的场景预期有收益。
  - TCP 新增 per-netns ehash：为每个 netns 创建 ehash，减少共享 global ehash 造成的 bucket 冲突引入的性能回退，高密容器且连接规模大的场景适用，对应用透明。
  - tcp zerocopy receive 性能优化：进一步提升 tcp zerocopy receive 性能，特别是中等大小（如 32K）的请求。
- virtio-net:
  - 完善的 RSS 功能支持；
  - 解决 XDP 重复验证 checksum 问题，提升网卡处理效率。
  - 支持 NetDIM - 动态调节云上网卡中断频率。
  - 支持调整网卡队列大小。
  - 支持网卡硬件收包信息统计，提升网卡运维监控能力。
- 依托 eBPF 的强大能力，提供了基于 eBPF 的 tcprt，vtoa 等功能的支持，相比于内核模块方案，eBPF 方案提供了更强大的能力。
- 支持 SMC 内核网络协议栈：配合硬件 RDMA 技术提供共享内存通信，透明加速 TCP 以获得更低的网络时延和 CPU 使用率。

perf

- 支持倚天 CMN 和 DDR PMU 的 perf metric 功能，可检测倚天710 slc miss rate、跨die流量和访存流量等指标。
- 添加 dwc_rootport PMU 驱动，使 perf 工具可以采集到硬件支持的 PMU 事件，用于分析 PCIe rootport 性能。

驱动

- 支持按照 net、block、gpu 领域的方式进行模块签名校验。
- 网卡驱动，支持 nfp、txgbe、ngbe 国产网卡驱动。
- OOT 驱动基线，当前 6.6 内核支持的 OOT 有 23.10 版本的 mellanox 驱动。

ARM

- 支持倚天 710 DDRC PMU 驱动。
- 支持倚天 710 CMN-700 PMU 驱动。
- ARM ras 内存隔离。
- 移植支持 coresight trbe。
- 支持 NMI watchdog 检查。
- 支持倚天710 mpam 驱动。
- 支持 PV spinlock。
- 支持 PV poll。

- 支持 PV preempt。
- 支持倚天 cpuinfo 信息增强。
- 支持 crash kernel。
- 支持 cpu die topo。
- 支持 CPU prefetch功能。
- 支持 live patch 功能。

x86

- 机密计算：
  - TDX guest 基础支持：启动适配、内存加解密基本接口适配、ve、tdvmcall、swiotlb 优化等。
  - TDX guest attestation 接口支持：支持完整的基于 tdvmcall 的 attestation 接口（get_quote、get_report 等），提供完整的远程证明能力。
  - TDX guest 的 unaccepted memory特性支持：支持了 efi stub 和 boot parameter两种方式上报的unaccepted memory，让 guest 内存 on-demand 的 accept，加速大规格内存实例启动速度提高 200%。
  - TDX guest 的启动速度优化：通过优化 cpu idle 的实现，加速 TDX guest 内核的启动速度50%。
  - 支持海光 csv/csv2/csv3 ：安全内存管理与初始化，虚拟机启动与运行，使能海光 CSV3 虚拟机 CMA 并发分配内存，支持 live migration 等。
- 可信计算：
  - 支持海光 TPM/TCM/TDM/TPCM。
  - 支持海光密钥管理（TKM）功能。
- PMU：
  - 支持 Intel GNR 的 Lbr log event，扩展了 branch stack 事件的信息。
  - 支持 Intel GNR 的 timed PEBS 能力，相关硬件事件的采集精度更高。
  - 支持 GNR/SRF Core/Uncore PMU event counter，支持新平台上 perf 事件采集。
  - 支持 IOMMU PMU, 提供了采集和分析 IOMMU 事件的能力。
  - AMD LBRv2 支持以及相关 Bugfix 支持，AMD Perf ibrs 或其他相关 Bugfix 支持。
- IO 虚拟化：
  - 完整支持了基于 PASID 的 SVA 能力，用户可以直接基于 VA 面向支持 SVA 的设备编程。
  - 完整支持了 scalable-iov，可以基于硬件提供更细粒度的 IO 资源虚拟化。
  - 支持了 Intel IOMMU 的 slad，为 IO 设备热迁移提供了脏页追踪的能力。
  - 支持了 vfio 的热迁移协议 v1 接口。

- AMD IOMMU IO page table支持。
- AMD IOMMU PCI segment 支持。
- AMD IOMMU X2AVIC 及相关 IOMMU Bugfix 支持。

- MISC：
  - 支持 split lock detectionc。
  - 支持 srso、retbleed、mds、spectre 相关 cpu 漏洞的缓解机制。
  - 支持 SRF 平台 Sched Cluster；SRF 取消了超线程，4 个 Core 共享 L2 cache。
  - 支持新平台指令，包括 AMX FP16，AVX10，CMPccXADD，SW Prefetch。
  - 支持新平台功率特性，支持新平台 TPMI，RAPL，ISS。
  - 支持 GNR IFS，用于线下检查硬件可用性。
  - AMD QoS BMEC 支持。
- 加速器：
  - 海光内核密码模块 CCP（Cryptographic Coprocessor）：
    - 适配支持海光 CCP ，并升级国密驱动支持海光 4 号 CPU。
    - 海光 CCP 添加 hct 模块用于支持海光 HCT 引擎。
    - 海光 CCP 支持透传虚拟机。
  - 支持 Intel GNR 平台 DSA/IAA 2.0 加速器。

随着 ANCK 5.10/6.6 版本的持续演进，会有越来越多的新特性合入，请持续关注龙蜥社区 Cloud Kernel SIG 的 ReleaseNote：https://openanolis.cn/sig/Cloud-Kernel。

6.1.2.3 ANCK 5.10 重要自研/回合特性如下：

调度

- core scheduling 特性：core sched 是以 core 为单位调度任务的一项技术，通过给任务打上不同的 cookie，可以避免不同 cookie 的任务同时运行在一个物理核的不同 HT 上，有效避免侧信道攻击。
- ACPU (assess CPU) 技术：统计任务运行时 HT 对端空闲的时间，并提供 per-cgroup 统计，可以用于评估任务运行时共享CPU核心的硬件资源竞争情况。
- HT aware quota 技术：在任务使用 core sched 特性的情况下，通过感知 HT 对端是否空闲来控制消耗 cfs quota 的速率，使任务在每个调度周期执行的指令数量大致相同，从而达到可预期算力交付的效果，适用于计算型任务。
- cgroup 级别的 SCHED_IDLE ： 可通过设置目标 cgroup 的 cpu.idle 属性来使得该 cgroup 的调度策略成为 SCHED_IDLE ，适用于批量管理离线任务。

- CPU 动态隔离技术：将不同 CPU 核心或 CPU 集群分配给不同任务，可避免任务间对 CPU 资源的相互竞争，提高系统性能和稳定性。该功能支持系统运行时动态更改 CPU 隔离配置，适应运行时关键任务变化场景，提升 CPU 资源利用率。
- Group Identity 负载均衡优化：通过对 Group Identity 的负载均衡策略进行优化，在混部场景中，可提高在线任务的性能，避免离线任务饿死。
- Group Balancer 特性：通过对拓扑结构的感知将整机 cpu 划分为若干分区，通过对 cgroup 规格和负载的感知将 cgroup 分配到一个合适的分区，减少跨 numa 访存以及 cache miss，提高任务的亲和性，从而提高任务性能。
- 异步unthrottle（解限流）特性：将 unthrottle 的操作异步化，减少单次 unthrottle 耗时，避免因单次 unthrottle 耗时过长导致内核自动调整 cgroup 的 quota 和 period。

内存

- mglru：支持了改进内存页回收能力的 mglru，能够在大数据场景改善内存回收的速率和准确性，提升 e2e 性能。
- THP ZSR：THP ZSR 可解决使用 THP 带来的内存膨胀问题。该功能会把透明大页拆分为小页面，并将其中的全零页面（zero subpage）回收，从而避免内存的快速膨胀引发 OOM。
- page cache（文件缓存）限制功能： 用于解决因 page cache 无限制使用带来的系统稳定性问题，例如业务抖动、预期外的内存溢出 OOM（Out Of Memory）等。支持以 memcg 为粒度对 page cache 使用进行限制，对超过限制的 page cache 进行异步回收或者同步回收。
- page cache（文件缓存）预留功能: 通过为 page cache（文件缓存）设置预留内存空间，解决因大量匿名页申请导致文件缓存颠簸的问题，从而缓解因持续回收 page cache 所引发的系统稳定性问题（例如无法触发 OOM）。
- OOT 内存定向隔离功能：OOT（Out Of Tree）驱动质量存在差异，常常出现 use after free 内存错误以及越界访问等问题。通过对 OOT 驱动的内存池进行隔离，确保被破坏的内存不会被分配至其他模块，使得 OOT 驱动破坏的仅为其自身的内存。从而能够快速界定问题，减少排查与修复所需的时间，进而降低业务损失。
- 内存资源隔离能力增强：
  - 锁竞争优化：将全局 LRU 锁细化为 Memcg 粒度锁（覆盖页移动、Swap 等场景），大幅降低锁竞争，多租户并发性能大幅提升。
  - 核心业务防抖动：支持代码段内存锁定功能，防止低水位时核心业务代码被频繁换入换出，保障关键任务响应延迟稳定。
  - 智能优先级管控：引入 OOM Kill 与 Swap 回收的优先级机制，确保高优先级 Cgroup 在资源紧张时最后被牺牲或回收。
  - 异步回收与分级水位：新增 Memcg 后台异步回收及全局最低水位线分级，替代同步直接回收，避免分配阻塞，并差异化保障延敏感型与吞吐型业务需求。
  - 精细回收与快速故障隔离：
    - 支持 THP 细分回收（拆分大页并回收零页），防止内存膨胀引发 OOM。
    - 自动清理僵尸 Memcg，释放系统资源。
    - 实现 Fast OOM 机制，快速终止异常容器，降低对整机及相邻业务的干扰。

- 内存成本优化：
  - Kidled 冷内存回收：回收冷内存，降低内存成本。
  - tmpfs 零页填充：当发生文件 hole 的 read fault 时，通过系统零页来填充，避免内存浪费，节省成本。
  - 页表共享：通过对共享内存的页表进行共享，可以避免大量重复页表页申请，节省内存成本。
- kfence 增强：KFENCE 功能进行了增强，支持灵活的动态开关 KFENCE 以及全面捕获内存污染问题，从而兼顾了线上探测与线下调试的需求。
- 代码多副本：在 NUMA 架构（尤其是 ARM 实例）中，不同 NUMA 节点具有各自的本地内存，当一个 NUMA 节点上的程序或进程需要访问其他 NUMA 节点的代码段时，就会引入额外的延迟和性能开销。通过代码多副本功能，可以将远程节点的代码段复制到本地节点，避免了跨节点访问，从而解决 NUMA 架构中因跨节点访问带来的性能延迟问题。
- Async fork：Async-fork 将 fork 调用过程中最耗时的页表拷贝部分从父进程移动到子进程，父进程因而可以快速返回用户态处理用户查询，子进程则在此期间完成页表拷贝。为了保持父子进程之间的数据一致性，我们设计了一套高效的主动同步机制。实验结果表明，与 Linux 中的默认 fork 相比，Async-fork 显著减少了快照期间到达请求的尾延迟。
- 代码大页：代码大页（Huge Pages）是基于透明大页 THP（Transparent Huge Pages）进行的优化扩展，支持将应用程序和动态链接库的可执行部分放入到大页（通常是 2 MB 或更大）中，有助于降低程序的 iTLB miss，并提升 CPU 的 2 MB iTLB 利用率，避免内存碎片化或内存膨胀问题，提高内存利用效率，适用于数据库、大型应用程序等大代码段业务场景。

  存储

  io_uring：
  - io_uring nvme passthrough：将实际执行的文件操作通过 io_uring uring_cmd 直接递交给 NVMe 驱动层处理，绕过文件系统和块层，大幅提升应用性能。
  - io_uring percpu sqthread polling：支持多个 io_uring 实例共享同一个 sqthread，优化 cpu 开销并降低上下文切换延时。
  - 高性能用户态块设备 ublk：支持基于 io_uring passthrough 机制的高性能用户态块设备 ublk，提供分布式存储 agent 的高效接入方案。
  - io_uring 整体 code base 更新至上游 5.15.85, 提升系统稳定性，有利于后期特性回合和维护。
- erofs：
  - 支持 erofs over fscache 按需加载。
  - 支持 erofs over fscache 按需加载故障恢复。
  - 支持块设备后端挂载或者文件作为后端挂载。
  - 支持 subpage 块大小，例如 512 字节块大小，可应用于容器镜像场景中 erofs 直接索引容器镜像的 tar 包，省去容器镜像生命周期管理中 tar 包的 untar 以及清理流程，从而提升性能和稳定性，并提升容器镜像作为 golden image 的安全性。
  - 支持 erofs 的 flattened block device 模式，使得 multi-blob images 存储于同一个 block device，该特性可以应用于块存储镜像直通以及机密容器的存储方案。

- 支持 FSDAX 直接内存访问，通过virtio-pmem 消除虚拟化 page cache。
- 支持透明压缩和全局压缩去重，支持LZ4、LZMA、deflate、Zstd 等压缩算法。
- 支持 DADI OverlayBD turboOCI 镜像加速。

- xfs：
  - XFS 16k 原子写：基于 XFS CoW 机制自研 16k 原子写特性，相比 MySQL 默认打开双写，有至多 50% 的性能提升，同时显著减少磁盘 IO。该方案的优势在于不依赖硬件，且无运行时 IO 链路配置依赖。
  - 支持延迟 inode inactivation 特性，该特性将回收放到后台的 kworker 中执行，降低前台应用因删除带来的卡顿。
  - 支持 dax reflink。
- ext4：
  - Ext4 append 写优化：优化 Ext4 delalloc append 写场景（典型的如业务写日志）多余的 i_disksize 更新操作，在 kafka 小包场景性能提升 10%。
  - ext4 修复 O_DIRECT + O_SYNC 语义问题。该问题将导致 append 写的场景文件落盘的长度没有及时更新，此时如果异常掉电将无法读取写入的数据。
- fuse：
  - fuse CTO 缓存一致性模型增强，适用于依赖强一致性的分布式文件系统（如 NFS）后端，并增强相关监控。
  - fuse fd passthrough 增强：常用场景的 IO 延迟降低到先前的 10%。
  - fuse fd attach 增强：可无损恢复 fuse 挂载点连接，提升生产环境的稳定性。
  - 跨 namespace 传递 fuse 挂载点支持：实现 fuse 挂载点在非特权容器间的传播，提供一种基于 fuse 的远端存储在云原生场景下的解决方案。
  - 支持 fuse daemon 在故障恢复之后，可以请求对故障期间的请求进行重发。
  - 支持 bg_queue 读写分离和公平性优化。
  - 支持 failover。
  - 支持多队列。
  - 支持 cache=none 模式下 shared mmap。
  - 支持 strict limit 特性动态开关。fuse 模块会设置 strict limit，在特定场景下可能导致回写非常慢甚至卡住，引入该 sysfs knobs 可以动态解决这种问题。
- block：
  - cgroup v1 支持 block throttle 限流和 iocost 权重控制。
  - IO hang 检测：通过扩展核心数据结构，提供相关接口导出 IO hang 信息，辅助快速分析和定位 IO hang 原因。
  - tcmu 支持 bypass tcmu data area 和 zero copy 特性。

- tcmu_loop 设备支持参数可配置，如 can_queue、nr_hw_queues、cmd_per_lun、sg_tablesize 等，当后端设备能力足够大的时候，调大这些参数能明显提升性能。
- 大块 IO 的 IOPS 限流功能优化：解决 block throttle 因大块 IO（典型的如 buffer IO 场景）可能存在的拆分导致 IOPS 限流不准确的问题。
- 避免预分配大的 SGL buffer，以优化 virtio-blk 设备的 IO 内存占用，对高密安全容器场景非常有用。

网络

- UDP 分段卸载（USO）技术：支持将大报文分割成携带传输头的小报文，从而提高在复杂网络环境下的转发效率和报文接收性能。
- 基于虚拟内存分配的 xdp socket 创建优化：通过 vmalloc 虚拟内存分配技术提升成功创建可能性，避免物理内存碎片化导致的分配失败问题。
- TSQ 性能优化：virtio-net 通过默认使能 tx napi，以优化 TCP Small Queue 的发送性能。
- virtio-net：
  - 完善的 RSS 功能支持。
  - 解决 XDP 重复验证 checksum 问题，提升网卡处理效率。
  - 支持 NetDIM - 动态调节云上网卡中断频率。
  - 支持调整网卡队列大小。
  - 支持网卡硬件收包信息统计，提升网卡运维监控能力。
- 依托 eBPF 的强大能力，提供了基于 eBPF 的 tcprt，vtoa 等功能的支持，相比于内核模块方案，eBPF 方案提供了更强大的能力。
- 支持 SMC 内核网络协议栈：配合硬件 RDMA 技术提供共享内存通信，透明加速 TCP 以获得更低的网络时延和 CPU 使用率。

perf

- 支持倚天 CMN 和 DDR PMU 的 perf metric 功能，可检测倚天 710 slc miss rate、跨 die 流量和访存流量等指标。
- 添加 dwc_rootport PMU 驱动，使 perf 工具可以采集到硬件支持的 PMU 事件，用于分析 PCIe rootport 性能。
- 支持以 perf metric 的形式对性能问题进行 topdown 分析，提升 CPU PMU 的易用性。
- 支持倚天 Arm-SPE 特性，完善 perf 工具，可支持检测倚天平台内存瓶颈和伪共享问题。

驱动

- 支持按照 net、block、gpu 领域的方式进行模块签名校验。
- 优化 PCI 驱动遍历 /proc/bus/pci/devices 的性能。
- 正式支持 NVMe over TCP、NVMe over RDMA 以及 nvme-multipath 特性， 打开相应 Kconfig 以及回合上游补丁增强稳定性。
- 阵列卡驱动，支持 sssraid 国产阵列卡驱动。
- 网卡驱动，支持 sssnic、nfp、txgbe、ngbe 国产网卡驱动。

- OOT 驱动基线建立，当前驱动基线已有 mellanox、ice、i40e、mpt3sas、mpi3mr 等 25+ 的驱动。
- 增强 softlockup 检测功能：可直接区分导致 softlockup 的原因是中断风暴、软中断执行时间长还是内核任务执行时间长这三种情况中的哪种，提高问题分析效率。

ARM

- 支持倚天 710 CMN-700 PMU 驱动。
- 支持倚天 710 DDRC PMU 驱动。
- 支持 SEA异常 内存隔离。
- 支持 mpam 隔离和监控。
- 支持 spe 驱动。
- 支持 coresight 采集程序跳转。
- 支持 NMI watchdog 检查。
- 支持倚天710 mpam 驱动。
- 支持 PV spinlock。
- 支持 PV poll。
- 支持 PV preempt。
- 支持倚天 cpuinfo 信息增强。
- 支持 crash kernel。
- 支持 cpu die topo。
- 支持 CPU prefetch功能。
- 支持 live patch 功能。

x86

- 机密计算
  - TDX guest 基础支持：启动适配、内存加解密基本接口适配、ve、tdvmcall、swiotlb 优化等。
  - TDX guest attestation 接口支持：支持完整的基于 tdvmcall 的 attestation 接口（get_quote、 get_report 等），提供完整的远程证明能力。
  - TDX guest 的 unaccepted memory 特性支持：支持了 efi stub 和 boot parameter 两种方式上报的 unaccepted memory，让 guest 内存 on-demand 的 accept，加速大规格内存实例启动速度提高 200%。
  - TDX guest 的启动速度优化：通过优化 cpu idle 的实现，加速 TDX guest 内核的启动速度 50%。
  - 支持海光 csv/csv2/csv3：安全内存管理与初始化，虚拟机启动与运行，使能海光 CSV3 虚拟机 CMA 并发分配内存，支持 live migration 等。

- 可信计算
  - 支持海光 TPM/TCM/TDM/TPCM。
  - 支持海光密钥管理（TKM）功能。
- PMU
  - 支持 Intel GNR 的 lbr log event，扩展了 branch stack 事件的信息。
  - 支持了 IOMMU PMU，提供了采集和分析 IOMMU 事件的能力。
  - 支持 GNR/SRF Core/Uncore PMU event counter，支持新平台上 perf 时间采集。
  - AMD LBRv2 支持以及相关 Bugfix 支持，AMD Perf ibrs 或其他相关 Bugfix 支持。
- IO 虚拟化
  - 完整支持了基于 PASID 的 SVA 能力，用户可以直接基于 VA 面向支持 SVA 的设备编程。
  - 完整支持了 scalable-iov，可以基于硬件提供更细粒度的 IO 资源虚拟化。
  - 支持了 Intel IOMMU 的 slad ，为 IO 设备热迁移提供了脏页追踪的功能。
  - 支持了 vfio 的热迁移协议 v1 接口。
- MISC
  - 支持了split lock detection。
  - 支持了 srso、retbleed、mds、spectre 相关 cpu vulnerability 的 mitigation。
  - 支持 SRF 平台 Sched Cluster；SRF 取消了超线程，4 个 Core 共享 L2 cache。
  - 支持新平台指令，包括 AMX FP16、AVX10、CMPccXADD、SW Prefetch。
  - 支持新平台功率特性，支持新平台 TPMI、RAPL、ISST。
  - 支持 GNR IFS，用于线下检查硬件可用性。
  - AMD QoS BMEC 支持。
  - AMD IOMMU IO page table支持。
  - AMD IOMMU PCI segment 支持。
  - AMD IOMMU X2AVIC 及相关 IOMMU Bugfix 支持。
- 加速器
  - 海光内核密码模块 CCP（Cryptographic Coprocessor）。
  - • 适配支持海光 CCP，并升级国密驱动支持海光 4 号 CPU。
  - • 海光 CCP 添加 hct 模块用于支持海光HCT引擎。
  - • 海光 CCP 支持透传虚拟机。

6.1.3 实现软硬件兼容性验证的自提交能力

6.1.3.1 概述

龙蜥社区以及广大 ISV / IHV / OSV 合作厂商，在发布各类软件、硬件、龙蜥操作系统及衍生版等产品时，均需要评估验证各自产品的兼容性。为此，龙蜥社区针对软件、硬件、操作系统三类对象，通过配套工具、流程、文档等，提供了高效、便捷、自动的兼容性评估验证手段，推动龙蜥社区各类软硬件产品的兼容适配，繁荣龙蜥社区软硬件生态。

6.1.3.2 软件兼容性

软件兼容性，包括引入龙蜥操作系统发行版的软件、业界主流开源软件和商业软件的兼容性，龙蜥社区提供了差异化的兼容性验证流程，致力于验证各类软件与龙蜥操作系统不同版本之间的兼容性，推动软件与龙蜥操作系统的兼容适配，围绕龙蜥操作系统建立完善的软件生态。

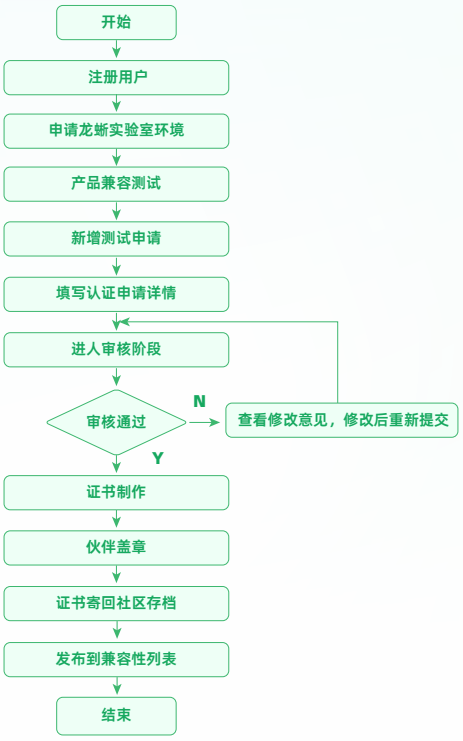
软件兼容性验证主要包含以下 3 类：

1. 对于引入到龙蜥操作系统发行版的 RPM 软件，基于 CI-Bot、ABS 和 T-One 等龙蜥社区基础设施，建立了一套统一专业的 CICD 流程，对引入的软件包进行源码扫描、SPEC 检查、LICENSE 检查、构建测试、安装卸载测试、ABI 检查和依赖检查等一系列检查，保障引入软件的兼容性。

2. 对于业界主流的上游活跃开源软件，以源码安装运行方式，在龙蜥操作系统上进行兼容性验证，保障开源软件最新发布版本在龙蜥操作系统上的兼容性，验证通过的软件会发布到龙蜥社区软件兼容性列表。

3. 对于各 ISV 厂商的商业软件，龙蜥社区提供了一套完整的软件兼容性认证规范、流程和平台，并基于龙蜥实验室基础设施提供了免费的验证环境。ISV 厂商可自助申请兼容性认证，认证通过后颁发龙蜥社区兼容性证书，并发布到龙蜥社区软件兼容性列表。

龙蜥操作系统用户可以通过龙蜥社区软件兼容性列表查询相关软件与龙蜥操作系统某个发行版本的兼容性。龙蜥社区软件兼容性 SIG 会持续发布和更新软件兼容性列表，持续推动龙蜥社区软件生态建设。

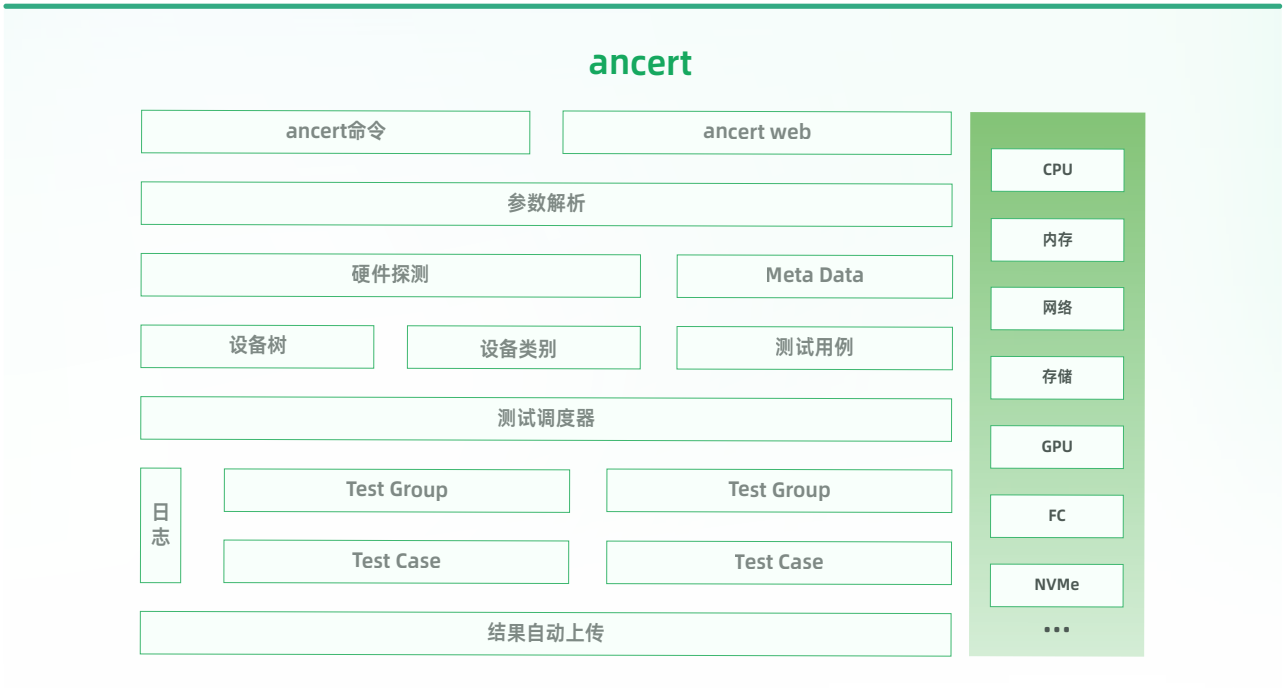


6.1.3.3 硬件兼容性

针对硬件兼容性，龙蜥社区提供了 ancert 硬件兼容性测试套件，致力于验证硬件设备集成商等厂商发布的整机服务器和各种板卡外设与龙蜥操作系统不同版本之间的兼容性，推动社区发行版在各种硬件设备上的适配，围绕龙蜥操作系统建立完善的硬件生态。

目前，ancert 支持服务器整机和 NIC、HBA、FC、GPU、NVMe 等多种外设，以及 DPU 与龙蜥操作系统的硬件兼容性测试。主要技术特点：

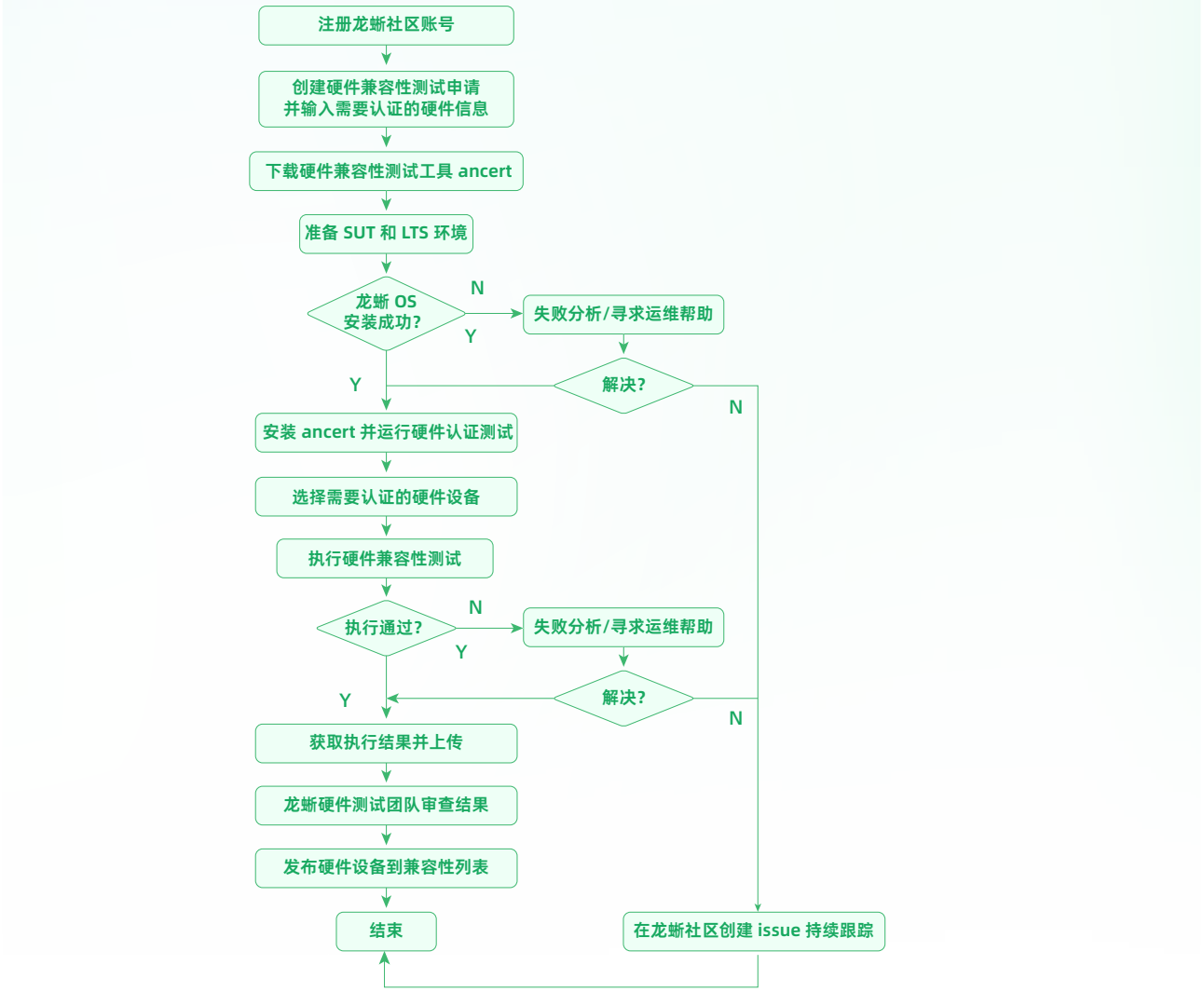
- 跨平台，支持 x86、ARM、LoongArch 等不同架构。
- 支持各种硬件设备的探测、识别、分类、展示。
- RPM 包形式发布，同时支持单机、多机测试环境。
- 一款自动化测试框架，支持自动化，并发测试等。
- 支持 Python，Shell 等语言编写的测试用例。



此外，龙蜥社区硬件兼容性 SIG 构建了完整的硬件设备兼容性测试申请流程，包括：验证标准、申请流程、硬件测试、结果验证、列表发布等流程。

社区用户，IHV 或者硬件设备集成商等通过安装龙蜥操作系统硬件兼容性验证工具 ancert，对相关硬件设备执行兼容性测试，并将测试结果提交到龙蜥社区硬件兼容性列表，测试结果经过龙蜥社区硬件兼容性 SIG 审核通过之后就可以发布到龙蜥社区硬件兼容性列表上。

龙蜥操作系统用户可以通过龙蜥社区硬件兼容性列表查询相关硬件设备与龙蜥操作系统某个版本的兼容性。龙蜥社区硬件兼容性 SIG 会持续发布和更新硬件兼容性列表，持续推动龙蜥社区硬件生态建设。



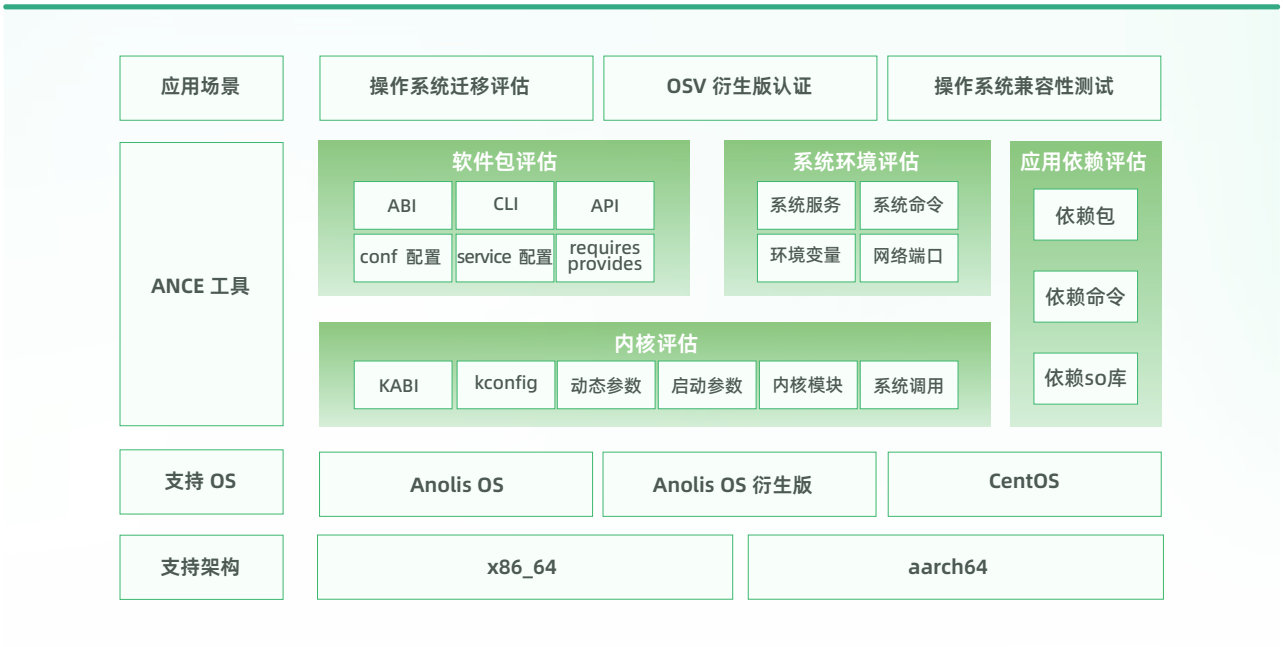
6.1.3.4 操作系统兼容性

针对操作系统兼容性，龙蜥社区提供了 ANCE（ANolis Compatibility Evaluation）操作系统兼容性分析评估工具，可被应用于操作系统迁移、龙蜥操作系统兼容性测试、OSV 衍生版认证等多种场景，致力于推动国产化操作系统的迁移和替代，保障龙蜥操作系统发行版的前向兼容性，以及衍生版与龙蜥社区技术路线的一致性，降低社区重复适配成本。

目前，ANCE 工具支持操作系统多种维度的评估，主要技术特点：

- 跨平台，支持 x86、ARM 不同架构，支持 Anolis OS、CentOS 等不同发行版及衍生版。
- 丰富的评估维度，包括软件包、内核、系统环境、应用依赖 4 大类共 20 项评估维度。
  - 软件包包括 ABI、API、CLI、conf 配置、service 配置、provides、requires。
  - 内核包括 kABI、kconfig、内核动态参数、内核启动参数、内核模块、系统调用。
  - 系统环境包括系统服务、系统命令、环境变量、网络端口。
  - 应用依赖包括依赖 RPM 包、依赖系统命令、依赖 so 库。

- 支持多种评估对象，包括静态的 ISO、RPM 文件，以及动态的 OS 运行时环境。
- RPM 包形式发布，支持离线部署使用，或被集成其他平台使用。



对于操作系统迁移场景，通过与 SysOM 平台结合，简化迁移前后的步骤，加速操作系统迁移和替代。

对于龙蜥操作系统兼容性测试场景，主要验证发行版小版本迭代的前向兼容性。

对于 OSV 衍生版认证场景，主要验证 OSV 厂商发布的龙蜥衍生版与龙蜥发行版的亲缘性，保障与龙蜥社区技术路线的一致性，并享有龙蜥社区既有软硬件兼容性生态，降低社区重复适配成本。

6.2 云原生场景

6.2.1 云原生场景下的计算核心 RunD

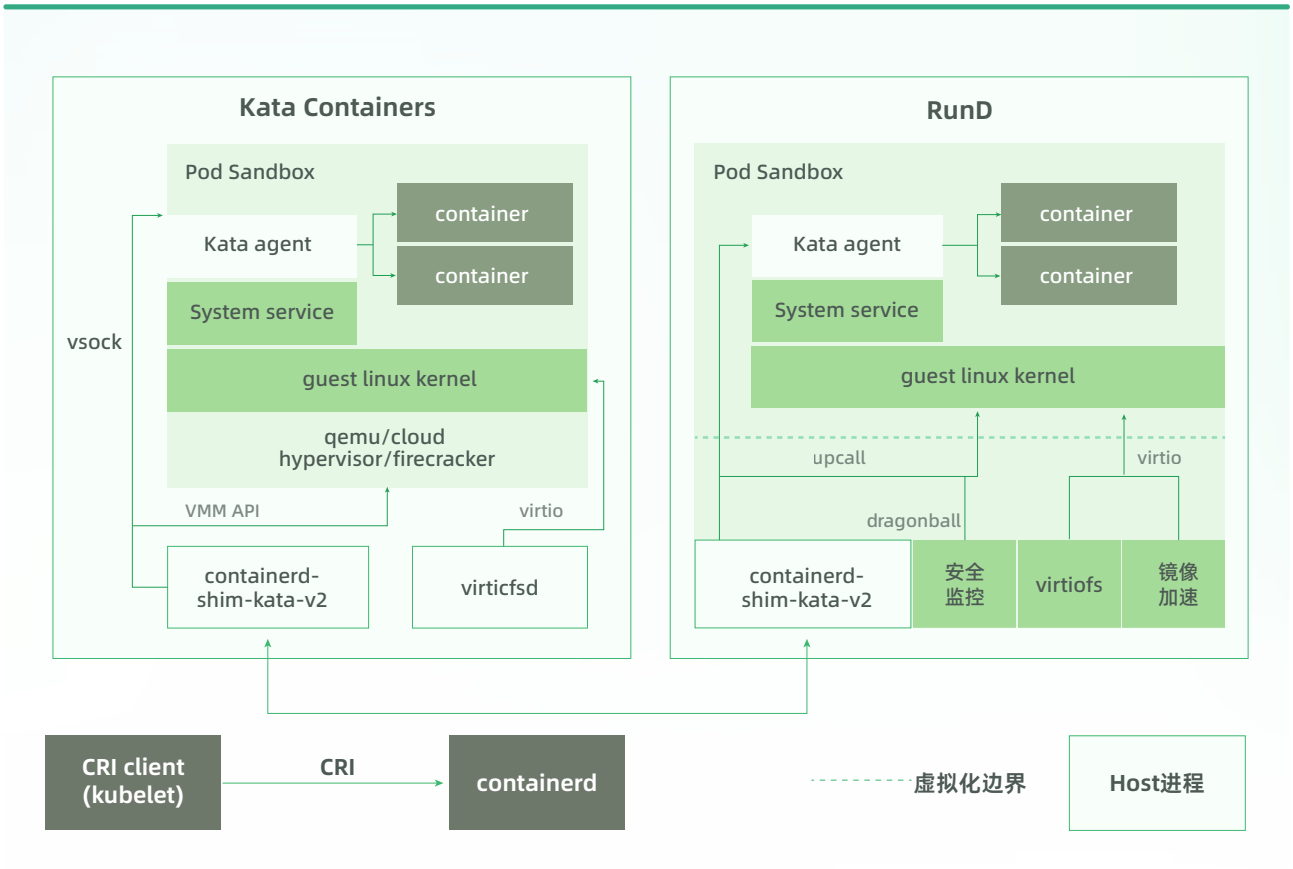
6.2.1.1 技术介绍

随着容器技术成熟，基于容器镜像部署应用，基于 k8s 管理应用逐步成为大家共识，越来越多企业和应用投身到了容器化部署改造升级进程。但是随着容器化使用越来越多，原有基于 namespace + cgroup 的隔离方式也面临着越来越多的挑战：

1. 多个容器共享内核，内核锁、访存竞争等系列干扰对容器应用性能的干扰

2. 多租场景共享内核带来的安全风险

基于这些问题，社区提出来了基于 kata 的安全容器架构，通过将虚拟化技术和容器技术结合，并直接对接 k8s 社区，很好的解决了 runc 容器共享内核带来的性能隔离和安全隔离难题。RunD 也是在这个背景下产生，用于构建阿里云内部安全容器。安全容器路径探索过程中我们碰到了很多问题，比如社区原有 kata containers 资源开销比较大，每个安全容器资源开销超过 100MB，启动也很慢，需要数秒，另外传统 OS 需要加载大量驱动来支持各类设备，还有 systemd 等重量级服务，使得整个安全容器很臃肿，面对 serverless 场景高密度、高并发、低延迟启动很难满足，同时也引入了一些非必要的安全风险。面对这些问题，我们对整体架构做了如下系统性的梳理和重构：



用 Rust 实现 VMM Dragonball：让原有VMM的内存资源开销从 100+ MB 下降到 3 MB。即使是128M 实例的开销也在可接受范围。同时支持设备直通、设备热插拔、NUMA 等特性，让 RunD 能够适配各种复杂业务场景；多进程融合设计：使 RunD 在生产环境维护升级更加便捷，提升安全容器稳定性的同时也让进程间通信转换为进程内通信，降低通信开销，优化启动性能；精简操作系统相关组件：有效剥离非容器场景相关特性和组件，降低资源开销，精简组件和接口数量，减小沙箱受攻击风险；VMM 和 Guest Kernel 的融合设计是安全容器的独特之处。传统虚拟机 Guest Kernel 由用户提供，VMM 要处理各类 Guest Kernel，因此大量问题要在宿主机侧解决。但在安全容器场景，Guest Kernel 和 VMM 的统一提供让两者可以深度融合。这一设计进一步提升了安全容器的启动速度，降低了安全容器的开销。

通过上述技术突破，RunD 目前已在生产环境部署并支持单节点 4000 安全容器、200/s 并发、200ms 启动。

6.2.1.2 技术应用场景

RunD 技术解决了微服务和 Serverless 场景下细粒度资源安全隔离和按需供给问题，该技术适合以下三类场景：

1. 多租户容器场景，例如：公共云对外容器服务场景；

2. 可信&不可信容器混合部署，例如：大数据场景里面部分 UDF 程序的安全隔离；

3. 不同 SLO 业务混合部署，例如：离线和在线业务的混合部署。

6.2.1.3 技术影响力

RunD 经历过大规模生产环境的考验，相关成果也已汇总为学术论文发表在 2022 年 USENIX ATC、2023 年 SOSP 上。当前，RunD 部分主体功能已经正式合并到 Kata 3.0 中，成为 Kata 3.0 架构的一部分。龙蜥社区的云原生 SIG 也在组内对 Kata3.0 架构做了重点介绍，并在社区构建了 RunD 预览版，为社区用户提供体验。

6.2.2 跨云-边-端的只读文件系统 EROFS

6.2.2.1 背景介绍

在云原生、桌面、终端等应用领域，为了高效可信构建，分发和运行镜像，解决方案一般倾向选择只读方案，其优势在于分发和签名校验、写保护、器件故障可靠恢复等。通用文件系统如 EXT4 和 XFS 往往不能充分满足镜像极致大小，压缩，去重及可复现构建等需求，且通用文件系统冷门特性会增加格式复杂度，影响分发和执行环节的安全性和可控性，因此打造 Linux下高性能自包含内核只读文件系统能更好地服务容器、终端、集群 OS 等业务场景。

6.2.2.2 技术方案

EROFS 是为高性能只读场景量身打造的内核文件系统，提供了多层镜像、透明压缩、块去重、原生按需加载、FSDAX 内存直接访问等特性，于 Linux 5.4 正式合入 Linux 主线。在容器镜像领域，通过与 CNCF Dragonfly Nydus 镜像服务、Composefs、阿里云 DADI OverlayBD 等方案深度融合，打造 FS-Cache, TarFS 等技术，服务容器 runC、Kata、gVisor 等场景。在终端领域已成为 Android Open Source Project 推荐的系统分区文件系统格式。

技术优势：

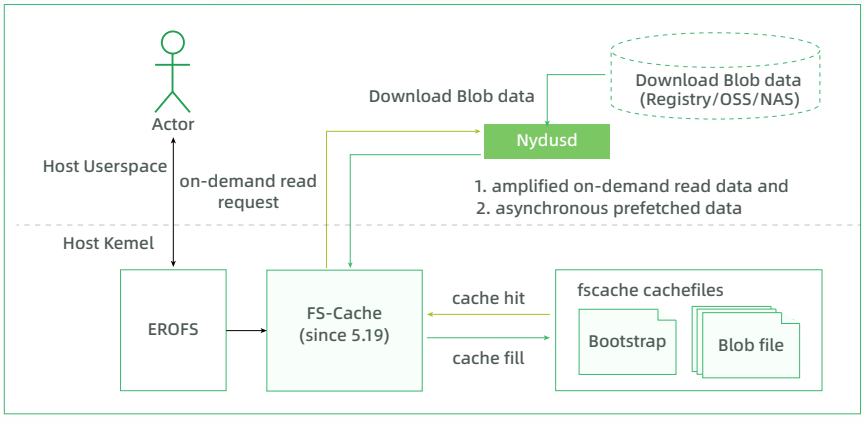
- Linux 内核原生，通过压缩（LZ4 / LZMA / DEFLATE / Zstd）、去重节省镜像存储空间。
- 原地解压等技术进一步优化运行态内存占用，提升性能。
- 继 XFS 文件系统后 Linux 社区上游第二个原生支持透明大页（large folios）的主流磁盘文件系统，领先 EXT4。
- 提供内核原生按需加载能力，从源头解决 FUSE 额外拷贝和上下文切换开销。
- 提供 OCI 标准镜像免解压 TarFS 技术提升性能，实现运行时数据保护及设备直通。
- 除传统块设备外，支持基于文件挂载，提升容器高密场景性能并降低虚拟块设备管理开销。

应用场景：容器/App/系统镜像，软件包管理，AI 数据分发，函数计算，机密计算，无盘启动，安装器等。

6.2.2.3 基于 EROFS + FS-Cache 优化 Nydus 镜像按需加载

EROFS over FS-Cache 是龙蜥社区牵头为 Nydus 和 EROFS 开发的下一代容器镜像按需加载技术，同时也是 Linux 内核原生的镜像按需加载特性，于 5.19 合入内核社区主线。

该方案将按需加载的缓存管理通过 FS-Cache 框架下沉到内核态执行，当镜像已在本地缓存时，相比用户态方案可有效避免内核态/用户态上下文切换和内存拷贝；当缓存未命中时，再通知用户态通过网络获取数据，做到真正的“按需”，非按需场景下实现几乎无损的性能和稳定性。



在按需加载场景，EROFS over FS-Cache 相比 FUSE 性能更优（注：数据为三次测试取平均值）：

	OCI	EROFS + FUSE	EROFS + FS-Cache
wordpress E2E 启动时间	11.562s	5.263s	4.619s

在非按需场景，EROFS over FS-Cache 相比 FUSE 性能也更优：

	OCI	EROFS + FUSE	EROFS + FS-Cache
本地 cache 4K 顺序读	387068KB/s	211767 KB/s	366291KB/s
本地 cache 4K 随机读	6153KB/s	5450KB/s	6170KB/s

6.2.2.4 基于 EROFS TarFS 技术优化 OCI 容器镜像

OCI 标准镜像是分层增量构建的 tar.gz 压缩包，当前 OCI 镜像需解包到 overlay 文件系统实现数据可访问，这存在如下问题：

- 小文件太多可造成大量元数据写入，影响启动性能；大量小文件也影响 GC 效率；
- 无法做运行时校验，丢文件/数据破损无法及时发现；
- 无法做只读/防写入保护。

TarFS 技术通过把 OCI 镜像透明转换为 EROFS 可以识别的格式，避免直接把 tar.gz 解包到 overlay 文件系统，其优势有：

- 避免大量文件创建/删除对底层文件系统的冲击；
- 可做只读保护和运行时校验；
- 转换后的格式可 P2P 分发，集群侧可制作镜像缓存，进一步优化启动速度。

在 containerd 非按需场景，对比当前默认的 overlayfs snapshotter，EROFS snapshotter 启动性能更优：

	overlayfs	EROFS
wordpress E2E启动时间	5.654s	4.838s

6.3 智能计算场景

6.3.1 AI 工作负载优化的龙蜥操作系统

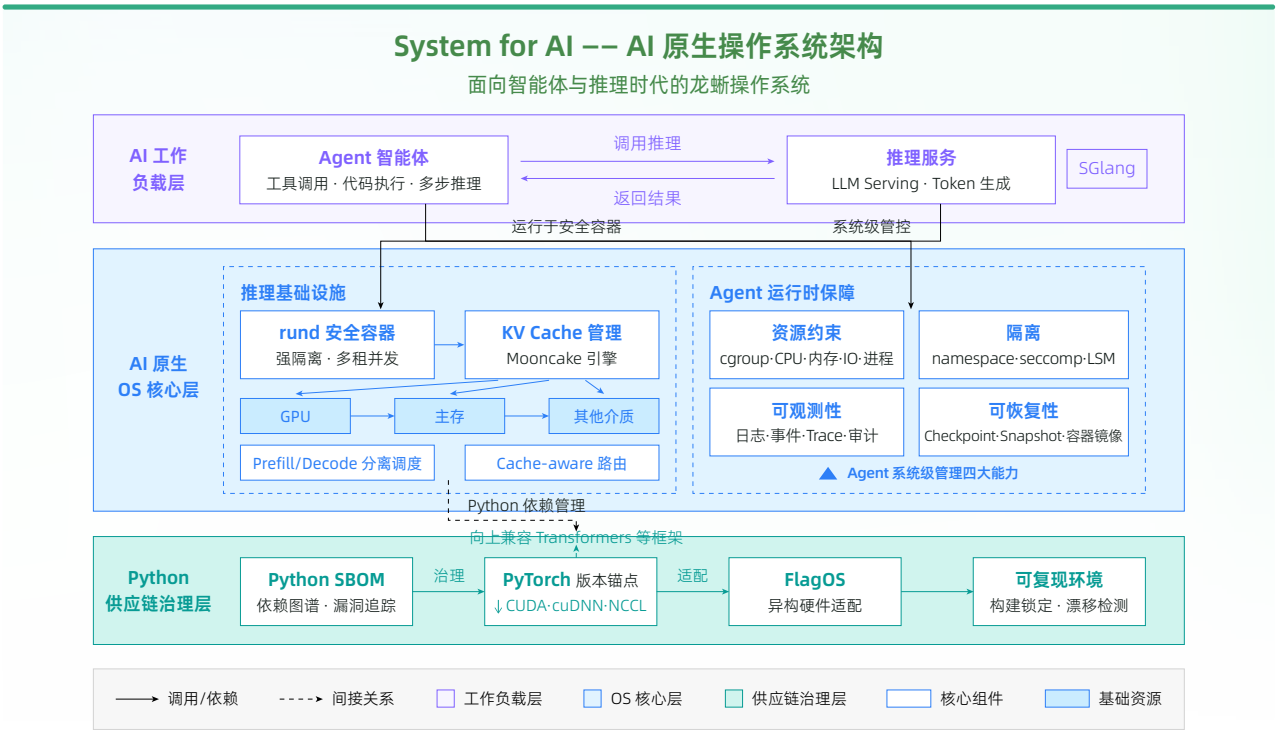
6.3.1.1 背景

AI 的发展正在把基础设施的重心从“通用计算”推向“面向模型与智能体的系统工程”。

一方面，大模型推理从零散试点走向规模化服务：并发上升、多模型多租户共存，平台需要在吞吐、推理延迟和成本上持续优化；另一方面，智能体（Agent）让应用从“调用接口”转向“调用工具与执行任务”，对系统提出了长生命周期运行治理、清晰的受控执行边界，以及可审计可追溯的治理能力等新要求；与此同时，多租户 MaaS 平台形态的普及要求运行时在性能与隔离之间取得新的工程平衡，形成可信 AI 所需的安全边界与可证明性。 此外，AI 也正在反向重塑操作系统自身的工程体系。对开源的社区操作系统而言，研发、测试与运维长期面临现实挑战：组件规模繁多、社区人力不足（高度依赖自愿贡献）、文档缺失与知识分散、以及内核等关键组件门槛过高，导致评审、验证、定位与长期维护的成本居高不下。社区需要把 AI 能力引入系统工程全流程，在代码评审、测试生成、问题定位、基线看护、漏洞处置以及版本选型与 SBOM 治理等环节持续提效，增强工程协作与交付确定性。其中，Python 生态已经是 AI 应用开发与交付的软件运行时与工具链生态的事实标准；其依赖漂移、包投毒与合规审计压力，使“Python 供应链治理”从应用层问题变成发行版与平台必须回答的问题，Python SBOM 也因此成为 AI 供应链治理的关键抓手。

基于此，龙蜥操作系统 Anolis OS 面向 AI 时代一直秉持两条主线协同演进，并逐步工程化落地为龙蜥操作系统的发行版特性和能力：

- System for AI：面向智能体与推理时代，规划推理基础设施、强隔离运行时与以 Python SBOM 为核心的供应链能力，逐步工程化落地为龙蜥操作系统的特性与基础能力。
- AI for System：用 AI 反哺系统工程，把运维从命令行走向自然语言，把研发从经验驱动走向证据链闭环，提升评审、测试、回归定位与 CVE 处置能力。



6.3.1.2 System for AI - 面向智能体与推理时代的 AI 原生 OS

6.3.1.2.1 让推理服务“可规模化运营”

推理服务要“可规模化运营”，长期面临一个很现实的约束：很难同时把成本、体验与稳定性都做到极致。更常见的状态是，追求更低成本会带来更激进的共享与更高的抖动风险；追求更强体验可能推高资源占用；追求更稳则往往意味着更高的冗余与更保守的策略。

对操作系统而言，关键在于把推理负载中最容易失控、最影响整机稳定与隔离边界的部分（多租场景下的相互影响、显存与 KV Cache 压力）尽可能沉淀为更强的底层能力：运行边界更清晰、故障域更明确、资源约束更扎实、以及系统证据更完整，从而支撑上层去持续改进成本、体验与稳定性这个“不可能三角”。

面向大模型推理的规模化运营，通常会反复遇到以下几类问题：

1. 体验指标容易“平均不错、偶发很差”：在高并发与流量波动下，需要保持 Token 生成速度稳定，同时避免因为资源争用、显存压力与执行抖动导致的尾部延迟放大；否则即便平均指标不错，体验仍会不稳定。

2. 成本压力来自多个层面，且难以持续收敛：推理成本不仅是算力时间，还包括显存占用、资源空转与峰值预留等。很多时候成本下不来根因不是“算力不够”，而是资源使用方式不可持续。

3. GPU 与显存压力放大系统不稳定因素：推理服务的瓶颈不只在算力，还包括显存占用与碎片化、KV Cache 膨胀、以及同机并发带来的资源争用。缺少有效的隔离与约束时，常见结果是“能跑但并发上不去、成本下不来、抖动很难压住”。

4. 工程复杂度叠加，线上表现更难复现：多模型、多版本、多后端并存会放大环境差异的影响，导致“同一套配置在不同节点表现不一致”，使问题定位与持续优化成本上升。

围绕上述问题，龙蜥操作系统在推理方向上的思路不是把所有优化都“做进操作系统”，而是优先抓住那些最容易影响多租并发持续运营，同时又最需要底层能力支撑的关键点：一类是多租场景下的强隔离运行边界，用来降低同机相互影响、明确故障域；另一类是 KV Cache 带来的显存压力与抖动，这是影响并发、成本与体验稳定性的核心变量之一。基于这两个关键矛盾，龙蜥操作系统当前更聚焦在 rund 安全容器与围绕 KV Cache / Mooncake 的技术栈建设上，先把“边界更清晰、故障域更明确、资源约束更扎实、系统证据更完整”打牢，再为上层持续优化留出空间。

- 首先，rund 安全容器提供强隔离机制，支撑 MaaS 多租并发下的可持续运营。 在共享算力资源池场景中，隔离边界是否足够清晰、资源争用是否可控，直接影响稳定性与尾部抖动，也影响可实现的共享程度与成本效率。龙蜥操作系统的 rund 安全容器提供更强的隔离运行机制；在此基础上结合混布能力，可以在多租户并发下更好地约束相互影响、缩小故障域，从而在保证 Token 生成速度的同时优化平台成本。

- 其次，围绕 KV Cache 膨胀问题，龙蜥操作系统规划以 Mooncake 构建关键推理技术栈能力。 在很多推理系统里，KV Cache 是影响并发与成本的关键变量：KV 增长会直接挤压 GPU 显存，从而降低可服务并发并推高成本，同时也容易引入抖动。Mooncake 作为生态中的开源 KV Cache 方案之一，龙蜥操作系统计划在推理技术栈中引入并工程化集成，将其纳入发行版交付与依赖体系，作为应对 KV 膨胀与显存压力的关键组件。

- 最后，结合业界“层次化 KV Cache”趋势，把 KV 按热度与生命周期分层管理，并与推理框架策略协同。 层次化 KV 可以理解为：在 offload 基础上进一步体系化，把 KV 按热度与生命周期分层管理（GPU /主存/其他介质），并配合推理框架侧的回收、复用与准入策略，使不同负载下的表现更稳定、更具一致性，从而减少抖动并改善成本效率。这里可以与 SGlang 等开源推理框架的策略协同，形成“运行时能力 + 框架策略”共同作用的工程闭环。

当强隔离运行边界（rund）与 KV Cache 关键瓶颈处理能力（Mooncake/层次化 KV）逐步工程化落地后，推理服务在多租与高并发压力下的运行形态会更稳定：同机相互影响更小，尾部抖动更容易收敛，资源共享更可持续；同时，故障域与系统证据更清晰，也更有利于持续定位与迭代优化，从而为“持续改进成本、体验与稳定性这个不可能三角”提供更扎实的底座。

6.3.1.2.2 让智能体成为系统级核心工作负载

> 从操作系统能力边界出发，讨论在当前 Agent 快速演进阶段，OS 层可能需要补齐的通用能力（资源、隔离、执行与可观测），用于更稳地承载“长任务 + 工具调用 + 多进程/多服务协同”的新型工作负载。该部分仍处在持续迭代与工程探索中，并未形成业界统一标准。

把 Agent 放到真实服务器环境里运行后，首先暴露出来的往往不是“模型够不够聪明”，而是一些非常基础的系统问题：任务更长、路径更不确定、外部交互更多、工具执行更多。一旦缺少系统层的资源约束、运行边界与可观测支撑，很容易演变成整机不稳、风险边界不清、问题难定位。从 OS 视角看，具体的例如：

1. 长任务与不确定执行路径带来的资源失控风险：Agent 类任务往往运行更久、分阶段执行，并可能因为外部环境（网络/依赖服务/工具返回）反复重试或改变路径，导致 CPU、内存、文件句柄、线程等资源长期占用甚至泄漏；在 OS 层表现为整机负载抖动、OOM、句柄耗尽、以及“单任务拖垮整机”。

2. 工具执行与外部交互使攻击面扩大，需要更强的进程级隔离与权限边界：Agent 任务常伴随命令执行、文件读写与网络访问。OS 需要提供更清晰、可组合的隔离手段（命名空间、cgroup、seccomp/LSM、sandbox/安全容器等），把“能做什么”变成可约束的运行边界，降低越权与逃逸风险。

3. 可恢复性：进程/作业中断后的“最小恢复能力”缺位，长任务更容易遇到进程崩溃、节点重启、依赖超时等问题。虽然业务语义恢复属于上层，但 OS 层需要为“可恢复”提供基本设施，更可靠的进程监督与退出原因采集、可持久化的状态与文件一致性保障、以及可用于断点续跑的检查点/快照/容器镜像层等基础能力。

4. 可观测性不足导致定位困难：当 Agent 任务引发性能退化或系统异常时，定位往往跨越进程、容器、文件系统、网络与权限层。OS 需要提供足够的可观测能力，统一的日志与事件、资源统计、调用链/trace 接口、以及可审计的系统调用与安全事件记录，帮助快速回答“哪个任务做了什么、消耗了什么资源、为什么失败”。

上面的挑战并不要求 OS 去理解任务语义，但确实要求 OS 把一些原本靠经验兜底的能力变成更可靠的基础设施。龙蜥操作系统在这一方向上的思路，是把 Agent 执行尽量收敛为“受控工作负载”的形态，在资源、隔离、可观测与可恢复性上提供更强的系统支撑；其中也需要与上层应用/框架协同才能发挥完整效果。

- 资源约束与异常回收：通过 cgroup 等机制强化 CPU / 内存 / IO / 进程数 / 句柄等资源的隔离与配额控制，并在异常时支持更及时的限制与回收，降低“单任务拖垮整机”的概率。
- 更强隔离的执行环境（rund 方向）：在容器之上引入更强隔离边界，形成更清晰的执行域与权限边界。rund 安全容器也可以作为 Agent 执行环境的承载底座之一，用于在多租与高风险工具执行场景下降低逃逸与越权风险，并缩小潜在影响面。
- 最小权限与系统级约束组合：利用 Linux 能力体系（namespace / cgroup / seccomp / LSM 等）对文件、网络、进程行为与系统调用面进行约束，使“工具能做什么”在 OS 层具备可落地的边界。
- 系统证据与可追溯性增强：强化任务级 / 进程级资源画像与异常信号（OOM、throttle、IO 放大、网络异常、权限拒绝等）的采集与关联，为定位提供更可靠的系统证据。
- 可恢复基础设施支撑：OS 层提供更好的崩溃原因与退出路径记录、文件系统一致性与隔离执行环境的可重建性，为上层实现断点续跑与任务恢复降低门槛。

如果上述方向能够逐步工程化落地，OS 层面希望带来的变化是：Agent 类长任务在真实服务器环境里运行得更稳，不容易因为资源泄漏或异常重试把机器拖垮；工具执行被限制在更清晰的运行边界内，风险外溢面更小；当问题发生时，系统层能够提供更完整的证据用于定位与复盘，从而让试点阶段“靠经验兜底”的成本逐步下降。

6.3.1.2.3 Python SBOM 供应链治理——让 AI 环境可复现、可审计、可回滚

> 本章所述能力目前在龙蜥操作系统侧仍处于规划与逐步建设阶段，目标与路线相对明确，整体能力后续会随着社区与生态进展逐步工程化落地。

Python 生态的繁荣为用户带来了效率，当然也提高了复杂度。对 AI 负载而言，Python 依赖往往更深、组合更多、与底层系统依赖耦合更强，因此“环境能否复现、升级能否回滚、风险能否审计”会更快变成发行版与平台层的问题。具体表现为：

- 环境与依赖漂移导致复现困难，“装得上≠跑得稳”；
- 多后端与系统依赖差异带来一致性挑战；
- 深层依赖漏洞或投毒扩散面广，风险暴露面扩大；
- 依赖组合升级后回滚路径不清晰，影响面大。

围绕上述问题，单靠应用侧“锁版本/写文档”往往难以长期维持一致性；更可持续的做法，是把关键依赖组合、来源与可回滚路径纳入发行版的交付链路，让环境在安装、升级、回退与问题复现时都有一致的依据与抓手。基于这一思路，龙蜥操作系统计划以发行版交付为牵引，逐步建设以 Python SBOM 为核心的供应链能力，并在框架与异构生态上选择若干“锚点”优先推进工程化落地。

- Python SBOM 作为供应链能力底座：把“装了什么、来自哪里、依赖是什么、风险在哪里、出了问题如何回滚”变成发行版交付链路的一部分，并与 OS/镜像 SBOM、漏洞处置与版本选型联动。
- PyTorch 作为“框架锚点”：把 PyTorch 版本、依赖与后端组合纳入更可控的交付与回滚体系，降低环境构建与线上漂移成本。

另外，龙蜥社区已成立智算联盟，并与 FlagOS 社区协同推进，面向多种 AI 芯片使用过程中的核心痛点问题，目标是打造更开放兼容的 AI 开源新基座。 在此协同背景下，借助 FlagOS 在“打破不同芯片软件栈之间生态隔离”的能力，龙蜥希望推动异构生态从“碎片化适配”走向“可复用”，通过“一次编译、多次运行”等方向提升算子/编译层复用性，减少重复适配投入，提高生态可持续性。

当 Python SBOM 能力逐步落地，AI 环境更容易做到可复现、可回滚、风险更可解释；用户更少被依赖组合与环境漂移消耗精力，系统维护也更容易形成长期可持续的工程节奏。

6.3.1.3 AI for System - 用 AI 发展龙蜥操作系统

6.3.1.3.1 自然语言运维与 AIOps——从“执行指令”到“表达意图”

系统与栈的复杂度提升，使运维越来越依赖经验与上下文：跨层定位难、重复劳动多、知识沉淀难，最终影响稳定性与效率。 龙蜥在运维侧的重要载体之一是由系统运维 SIG 建设的自动化运维平台 SysOM。SysOM 希望在统一入口下覆盖主机管理、系统监控、异常诊断、日志审计与安全管控等典型运维链路，降低工具碎片化带来的成本，并降低 OS 问题分析门槛。

截至目前，SysOM 已形成一站式运维能力底座，集成主机管理、监控、诊断、审计、安全等关键能力，为进一步叠加自然语言交互与 AIOps 提供基础承载面。 围绕“自然语言运维与 AIOps”的方向，SysOM 进一步发布了 SysOM Agent 作为 SysOM 的智能助手，SysOM Agent 接入 SysOM MCP 的诊断能力，通过 MCP 把复杂的运维操作封装为 AI 可直接调用的标准工具，使 AI Agent 能够在真实环境中执行诊断任务并输出系统级分析。用户无需掌握命令行，仅通过自然语言提问即可获得更接近工程师视角的诊断结论与分析过程。

6.3.1.3.2 AI 辅助研发——面向开源 OS 研发的智能工程助手

开源操作系统研发具有典型特征：代码规模大、子系统多、并发修改频繁、回归路径长，且缺陷形态复杂（并发、内存、边界条件、错误处理、ABI/兼容性等）。维护者与贡献者普遍面临以下挑战：

- 代码评审负担重：review 需要理解上下文、历史变更、子系统约束与潜在安全影响，且容易漏掉细微但高风险的问题。
- 测试构建与覆盖不足：补丁引入的问题往往出现在极端配置、特定架构或罕见时序下，常规测试难以覆盖。
- 跨版本线移植复杂：同一修复在不同版本/分支上存在 API 差异、依赖顺序与冲突点。
- 知识分散：内核/发行版经验散落在邮件列表、提交记录、issue 与文档中，新贡献者学习曲线陡峭。

龙蜥社区成员借助 AI 致力于让维护者把更多精力用于关键判断与架构把控，减少重复性的检索、摘要与机械检查工作；让贡献者更容易写出符合子系统约束的补丁，并在提交前更充分地自测与自检；追求测试覆盖与问题定位更系统化，减少后期或线上才暴露的低级回归；降低跨分支维护与长期支持的工程负担，提升社区协作效率。

目前在社区内部的构建发布测试基础设施中，已经在试点使用 CodeReview agent、测试用例生成 Agent、稳定性测试 Agent、测试看护 Agent 等能力，以提升评审与测试效率并降低回归定位成本。

6.3.1.3.3 AI 辅助安全——让 CVE 处理形成可追溯的工程闭环

CVE 处理不仅要修复，还要及时、正确、可验证、可审计；多版本线与多栈叠加使影响面评估与回归验证成本高，且人工流程容易出现遗漏或重复劳动。社区维护者规划在 CVE 处理的不同阶段引入 AI 能力：

- 从披露到评估：AI 辅助影响面分析（版本线/组件/SBOM）。
- 从修复到发布：AI 辅助补丁候选与适配建议、回归验证与风险提示。
- 从发布到审计：与 SBOM 联动形成可追溯闭环。

目前在社区内部的 CVE 跟踪、导入及自动化评估等阶段，已经在试点使用相关 AI 能力，以改善平均修复时延并降低回归率。

6.3.1.3.4 SBOM 与版本选型智能化——让“选什么、为什么、风险是什么”可解释

对于每个龙蜥发行版本来说，社区维护者需要考虑面向未来 10 年的演进，内核、关键库、Python 与 AI 栈版本选型在稳定性、安全性、性能与生态兼容之间权衡复杂，需要全链路可追溯与可解释，支撑长期维护与治理。为了让版本矩阵更透明、选择更可解释，升级更稳、回滚更明确，供应链治理成本下降。龙蜥社区也在探索借助 AI，实现：

- OS SBOM 与 Python SBOM 协同治理，支撑版本选型、升级/回滚计划与审计输出。
- 用 AI 辅助做版本风险评估与推荐矩阵生成，降低决策成本。

该方向目前仍处于社区调研规划阶段，初步计划在下一个发行版大版本中借助相关能力开展版本选型与风险评估方面的实践与落地。

6.3.2 LLM 推理技术栈

6.3.2.1 SGLang 高性能 LLM Serving 框架

SGLang 是一个专为大规模语言模型推理设计的高性能 serving 框架，通过结构化生成语言（Structured Generation Language）和先进的调度优化技术，提供高效的模型服务能力。随着大语言模型应用场景的不断扩展，从简单的文本生成到复杂的多模态理解，推理系统需要处理越来越复杂的任务类型和多样化的性能需求。SGLang 通过其独特的设计理念，将结构化生成与高性能推理相结合，为开发者提供了一个既强大又易用的推理平台。该框架不仅关注吞吐量和延迟等传统性能指标，还特别强调输出的可控性和一致性，这对于生产环境中的实际应用至关重要。

6.3.2.1.1 架构概述

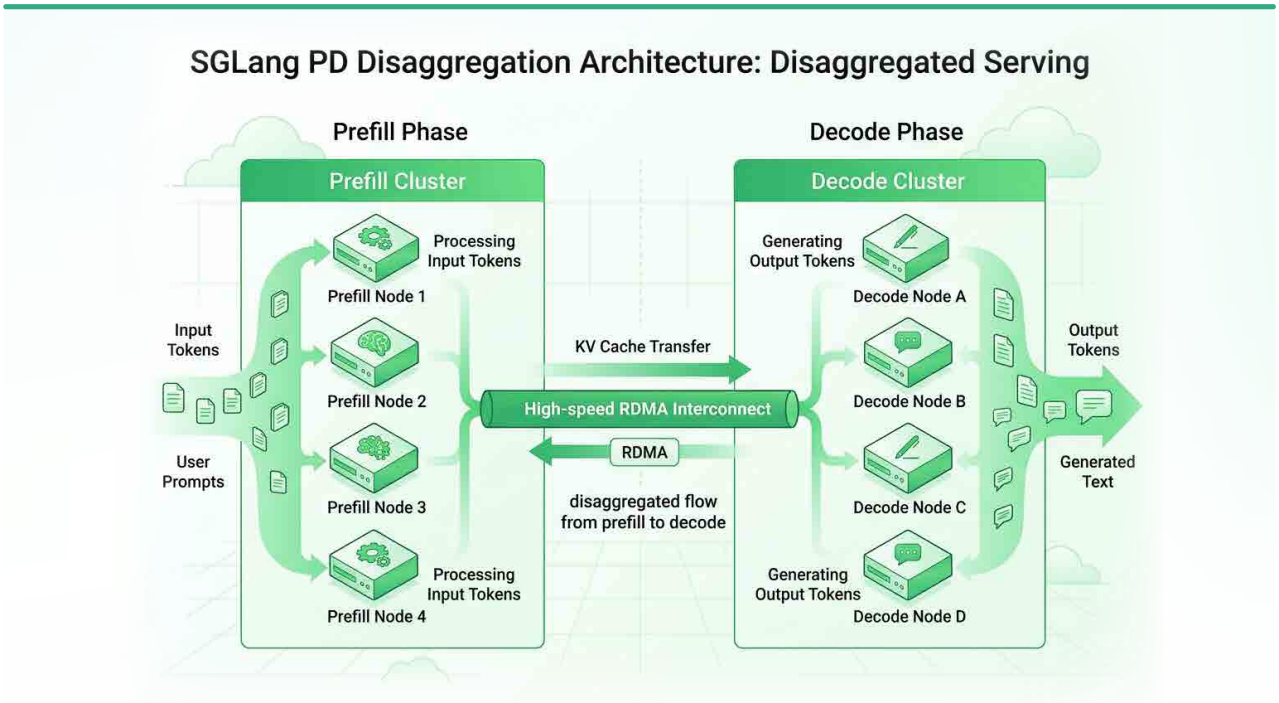
SGLang 采用模块化设计，这种设计哲学使得框架具有高度的可扩展性和可维护性。每个模块负责特定的功能，模块之间通过定义良好的接口进行通信，这种松耦合的架构使得开发者可以轻松地替换或扩展特定组件。

Frontend 模块负责处理请求接收和结构化生成。它是用户与 SGLang 交互的主要入口，负责解析用户请求，执行结构化生成逻辑，并将结果返回给用户。Frontend 支持多种输入格式，包括纯文本、结构化模板和程序化的生成逻辑。

Scheduler 是智能调度器，支持多种调度策略。它是 SGLang 性能优化的核心，负责决定何时以及如何在 Worker 上执行请求。Scheduler 需要平衡多个目标：最大化吞吐量、最小化延迟、保证公平性。SGLang 的 Scheduler 采用了先进的连续批处理技术，可以动态调整批处理大小以适应变化的负载。

Worker 模块执行实际的推理计算。每个 Worker 管理一个或多个 GPU，负责加载模型权重并执行前向传播。Worker 实现了多种优化技术，包括 PagedAttention、FlashAttention 等，以最大化 GPU 利用率。

Communication Layer 提供高效的跨节点通信能力。对于分布式部署，这是至关重要的组件。SGLang 支持多种通信后端，包括基于 RDMA 的高性能传输和基于 TCP 的通用传输，用户可以根据硬件环境选择合适的后端。



6.3.1.1.2 PD 分离（Prefill-Decode Disaggregation）

PD 分离是 SGLang 的核心优化技术，将预填充和解码阶段分配到不同的计算资源，实现独立扩展。这一技术的出发点是对 LLM 推理两个阶段的深入理解：预填充阶段和解码阶段具有截然不同的计算特征和资源需求。

预填充阶段是计算密集型的。在这个阶段，模型需要处理输入提示（prompt），计算所有 token 的注意力权重，并生成初始的 KV Cache。这个阶段涉及大量的矩阵乘法运算，对计算能力有很高的要求。然而，预填充阶段的持续时间相对较短，一旦完成，就不会再有新的计算需求。

解码阶段是内存密集型的。在这个阶段，模型以自回归的方式逐个生成输出 token。每个新生成的 token 都需要访问之前所有 token 的 KV Cache，因此解码阶段主要受限于内存带宽而非计算能力。解码阶段可能持续很长时间，特别是对于长输出序列。

传统的合署部署方式将两个阶段放在同一组 GPU 上，这导致了资源利用率的问题。当预填充阶段运行时，解码阶段必须等待；当解码阶段运行时，预填充阶段无法执行。这种互斥执行模式限制了系统的整体吞吐量。

架构优势

PD 分离通过资源专精化解决了上述问题。预填充节点针对计算密集型预填充阶段优化，配置更多的计算单元以最大化吞吐。这些节点可以处理更大的批处理，因为预填充阶段的计算可以很好地并行化。

解码节点针对内存密集型解码阶段优化，配置更大的显存以支持更大批处理。解码阶段的批处理受限于 KV Cache 的显存占用，因此解码节点需要更大的显存容量。通过专门的优化，解码节点可以在内存带宽限制下实现最高的 token 生成速度。

独立扩展是 PD 分离的另一个重要优势。根据负载特征，可以分别扩展预填充和解码集群。如果系统收到大量短请求，可能需要更多的预填充节点；如果系统需要生成长文本，可能需要更多的解码节点。这种灵活性使得系统可以适应不同的工作负载模式。

KV Cache 传输

PD 分离的关键技术挑战是如何高效地将 KV Cache 从预填充节点传输到解码节点。KV Cache 可能非常大，特别是对于长序列，因此传输效率直接影响整体性能。

SGLang 通过高速 RDMA 网络实现高效的 KV Cache 传输。RDMA 允许网络适配器直接读写远程内存，无需 CPU 参与，这大大降低了传输延迟和 CPU 开销。对于跨节点的 PD 分离部署，RDMA 是必不可少的。

SGLang 支持多种传输后端以适应不同的部署环境。zmq_to_scheduler 是默认的 RDMA 后端，提供了最佳的性能。zmq_to_tokenizer 是一个更通用的后端，适用于没有 RDMA 的环境。mooncake 后端集成了 Mooncake 的高性能通信原语，适用于超大规模部署。

优化的序列化/反序列化减少了传输开销。SGLang 使用高效的二进制格式，避免不必要的内存拷贝。流水线重叠传输与计算进一步隐藏了传输延迟，预填充节点可以在传输前一个请求的 KV Cache 的同时，开始处理下一个请求。

与 Mooncake 集成

SGLang 与 Mooncake 团队深度合作，集成了 Mooncake 的高性能通信原语。这种集成使得 SGLang 可以充分利用 Mooncake 在超大规模部署方面的优势。

通过使用 Mooncake EP，SGLang 实现了高效的专家并行通信。这对于服务 MoE 模型至关重要，因为 MoE 模型需要频繁的专家间通信。Mooncake 的优化通信内核可以将通信开销降到最低。

SGLang 还支持 Mooncake 的容错机制，提升服务可靠性。在大规模部署中，硬件故障是不可避免的。Mooncake 的 EEP 技术可以在 GPU 故障时快速恢复，确保服务不中断。

此外，SGLang 利用 Mooncake 的 RDMA 优化加速 KV Cache 传输。Mooncake 在 RDMA 方面做了大量优化，包括零拷贝传输、自适应路由等，这些优化都被 SGLang 所继承。

6.3.1.1.3 EPD（Encoder-Prefill-Decode）三层分离

EPD 是 SGLang 针对多模态模型的扩展架构，将视觉编码、语言预填充和解码三个阶段完全分离。随着多模态大模型的兴起，传统的 PD 分离已经不能满足需求，因为多模态模型还包含视觉编码阶段。

视觉编码阶段使用 Vision Transformer（ViT）处理输入图像，生成视觉嵌入。这个阶段是计算密集型的，但计算模式与语言预填充不同。ViT 通常比语言模型小得多，但需要对每个图像进行完整的前向传播。

语言预填充阶段接收视觉嵌入，将其与文本 token 结合，进行语言模型的预填充计算。这个阶段既要处理文本，也要处理视觉信息，计算复杂度较高。

解码阶段与纯语言模型相同，负责自回归地生成输出 token。

架构设计

EPD 架构包含三个独立的 Server 类型。Encoder Server 负责 ViT 前向传播，生成视觉嵌入。它通过 `--encoder-only` 参数启动，只执行视觉编码任务。Encoder Server 支持前缀多模态缓存，可以缓存常见图像的嵌入，避免重复计算。

Prefill Server 接收来自 Encoder Server 的视觉嵌入，将其与文本 token 结合，处理预填充计算。它通过 `--language-only` 参数启动。Prefill Server 需要与 Encoder Server 协调，确保视觉嵌入及时到达。

Decode Server 是标准的解码节点，接收来自 Prefill Server 的 KV Cache，负责生成输出 token。它是 decode-only 的，不处理视觉信息。

关键发现

EPD 研究揭示了一个反直觉的发现：视觉 Transformer（ViT）不会从增加张量并行度中受益。这与语言模型的情况截然不同，语言模型通常可以从更高的张量并行度中获得显著加速。

基准测试显示，TP=8 配置下视觉编码需要 523.80ms，而 TP=4 配置下只需要 465.80ms。更高的并行度反而更慢，这是因为 ViT 的权重相对较小，通信开销在总时间中占比较大。当增加并行度时，通信开销的增长超过了计算收益。

因此，EPD 采用水平数据并行策略，将图像分布到多个编码器实例，而非增加张量并行度。这种策略可以更好地扩展视觉编码能力，而不会受到通信开销的限制。

性能表现

对于图像密集型工作负载（约 4 张图像/请求），EPD 展现出显著的性能优势。在 1 QPS 时，EPD 比同地部署降低 6-8 倍延迟。这是因为三个阶段可以并行执行，而不需要等待前一个阶段完成。

在高 QPS 时，吞吐量大约翻倍。独立扩展使得每个阶段都可以根据需求进行优化，消除了合署部署中的资源争用。

独立扩展还意味着视觉编码容量可独立扩展，不影响语言模型部署。如果应用需要处理大量图像，可以简单地增加 Encoder Server 的数量，而不需要调整语言模型的配置。

6.3.1.1.4 调度优化

连续批处理（Continuous Batching）

SGLang 实现了高效的连续批处理机制，这是提升吞吐量的关键技术。传统的静态批处理等待一批请求全部到达后才开始处理，这导致了延迟增加和 GPU 空闲。

SGLang 的动态批处理大小调整可以适应不同请求长度。当一个新请求到达时，如果当前批处理还有空间，它可以被立即加入正在进行的批处理中。这种动态调整使得 GPU 可以保持高利用率，而不需要等待完整的批处理行程。

抢占和恢复机制支持优先级调度。高优先级请求可以抢占低优先级请求的资源，确保关键任务得到及时处理。被抢占的请求会被保存状态，稍后恢复执行。

内存高效管理减少了显存碎片。SGLang 使用 PagedAttention 技术，将 KV Cache 分页管理，避免了传统连续内存分配导致的碎片问题。这使得系统可以支持更大的批处理和更长的序列。

投机解码（Speculative Decoding）

投机解码是降低延迟的有效技术，特别适用于低并发场景。其核心思想是使用较小的草稿模型预测多个 token，然后用原始模型并行验证这些预测。

草稿模型通常比原始模型小得多，因此可以快速生成候选 token。原始模型并行验证这些候选，如果验证通过，就可以一次接受多个 token，而不是逐个生成。

这种方法显著降低了端到端延迟，特别是在低并发场景。当只有一个或少数几个请求时，批处理的优势无法发挥，投机解码提供了一种替代方案来加速生成。

前缀缓存（Prefix Caching）

前缀缓存针对多模态场景进行了优化。视觉嵌入缓存避免了重复编码相同图像，这对于包含重复图像的请求序列特别有效。

文本前缀缓存加速了共享上下文的请求。在多轮对话或批量处理相似请求时，前缀缓存可以避免重复计算相同的初始 token。

多级缓存策略平衡了内存使用和命中率。SGLang 使用 LRU 等策略管理缓存，确保最常用的前缀保持在缓存中，同时限制总内存使用。

结构化生成

SGLang 提供强大的结构化生成能力，这是其区别于其他推理框架的重要特性。结构化生成确保模型输出符合预定义的结构，这对于将 LLM 集成到应用程序中至关重要。

JSON Schema 约束确保输出符合预定义结构。开发者可以指定输出的 JSON 格式，包括字段类型、必填字段等。SGLang 会在解码过程中强制执行这些约束，确保生成的 JSON 是有效的。

正则表达式约束通过正则控制输出格式。这对于需要特定格式的输出（如电话号码、邮箱地址等）非常有用。SGLang 使用高效的正则引擎，在解码过程中实时检查约束。

上下文无关文法（CFG）支持复杂语法规则。对于更复杂的结构化需求，开发者可以使用 CFG 定义输出语法。SGLang 的 CFG 支持使得它可以处理嵌套结构、递归定义等复杂情况。

高效解码算法避免无效生成。传统的约束解码方法会生成大量无效 token，然后过滤掉。SGLang 使用约束感知的解码算法，只生成满足约束的 token，大大提高了效率。

6.3.1.1.5 在龙蜥操作系统上的优化

龙蜥操作系统为 SGLang 提供了全面的优化支持，使得 SGLang 在龙蜥上可以发挥出最佳性能。

网络优化

RDMA 内核优化降低了 RDMA 通信延迟，提升跨节点传输效率。龙蜥内核针对 RDMA 进行了深度优化，包括优化的驱动程序、减少的中断处理等。这些优化对于 PD 分离部署至关重要，因为频繁的 KV Cache 传输对网络性能非常敏感。

GPUDirect RDMA 支持 GPU 直接访问远程内存，减少数据拷贝。在传统的 RDMA 传输中，数据需要从 GPU 内存拷贝到 CPU 内存，然后通过 RDMA 发送，接收方也需要反向拷贝。GPUDirect RDMA 允许 GPU 直接发起 RDMA 操作，消除了这些不必要的拷贝。

网络拓扑感知调度器可以感知网络拓扑，优化任务放置。在多机架部署中，同一机架内的节点之间的网络延迟通常低于跨机架的延迟。龙蜥的拓扑感知调度可以将通信频繁的节点放置在同一机架，减少网络延迟。

内存管理

大页支持优化了大页配置，减少 TLB miss。大页可以减少页表项的数量，降低 TLB 压力，提高内存访问性能。龙蜥提供了方便的大页配置工具，简化了大页的设置。

NUMA 优化实现了智能内存分配，提升多节点性能。龙蜥内核可以感知 NUMA 拓扑，将内存分配在访问速度最快的节点上。对于多 GPU 服务器，这意味着 GPU 和 CPU 内存之间的数据传输可以走最快的路径。

显存优化与龙蜥内核协同，优化 GPU 显存使用。龙蜥提供了显存监控和管理工具，帮助用户了解显存使用情况，优化模型配置。

容器化部署

AI 容器镜像提供了预配置 SGLang 的容器镜像。龙蜥提供了包含 SGLang 及其所有依赖的容器镜像，用户无需手动安装复杂的软件栈，可以快速开始部署。

资源隔离基于龙蜥容器技术实现 GPU 资源隔离。在多租户环境中，资源隔离确保不同用户的任务不会相互干扰。龙蜥的容器技术支持细粒度的 GPU 资源分配。

弹性伸缩与 Kubernetes 集成，支持自动扩缩容。SGLang 可以部署在 Kubernetes 集群上，利用 Kubernetes 的自动扩缩容能力，根据负载自动调整实例数量。

性能基准

在龙蜥操作系统上的典型性能表现如下表所示。这些基准测试在标准硬件配置下进行，展示了 SGLang 不同配置相对于基线的性能提升。

配置	吞吐 (tokens/sec)	延迟 (ms)
单节点 PD 合署	基准	4.838s
PD 分离（4P+12D）	+120%	-45%
EPD 多模态	+85%	-60%（图像密集）

PD 分离配置展示了显著的性能提升，吞吐量提升 120%，延迟降低 45%。这证明了分离式架构在高负载场景下的优势。

EPD 多模态配置在图像密集型工作负载中表现尤为出色，延迟降低 60%。这验证了 EPD 三层分离架构在处理多模态任务时的有效性。

参考链接

- SGLang GitHub：https://github.com/sgl-project/sglang
- EPD: Encoder-Prefill-Decode Disaggregation：https://www.lmsys.org/blog/2026-01-12-epd/
- Mooncake EP Integration：https://lmsys.org/blog/2025-05-05-large-scale-ep/

6.3.2.2 Mooncake 大模型推理加速框架

Mooncake 是专为大规模语言模型（LLM）推理设计的分布式 serving 框架，由 Moonshot AI 与清华大学联合研发，现已成为 PyTorch 生态系统正式成员。该框架通过创新的 Expert Parallelism（EP）技术和 PD 分离架构，显著提升了 MoE（Mixture of Experts）模型的推理性能和资源利用率，专门解决 LLM 服务中的"内存墙"（memory wall）问题。

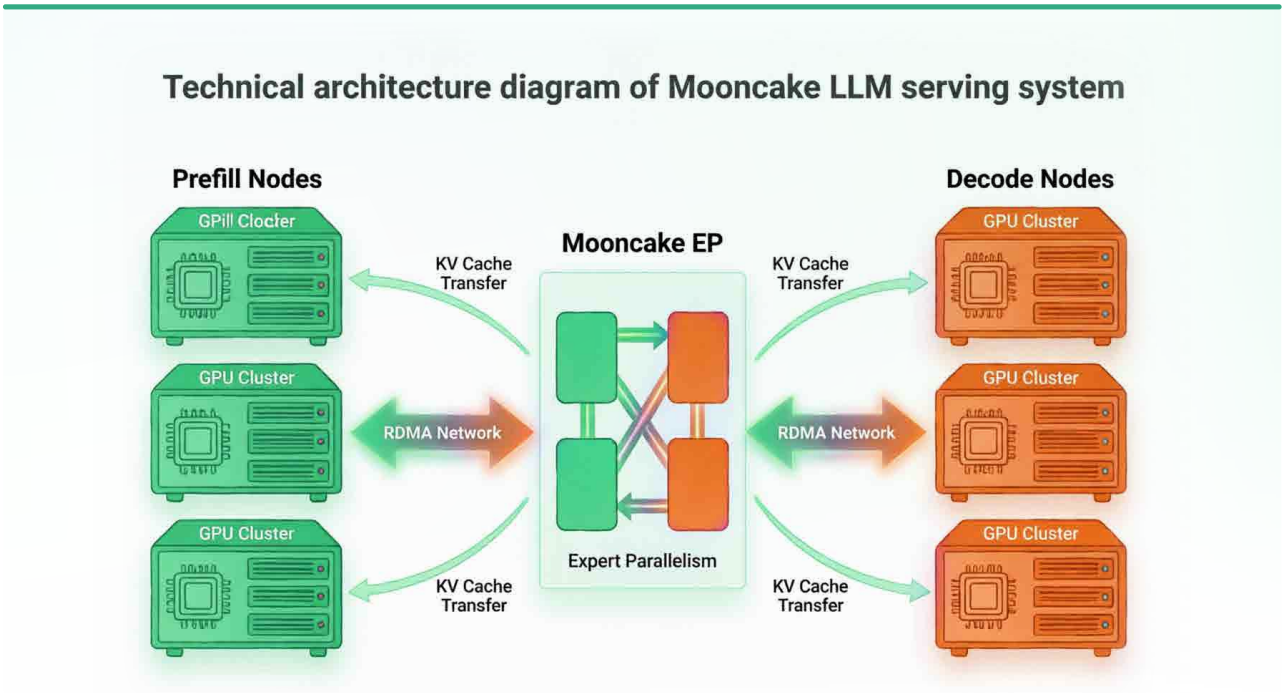
随着大语言模型规模的不断增长，尤其是 Mixture of Experts（MoE）架构模型的兴起，如何在保证低延迟的同时实现高吞吐量成为推理系统面临的核心挑战。传统的张量并行（Tensor Parallelism）和流水线并行（Pipeline Parallelism）方案在处理 MoE 模型时面临显存瓶颈和通信开销过大的问题。Mooncake 通过以 KV Cache 为中心的分离式架构设计，结合 Expert Parallelism 技术，为大规模 MoE 模型推理提供了一套完整的解决方案。

Mooncake 的核心定位是分布式 KV Cache 传输与存储系统，其五大核心功能包括 Prefill-Decode 分离、全局 KV Cache 复用、弹性专家并行（EP）、PyTorch 分布式后端支持以及权重快速更新。这些功能使 Mooncake 能够在大规模生产环境中提供高性能、高可用的推理服务。

6.3.2.2.1 架构概述

Mooncake 采用以 KV Cache 为中心的分离式架构，这是其区别于传统推理框架的核心理念。在传统架构中，预填充（Prefill）和解码（Decode）阶段通常运行在同一组 GPU 上，这导致资源利用率低下，因为两个阶段具有截然不同的计算特征：预填充阶段是计算密集型的，需要处理长序列的矩阵运算；而解码阶段是内存密集型的，主要受限于 KV Cache 的读写带宽。

Mooncake 将预填充和解码阶段分配到不同的计算节点，通过高速 RDMA 网络实现高效的 KV Cache 传输。这种设计使得两个阶段可以独立扩展，最大化 GPU 利用率。预填充节点可以配置为拥有更强的计算能力以处理计算密集型任务，而解码节点则可以配置更大的显存以支持更大的批处理大小。



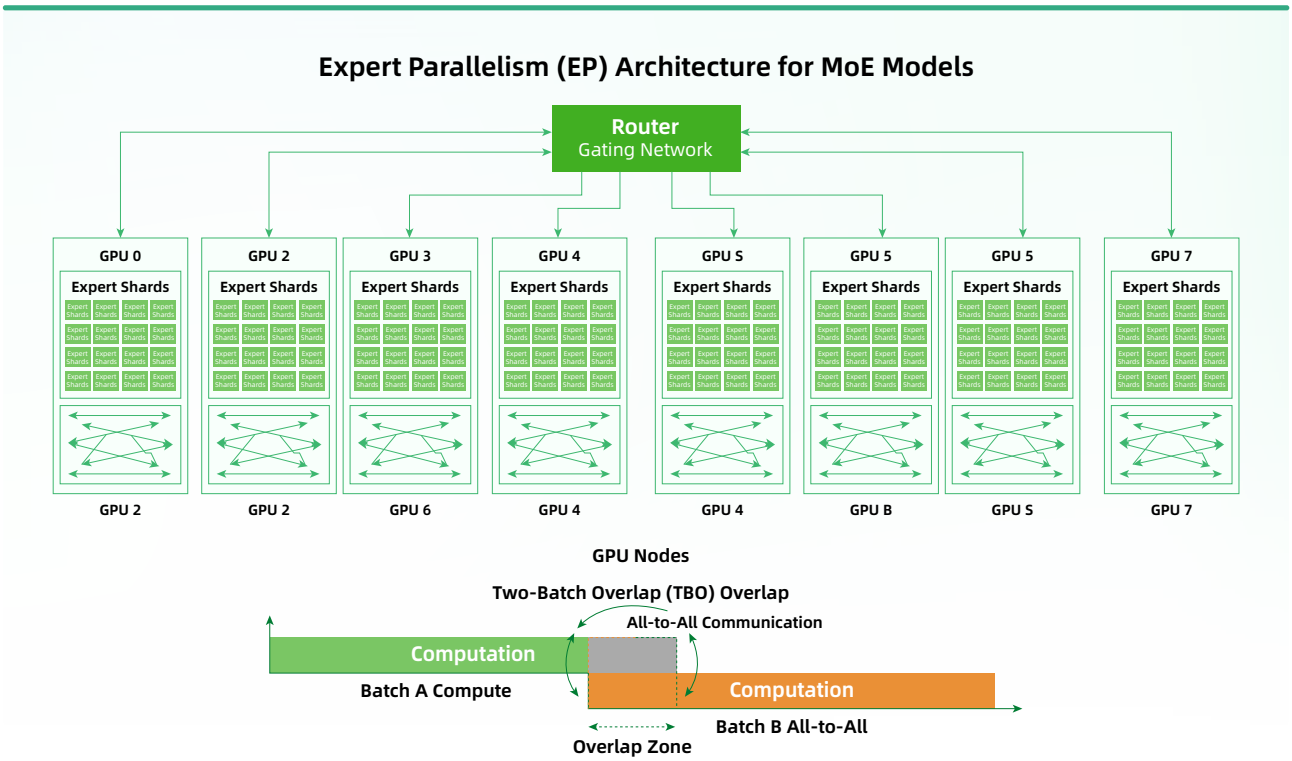
在 Mooncake 的架构中，请求首先到达预填充节点，完成初始的注意力计算和 KV Cache 生成。随后，生成的 KV Cache 通过 RDMA 网络传输到解码节点，解码节点负责自回归地生成输出 token。这种分离不仅提高了资源利用率，还使得系统可以根据实际负载动态调整预填充和解码节点的比例，实现弹性扩展。

Mooncake 架构包含三个核心组件：Mooncake Transfer Engine（TE）、Mooncake Store 和 Mooncake Master。Transfer Engine 负责高效的 KV Cache 传输，支持 RDMA 和 NVLink 零拷贝传输；Mooncake Store 作为分布式共享内存，实现跨请求、跨引擎实例的全局缓存复用；Mooncake Master 是集中式元数据服务器，管理 TE 和 Store 的元数据。这种分层架构使得 Mooncake 可以灵活地适应不同的部署环境和应用场景。

6.3.2.2.2 Expert Parallelism（EP）技术

Expert Parallelism 是 Mooncake 的核心技术，专门用于解决 MoE 模型中的显存瓶颈问题。在 MoE 模型中，每个输入 token 只会激活一小部分专家（experts），但完整的专家权重需要加载到显存中。对于拥有数百个专家的大规模 MoE 模型，这会导致显存需求超出单卡容量。

EP 将专家权重分布到多个设备上，通过优化的通信内核实现高效的全对全（All-to-All）通信。具体而言，每个 GPU 只存储一部分专家的权重，当 token 需要路由到存储在其他 GPU 上的专家时，通过 All-to-All 通信交换 token 和对应的激活值。这种设计使得系统可以支持远超单卡显存容量的专家数量。



双模式调度机制

Mooncake EP 支持两种调度模式以适应不同阶段的需求。Normal Dispatch 模式针对预填充阶段优化，追求最大吞吐量，适用于计算密集型的预填充任务。该模式会聚合更多的 token 进行批量处理，减少通信开销。Low-Latency Dispatch 模式则针对解码阶段优化，支持 CUDA Graph 技术，显著降低延迟。解码阶段对延迟极为敏感，因此该模式采用轻量级的通信原语，确保每个请求都能得到快速响应。此外，Auto Mode 可以根据工作负载的特征动态选择调度模式，在保证延迟的同时最大化吞吐量。

两批次重叠技术

TBO（Two-batch Overlap）技术是 Mooncake 提升 GPU 利用率的关键创新。传统的 EP 实现中，通信和计算是串行执行的：先进行 All-to-All 通信将 token 发送到目标 GPU，然后等待通信完成后才开始专家计算。这种串行模式导致 GPU 在通信期间处于空闲状态。

TBO 技术将单个批次拆分为两个微批次，使计算和通信能够重叠执行。具体而言，当第一个微批次在进行专家计算时，第二个微批次可以同时进行 All-to-All 通信。这种流水线并行策略显著提高了 GPU 利用率，减少了空闲等待时间。在实际部署中，TBO 可以将 GPU 利用率从 60% 提升到 85% 以上。

专家并行负载均衡器

EPLB（Expert Parallelism Load Balancer）解决了 EP 中的负载不均衡问题。由于不同专家的激活频率不同，简单的轮询分配会导致某些 GPU 过载而其他 GPU 空闲。EPLB 使用 256 个活跃专家加 32 个冗余专家的配置，通过智能路由算法最小化工作负载不平衡。

冗余专家机制允许系统将过载的专家复制到冗余槽位，从而分散负载。更重要的是，EPLB 支持非 2 的幂次并行度，如 EP12、EP72 等，这提供了更灵活的扩展能力。传统的 EP 实现通常要求专家数量是并行度的整数倍，这限制了部署的灵活性。EPLB 打破了这一限制，使得用户可以根据实际硬件配置选择最优的并行度。

性能表现

在大规模部署场景下，Mooncake EP 展现出卓越的性能。EP72 配置下，解码阶段单节点可以达到 22,282 tokens/sec，相比 TP16 基线提升 5.2 倍。这一提升来自于 EP 对显存瓶颈的有效缓解，使得更大的批处理成为可能。

EPLB 负载均衡器进一步提升了性能，预填充阶段加速 1.49 倍，解码阶段加速 2.54 倍。负载均衡对于 EP 的性能至关重要，不均衡的负载会导致部分 GPU 成为瓶颈，拖慢整个系统的速度。通过智能的冗余专家分配，EPLB 确保了所有 GPU 的负载大致均衡。

K2 部署实践展示了 Mooncake 支持超大规模专家的能力，支持高达 384 个专家的动态路由，使用 96 个冗余专家平衡负载。这种规模的支持使得 Mooncake 可以服务于当前最大规模的 MoE 模型。

6.3.2.2.3 Elastic Expert Parallelism（EEP）容错机制

EEP 是 Mooncake 的高可用性扩展，实现了部分故障容错能力，确保在 GPU 故障时服务不中断。在大规模 GPU 集群中，硬件故障是不可避免的。传统的容错方案通常需要重启整个服务，导致数分钟的中断时间，这对于在线服务是不可接受的。

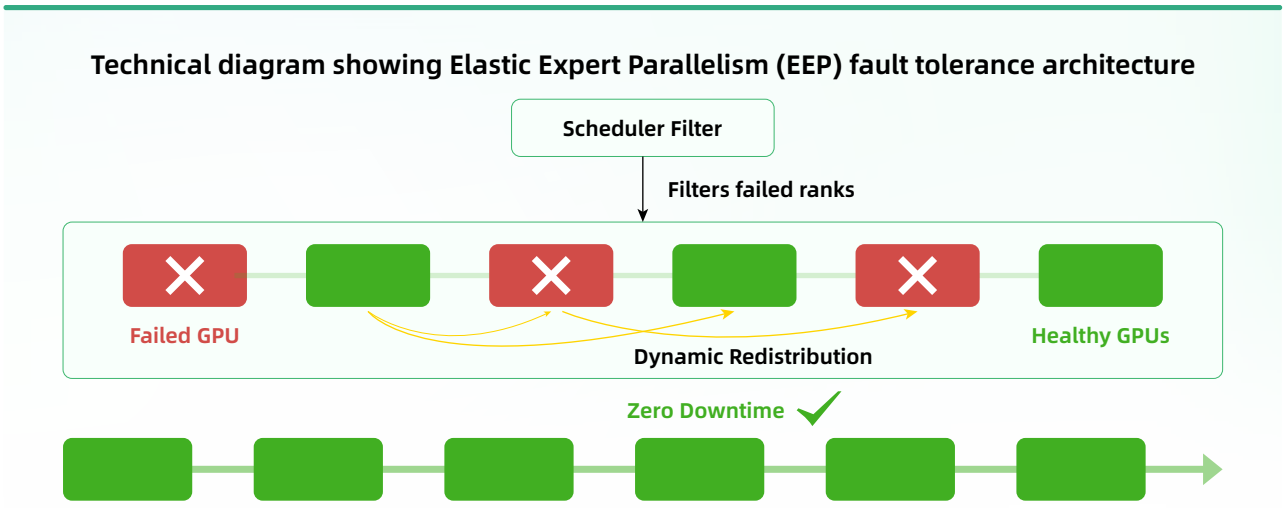
核心创新

EEP 打破了专家与特定 GPU 之间的刚性映射，这是实现快速故障恢复的关键。在传统 EP 中，每个专家固定映射到一个特定的 GPU，当该 GPU 故障时，对应的专家就无法访问。EEP 通过动态映射机制，允许专家权重在多个 GPU 之间迁移。

这种设计实现了秒级恢复能力，故障恢复时间从传统的 2-3 分钟缩短至 10 秒以内。恢复过程中，系统首先检测到故障 GPU，然后从备份中恢复受影响的专家权重到健康的 GPU，最后更新路由表将请求导向新的位置。整个过程对用户透明，正在处理的请求可能会经历短暂的延迟增加，但不会失败。

EEP 还实现了线性降级能力，吞吐量随剩余节点数线性缩放。测试数据显示，在 1-16 个 GPU 故障时，吞吐量从 5552 tokens/sec 平滑下降到 2825 tokens/sec。这种线性降级特性使得运维人员可以在部分硬件故障的情况下继续提供服务，而不是立即进行紧急维修。

重要的是，EEP 实现了零性能损耗。与标准 DeepEP 相比，正常工作时没有额外的性能开销。容错机制只在故障发生时才会激活，日常运行中不会消耗额外的资源。



架构分层

EEP 的架构分为两个主要层次。调度层（Scheduler Layer）负责过滤故障的 DP Rank，防止将请求路由到不健康的资源。调度器维护一个健康状态表，定期从各个节点收集心跳信息。当某个节点在指定时间内没有响应时，调度器将其标记为不健康，并停止向该节点发送新的请求。

专家并行层（EP Layer）负责动态将专家权重重新分布到存活的 GPU 上，实现无缝故障转移。每个专家权重在内存中维护多个副本，当主副本所在的 GPU 故障时，系统可以快速切换到备用副本。权重迁移通过 RDMA 网络进行，充分利用高速网络带宽。

关键配置参数

EEP 提供了灵活的配置选项以适应不同的可靠性需求。--ep-num-redundant-experts 参数控制容错容量，指定冗余专家数量。更多的冗余专家意味着更高的容错能力，但也会消耗更多的显存。用户需要在容错能力和资源消耗之间做出权衡。

--enable-elastic-expert-backup 参数启用内存中权重备份，实现快速恢复。当启用该选项时，系统会在多个 GPU 上维护专家权重的副本，故障时可以立即切换到备用副本，无需从磁盘或远程存储加载权重。

6.3.2.2.4 K2 大规模 EP 部署实践

K2 是 Mooncake 在大规模集群上的部署实践，展示了如何在生产环境中部署超大规模 EP。K2 采用 1P1D（1 Prefill + 1 Decode）架构，这是经过实践验证的高效配置。

> 具体配置包括 4 个 Prefill 节点和 12 个 Decode 节点（即 4P12D 配置），这种比例是根据典型的工作负载特征确定的。预填充阶段计算密集但持续时间较短，解码阶段内存密集但持续时间较长，因此需要更多的解码节点来处理持续的生成任务。

在批处理方面，K2 的 Decode 批大小可达 480，这是 EP 技术缓解显存瓶颈后的结果。大批处理显著提高了吞吐量，降低了每个 token 的平均处理成本。

K2 采用了多项优化技术。NUMA-aware GPU 分组优化了 NVLink 和 PCIe 的利用率，确保同一 NUMA 节点内的 GPU 之间通信走最快的路径。ep-dispatch-algorithm=dynamic 实现了动态专家负载均衡，系统会根据实时的负载情况调整专家分布。deepep-mode=low_latency 降低了解码延迟，对于延迟敏感的应用至关重要。RDMA 加速 KV 缓存传输，确保预填充和解码节点之间的高效数据流动。

6.3.2.2.5 与主流推理引擎集成

Mooncake 与 PyTorch 原生推理引擎深度集成，目前已在多个主流推理框架中得到支持，包括 SGLang、vLLM 和 Tensor-RT-LLM。这种广泛的集成使得用户可以在不同的技术栈中享受 Mooncake 带来的性能提升。

SGLang + Mooncake 集成

Mooncake 团队与 SGLang 社区深度合作，贡献了 PD 分离（Prefill-Decode Disaggregation）实现。SGLang 原生支持 `--disaggregation-transfer-backend mooncake` 和 `--hicache-storage-backend mooncake` 参数，可以方便地启用 Mooncake 作为传输和存储后端。

在生产级部署中，SGLang、Mooncake 和 SMG（Shepherd Model Gateway）可以组成联合方案。SMG 作为智能 PD 分离路由层，负责将请求路由到合适的预填充或解码节点。预填充节点通过 Mooncake Transfer Engine 将 KV Cache 推送到 Mooncake Store，解码节点从 Store 中拉取共享的 KV Cache 进行生成。这种架构实现了高吞吐预填充和低延迟生成的完美结合。

vLLM + Mooncake 集成

vLLM 通过 `MooncakeConnector` 作为 KV 传输连接器与 Mooncake 集成。用户可以通过配置 `kv-transfer-config` 参数启用 Mooncake 功能。

预填充节点配置为 KV 生产者，负责生成 KV Cache 并通过 Mooncake 传输；解码节点配置为 KV 消费者，从 Mooncake 接收 KV Cache 进行解码。这种配置方式使得 vLLM 用户可以无缝迁移到 Mooncake 架构，享受 PD 分离带来的性能提升。

TensorRT-LLM 支持

TensorRT-LLM 官方支持 Mooncake，用户可以在 NVIDIA 优化的推理引擎中使用 Mooncake 的分布式 KV Cache 管理能力。这为追求极致性能的用户提供了更多选择。

6.3.2.2.6 生产环境验证与行业应用

Mooncake 已在多家顶级科技公司和机构的生产环境中得到验证，包括 Moonshot AI（Kimi）、阿里云、蚂蚁集团以及 Approaching.AI、LightSeek Foundation 等。这些生产级部署证明了 Mooncake 在大规模、高并发场景下的稳定性和可靠性。

在这些实际应用中，Mooncake 成功支撑了日均数十亿次的推理请求，服务了数亿用户。特别是在长文本处理场景（如 Kimi 的智能助手），Mooncake 的全局 KV Cache 复用能力显著降低了重复计算，提升了用户体验。

6.3.2.2.7 在龙蜥操作系统上的部署优势

龙蜥操作系统为 Mooncake 提供了优化的底层支持，使得 Mooncake 在龙蜥上可以发挥出最佳性能。作为 PyTorch 生态系统成员，Mooncake 与龙蜥的 AI 软件栈深度集成，为用户提供开箱即用的部署体验。

高性能网络是龙蜥的一大优势。龙蜥内核针对 RDMA 和 GPUDirect RDMA 进行了深度优化，降低了跨节点通信延迟。这对于 Mooncake 至关重要，因为 Mooncake 的分离式架构依赖频繁的跨节点 KV Cache 传输。龙蜥的 RDMA 优化可以将通信延迟降低 20% 以上，显著提升端到端推理性能。

NUMA 亲和性是另一个关键优化点。龙蜥内核提供 NUMA 感知调度，确保进程和内存分配在同一 NUMA 节点，减少跨节点内存访问。对于多 GPU 节点，这意味着 GPU 和 CPU 内存之间的数据传输可以走最快的路径，最大化内存带宽利用率。

容器化部署方面，Mooncake 与龙蜥 AI 容器镜像无缝集成。龙蜥提供了预配置 Mooncake 依赖的容器镜像，包括优化的 RDMA 驱动、CUDA 运行时和 Mooncake 库。用户无需手动安装复杂的驱动和库，只需拉取镜像即可开始部署，大大简化了运维复杂度。

故障检测方面，龙蜥提供了快速的故障检测机制，配合 EEP 实现高可用服务。龙蜥内核可以更快地检测到 GPU 故障并通知用户空间程序，缩短故障发现时间，加快恢复速度。这对于需要 7x24 小时在线服务的生产环境至关重要。

6.3.2.2.8 权重快速更新与 RL 训练支持

Mooncake 支持张量原生、零拷贝的权重快速更新，这一特性对于 RL（强化学习）训练和 checkpoint 场景尤为重要。在大模型 RL 训练中，策略模型需要频繁更新，传统的权重加载方式涉及大量的数据拷贝，耗时且低效。

Mooncake 的权重快速更新机制允许直接在网络传输层更新模型权重，无需经过 CPU 内存中转。这种零拷贝更新方式可以将权重更新时间从数分钟缩短到数秒，使得在线学习、实时策略更新等场景成为可能。

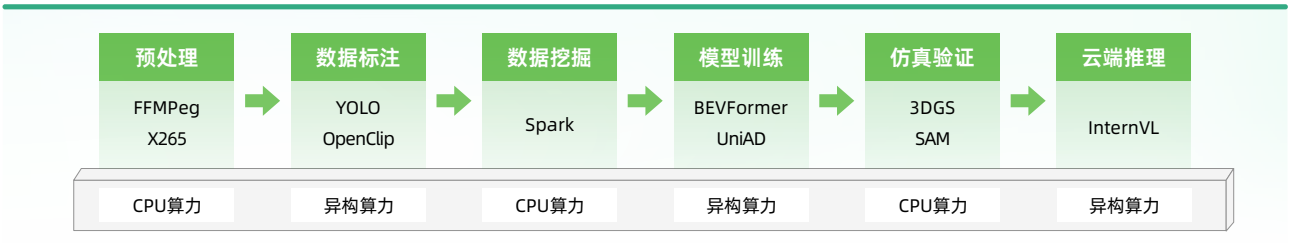
在 checkpoint 场景下，Mooncake 支持快速的模型状态保存和恢复。通过分布式存储后端，checkpoint 可以并行写入多个存储节点，大幅提升保存速度。恢复时，权重可以从最近的存储节点并行加载，缩短服务启动时间。

参考链接

- Mooncake：Large-Scale EP for MoE Models-https://lmsys.org/blog/2025-05-05-large-scale-ep/
- EEP：Partial Failure Tolerance-https://www.lmsys.org/blog/2026-03-25-eep-partial-failure-tolerance/
- K2：Large-Scale EP Deployment-https://www.lmsys.org/blog/2025-07-20-k2-large-scale-ep/
- Mooncake Joins PyTorch Ecosystem：https://pytorch.org/blog/mooncake-joins-pytorch-ecosystem/
- Mooncake GitHub：https://github.com/kvcache-ai/Mooncake
- Mooncake 文档：https://kvcache-ai.github.io/Mooncake/

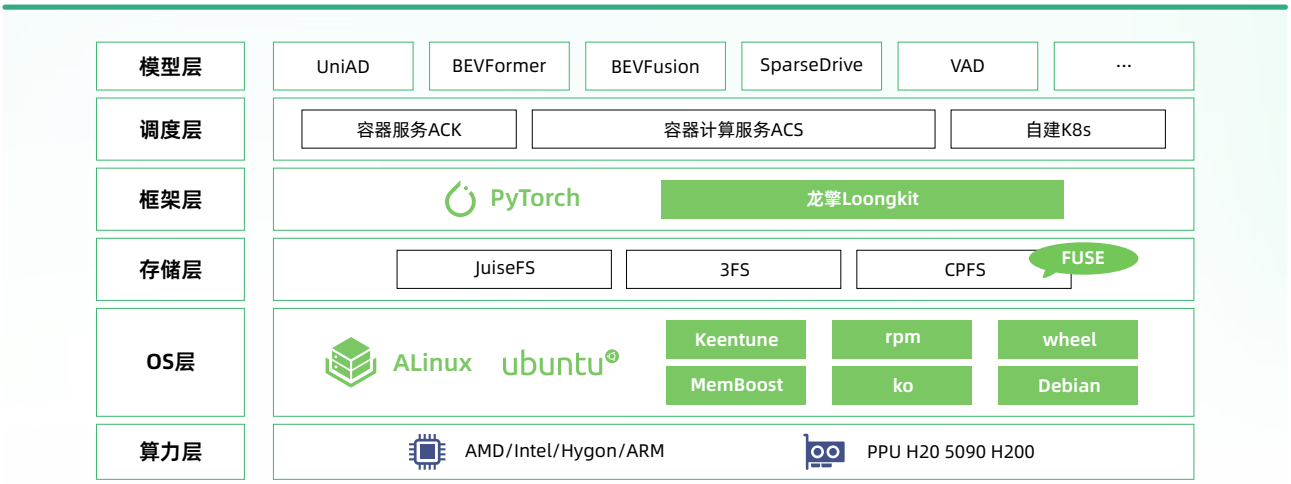
6.3.3 智驾场景技术栈

在 AI 时代，智能驾驶迎来了爆发式发展，也给阿里云带来了新的机遇和挑战。智能驾驶的技术链路很长，包括数据预处理、标注、挖掘、模型训练、仿真验证以及推理等关键步骤。



其中智驾模型训练由于算力需求强，数据量大等特点，特别适合云上环境，是与友商 PK 的主战场之一。基础软件构建了面向智驾模型训练全栈的技术赋能体系，提供了全栈优化方案，加速模型训练，提升阿里云竞争力。

智驾模型训练的整体方案架构图如下（绿色为基础软件特别优化的组件）：



整体上可以分为计算层和存储层。

计算层(软件栈)由以下几个主要层次组成：

- OS：操作系统是软件优化的底座，当前阿里云上智驾场景用的比较多的 OS 有 Alibaba Cloud Linux、Ubuntu 等。
  - 基础软件为 Alibaba Cloud Linux 3 以及 Ubuntu 提供了 Keentune，一款开机自动启动的 OS 服务插件，部署了 Keentune 的操作系统会自动将关键参数设置为最适合智驾训练场景的数值，提升系统性能。
- 调度层：主要进行容器管理和任务调度等工作。通常用户会选择 ACK 或 ACS，但也有客户会自建 K8S 集群管理资源。
- 框架层：
  - PyTorch：一个深度学习框架，具备动态计算图、易于使用、社区支持强等特点，因此被主流智驾模型广泛采用。Alibaba PyTorch 是阿里云自研的 PyTorch 发行版，针对开源 PyTorch 常见的 torch.compile 编译失败问题做了全面修复，提升性能和易用性。另外大幅改善了 PyTorch Profiler 的运行时开销以及 dump 数据时的应用暂停时间
  - 龙擎（Loongkit）：是阿里云推出的在基础软件视角下为 AI 框架层进行深度优化的软硬协同加速的 OS 扩展组件。专注于：
    - 专为 PyTorch 打造：LoongKit 专为与 PyTorch 无缝集成而设计，消除框架层特定场景的性能瓶颈。
    - 专为 PPU 实例打造：针对阿里云自研的 PPU 实例，提供针对性的性能优化，提升国产 GPU 的性价比。
    - 智驾场景的特别优化：在常见的智驾开源模型（Bevformer、UniAD、VAD 等）上实现 20%~100% 的训练加速。
- 模型层：智驾领域的模型种类繁多，有的负责感知与决策（BEVFormer/UniAD），有的负责识别标注（YOLO/OpenClip），也有目前比较新的 VLA 模型。基础软件对这些模型主要进行了下面几点优化：
  - 环境参数设置
  - 使用性能更优的组件
  - 优化关键算子
- 存储层：由于训练数据往往非常庞大，因此存储的性能非常关键。基于多机分布式训练诉求，推荐使用 CPFS。基础软件为 CPFS 提供了优化过的 FUSE（Filesystem in Userspace）特性，允许非特权用户在用户空间（userspace） 实现自定义的文件系统，大幅提升存储带宽及文件系统 IOPS 。

6.3.4 编译优化

6.3.4.1 Pytorch 及 torch.compile

PyTorch 是全球学术界和工业界最流行的深度学习框架之一，广泛用于开发、训练和部署各种深度学习模型。特别是在大语言模型的构建与优化过程中，PyTorch 提供了灵活且高效的支持,使得研究人员和工程师能够快速迭代和实现复杂的模型架构。例如，通义千问(Qwen)和 OpenAI 的 GPT 系列模型都是使用 PyTorch 开发和训练的。

torch.compile 是 PyTorch 2.0 中引入的图编译优化，它通过即时编译（JIT）技术，在运行时自动将你的 Python 代码转换为优化的机器代码,从而在最小化用户代码修改的情况下加速用户的模型性能。

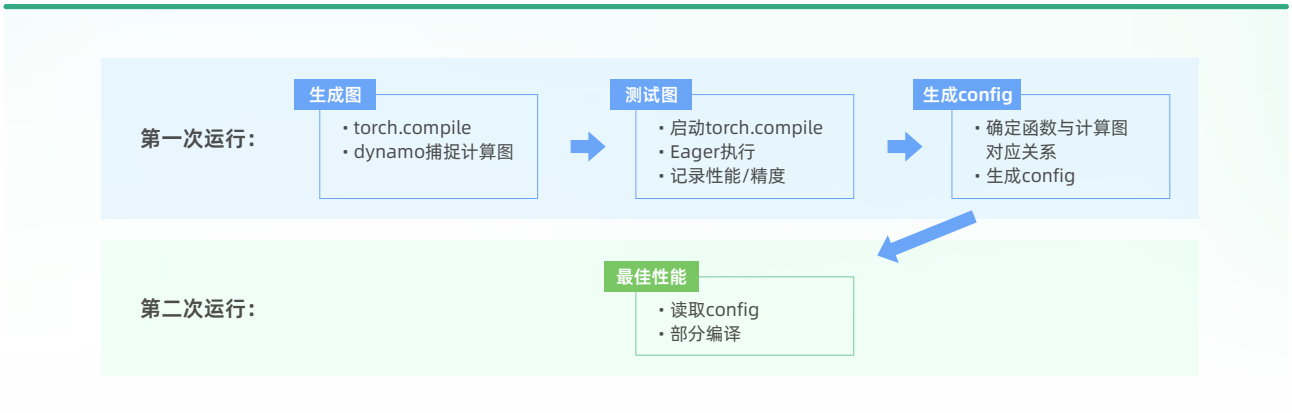
虽然 torch.compile 已经成为众多 AI 场景的标配优化方案，但它仍然存在一些问题，如：

- dynamic shape 造成的性能不及预期。
- 编译时间长，config 需要预热。

龙蜥社区提供了 Regional Compile 以及 prefilled 模式等编译优化来解决以上问题。

6.3.4.2 Regional Compile

torch compile 编译的过程中存在某些 graph 由于 graph break 过多等原因使得它们编译后的性能在加入 dynamo 开销后效率低于 eager 模式，因此我们希望通过 regional compilation 发现模型中编译收益大的模块以及仅编译这些模块，从而获得最佳性能。 Regional Compile 功能需要一次预执行来确定 torch.compile 编译收益大的模块并形成 config，之后的运行只需要加载 config 即可获得最佳性能。 预执行的工作流程如下图所示：



6.3.4.3 prefilled 模式

对于较大的模型，使用 torch.compile 往往会引入较长的停顿，可能导致超时。

`prefilled` 模式支持在低编译开销的情况下获得和 `max-autotune` 近似的性能。

6.3.5 面向 AI Agent Sandbox 场景的操作系统优化

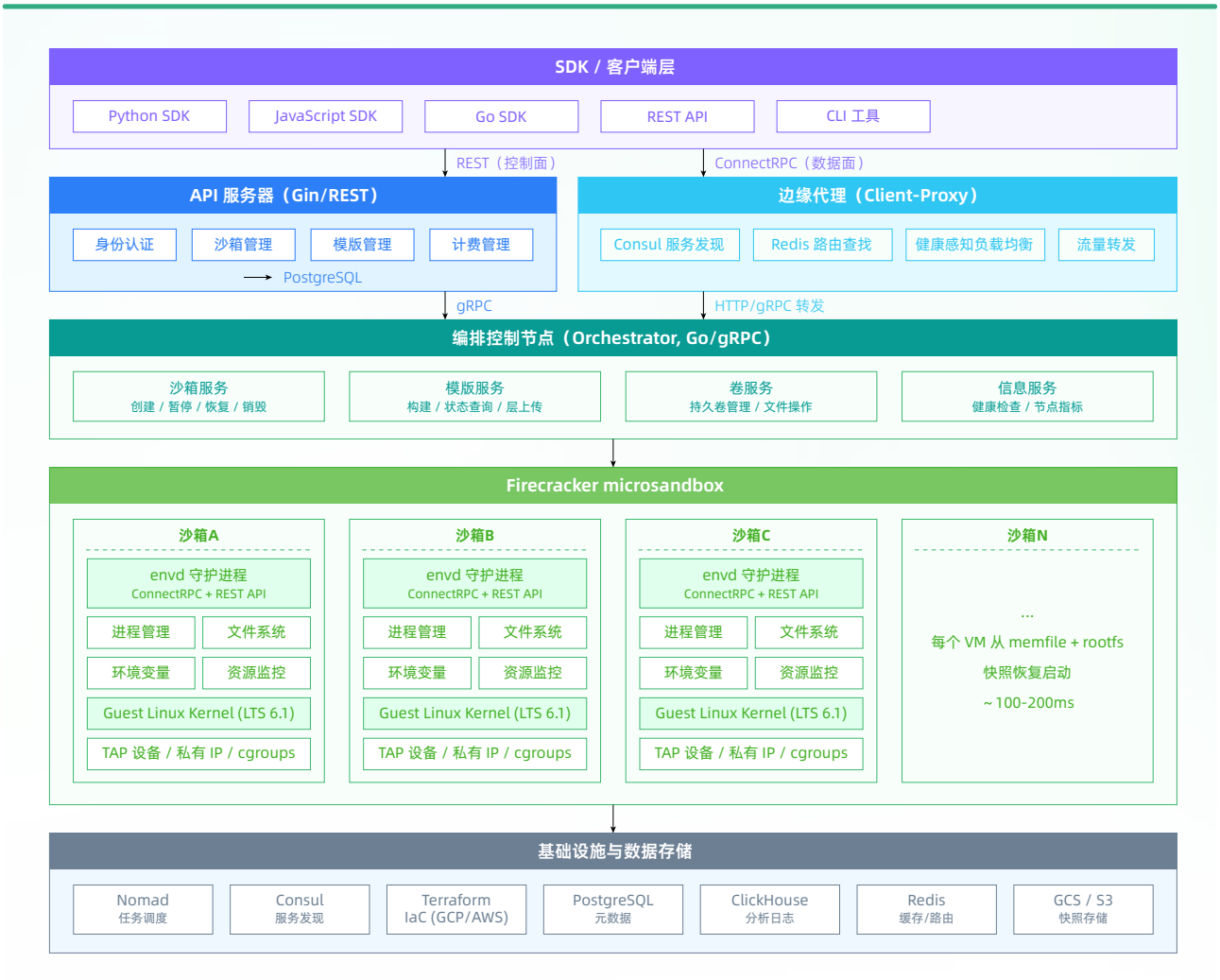
6.3.5.1 背景

随着大语言模型 LLM 技术的快速发展，AI Agent 正从简单的对话交互演进为具备自主行动能力的智能体。现代 AI Agent 能够调用外部工具（如代码解释器、浏览器、文件系统等）与 LLM 进行多轮交互，自主完成复杂任务。这一趋势催生了对安全、高效执行环境的迫切需求，即 Agent 需要一个既能安全运行不可信代码，又能快速弹性扩缩的沙箱运行时。

然而，传统的 Serverless 和 FaaS 方案在应对 Agent 工作负载时面临诸多挑战。Agent 的调用模式具有高度不规则性：请求可能在短时间内突发式到达，也可能长时间处于空闲状态。传统方案在此场景下存在冷启动延迟高（通常数百毫秒至数秒）、资源利用率低、隔离粒度不足等问题，导致高昂的使用成本和不理想的用户体验。

6.3.5.2 E2B：AI Agent 沙箱的行业实践

E2B 是目前业界较为知名的 AI Agent 沙箱平台，为 Agent Sandbox 场景提供了一个有代表性的参考实现。E2B 基于 Firecracker microVM 提供隔离的代码执行环境，支持多语言运行时和丰富的 SDK 生态，会话时长可达 24 小时，已被广泛用于代码解释器、浏览器自动化等 Agent 应用场景。



E2B 的核心思路是为每个 Agent 任务分配一个独立的 microVM 沙箱，通过虚拟机级别的隔离来保障安全性。它同时也采用了基于 microVM 的 checkpoint/restore 技术来加速沙箱的启动过程。这种架构为 Agent 提供了一个安全、可预测的执行环境，但在大规模部署场景下仍面临两方面的优化空间：

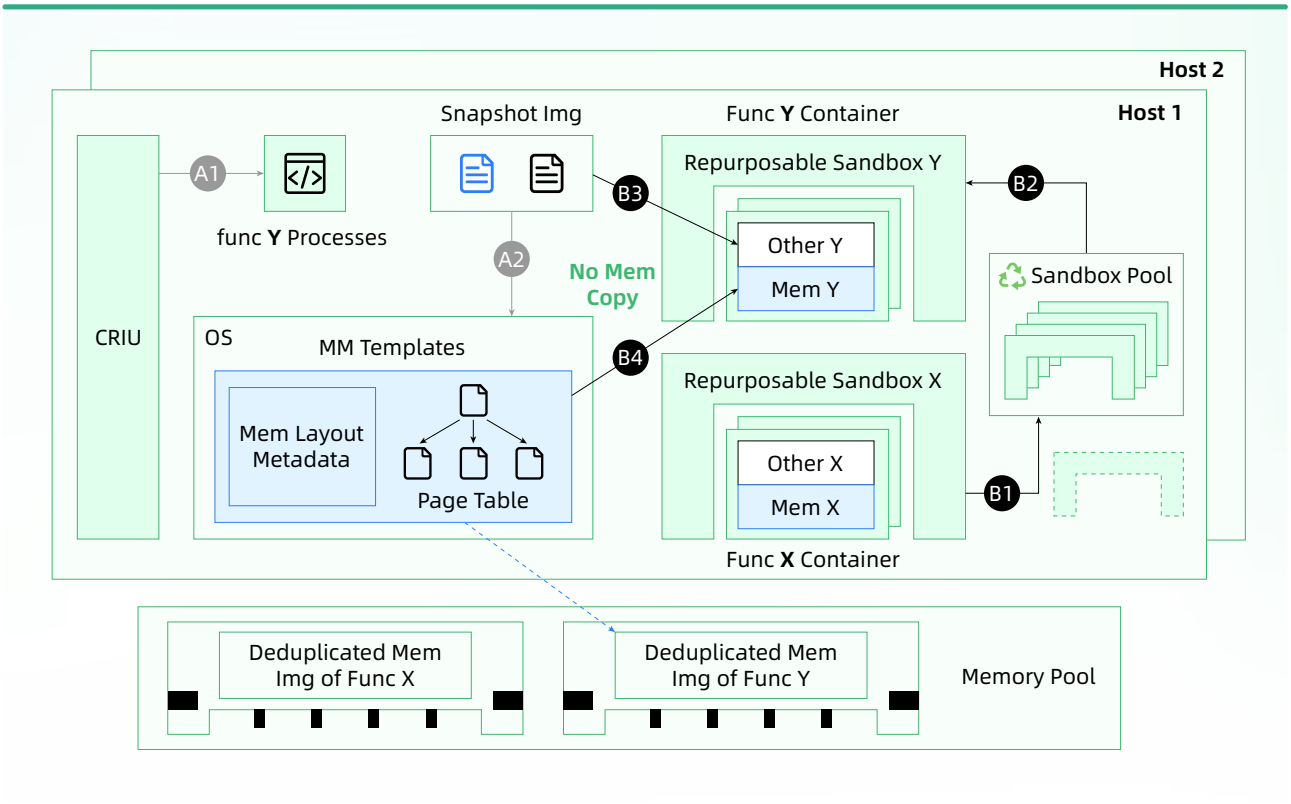
- 启动延迟方面，E2B 的冷启动延迟通常在百毫秒量级，对于高频突发调用的 Agent 工作流而言仍有优化空间。
- 内存效率方面，由于每个 microVM 独立维护完整的内存状态和 Guest 页缓存，在单台物理机运行数百个并发沙箱时，内存开销成为部署密度的主要瓶颈。

6.3.5.3 TrEnv：操作系统层的沙箱优化

研究背景与成果

针对上述操作系统层面的性能瓶颈，龙蜥社区与清华大学联合开展了深入的研究工作，提出了 TrEnv 技术方案。该工作的核心研究成果以论文《TrEnv: Transparently Share Serverless Execution Environments Across Different Functions and Nodes》发表于操作系统领域顶级会议 SOSP 2024。

TrEnv 从操作系统内核的视角出发，深入分析了 Serverless 和 Agent 工作负载的内存访问特征与执行环境特性，发现不同函数实例之间存在大量可共享的执行环境状态。基于这一洞察，TrEnv 提出了两个核心的内核级优化机制：可复用沙箱（Repurposable Sandbox）和内存模板（mm-template），从根本上降低了沙箱的创建开销和内存占用。



核心技术一：基于 CRIU 的可复用沙箱

TrEnv 的第一个关键创新是可复用沙箱（Repurposable Sandbox）机制，该机制与 CRIU（Checkpoint/Restore In Userspace）深度结合。CRIU 是 Linux 生态中成熟的进程级保存与恢复工具，能够将运行中的进程状态完整保存为镜像，并在之后恢复执行。TrEnv 在 CRIU 的基础功能上进行了扩展，实现了执行环境的跨函数透明共享。

传统方案在函数执行完毕后直接丢弃沙箱，导致后续请求必须重新创建。TrEnv 的可复用沙箱则将用完的沙箱"清理"后放入通用沙箱池，任何类型的函数都可以复用这些沙箱。在实现层面，TrEnv 对沙箱的各个组成部分进行了精细拆解优化：rootfs 通过 overlayfs 的挂载替换策略实现快速切换，仅需 2 次系统调用（传统方案需要 9+ 次 mount、6+ 次 mkdev）；cgroup 利用 Linux 的 `CLONE_INTO_CGROUP` 特性，在进程创建时直接指定 cgroup 归属，绕过了传统 cgroup 迁移（将进程移动到新 cgroup）中全局读写信号量导致的 10-50ms 同步延迟，将开销降至 100-300 微秒。

核心技术二：内存模板（mm-template）

TrEnv 的第二个关键创新是内存模板（mm-template），这是 TrEnv 在 OS 内核中实现的一个新抽象。mm-template 的设计解决了传统内存恢复方式需要完整复制内存镜像的性能瓶颈。

mm-template 的核心思想是：同一模板创建的多个沙箱实例，其内核代码段、共享库、运行时环境等内容在内存中高度相似。TrEnv 通过对 Serverless 函数的实测分析发现，执行过程中 24% 到 90% 的内存页面是只读的。基于这一洞察，mm-template 将这些公共只读页面组织为共享的内存模板，所有同模板的沙箱实例共享同一份物理内存页面，仅对各实例私有的数据页面通过 Copy-on-Write 机制进行独立分配。

在恢复效率方面，传统 CRIU 恢复需要完整复制内存镜像，如一个 JavaScript 运行时的内存镜像超过 70MB，而 mm-template 只需拷贝不到 400KB 的元数据，并通过一次 `mmt_attach` 系统调用将模板应用到目标进程。配合 CXL 内存的字节寻址能力，mm-template 可以为只读页面预配置有效的页表项，CPU 通过普通 load 指令直接读取，完全消除了缺页开销。

性能表现

在实验环境下，TrEnv 展现了显著的性能优势。

- 启动速度：TrEnv 可以在约 10ms 内启动大多数函数实例，即使是包含超过 140 个线程的复杂应用也仅需 18ms。相比传统 CRIU 的 1.7 秒（P99），实现了超过 100 倍的加速。
- 端到端延迟：在并发负载下，TrEnv 相比行业内先进的延迟恢复（lazy restore）方案（FaaSnap 与 REAP），P99 启动延迟最高可提升 7 倍。
- 内存节省：通过跨函数去重和内存共享，平均减少 48% 的峰值内存使用。

6.4 一云多芯硬件生态

6.4.1 AMD Turin 前沿计算平台的适配和支持

6.4.1.1 背景介绍

AMD Turin 是基于“Zen5”架构推出的第五代 AMD EPYC 9005 服务器处理器平台。Turin 新平台为 AI 和关键业务工作负载而打造，是目前 x86 架构上最先进的服务器芯片之一。该系列处理器比上一代产品具备更多的核心和更高的频率，还支持更快的 DRAM。龙蜥社区与 AMD 公司展开了深入合作，完成了该平台的全面产品化支持。

6.4.1.2 关键技术

龙蜥社区与 AMD 公司展开了深入合作，完成了该平台的全面产品化支持，在 Anolis OS 8/Anolis OS 23 上支持 Turin 平台最新硬件特性的全面使能与深度优化。

硬件 Profiling 能力

Anolis OS 支持 AMD Turin 平台新增的硬件 PMU 特性：

- Core PMU & Uncore PMU：支持 Zen5 架构的 core/uncore PMU events，提供精确、细粒度的性能分析能力。
- LBRv2：支持最后分支记录 v2 特性，用于分析程序的控制流并帮助识别性能瓶颈。
- IBS：支持基于指令的采样（Instruction Based Sampling），提供更高精度、性能开销更小的 Core 事件采样，解决基于中断的采样存在的偏差（skid）问题。

资源隔离能力（QoS）

Anolis OS 提供了对 AMD 所有代际处理器（包括 Turin）完整的 PQoS（Platform Quality of Service）能力。该技术在 AI 混部、ECS 多租等场景属于必备能力，能够有效缓解共享资源竞争问题。

- PQM（监控）：支持 L3 cache occupancy 和 L3 cache bandwidth 监控。
- PQE（执行）：支持 L3 CAT、CDP、L3 BE、L3 SMBE 等资源限制能力。
- BMEC：带宽监控事件配置，允许软件选择性配置监控特定的 L3 内存带宽事件类型（包括本地/远端 NUMA 域的读写、慢速内存访问等）
- ABMC：可分配带宽监控计数器，提供将 RMID 固定到硬件计数器的选项，确保长期带宽监控数据的有效性。

电源管理

Anolis OS 支持 AMD Turin 平台的全面电源管理能力：

- AMD P-States 驱动：基于 CPPC（Collaborative Processor Performance Control）技术，提供更细粒度的频率管理，支持 active、passive、guided 三种模式，可使 CPU 最低工作频率下探至数百 MHz 级别，极大优化空闲状态下的能耗。
- HSMP：支持主机系统管理接口（Host System Management Port），提供功耗管理、频率调节、互联优化、内存子系统监控、CPU 利用率统计等精细控制能力。

指令集增强

Anolis OS 支持 AMD Turin 平台的指令集增强特性：

- AVX-512 完整支持：Turin 作为首款 Zen5 架构的 EPYC 服务器处理器，提供完整的 512 位数据位宽的 FPU 单元，硬件可直接执行 AVX-512，相比上一代 Zen4 的 AVX-512 性能大幅度提升。
- INVLPGB 指令：支持广播 TLB 无效操作，无需发送 IPI 中断即可无效 TLB 条目，在 TLB 刷新频繁的场景下可显著降低开销。

IO 加速

Anolis OS 支持 AMD Turin 平台首次引入的 SDCI（Smart Data Cache Injection）技术。SDCI 是一种将数据从 IO 设备直接插入到 L3 缓存的机制，通过直接缓存来自 I/O 设备的数据，大幅降低对 DRAM 带宽的需求以及处理器消耗 I/O 数据的延迟，显著提升网络通信等 I/O 密集型应用的性能。

安全漏洞修复

AMD Turin 平台作为 Zen5 架构，主要受到 speculative execution 推测执行漏洞的影响，如 Speculative Store Bypass、spectre_v1、spectre_v2 等，Anolis 已默认支持并打开相关漏洞的软件安全防护措施。

值得一提的是，相比 AMD 上一代 Zen4 Genoa 处理器平台，Turin 平台在硬件层面上不再受到 SRSO（Speculative Return Stack Overflow）漏洞的影响，因此软件层面无需打开相关修复，避免了 SRSO 漏洞防护带来的性能开销。

RAS 能力

Anolis OS 在 Turin 平台上支持 AMD SMCA（Scalable Machine Check Architecture），可实现对各类硬件故障的自动检测、记录与上报，涵盖系统总线、内存、缓存、TLB 以及数据通路中的 ECC 和奇偶校验错误等，兼容 EDAC 和 Rasdaemon 特性。

虚拟化能力加强

Anolis OS 在 AMD Turin 平台上完整支持 QEMU/KVM 虚拟化能力，并解决了多个 AMD IOMMU 相关特性的稳定性问题。支持 x2AVIC（Second Generation Advanced Virtual Interrupt Controller），提高处理中断请求时的效率和性能。支持开关 IOMMU irtecache 功能，解决不同虚拟化实例规格场景下 irtecache 导致的性能问题。

安全虚拟化（机密计算）

Anolis OS 支持 AMD SEV（Secure Encrypted Virtualization）技术，将物理机密计算能力传导至虚拟机实例，在云上打造立体化可信加密环境：

- SEV：基础内存加密，通过嵌入式 AES 引擎对 VM 内存进行加密，每个 VM 拥有独立的加密密钥。
- SEV-ES：寄存器加密，当 VM 停止运行时加密所有 CPU 寄存器内容，防止信息泄露给 Hypervisor。

6.4.2.3 应用和生态

Anolis OS 在 AMD Turin 平台上已完成全栈软件生态的适配与验证，涵盖云计算、AI 训练与推理、数据库、大数据、容器化部署等主流服务器应用场景。Turin 平台作为 AMD 最新一代 EPYC 服务器处理器，x86 软件生态高度成熟，现有应用可无缝运行。Anolis OS 充分发挥 Turin 平台 Zen5 架构的原生 AVX-512 算力、SDCI IO 加速及 SEV 机密计算等新特性，为用户在高性能计算、网络密集型应用及安全可信等场景提供开箱即用的增强体验。

6.4.2 Intel GNR 革新算力平台的适配和支持

6.4.2.1 背景介绍

Intel Granite Rapids（简称 GNR）即第六代至强处理器，采用全新的 Intel 工艺制造，在可扩展性、加速器、内存、I/O、安全等方面均有显著增强。GNR 平台面向 AI 和关键业务工作负载而打造，是目前 x86 架构上最先进的服务器芯片之一。

6.4.2.2 关键技术

龙蜥社区基于 Intel Arch SIG 第一时间完成了该平台的全面产品化支持，在 Anolis OS 8/ Anolis OS 23 上提供 GNR 平台硬件特性的全面使能与深度优化。

硬件 Profiling 能力

Anolis OS 支持 GNR 平台新增的硬件 PMU 特性；包括 Core/Uncore Event Counter、LBR Event Logging、Timed PEBS、IOMMU Perf 等，用户可以使用 Anolis 提供的 perf 工具采集更多且更详细的硬件事件信息。

此外，Anolis OS 支持 GNR 平台的 TPMI（Topology Aware Register and PM Capsule Interface）功能，提供可枚举的 MMIO 接口用于电源管理特性，实现对 CPU 电源管理的精细化控制，帮助更高效地优化运营成本。

电源管理

Anolis OS 在 GNR 平台完整支持 Intel SST（Speed Select Technology）系列功能，包括：

- SST-PP（Performance Profile）：提供多种预定义的处理器配置方案，允许用户根据工作负载需求选择性能、功耗和 CPU Core 数量的最佳平衡。
- SST-BF（Base Frequency）：为不同 CPU Cores 设置独立的基础频率，提供差异化的基础性能保证。
- SST-TF（Turbo Frequency）：精细化管理 CPU Cores 的睿频，为高优先级核心分配更高的睿频频率。
- SST-CP（Core Power）：在 CPU 功耗受限时动态优先级分配功耗和频率资源，实现核心级的电源服务质量（Power QoS）。

Anolis OS 在 GNR 平台支持 5 种 C-states：POLL、C1、C1E、C6、C6P。其中 C6P 为 GNR 新增类型，通过要求所有核心协同请求来触发整个处理器进入最深的 Package 级 C6 状态，实现最大化节能效果。

资源隔离能力

在原有 RDT（Resource Director Technology）的基础上，Anolis 支持 GNR 平台 L2/L3 Non-contiguous way masks，即在使用 CAT（Cache Allocation Technology）时可指定非连续的 Bit Mask。Anolis OS 在启动时会检查 CPUID.0x10.1:ECX[3] 和 CPUID.0x10.2:ECX[3] 是否置位，在支持该特性的平台上向用户开放设置 CAT 非连续掩码的能力。

加速器增强

在支持 SPR/EMR 平台 QAT/DSA/IAA/DLB 的基础上，Anolis OS 进一步支持了 GNR 平台加速器引入的新特性：

- DSA 2.0：支持 Event log、completion record faulting、跨地址空间数据搬移、CRC64、DIX、Fill16 等特性，适用于内存拷贝、内存清零等高吞吐场景。

- IAA 2.0：增强 Deflate 算法，为数据查询分析提供高吞吐的数据压缩/解压缩和数据过滤能力。
- QAT：支持传统 Web Server、云原生（Service Mesh）、QUIC/HTTP3 等场景的加解密和压缩加速，实测性能可提升 2～3 倍。

另外，Anolis OS 支持 GNR 平台 AMX FP16 指令，支持更高的数据精度，可以更快地运行推理模型。Anolis OS 集成的 oneDNN、Pytorch、TensorFlow、OpenVINO 等可无感使用该指令进行加速。

虚拟化能力加强

Anolis OS 完整支持 TDX Guest 能力；包括启动适配、内存加解密基本接口适配、VE 和 tdvmcall 接口、swiotlb 优化；并且支持完整的基于 tdvmcall attestation 接口，另外支持了 efi stub 和 boot parameter 两种方式上报的 unaccepted memory，让 Guest 可以按需 Accept 内存，加速大规格内存实例启动速度。

GNR 平台支持 TDX Gen 2.0，带来显著增强：

- 加密密钥规模提升：能够同时支持并管理多达 2048 个独立的加密密钥。
- 加密强度增强：每个密钥都采用业界领先的 AES-256 位加密强度。
- 简化部署：现有应用程序通常无需修改代码即可在 TDX 环境中运行，并支持虚拟机热迁移。

另外 Anolis OS 支持 GNR 平台引入的 IO 虚拟化加速能力；包括以往在执行中断重映射禁用操作或更改 PASID 相关的表项时需要软件对所有相应的缓存执行全局清空的操作，现在可由硬件自动完成。

6.4.2.3 应用和生态

Anolis OS 在 Intel GNR 平台上已完成全栈软件生态的适配与验证，涵盖云计算、AI 推理、数据库、大数据、容器化部署等主流服务器应用场景。GNR 平台作为 Intel 最新一代至强处理器，x86 软件生态高度成熟，现有应用可无缝运行。Anolis OS 充分发挥 GNR 平台的 DSA/IAA/QAT 加速器、AMX FP16 指令及 TDX 机密计算等新特性，为用户在数据压缩、加解密加速、AI 推理及安全可信等场景提供开箱即用的增强体验。

6.4.3 Intel CWF 平台支持高能效算力平台的适配和支持

6.4.3.1 背景介绍

Intel Clearwater Forest（简称 CWF）是与 Granite Rapids（GNR）同代的第六代至强处理器，采用全新的 Intel 工艺制造。与 GNR 的 P-Core（性能核心）定位不同，CWF 基于 E-Core（能效核心）架构设计，在保持高核心密度的同时提供卓越的能效比，特别适合高密度计算、云原生微服务及能效敏感型工作负载。

6.4.3.2 关键技术

龙蜥社区基于 Intel Arch SIG 完成了该平台的全面产品化支持，在 Anolis OS 23 支持 CWF 平台最新硬件特性的全面使能与深度优化。

指令集增强

CWF 平台新增了多项指令集扩展，Anolis 已完成全面适配：

- SM3/SM4：硬件级国密算法指令加速，支持 SM3 哈希算法和 SM4 对称加密算法的硬件直接执行，显著提升国密合规场景下的加解密性能。
- SHA512：SHA-512 哈希算法硬件加速指令，提升加密与数据完整性校验场景的处理效率。
- AVX-VNNI-INT16：向量神经网络整数 16 位运算指令，面向 AI 推理场景提供更高效的低精度整数计算能力。

硬件 Profiling 能力

Anolis OS 支持 CWF 平台新增的硬件 PMU 特性；包括 Core/Uncore Event Counter、LBR Event Logging、Timed PEBS、IOMMU Perf 等，用户可以使用 Anolis 提供的 perf 工具采集更多且更详细的硬件事件信息。

此外，Anolis OS 支持 CWF 平台的 TPMI（Topology Aware Register and PM Capsule Interface）功能，提供可枚举的 MMIO 接口用于电源管理特性，实现对 CPU 电源管理的精细化控制，帮助更高效地优化运营成本。

电源管理

Anolis OS 在 CWF 平台完整支持 Intel SST（Speed Select Technology）系列功能，包括：

- SST-PP（Performance Profile）：提供多种预定义的处理器配置方案，允许用户根据工作负载需求选择性能、功耗和 CPU Core 数量的最佳平衡。
- SST-BF（Base Frequency）：为不同 CPU Cores 设置独立的基础频率，提供差异化的基础性能保证。
- SST-TF（Turbo Frequency）：精细化管理 CPU Cores 的睿频，为高优先级核心分配更高的睿频频率。
- SST-CP（Core Power）：在 CPU 功耗受限时动态优先级分配功耗和频率资源，实现核心级的电源服务质量（Power QoS）。

CWF 平台基于 E-Core 架构，在相同功耗预算下可提供更高的核心密度，特别适合多线程并发场景。

资源隔离能力

在原有 RDT（Resource Director Technology）的基础上，Anolis OS 支持 CWF 平台 L2/L3 Non-contiguous way masks，即在使用 CAT（Cache Allocation Technology）时可指定非连续的 Bit Mask。Anolis 在启动时会检查 CPUID.0x10.1:ECX[3] 和 CPUID.0x10.2:ECX[3] 是否置位，在支持该特性的平台上向用户开放设置 CAT 非连续掩码的能力。

加速器增强

在支持 SPR/EMR 平台 QAT/DSA/IAA/DLB 的基础上，Anolis OS 进一步支持了 CWF 平台加速器引入的新特性：

- DSA 2.0：支持 Event log、completion record faulting、跨地址空间数据搬移、CRC64、DIX、Fill16 等特性，适用于内存拷贝、内存清零等高吞吐场景。
- IAA 2.0：增强 Deflate 算法，为数据查询分析提供高吞吐的数据压缩/解压缩和数据过滤能力。
- QAT：支持传统 Web Server、云原生（Service Mesh）、QUIC/HTTP3 等场景的加解密和压缩加速。

6.4.3.3 应用和生态

Anolis OS 在 Intel CWF 平台上已完成全栈软件生态的适配与验证，涵盖云计算、容器化微服务、Web 服务、数据库等主流服务器应用场景。CWF 平台与 GNR 共享同代 x86 软件生态，现有应用可无缝运行。CWF 基于 E-Core 架构提供更高的核心密度与能效比，特别适合高并发、多租户的云原生部署场景。

6.4.4 海光平台支持

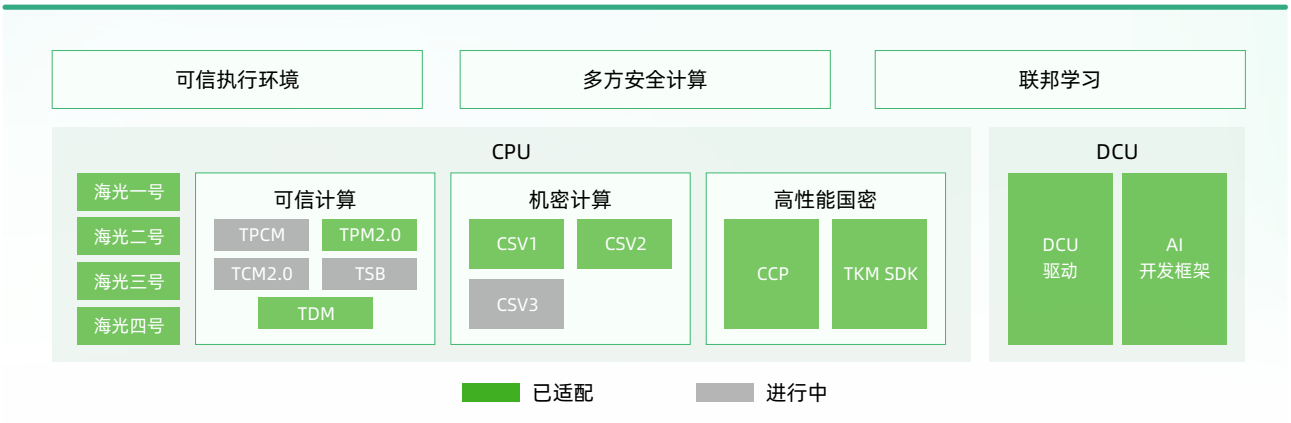
6.4.4.1 背景介绍

为了满足金融等行业 IT 基础架构日益增长的需求，龙蜥社区联合海光平台致力于构建国产 x86 架构的软件生态， 包括国密、可信计算、机密计算和虚拟化等，提供安全可信的环境来承载云上业务需求。

紧随海光研发节奏，龙蜥社区积极推进海光新特性的适配，对海光平台的适配领跑业界。龙蜥社区已经完成海光平台多款机

型的适配工作，并且完成了机密计算，可信计算等多个特性与硬件的适配工作，助力龙蜥客户以开箱即用的方式，享用新技术特性。

龙蜥社区现已适配海光平台 CPU （海光一号、海光二号、海光三号、海光四号）以及各种海光平台特性，整体支持情况如下图：



6.4.4.2 关键技术

龙蜥社区积极推进海光新特性的适配，对海光平台的适配领跑业界。龙蜥社区已经完成海光平台多款机型的适配工作，并且完成了机密计算，可信计算等多个特性与硬件的适配工作，助力龙蜥客户以开箱即用的方式，享用新技术特性。龙蜥社区在 Anolis OS 8/Anolis OS 23 上支持海光平台最新硬件特性的全面使能与深度优化。

指令集增强

- 海光四号微架构升级：前端显著扩充分支预测单元缓存容量与数据带宽，引入更先进预测算法；操作码缓存（OP Cache）规模大幅扩展。后端地址生成单元计算吞吐增强；浮点运算执行通道宽度拓宽；乱序执行窗口加深。
- 新增 SM3/SM4 指令集：海光四号新增国密 SM3/SM4 指令集，相比纯软件实现性能大幅提升。
- Glibc 优化：优化用户态库针对海光平台特性，使用 AVX2 替代 AVX512，显著降低 MySQL 测试中的平均、95 分位及最大延迟。

IO加速

- Cache Stashing IO特性：海光四号首次引入该特性，优化 I/O 数据直达缓存的路径，减少内存访问开销。
- 大内存拷贝优化：优化内核态与用户态之间的大内存拷贝效率，FIO 大 IO 性能提升。
- 密码加速引擎：集成 CCP (Cryptographic Co-Processor硬件模块，硬加速 SM2/SM3/SM4 及国际算法；支持外置加速引擎类似 Intel QAT，构建全栈加密高性能方案。

虚拟化能力加强

- CCP 虚拟化支持：Alibaba Cloud Linux 4 支持 CCP 虚拟化（vCCP）， enabling CSV3 虚拟机的 CMA 并发分配及热迁移。
- 密钥管理虚拟化：TKM 支持虚拟化（vTKM）。
- 代码多副本优化：将远端节点代码段在本地节点添加副本，避免跨节点访问，提升 PostgreSQL 性能。

安全虚拟化（机密计算）

- CSV (China Secure Virtualization) 技术演进：

- CSV1：基础内存加密，写内存自动加密，读内存自动解密，每个虚拟机使用不同密钥。
- CSV2：在 CSV1 基础上增加虚拟机状态加密功能。
- CSV3：实现 CPU 内部数据强隔离，禁止主机操作系统对虚拟机内存读写，保证数据完整性与机密性。

- CSV 核心特性：
  - 资源隔离：基于 ASID 区分虚拟机与主机，独立 Cache/TLB，密钥由安全处理器管理。
  - 实时加解密：硬件自动完成，对业务透明，性能开销 <1%。
  - 抗攻击能力：抵御重放攻击、DMA 攻击及嵌套页表重映射攻击。
  - 启动度量与认证：安全处理器校验文件度量值，支持基于芯片唯一密钥的远程认证报告。
  - 生态支持：Alibaba Cloud Linux 4 支持 CSV 全版本，提供基于 CSV 的 TEE 化 KMS 方案及袋鼠机密容器支持。

其他相关优化（内核与用户态定制）

- ccx-awareness qspinlock：优化 qspinlock，将海光 CCX（Core Complex）纳入考量，MySQL OLTP 读写性能提升。
- AI 加速支持：面向 AI 与科学计算的 GPGPU 加速卡，基于 DTK 软件栈，原生支持 TensorFlow/Pytorch/Paddle 等框架，提供多精度支持

6.4.4.3 应用和生态

Anolis OS 在海光平台上已完成全栈软件生态的适配与验证，涵盖云计算、数据库、Web 服务、容器化部署、Java 应用等主流通用计算场景。海光平台作为国产 x86 架构，软件生态与国际 x86 平台高度兼容，现有应用可无缝迁移运行。

在国密与安全领域，Anolis OS 充分发挥海光平台的 SM3/SM4 指令级硬件加速、CCP 密码协处理器及 HCT 密码加速套件，结合 CSV 安全虚拟化、可信引导（TPM/TCM/TPCM）等安全特性，为金融、政务、医疗等对安全合规有严格要求的行业用户提供开箱即用的全栈国密与可信计算解决方案。

在 AI 与科学计算领域，Anolis OS 支持海光 DCU 加速卡及 DTK 软件栈，原生适配 TensorFlow、Pytorch、Paddle 等主流深度学习框架，提供 FP32/FP16/INT8 多精度训练和推理能力，覆盖计算机视觉、智能语音、推荐系统等典型 AI 应用场景。

6.4.5 龙芯自主指令集的支持

6.4.5.1 背景介绍

LoongArch 是由龙芯中科推出的新一代指令系统，包括基础架构部分和向量指令、虚拟化、二进制翻译等扩展部分，近 2000 条指令。该指令系统具有较好的自主性、先进性与兼容性，对二进制翻译、虚拟化、向量化的支持能够为操作系统、虚拟机的开发降低成本。

基于 LoongArch 指令集的处理器芯片如 3A5000、3C5000、3D5000、3C6000、3D6000、3E6000 等已经研发成功并量产。Anolis OS 完美地支持基于 LoongArch 指令集的龙芯处理器，并为基于龙芯处理器打造的硬件平台提供了操作系统生态。

6.4.5.2 关键技术

- 内核支持

基于 6.6 内核进行了全新的 LoongArch 架构支持，包括 LoongArch 架构的基础指令支持，扩展向量指令支持，扩展二进制翻译支持，扩展虚拟化支持；同时实现了基于 LoongArch 架构研发的 3A5000、3C5000、3C5000L/LL、3D5000、3C6000、3D6000、3E6000 处理器的支持和相关配套桥片 7A1000、7A2000 的支持，新增虚拟化 PMU、IOMMU 支持，新增龙芯平台 SE/SDF 安全认证模块、GMAC 网卡驱动等支持，并在各个平台进行了完善的测试，相关技术指标也进行了优化。

- 虚拟化技术

QEMU/KVM 是目前非常流行的虚拟化技术，它基于内核提供的 kvm 模块，结构精简，性能损失小。Anolis OS 在龙芯平台上支持 qemu 以及 libvirt，并提供基于龙芯 CPU 的虚拟化、管理平台一体化方案，为客户提供全栈的云服务体系。针对 QEMU8 龙芯平台的优化、支持代码已经合入 Anolis OS 主线分支。qemu 新增 cpu热插拔、PMU 功能等支持，libvirt 使能对 numa 的支持。

- 语言平台 GCC/LLVM/Golang/Rust/Java/JavaScript

针对龙芯平台的优化、支持代码已经合并进入 Anolis OS 社区主线分支。这些改动除新增功能支持，专注通用性优化外，也包括针对龙芯处理器的深度优化，比如使用龙芯的专有指令。如此可以充分挖掘指令特点，最大限度利用硬件。其中龙芯平台的 JVM 虚拟机和 V8 引擎优化后已经能够承担量级可观的日常测试和开发任务。

6.4.5.3 应用和生态

Anolis OS 操作系统环境及软件均已移植完成，成为了 LoongArch 的原生版本。行业应用方面，面向 LoongArch 的移植工作也在有条不紊地进行，LoongArch 的原生生态已经不输于原本的 LoongISA。

3 个二进制翻译系统 x86、Arm、mips 翻译能力使得龙芯平台可以短时间内兼容其他平台成熟的应用软件。翻译运行效率也在持续提升，已经接近 90%。

6.4.6 开源硬件 RISC-V 支持

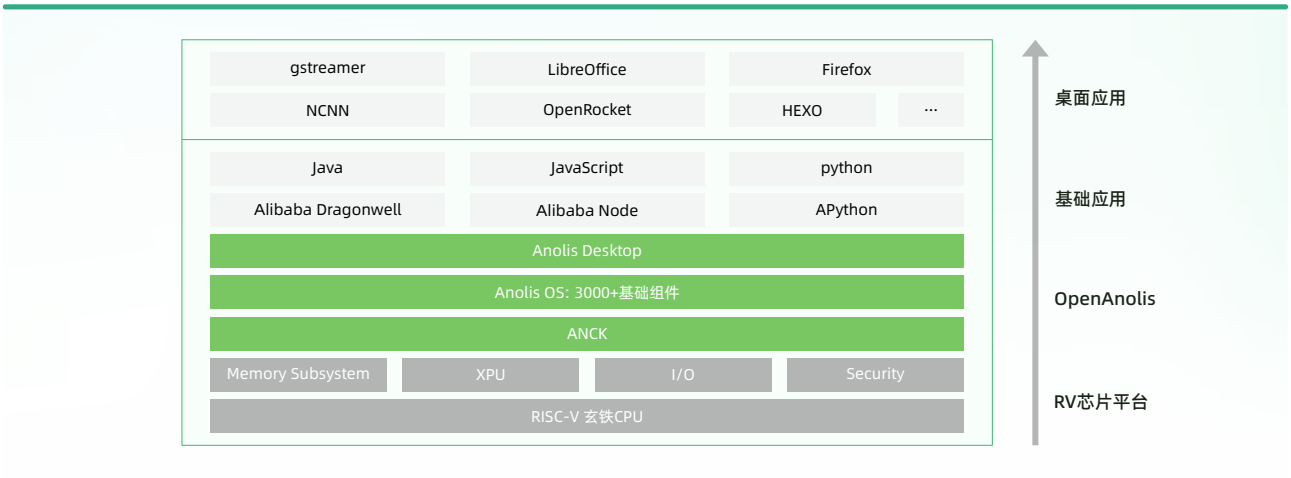
6.4.6.1 背景介绍

RISC-V 是一套开源指令集。为拥抱更加开放的芯片生态，指令集标准和扩展标准采用全球共享共治的模式，与 Arm 和 x86 有很大不同，因此也受到了业内人士的普遍关注，在未来有着很大应用潜力。龙蜥社区正式成立了 RISC-V ARCH SIG（Special Interest Group），全面兼容并促进 RISC-V 生态发展。 随着 RISC-V 硬件算力的不断发展，RISC-V 硬件的性能正在不断提升，而 RVA23 标准的发布成为 RISC-V 向服务器领域迈进的标志性事件，在新的 AI 智算浪潮中，RISC-V 凭借其低成本、可扩展性等优势引起了国内外业界重视，RISC-V 的新一波浪潮正在孕育中。

6.4.6.2 关键技术

龙蜥社区 RISC-V SIG 秉承 RISC-V 共享共治的模式，与达摩院、中兴、如意社区、中科院软件所 PLCT 实验室、统信等高校和公司共同建设龙蜥社区 RISC-V 软硬件生态，在 Anolis OS 8 上支持 RV64GC 指令集及软件生态，在 Anolis OS 23 上支持最新的 RVA23 指令集及软件生态

Anolis OS 8



龙蜥操作系统 5.10 内核全面支持 RISC-V RV64GC 指令集，在 arch、mem、ftd、GPU、VPU 等内核子系统方面合入补丁 70+。在 BaseOS 方面，龙蜥社区完成了 3000+ 个软件包在 RISC-V 架构上的适配，极大的丰富了 RISC-V 软件包的生态。在桌面镜像方面，龙蜥社区提供了基于 XFACE 的桌面镜像，并全面支持 RISC-V 架构。在生产力应用方面，支持了 JAVA、Python、NodeJS 等主流语言。

除此之外，Anolis OS 8 还完成了 Alibaba Dragonwell、Alibaba Node、APython 等云上应用，以及 LibreOffice、Firefox、Open-Rocket 等办公套件，和 NCNN 等 AI 应用在 RISC-V 架构上的适配。帮助 RISC-V 在桌面和数据中心领域迈出了关键一步。

目前，龙蜥社区已经联合达摩院、统信软件、中科院软件所 PLCT 实验室基于 Anolis OS 8 共同打造了围绕 RISC-V 芯片、OS 和生态应用的软硬件全栈平台，帮助 RISC-V 架构继续在嵌入式领域发光发热，并逐步迈入桌面和数据中心领域。

Anolis OS 23

Anolis OS 23 使用的 6.6 内核回合了 1500+ 补丁，已全面支持 RVA23 Mandatory 指令集，及部分 Optional 指令集。此外正在逐渐完善 PMU、RAS、QoS、NUMA、Kdump、Hotfix 等服务器基础功能。

Anolis OS 23 RISC-V 版本软件选型是由国内多家社区基于如意社区平台合作完成的软件版本选型，充分考虑了内核、工具链、核心软件包和生态软件包的功能完整性、可维护性。Anolis 23.5 已经统一选型完成，工具链（GCC、binutils、glibc）、核心软件包（libvirt、openssl 等）已完成版本升级和 RVA23 适配，其他生态软件包均支持完整的 RVA23U64 指令集标准。

除基础支持外，Anolis OS 23 还积极合入最新的性能优化补丁，如 zlib、zstd、openssl、lz4 等关键基础软件包中合入了基于 RVA23 的向量指令优化，性能预计提升 20%~200%。

6.4.6.3 应用和生态

目前，龙蜥社区提供的 RISC-V Anolis OS 8 已经支持达摩院高性能 RISC-V平台：曳影 1520，能够在曳影 1520 上流畅运行 Anolis OS 桌面环境，在此之上还能运行 LibreOffice、Firefox、Droganwell、NCNN 等生产力软件和云上应用。Anolis OS 8提供的 GPU、VPU 等驱动能够完美释放曳影 1520 在音视频领域的硬件潜能。 Anolis OS 23 preview 版本已支持 RISC-V 服务器开发版：算能 SG2042。 Anolis OS 23.5 正式版本已全面支持 RVA23。

6.4.7 GPU 异构硬件支持

6.4.7.1 背景介绍

Anolis OS 23 为全面拥抱智算的最新一代开源操作系统版本，目标打造智算时代支持最佳的国产操作系统。Anolis OS 23 内核在支持南向 GPU 异构硬件方面，除了在 Anolis OS 8 版本已支持的对 GPU 硬件的抽象层 DRM（Direct Rendering Manager）、

DMA-BUF、IOMMU 以及 VFIO/vGPU 虚拟化等核心基础特性外，还引入 AI 计算芯片专用的 accel 子系统，大大降低了国产 GPU 适配门槛；针对智算服务器大内存特点支持大内存管理，优化了内存管理开销。

6.4.7.2 关键技术

Anolis OS 23 在 GPU 异构硬件支持上，依托 Linux 开源上游内核特性，提供统一硬件抽象层、轻量级设备注册和管理接口以及标准化生命周期管理，能够以最小的内核侵入性支持多样化的国产 GPU 异构硬件。

DRM（Direct Rendering Manager）

DRM 已演化为 GPU 的通用资源管理器，三大核心职责： 显存管理、命令提交与调度、模式设置（KMS），是现代 GPU、AI 加速卡、视频编解码卡的底层基石。

DMA-BUF

DRM 的 DMA-BUF 框架，允许不同设备（如网卡、编解码器、GPU）共享同一块物理内存页，而无需复制数据。用户态程序可以通过 mmap 直接将显存映射到进程地址空间，或者让其他设备直接访问显存，从而实现零拷贝数据传输。在 AI 推理流水线中，数据可以从采集卡直接“飞”入 GPU 显存，CPU 几乎不介入，可显著延迟降低，大幅提升吞吐量。

IOMMU 及其虚拟化

iommufd 是 Linux 内核新一代 IOMMU 用户态管理接口，也是上游在设备直通与虚拟化方向的重要演进趋势。相比传统 VFIO type1 模型，iommufd 在对象模型、扩展性、新硬件特性支持如硬件脏页追踪、PAISD 等方面更具优势。Anolis OS 23 对 iommufd 提供了较为完善的支持，可更好满足国产 GPU 和 AI 加速设备在虚拟化、容器化及云场景中的部署需求。

accel（Accelerate Subsystem）

accel 子系统是 Linux 内核在 6.1 版本正式引入的一个全新核心子系统，是 DRM 的轻量化实现，专门用于管理纯计算类加速器（如 GPGPU、NPU、AI 芯片等），剥离了 DRM 中与图形显示（KMS, Framebuffer）强绑定的代码，大幅降低新硬件的上手门槛。

同时，优化了容器化和云原生支持，Kubernetes Device Plugins 可以更容易地编写通用的 Accelerate Device Plugin，自动发现 /dev/accel/* 设备并挂载到容器中，无需针对每家厂商写特殊逻辑。

在促进开源驱动生态方面，accel 的简化模型鼓励更多厂商将计算部分的驱动开源，融入主线 Linux 内核。

大内存管理

目前，越来越多的场景对内存大页的需求变得强烈，一方面是随着业务场景所需内存的 Size 越来越大，大页使用可以有效地降低大内存管理成本；另一方面大页软硬协同在性能提升上更加明显，尤其在虚拟化场景，比如 Arm contiguous PTE，AMD THP 等可大大降低 TLB miss。另外在当前火热的 AI 场景，大页已经证明可以带来性能提升。

Large Folio 是 Linux 内核内存管理子系统（MM）在 6.1 版本引入的一项重大架构革新，它的核心目的是彻底解决“透明大页”（THP, Transparent Huge Pages）长期存在的性能抖动和碎片化问题，为数据库、AI 大模型训练/推理等大内存应用提供更稳定、更高效的内存分配机制。Anolis OS 23 版本除了包含 6.6 内核社区自带的特性，还主动从开源社区上游主动回合重要特性，以及针对开源实现的进一步优化。

6.4.7.3 应用和生态

龙蜥社区已加入开放智算产业联盟（简称“COIA”），致力于促进智算产业开源社区的交流与生态合作。目前，已引导和推动多家国内 GPU 厂商（燧原科技、摩尔线程、海光、平头哥、天数智芯、壁仞科技等）针对 Anolis OS 23 做兼容性适配和互认证。

Anolis OS 23 及其衍生版本在实际智算业务项目中，已实现对国内外 10+ 以上GPU厂商，共计 30+ GPU 型号的兼容和支持。

6.5 运维与性能

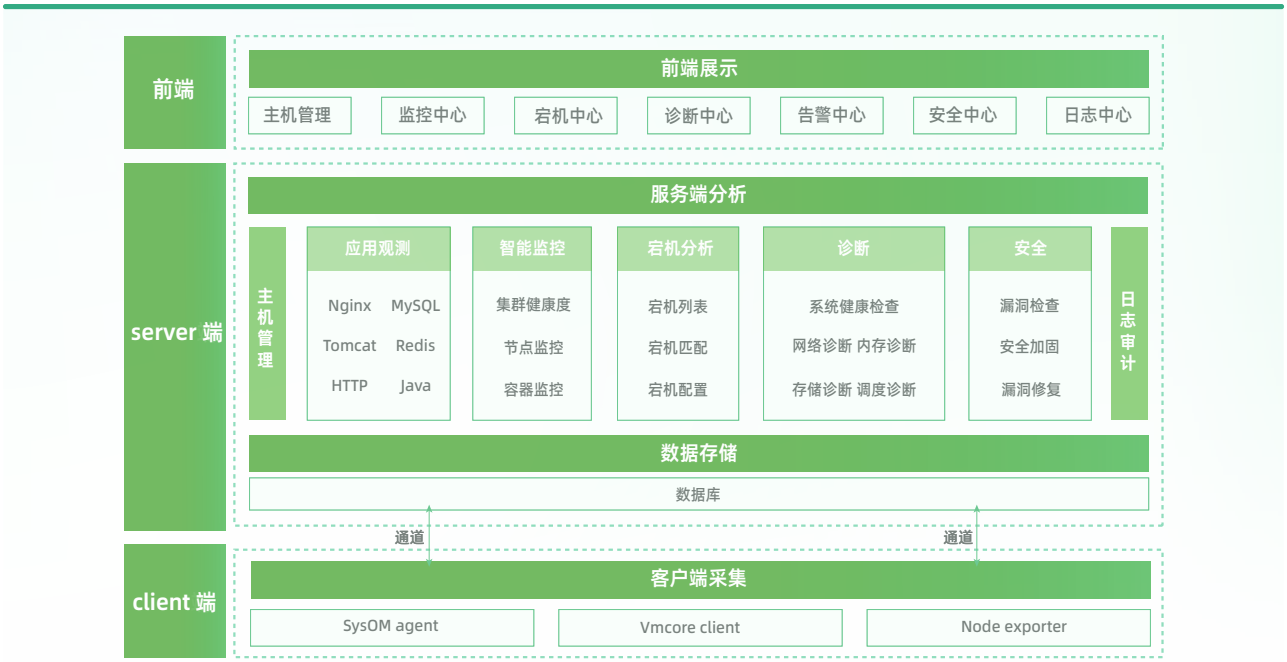
6.5.1 SysOM：一站式运维管理平台

6.5.1.1 概述

SysOM（System Operation & Maintenance）是由龙蜥社区系统运维 SIG 打造的一站式操作系统运维平台，致力于解决业内相关运维工具碎片化，门槛高的挑战。

6.5.1.2 技术方案

SysOM 的整体架构分为前端、服务端、客户端三部分，其系统架构图如下所示：



SysOM 是一款集成了应用观测、系统监控、告警、诊断和安全运维的全流程解决方案。它利用 SysOM agent 对内核行为进行深入监控和分析，结合服务端的大数据和机器学习技术，不仅能够实时观测应用的性能和状态，还能智能地监控系统的运行情况，发现并定位问题的故障原因，从而实现“一键式”的运维体验。整体平台具备以下特点：

- 统一平台：SysOM 将应用观测，主机管理、监控、诊断、审计、修复、安全能力 集于一体，核心的功能采用模块化设计，界面与核心服务分离，方便客户的二次集成，解决常规操作系统监控各类专业看板和告警无法与用户自身编写的代码关联的核心痛点，降低运维的门槛简单易用。
- 应用观测：实时观测和分析应用的行为和性能，追踪和展示应用请求在不同组件之间的传递和处理过程，并利用 eBFP 技术对应用进行深入监控和分析，定位应用问题根因，目前支持 HTTP、HTTPS、DNS、gRPC、MySQL、Redis、Dubbo、Java、Nginx、Tomcat 等协议的解析。
- 智能监控：联动监控、告警、诊断和分析等多个环节，全方位地监控容器，节点以及集群资源，以健康度来展示系统异常事件，并实现自动化根因分析，让用户一目了然地看到问题原因和解决方案。让每一次告警都“知其所以然”。

- 持续性能追踪：提供一站式性能追踪方案，支持容器、主机、集群、应用等维度的性能追踪，涵盖 C、C++、golang、Rust、Java 等多种主流编程语言，并支持 AI 智能诊断，帮助用户快速定位性能瓶颈。
- 稳定安全：提供统一的安全中心，为用户所管理主机提供全方位的漏洞监控、管理、修复，保障系统的安全性；同时提供各类安全加固能力，满足不同应用不同程度的安全要求。

6.5.2 Coolbpf：基于 libbpf 跨平台的跟踪诊断增强框架

6.5.2.1 概述

SysOM 运维环境复杂，面临多种硬件平台和多种内核版本的代码维护困难以及跨平台兼容性问题。为了解决这些问题，我们开发了基于 libbpf 的跨平台跟踪诊断增强框架-Coolbpf。Coolbpf 主要有以下三个特点：

- 卓越的跨平台支持：适配 4.19 及以上内核版本，兼容 ARM、x86 等多种硬件平台。
- 功能全面：Coolbpf 不仅提供高级网络功能，全面监控网络性能指标，还集成了火焰图性能分析工具，能够对 Rust、C/C++、Go、Java、Lua/LuaJIT、Python 等多种编程语言进行性能分析。
- 易于使用和集成：Coolbpf 通过 C 库简化了第三方应用的集成过程，降低了技术门槛。

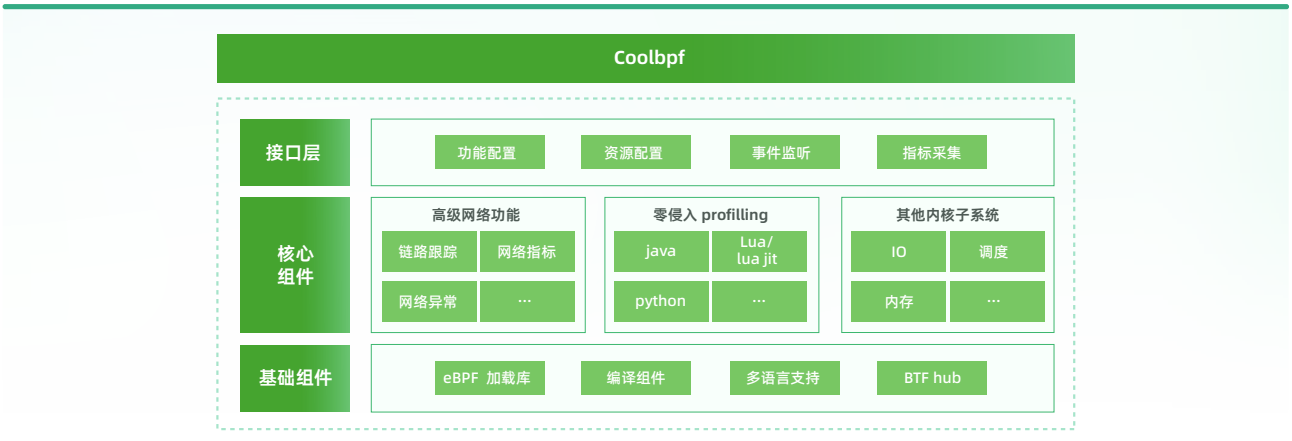
6.5.2.2 技术描述

Coolbpf 的功能架构如下图所示，主要分为三个层次：

1. 接口层：提供给用户的接口，目前主要有 4 类接口，分别是：①功能配置：配置和管理不同BPF功能或任务。可以通过用户接口或配置文件进行配置。②资源配置：获取系统或应用程序的监控数据，这通常由BPF探针执行，并用于收集特定的性能指标或事件。③事件监听：处理和转发采集到的事件到指定的处理模块。④指标采集：协调和管理采集到的数据，确保它们按照计划进行分析和处理。

2. 核心组件：由三个核心模块组成，高级网络功能、零侵入式性能分析以及其他内核子系统。高级网络功能模块涵盖了链路追踪、网络性能指标监测以及网络异常检测等关键特性。无侵入式性能分析是 Coolbpf 的亮点之一，它基于 eBPF 技术，提供了强大的性能分析工具，并且已经实现了对 Java、Lua/LuaJIT、Python、C/C++、Rust、Go等多种编程语言的支持。此外，其他内核子系统模块则包括了对 I/O、调度和内存等关键性能指标的监控。

3. 基础组件：它提供了 Coolbpf 运行所需的底层支持，主要有：①eBPF 加载库：加载和管理 eBPF 程序。②编译组件：提供了编译环境。③多语言支持：支持多种语言来扩展 Coolbpf 功能。④BTF hub：BTF 集中库，用于管理和查找内核数据结构定义，确保 eBPF 字节码与内核数据结构的一致性。



6.5.2.3 应用场景

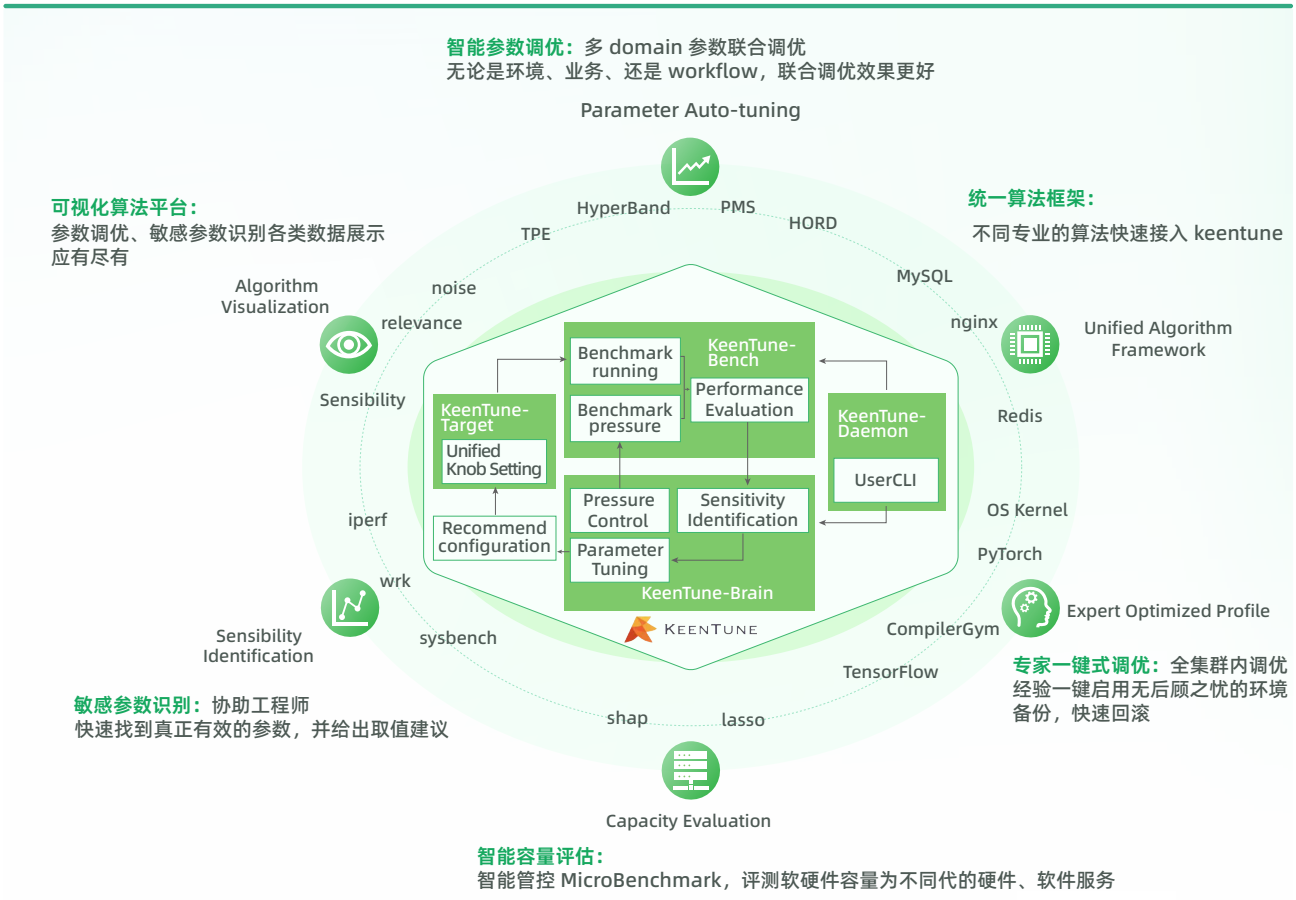
Coolbpf 正如其名，非常”酷“地提升了 eBPF 的开发效率，同时保证工具的跨平台兼容性。其主要应用场景为系统故障诊断、网络优化、系统安全和性能监控。

未来，Coolbpf 还将探索新技术和新特性，例如提升字节码翻译效率和内核运行时安全等，进一步丰富应用场景。

6.5.3 KeenTune：智能化全栈调优&容量评估工具

在进行业务的全栈调优中，往往遇到业务场景复杂、参数关系繁杂、调优成本高周期长、调优经验不应固化和扩散等问题。同样，在硬件、软件的容量评估中，也存在类似的问题。为了解决上述问题，KeenTune 基于 AI 算法与专家知识库，形成了以“专家一键式调优”和“智能参数调优”为主，“敏感参数识别”、“智能容量评估”为辅，“可视化算法平台”、“统一算法框架”配合，一套完整的智能化性能调优&容量评估的能力。

如下图中心框图所示，KeenTune 有五个模块：Daemon、Brain、Target 、Bench 和 UI。各模块可分可合，保证了部署的多样性，以应对不同的业务需求。

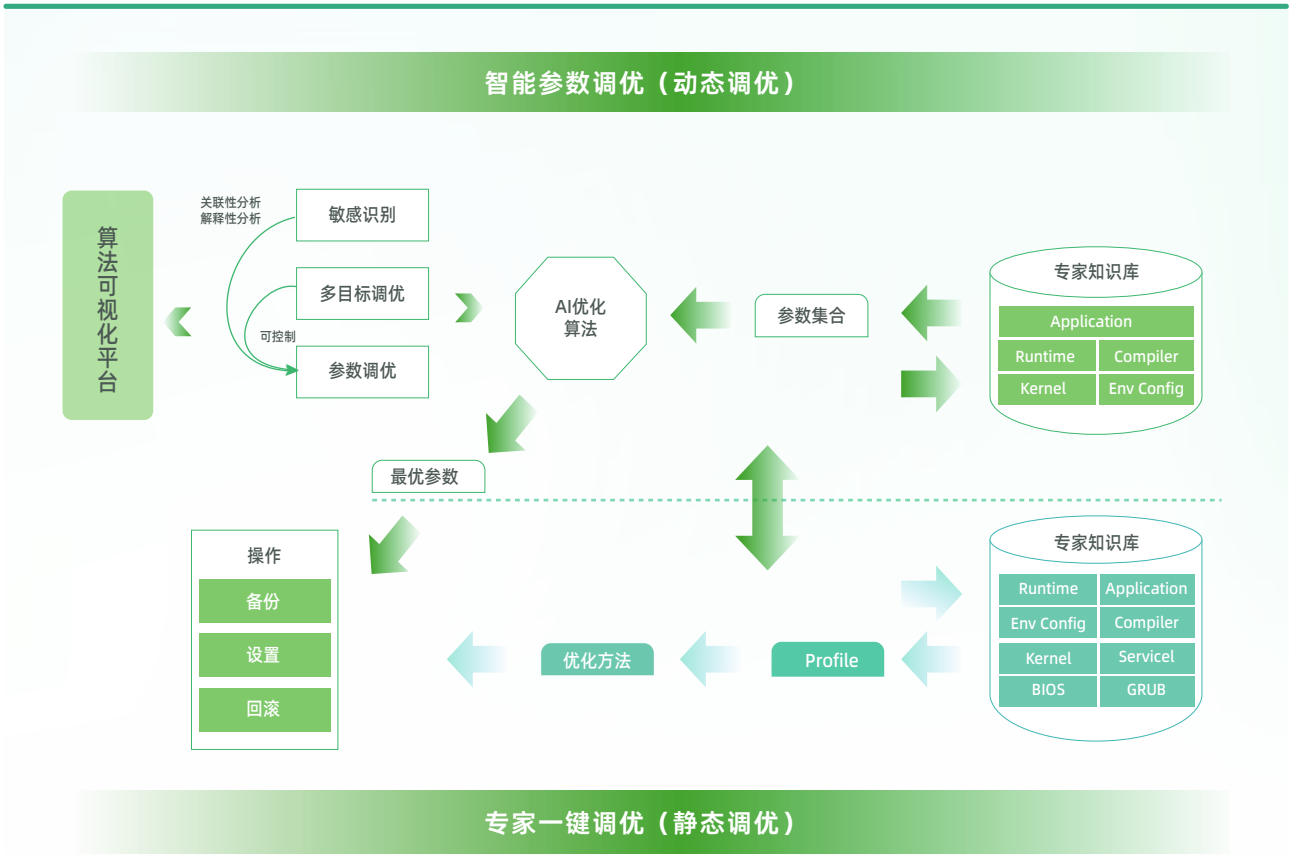


6.5.3.1 智能化性能调优

KeenTune 通过 AI 算法与专家知识库的配合使用，通过智能参数调优（动态）与专家一键式调优（静态）两项能力来实现 OS 上应用的高效调优。

- 智能参数调优：集成自研及主流开源的高效算法，提供内核、编译器、运行时、应用全栈的智能参数调优，全方位提高业务性能；同时，高效可信的参数可解释性算法辅助人工决策线上业务的最佳配置。

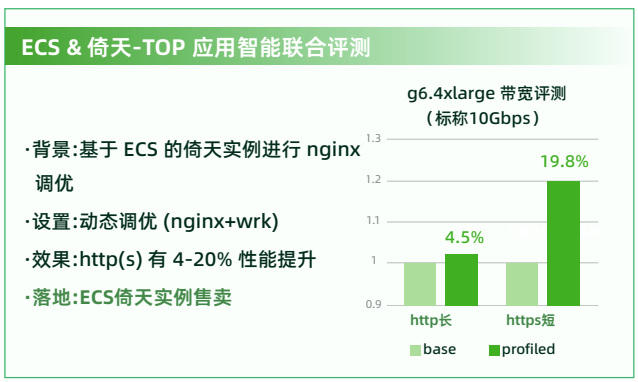
- 一键专家调优：针对基础业务负载和云上 TOP 应用，KeenTune 都提供了典型场景的调优专家库，一键设置即可提升该场景运行环境的性能。
- 动、静态的联合调优：动态调优的结果固化为静态调优配置，一套环境调优可以有效扩展到多套；静态调优配置成为动态调优的初始参数集合，有效保证应用运行在定制化的最佳设置环境。

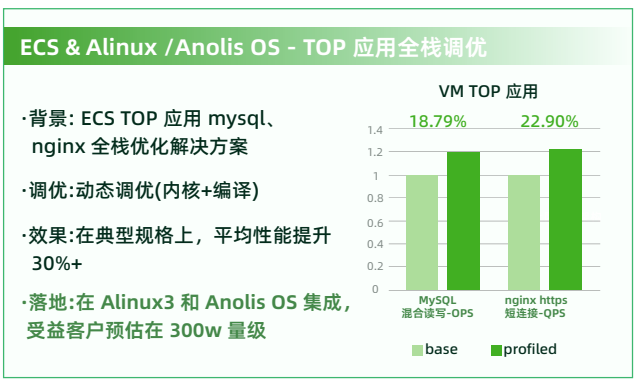


6.5.3.2 智能化容量评估

KeenTune 借助参数调优算法的能力，实现了对于 benchmark 参数的管控，从而达成快速进行系统容量评估的目的。目前已经形成了硬件容量评估的体系，覆盖 CPU、内存、IO、网络。后续也会持续在系统、业务容量评估持续发力。

KeenTune 在公有云、私有云、物理机上的多种业务场景都有比较好的效果，典型的一些如下：





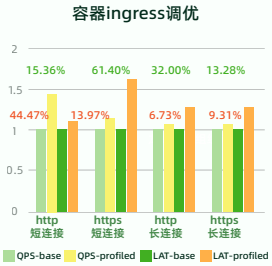
ACK-ingress-nginx 调优

·背景: ACK TOP 业务 ingress，大客户降本增效

·调优:动态调优(nginx+内核)

·效果:http(s) 长短连接场景均有 5 %-45 % 的性能提升

·落地: ACK 大客户解决方案提供定制化配置，受益节点 200+



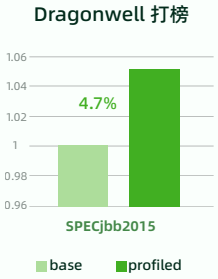
Dragonwell-SPECjbb 调优

·背景:Dragonwell 进行 SPECjbb 2015 打榜

·调优:静态调优(内核)

·效果: Dragonwell 在已经人工调优的基础上，性能提升 4.7%

·落地:dragonwell 打榜



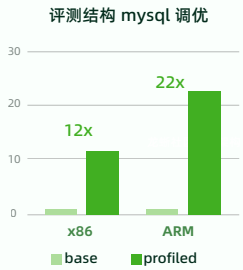
Anolis OS & 统信- mysql 调优

·背景:评测机构送检

·调优:静态调优 (BIOS、内核、disk、编译、应用配置等)

·效果:现场一键调优，x86 和 arm 上性能提升10 倍+

·落地:通过发改委评测



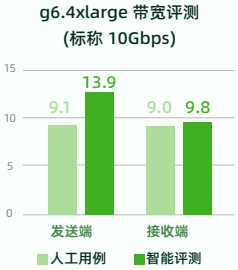
统信&私有云-智能容量评测

·背景:不同硬件、OS 上 CPU、内存、IO、网络重要资源的容量评测

·设置:动态调整，benchmark 配置

·效果:iperf3 在裸机、VM 不同规格均能有效评测网络收发端的容量

·落地:统信商业打榜+倚天容量评估 +SIG 持续合作输出



6.6 软硬件协同

6.6.1 多平台全链路 RAS 能力

6.6.1.1 背景介绍

随着云计算、大数据、物联网、人工智能大模型等领域的蓬勃发展，互联网数据存储规模和计算需求呈现爆炸式增长，大规模数据中心已成为支撑互联网服务的标准基础设施。数据中心的规模化部署，主要体现在：

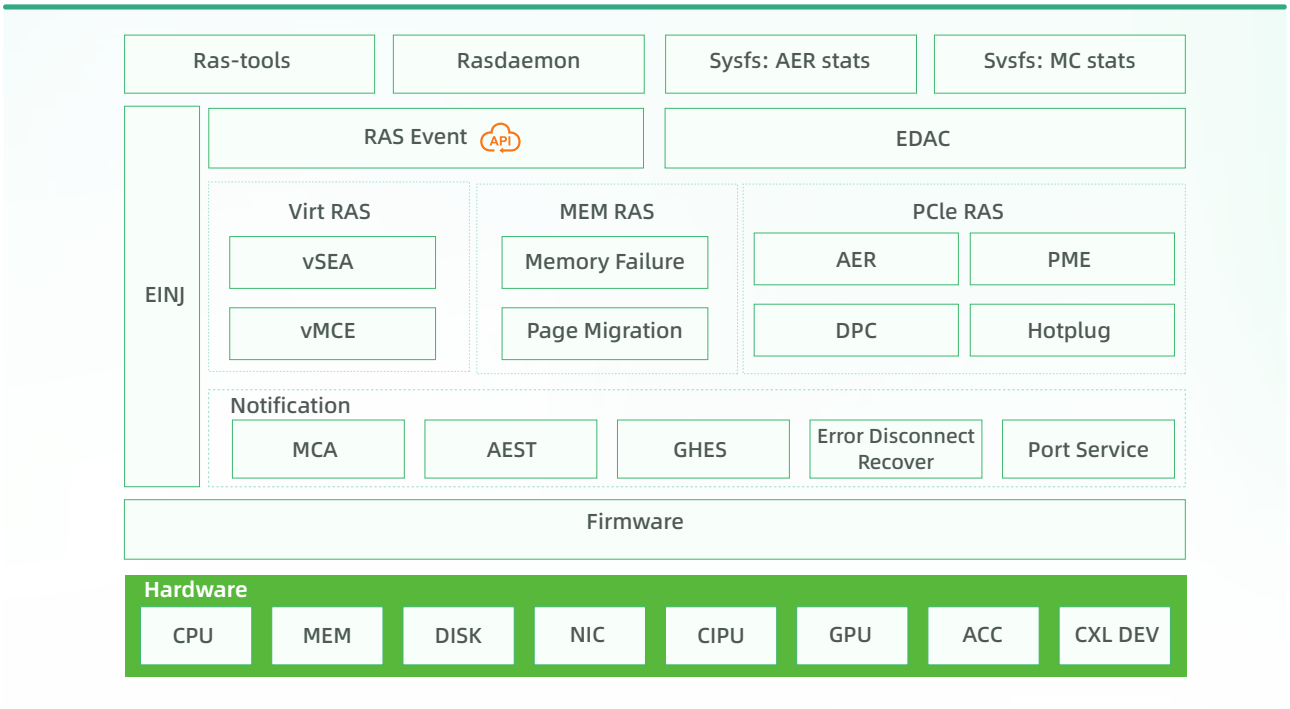
1. 数据中心的规模横向增长（scale out）：随着服务器组件的增多，以及数据存储和传输的总量不断增加，传统上很少发生的错误开始变得可见。任何材料磨损、环境影响、制造和设计缺陷等边界影响，都会被放大，导致系统的可用性或性能受到影响。

2. 数据中心的算力纵向增长（scale up）：在计算集群和数据中心，单台物理机部署的硬件和软件密度越来越高。物理机的 CPU 核心由数十向数百增长，内存由 GB 级向 TB 级发展，GPU 向 switch 级联拓展发展，可部署上百个 VM[3]和 2500 个容器实例。任何一台服务器宕机，造成的业务影响和成本损失被进一步放大。

3. 数据中心的负载多元化发展：人工智能训练、推理、图像视频处理和 AIGC 大模型等各种不同需求和形态的高性能计算类应用发展，驱动算力需求不断攀升，传统的单一计算类型和架构的处理器已经无法处理更复杂、更多样的数据。跨越标量（CPU）、矢量（GPU）、矩阵（ASIC）、空间（FPGA）的异构计算，如今已经成为企业推动 IT 基础设施重构的重要力量。 在超大规模云计算数据中心中，服务器宕机率一直是衡量 RAS（可靠性、可用性和可服务性）的一个关键指标，也是满足云计算终端用户 SLA（服务水平协议）的首要问题。稳定性是操作系统的核心竞争力之一，随着云服务架构、智能计算的持续发展，未来在基于通用和异构算力的自研芯片（如 CPU、GPU、CIPU、DPU 等）、一云多芯（如X86、ARM 等）场景下进行软硬件协同创新优化，充分利用硬件的 RAS 能力和软件恢复性技术，构建全面的端到端 RAS 解决方案，以确保操作系统的稳定性和运维能力。

6.6.1.2 技术方案

技术大图覆盖：



RAS 技术包括：

1. 硬件故障预防、检测、移除和注入

2. 固件错误收集、处理和上报

3. OS/VMM 错误恢复和处理

4. 错误记录、分析和预测

6.6.1.3 技术优势

全平台错误注入、验证和收集

1. ras-tools 自底向上验证和评估硬件、固件、OS、虚拟化以及上层应用的RAS上报、隔离和恢复能力，支持 CPU、内存、PCIe 错误注入，丰富测试用例用于覆盖测试。

2. rasdaemon 跨平台错误日志收集，同时支持 CPU/内存/PCIe/GPU 错误日志的收集。

增强内存隔离与恢复

1. 进程的用户地址空间由页表提供隔离，对于用户态进程消费 UCE 的情况，利用同步异常的特性，内核通过将错误的影响范围控制到进程粒度，从而实现对 UCE 错误的隔离和恢复，避免系统宕机。

2. 对于被内核消费的 UCE 错误，进行更细粒度的区分，内核通过 copy_from_user 和 get_user 等 uaccess 接口，从用户地址空间拷贝数据到内核地址空间，如果拷贝的数据中包含 UCE 错误，按照 POSIX 标准，只需要返回错误码 EFAULT 或已拷贝的长度，从而避免内核态消费 UCE 错误宕机。

3. 在 Scrubber 发现的 UCE。由于这个 UE 并不是进程的访存行为导致的，与进程上下文无关（Non-execution Path），不会导致错误传播，因此，并不需要立马宕机。内核只需要将 poison 页，打上 poison 标记，并解除页表映射即可。如果在后面进程的生命周期内，读取 poison 页，则在触发 page fault 时，向进程发送 SIGBUS 信号。

设备隔离与恢复

ANCK 内核完整支持 PCIe 设备 AER 和 eDPC 功能，提供 PCIe 设备的错误上报和隔离能力，此外 ANCK 还支持 CXL 设备的错误上报和隔离能力，其能力一部分遵循 PCIe AER/eDPC 标准, 另一部分遵循内核隔离与恢复。

统一错误监控与告警

使用用户态 rasdaemon 组件实现硬件错误的监控能力，将操作系统运行时的硬件错误全部收集到一起，再根据错误的类型和严重程度进行划分，最终转化为不同严重等级的错误日志记录在操作系统日志中。

6.6.1.4 应用场景

RAS 能力是云厂商满足云计算终端用户 SLA（服务水平协议）的重要技术，可以应用在所有的云架构中，尤其是随着智能计算的发展，应用对整机内存的需求越来越大，RAS 是保障内存容错能力的重要技术。

6.6.2 面向 DPU 场景的软硬协同协议栈

6.6.2.1 背景概述

随着传统数据中心中的应用大规模上云，对云上 VPC 网络的时延和吞吐都提出了更高的要求；与此同时云计算场景下 DPU 的能力 进一步增强，开始在云上提供普适的 RDMA 能力。此时传统的虚拟网卡和 TCP 协议栈已不能满足其对于网络吞吐、传输时延和增效 降本的要求。因此一个能向下充分发挥 DPU 硬件性能，向上能支撑大规模云上应用场景，开发部署和运维友好，兼容主流的云原生 等业务架构的软硬件协同协议栈，能大幅提升云上应用网络性能，进一步提升云的竞争力。

6.6.2.2 技术方案

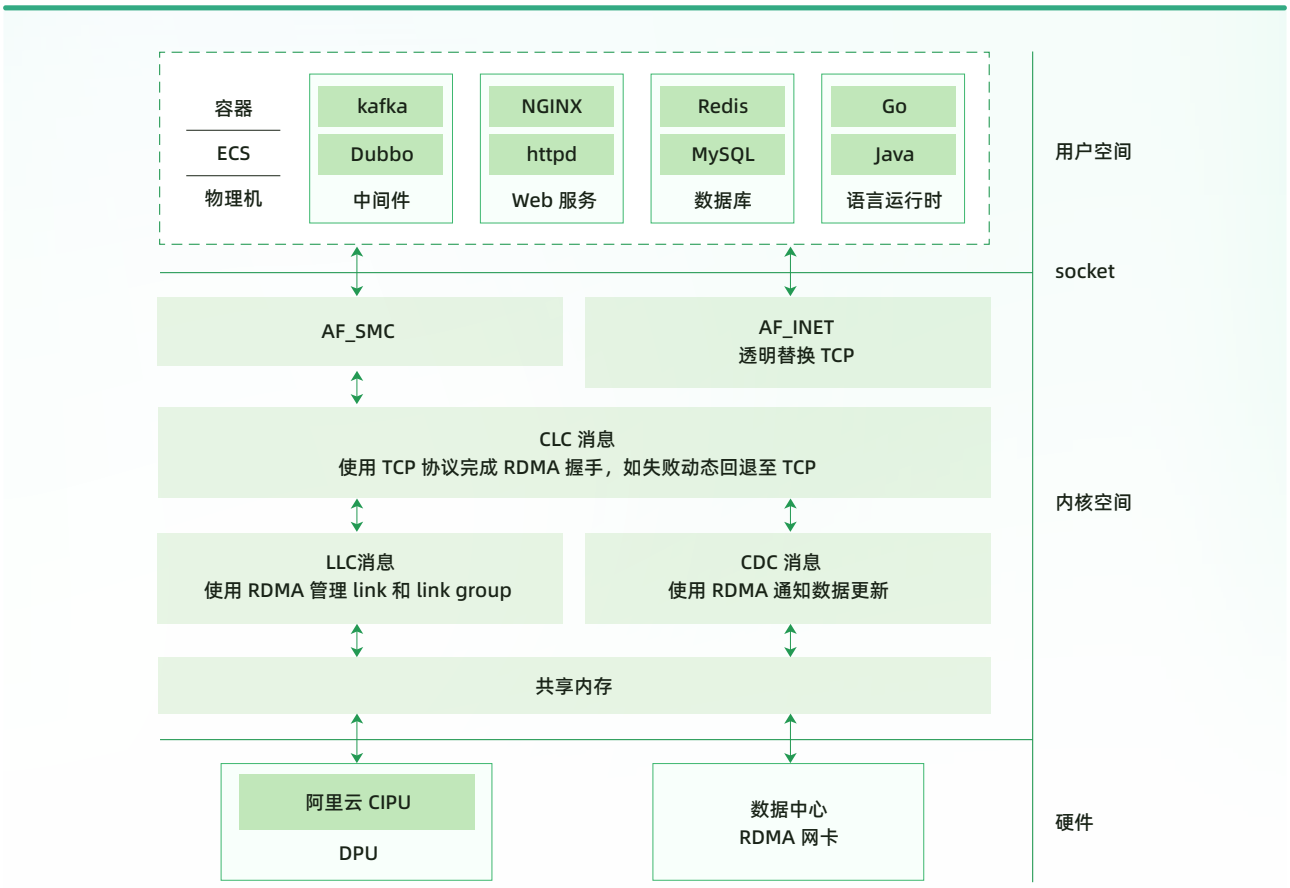
共享内存通信 SMC 是由 IBM 首次贡献至 Linux 社区，并由龙蜥增强和维护的软硬协同的高性能协议栈。针对不同的规模场景、硬件和应用模型，SMC 提供多位一体的方案以解决当前传统协议栈的问题：

1. 借助云厂商 VPC 或者数据中心 RDMA，实现不同规模和场景下的高性能通信，支撑不同的业务规模和场景。

2. 兼容 RDMA verbs 生态，实现协议栈卸载至硬件，提升网络性能，降低 CPU 资源使用，支持多种硬件。

3. 透明替换网络应用，SMC 完全兼容 TCP socket 接口，并可快速回退 TCP。

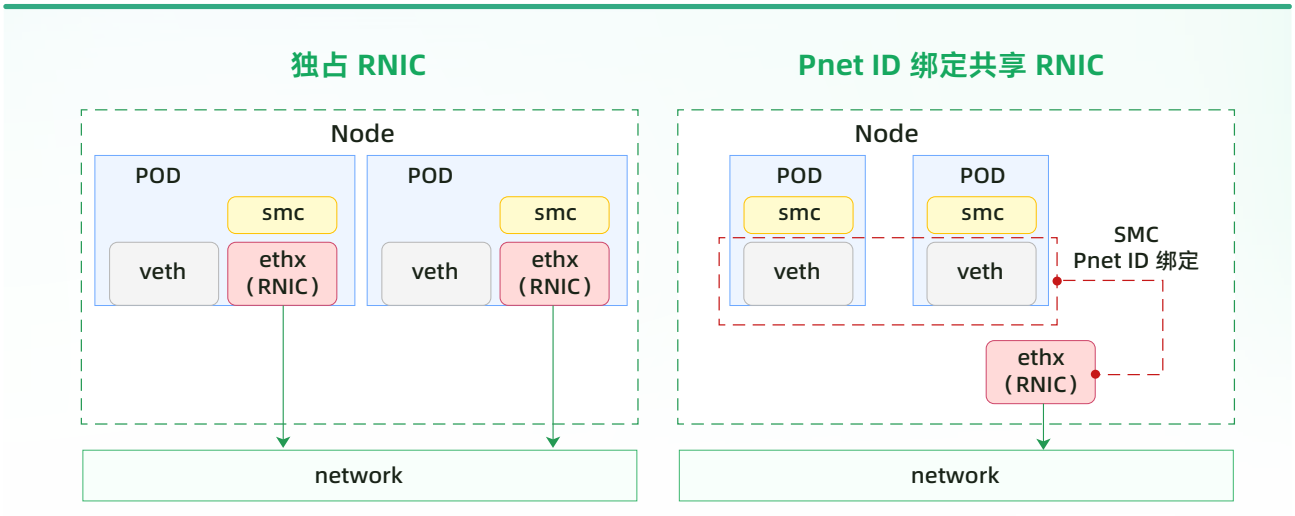
4. 使用统一高效的共享内存模型，借助硬件卸载实现高性能的共享内存通信。



在容器场景中使用 SMC

容器场景使用 SMC 可以有以下两种选择：

- 对于追求极致性能且容器密度不高的场景，可以配置 POD 独占 RDMA 设备。POD 内开启 SMC 透明替换后，在 RDMA 设备对应以太网接口上的 TCP 流量将被 SMC+RDMA 加速。
- 对于容器密度较高的场景，可以通过 PnetID 配置将 POD 内以太网接口与 Node 上的共享 RDMA 设备绑定。POD 内开启 SMC 透明替换后，被绑定以太网接口上的 TCP 流量将被 SMC+RDMA 加速。



6.6.2.3 技术优势

1. 透明加速传统 TCP 应用，对于应用程序、运行环境镜像、部署方式无侵入，对 DevOps 和云原生友好；

2. DPU 软硬协同的网络协议栈，更高的网络性能和更低的资源使用；

3. Linux 原生支持的标准化、开源的网络协议栈，SMC-R 实现自 IETF RFC7609，由社区共同维护；

6.6.2.4 应用场景

SMC 是一个内核原生支持的通用高性能网络协议栈，支持 socket 接口和快速回退 TCP 的能力，任何 TCP 应用均可实现透明替换 SMC 协议栈。由于业务逻辑与网络开销占比的差异，不同应用的加速收益存在差异。下面是几个典型的应用场景和业务最佳实践：

1. 内存数据库，Redis 和部分 OLAP 数据库，Redis QPS 最高提升 50%，时延下降 55%。

2. 分布式存储系统，云原生分布式存储 Curve 在 3 volume 256 depth randwrite 场景下性能提升18.5%。

3. Web service，NGINX 长链接下 QPS 最高提升 49.6%，时延下降 55.48%。

4. 事件流处理平台 Kafka，在不同流量压力下降低 3%-40% 的网络时延。

5. 服务网格 ServiceMesh，sidecar 模式下 fortio HTTP 负载 QPS 提升 10%-18%, TCP 负载 QPS 提升 30%-50%。

6. 搜推训练，分布式 ps-worker 架构 Wide&Deep 模型训练吞吐提升10%-15%。

6.6.3 面向 HTTP 3.0 时代的高性能网络协议栈

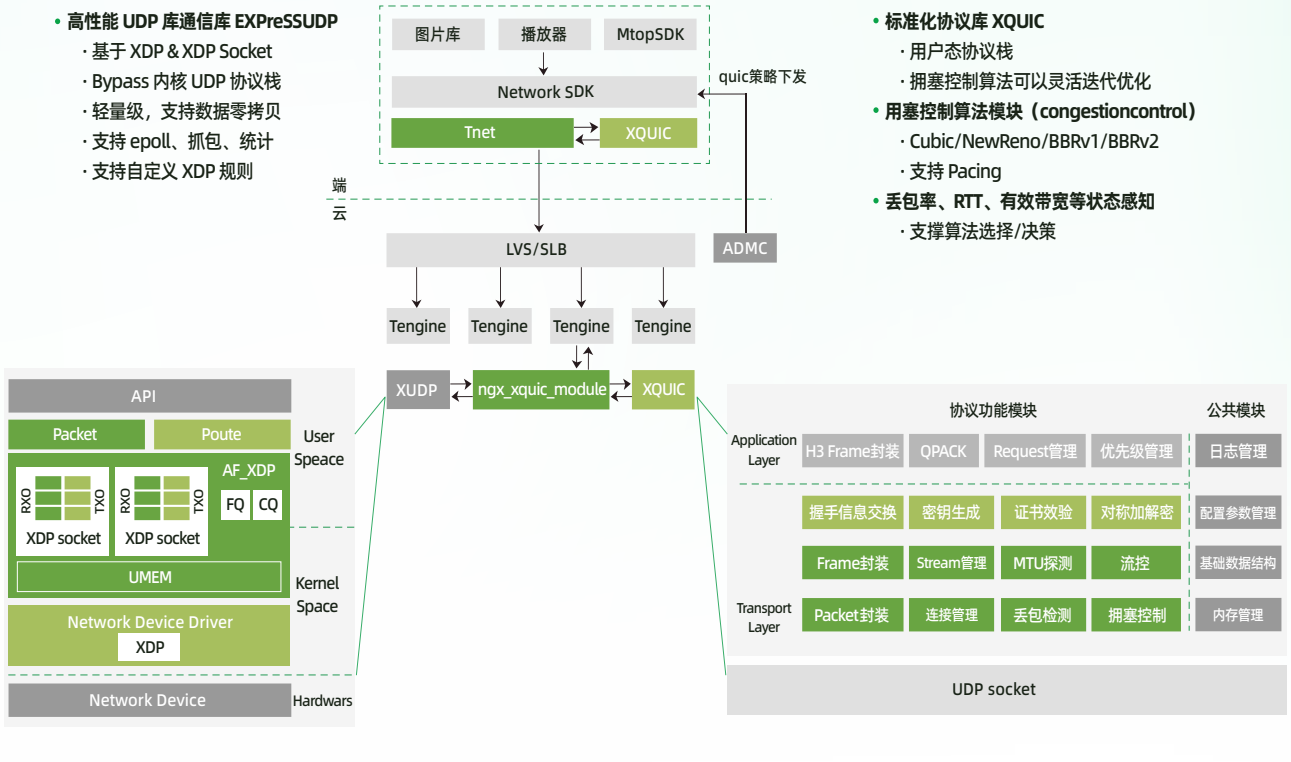
6.6.3.1 应用场景

随着互联网特别是移动互联网的快速发展，对互联网通信协议提出了新的诉求。经过多年的发展，QUIC 协议在 2021 年正式被 IEFT 标准化，成为 HTTP 3 的标准传输层协议。QUIC 是基于 UDP 实现的面向连接可靠有序的传输协议。相比于 TCP 在内核态实现，QUIC 基于 UDP 在用户态实现大大降低了部署成本，并且可将拥塞控制算法/参数调控到连接的粒度，灵活适应不同业务场景的网络需求。此 外，QUIC 还具备多路复用/0-RTT 握手/连接迁移等多种优点。

6.6.3.2 轻量、高性能、标准化的 HTTP 3.0 解决方案：XQUIC + ExpressUDP

XQUIC 是一个轻量、高性能、标准化的跨平台 IETF QUIC 协议库，内部包含了 QUIC-Transport（传输层）、QUIC-TLS（加密层、与 TLS/1.3 对接）和HTTP/3.0 （应用层）的实现，为应用提供可靠、安全、高效的数据传输功能，在 IETF 开发者社区进行了比较充分的互通性验证，支持 Linux/Android/iOS/Mac/Windows 等平台，在 Android/iOS 双端的编译产物均小于 400KB，适用于需要高性能但同时又对包大小敏感的移动端 APP 场景，可以方便地部署到移动设备和各种嵌入式设备中。在服务端方面，基于阿里内部广泛 使用的 Tengine，开发了 ngx_xquic_module 和 ngx_xudp_module 模块用于适配 Tengine 服务端。

图 XQUIC+ExpressUDP 整体架构和传输框架



XQUIC 还添加了对多路径传输的能力支持，可以同时利用 cellular 和 wifi 双通道进行数据传输。多路径 QUIC 草案已经被 IETF 工作组正式接收纳入标准，我们与达摩院 XG 实验室共同研发的 XLINK 多路传输技术，相关论文「XLINK: QoE-driven multi-path QUIC transport in large-scale video services」已经发表在网格顶会 SIGCOMM 2021 上。

ExpressUDP 是一个基于 Linux XDP Socket 和 XDP 实现的高性能用户态 UDP 通信库。利用 XDP 和 XDP Socket 封装了一套高性能 UDP 通信接口，为应用提供了一种高性能 UDP 通信编程框架。可以为高 PPS UDP 通信场景带来显著的网络性能提升。ExpressUDP 具有与 软件结合简单，不影响软件本身框架的特点。具备高吞吐，低延时，开发方便，部署简单等优点。利用龙蜥 OS 在业界首次实现了 virtio_net的发送方向的零拷贝，UDP PPS 相比非零拷贝提升 4 倍，相比普通内核 UDP 提升 7 倍以上，接收方向相比内核 UDP 提升 3 倍 以上。

6.6.3.3 应用场景

XQUIC 已经在手淘 Android/iOS 双端正式版本，XQUIC + ExpressUDP 已经在阿里巴巴集团统一接入网关大规模应用。在 RPC 请求场 景降低网络耗时 15%，在短视频场景下降低 20% 卡顿率，在上传场景提升 25% 上传速率（相对于 TCP）。在服务器端性能上，经过 ExpressUDP 适配改造，E2E QUIC 性能提升了 30%~50%。

6.7 安全可信

6.7.1 商密软件栈与后量子密码支持

从商密算法标准公布到现在已有十多年时间，与 AES、SHA 等主流国际算法相比，目前商密在基础软件中的支持和优化仍然不完善，甚至有较大的差距，商密算法的软硬件生态也处于碎片化状态，密码算法作为网络和数据安全的基石，应该且有必要在基础软件中具备开箱即用的能力；另一方面，密码算法是保障信息和数据安全的核心技术，随着近年来外部的国际贸易冲突和技术封锁的加深，内部互联网的快速发展，我们不能单一依赖国外的技术标准和产品，增强我国行业信息系统的安全可信显得尤为必要和迫切。商用密码算法给我们提供了一个新的选择，使得我们可以完全使用商密技术来构建网络和数据安全环境。

6.7.1.1 商用密码生态

商密软件栈 SIG 依托基础软件上游，秉承为已有轮子支持商密的原则，在全栈范围内的多个基础组件中支持了商密算法，包括 Linux 内核、OpenSSL、libgcrypt、gnulib、nettle 和 Dragonwell-Security-Provider 等在内的基础组件，涵盖了 C/C++和 Java 程序语言生态，实现了商密算法以及大量的性能优化；其次在诸多的密码应用场景中也使能了商用密码，比如文件加密、身份认证场景等。同时这些实现得到了上游社区的支持进入主线，基本补齐了商密算法在基础软件中的短板，在兼容已有 API 和生态的情况下，提供给普通开发者开箱即用和平滑的商密使用体验。

6.7.1.2 商用密码应用场景

在以下的场景中，通过在各基础软件中支持的商密实现，可以平滑地从国际主流算法切换到商密算法上来，在提供高安全性的同时，也有效避免了国外技术封锁带来的风险

- 文件加密 fscrypt 支持使用 SM4 算法，以及在 SM4-XTS，SM4-CTS 模式下的优化加速；
- 磁盘加密 LUKS 支持使用 SM4 商密算法；
- Kernel TLS 支持使用 RFC8998 规范中定义的商密算法套件（TLS_SM4_GCM_SM3 / TLS_SM4_CCM_SM3）；
- 基于身份认证的机制，比如 IMA、modsign，支持使用 SM2/SM3 算法组合的签名验签；
- 支持 GB/T 38636-2020 TLCP 标准，即双证书商密通信协议；

- 极致的商密算法性能优化，最大性能提升近 40 倍，适用于对算法性能要求较高的场景；
- Java 生态 Dragonwell-Security-Provider 提供极高性能国密(SM2/SM3/SM4)算法库，且支持TLS1.3+国密算法套件（TLS_SM4_GCM_SM3/TLS_SM4_CCM_SM3）协议标准 RFC8998，完全兼容 JCA/JSSE 框架接口标准；

6.7.1.3 后量子密码支持

量子计算的出现，为计算科学带来了范式级别的转换，借助量子叠加与量子纠缠的独特性质，量子计算机能够实现高度的计算并行性，使其在处理特定类型的数学问题时，展现出相较于经典计算机的指数级优势。理论上的强大算力已对现行密码学的安全根基构成了根本性的威胁，尤其是以 Shor 和 Grover 为代表的算法对当前主流的密码体系构成了颠覆性威胁。使得“store now, decrypt later”先收集后解密类型的攻击变得可能。

在这样的背景下，龙蜥依托 Tongsuo 密码库提供基于 PQC 的后量子密码支持，比如 ML-KEM、ML-DSA 等用于后量子时代的密钥封装和签名算法，并结合已有的商用密码算法形成了混合密钥协商和混合密钥签名能力，作为从传统密码向后量子密码过渡时期的解决方案。

6.7.2 龙蜥软件物料清单

Software Bill of Materials (SBOM)，软件物料清单，是一种正式的、机器可读的软件元数据信息，包含了软件的基本信息和软件依赖的所有开源代码和第三方组件的列表。SBOM 中还列出了管理这些组件的许可证、代码库中使用的组件的版本以及它们的补丁状态等信息，为研发人员提供了快速识别任何相关的安全或许可证风险的能力。SBOM 可以用于供应链安全管理、安全漏洞管理、软件合规管理、应急响应等场景，帮助软件的生产者、运营方和使用方高效快速地识别软件成分，排查软件合规风险，快速识别安全漏洞的影响范围等。

6.7.2.1 龙蜥 SBOM 支持

龙蜥社区已经实现了软件包、操作系统镜像、容器镜像的 SBOM 支持。龙蜥 SBOM 采用 SPDX v2.3 格式。SPDX，全称 Software Package Data Exchange，是 Linux Foundation 社区主导发起的一种开放的软件物料清单信息标准，并通过ISO 认证（ISO/IEC 5962:2021），成为正式的国际标准。其它流行的 SBOM 格式标准包括 CycloneDX、SWID 等。

通过对社区软件供应链安全基础设施的构建，包括社区 SCA 工具 abom（Anolis OS BOM）、社区软件信息中心（Package Center）、社区软件供应链风险管理平台（OSCM）等，龙蜥社区实现了对社区发布制品的自动化 SBOM 生成能力，并且能够通过 SBOM 提供软件研发周期的完整依赖信息，提升社区软件的透明性。

社区用户可以通过龙蜥社区软件信息中心（Package Center）（链接：https://package.openanolis.cn/）或者龙蜥社区安全中心（ANAS）（链接：https://anas.openanolis.cn/data）获取软件包 / ISO 镜像的 SBOM 文件。

6.7.2.2 龙蜥 SBOM 字段

以 NITA 推荐的 SBOM 最小字段集为基础，结合龙蜥社区本身的需求，并参考了不同 SBOM 格式的字段内容，龙蜥社区定义了自己的 SBOM 最小字段集合，能够充分满足社区的需求，并为后续支持其他 SBOM 格式提供了扩展能力。

SBOM 最小字段集合：

- 供应商/ Supplier
- 组件名/ Component Name
- 组件版本/ Component Version
- 唯一标识符/ Unique Identifiers，purl/cpe/swid

- 依赖关系/ Dependency relationship
- SBOM 作者/ Author of sbom data
- SBOM 生产时间戳/ Created Timestamp of SBOM data
- SBOM 版本/ Version of SBOM data
- SBOM 更新时间戳/ Updated Timestamp of SBOM data
- 组件的开源证书信息/ License of the Component
- 组件 Copyrights 信息/ Copyright of the Component
- 组件的 Hash 值/ Hash of the component
- 组件上游地址/ Source URL of the Component
- 组件官网/ Mainpage for the Component
- 组件下载地址/ Download Location of the Component
- 组件描述 / Description of the Component

推荐字段（Optional）：

- 组件的开发依赖/ Dev dependencies
- 组件的构建依赖/ Build dependencies
- 组件的构建工具依赖/ Build tools
- 组件的运行依赖/ Runtime dependencies
- 组件的补丁信息/ Patch info
- 组件的构建详情链接/ External ref of build info

6.7.2.3 基于 SBOM 的龙蜥供应链安全体系构建

依托 SBOM，龙蜥社区正在逐步构建供应链安全体系，保证交付制品的稳定、安全、合规。

6.7.3 机密计算技术

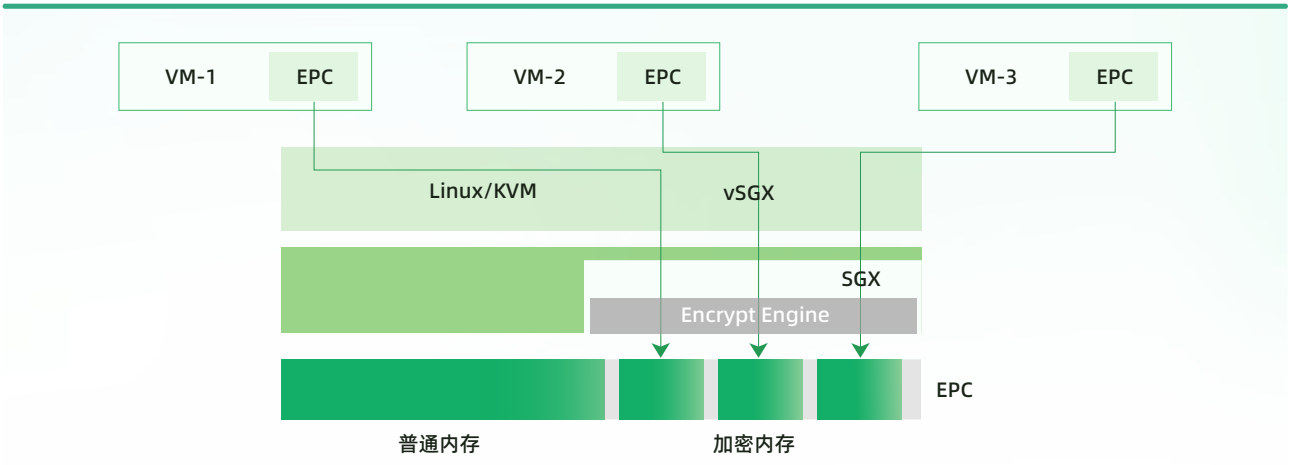
机密计算是一种依赖于硬件的使用中的数据保护技术。芯片厂商通过提供特殊的硬件指令、受保护的加密内存区域等手段，辅以基于硬件的密钥管理和密码学操作，为使用中的数据提供了一个受保护的可信编程环境，通常称之为可信执行环境（Trusted Execution Environment，简称 TEE）。

利用最底层硬件所能提供的安全性，在保持最小信任依赖的情况下，机密计算技术可以将操作系统和设备驱动程序供应商、平台和设备供应商、服务提供商及系统管理员从用户需要信任的实体列表中移除，从而大大降低了可信计算基（TCB，Trusted Computing Base）的大小。龙蜥社区为推动机密计算技术的应用，提供若干机密计算创新项目，目的是降低机密计算技术的使用门槛。

6.7.3.1 技术方案

SGX 虚拟化

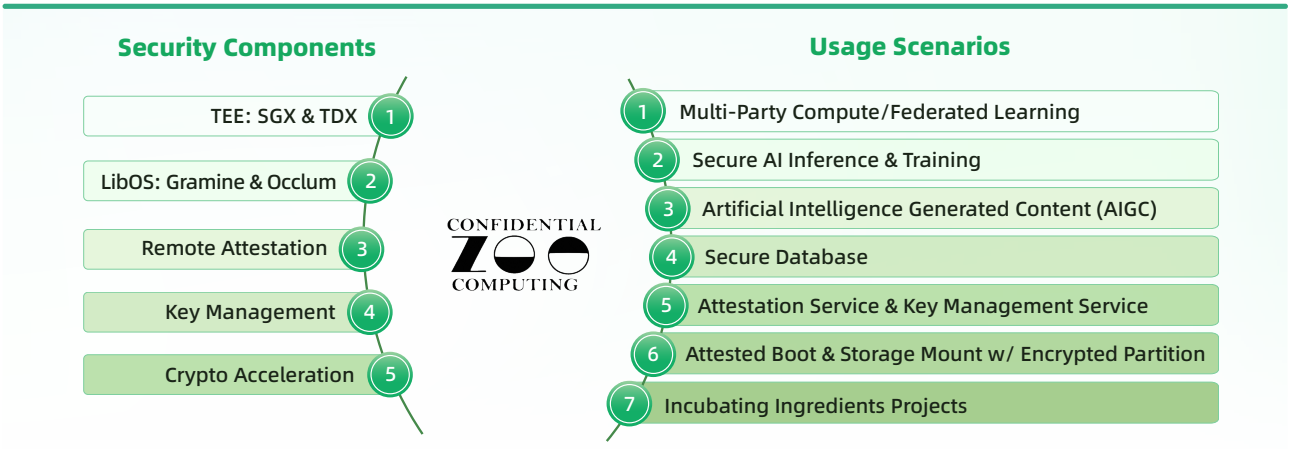
SGX 虚拟化允许将 SGX 硬件能力透传给虚拟机和容器，以允许用户将敏感工作负载运行在基于 Intel SGX Enclave 的 TEE 中。



目前 SGX 虚拟化已支持 Anolis OS 8，可为云上用户提供基于 Intel SGX Enclave 技术的应用级安全防护能力。

CCZoo

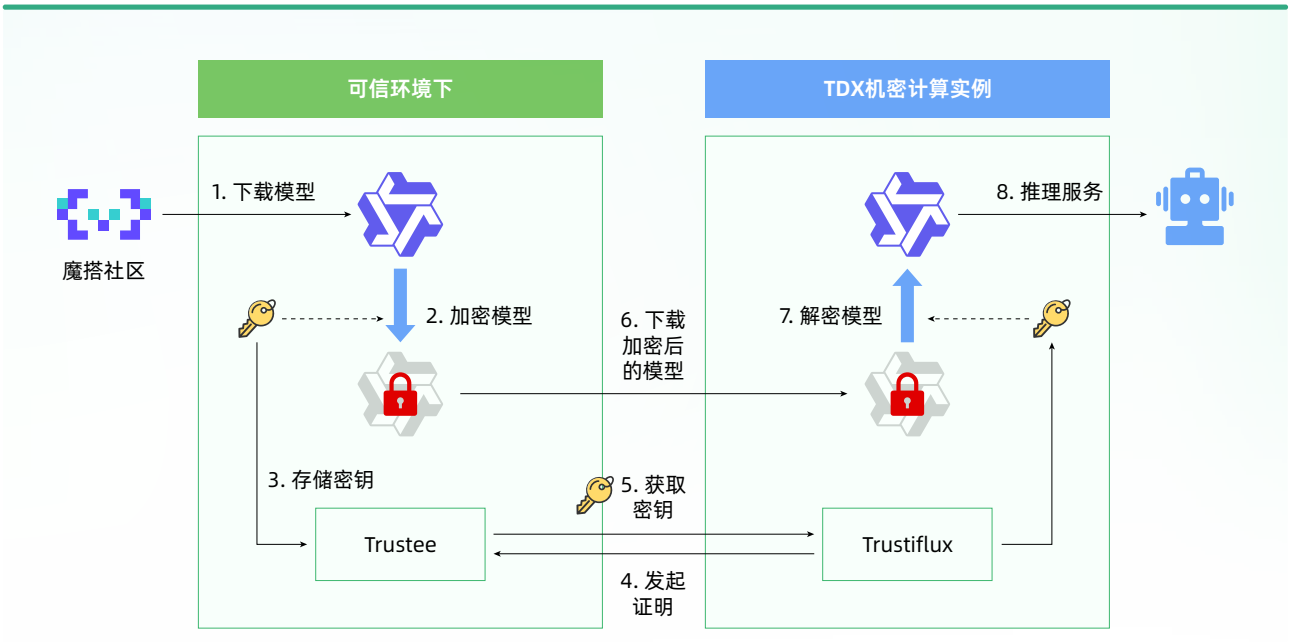
Intel 发起并开源 了Confidential Computing Zoo (CCZoo)。CCZoo 提供了不同场景下各种典型端到端安全解决方案的参考案例，增强用户在机密计算方案实现上的开发体验，并引导用户结合参考案例快速设计满足自己需求的机密计算解决方案。



CCZoo 目前提供了基于 LibOS Gramine + Intel SGX + OpenAnolis 容器的 E2E 安全解决方案参考案例，其中包括在线推理服务和横向联邦学习等。后续，CCZoo 计划基于 OpenAnolis 提供更多的机密计算参考案例，为用户提供相应的容器镜像，实现敏捷部署。

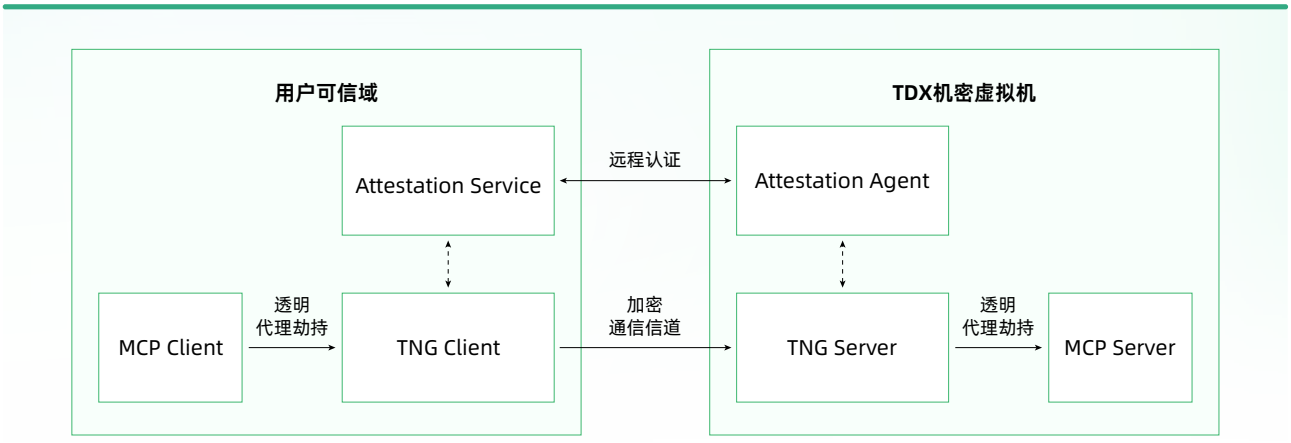
Confidential AI

Confidential AI 方案提供了在 Intel TDX 与海光 CSV 机密计算平台上部署隐私推理模型的标准化验证指南，支持 Qwen3、DeepSeek 等主流模型。方案采用“用户侧 Trustee 密钥托管 + 云端 Trustiflux 可信执行”的双端架构，演示了从模型加密存储、远程环境认证、动态密钥获取到安全解密加载的全流程。通过该实践，用户可快速构建基于远程证明的可信 AI 推理环境，确保敏感模型仅在通过严格身份验证的硬件隔离域中运行，实现端到端的数据隐私保护。



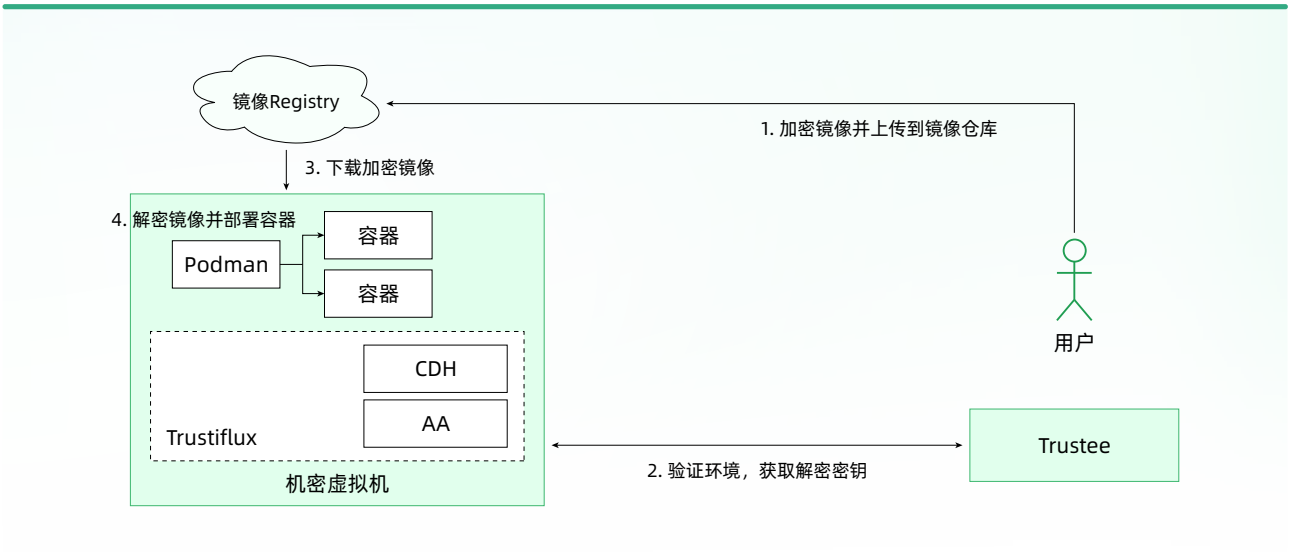
Confidential MCP

通过在 MCP 服务上使用 TNG 构建基于硬件远程认证的安全通信信道，一方面可以满足网络通信中数据的机密性和完整性保护的要求，另一方面可以实时对 MCP 服务运行的软硬件环境进行验证，保障其真实性和完整性。



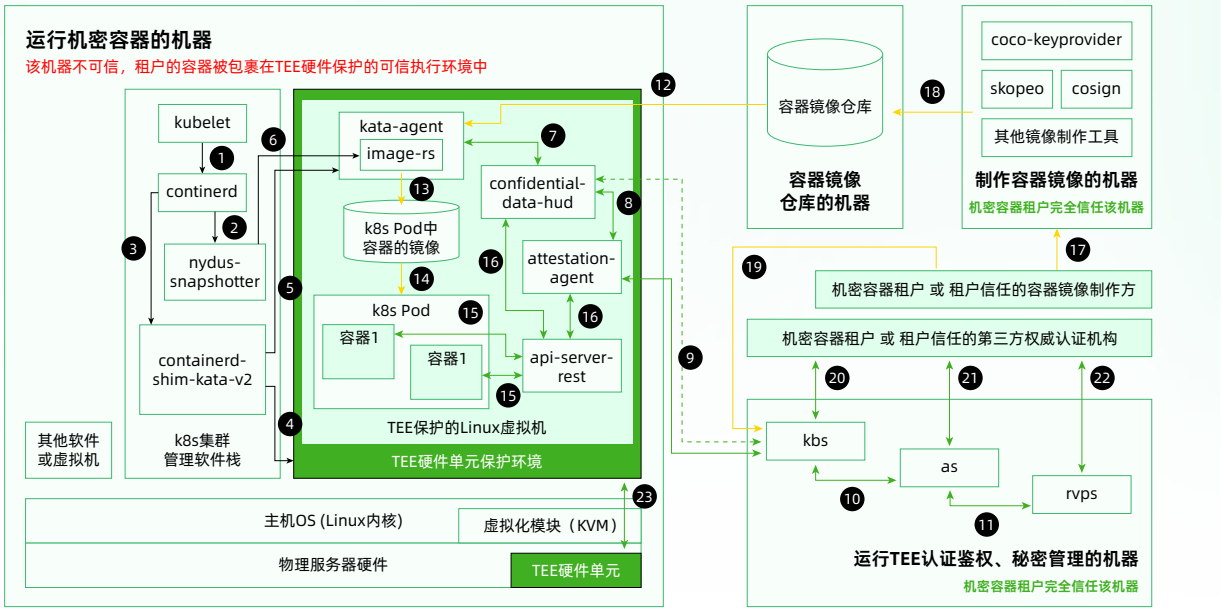
Trustiflux Docker 机密容器

通过在机密虚拟机的硬件级可信执行环境内部署 Trustiflux 机密运行时，配合 docker 做容器镜像解密，实现镜像的仓库中加密存储、网络中加密传输，仅在受信任的机密计算环境内解密与运行，从而实现“云侧不可见、平台不可窥”的敏感镜像全生命周期保护。



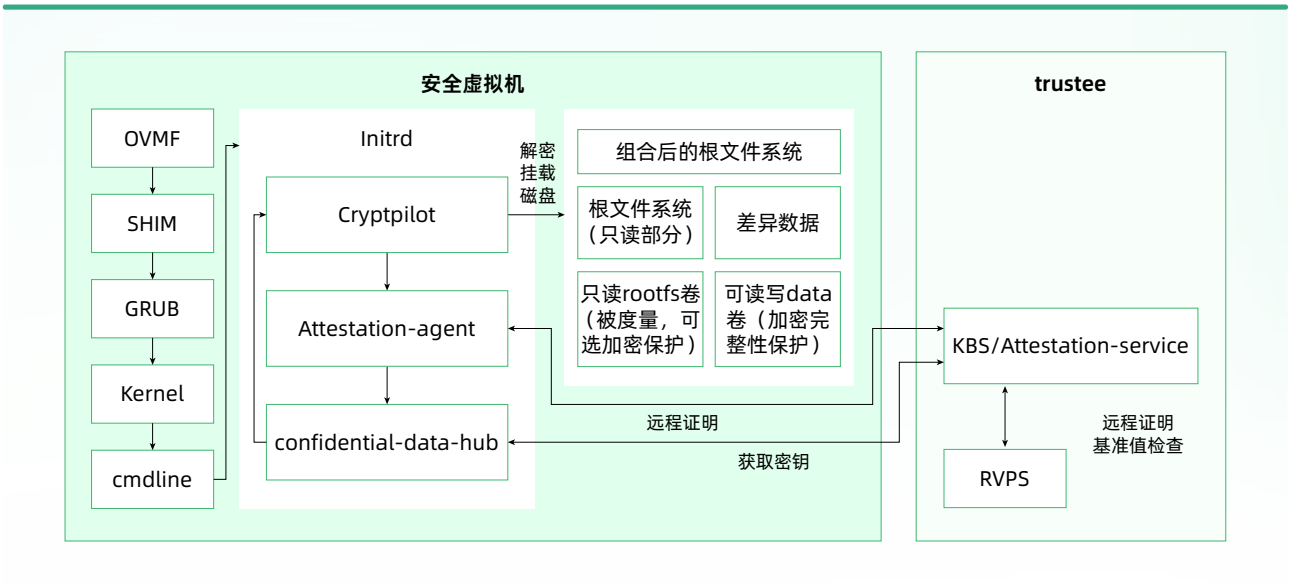
Kata 机密容器

Kata 机密容器是一种融合轻量级虚拟机隔离与可信执行环境（TEE）硬件技术的云原生安全方案，旨在实现“使用中数据”的端到端保护。它通过将容器运行于独立的机密虚拟机内，并结合远程认证、镜像加密签名及动态密钥注入机制，确保容器根文件系统对宿主机完全不可见且防篡改。该架构在提供虚拟机级别强隔离边界的同时，保持了与 Kubernetes 生态的无缝兼容，有效解决了传统容器在内存暴露和数据隐私方面的安全短板。Kata-3.13 进一步将组件边界与工作流云原生化（如通过 snapshotter 引导镜像获取、KBS/AS/RVPS 分层服务、Pod 注解驱动配置、LUKS 加密盘落盘），在提升安全性的同时改进部署与运维可用性。



基于远程证明的 CSV 虚拟机磁盘加密技术

阿里联合合作伙伴推出的基于远程证明的 CSV 虚拟机磁盘加密技术，旨在实现系统从引导到落盘的全生命周期数据保护。该方案融合 CSV 运行时度量（RTMR）、dm-verity 完整性校验及 LUKS2 加密标准，构建了“度量 - 认证 - 解密”的安全启动闭环：仅在虚拟机通过远程证明且度量值与基准一致时，可信密钥服务（Trustee）才下发解密密钥。通过分层密钥体系与硬件级度量绑定，该技术有效防御了离线篡改、密钥窃取等攻击，确保根文件系统与数据分区在存储态的机密性、完整性及可验证性。



6.7.4 Lua-LSM：内核安全小程序框架

在操作系统安全治理中，LSM 机制提供了强制访问控制能力，但在实际落地过程中，安全策略仍然面临“开发门槛高、策略迭代慢、响应周期长”的共性问题。尤其在云原生、多租和动态业务场景下，攻击面变化快、策略调整频繁，传统静态方式很难兼顾安全强度与交付效率。

Lua-LSM（Lua based Linux Security Module）围绕这一现实需求提出了“内核安全能力 + 脚本化策略”的实践路径。它保留内核侧安全控制的基础能力，同时引入更灵活的策略表达方式，使安全规则能够以更低成本完成开发、验证、发布和回滚。

6.7.4.1 技术方案

Lua-LSM 的核心思路是把 LSM 访问控制逻辑从“重内核开发”转为“策略化开发”。在关键安全检查点，框架将系统上下文交给策略层处理，再将判定结果返回给系统执行链路，实现对访问行为的动态约束。

从能力组织上看，Lua-LSM 将策略模块化为可独立管理的“安全小程序”。每个模块围绕一类目标问题实现策略逻辑，例如文件分类分级、进程行为约束、网络连接控制等。策略层重点关注“该不该放行”，无需深度介入底层实现细节，显著降低了安全能力建设门槛。

在状态管理方面，Lua-LSM 提供了内核对象属性与模块级共享状态能力，用于支撑策略在复杂场景下的上下文判断。

6.7.4.2 核心能力

Lua-LSM 形成了面向安全运营的三类核心能力：

1. 策略快速迭代能力：将安全规则从内核改造流程中解耦出来，缩短从需求提出到策略生效的周期。

2. 细粒度访问控制能力：可围绕进程、文件、网络等对象组合策略条件，实现更贴近业务语义的访问控制。

3. 运行时治理能力：支持策略的灰度发布、快速回滚与持续优化，提升安全运营的响应效率与可维护性。

6.7.4.3 应用场景

基于上述能力，Lua-LSM 适用于以下典型场景：

1. 主机关键资产保护：对敏感文件、核心进程和关键操作建立差异化访问约束。

2. 安全事件快速止血：在风险暴露阶段快速上线临时策略，控制影响面并争取修复窗口。

3. 策略持续运营优化：通过长期运行反馈不断调整策略，提升命中准确率并平衡性能开销。

6.7.4.4 总结

Lua-LSM 并不替代现有安全机制，而是为其提供更灵活的策略工程化能力。它以可编程方式提升了内核安全能力的可落地性和可运营性，在安全强度、迭代效率与运维复杂度之间提供了更均衡的实践方案，适合在龙蜥安全可信体系中作为内核动态策略能力底座持续演进。

6.8 编程语言

6.8.1 C++ 编译器和基础库

6.8.1.1 背景概述

Alibaba Cloud Compiler（ACC）编译器，相比 GCC，或其他 Clang/LLVM 版本在编译、构建速度上有很大的提升;利用 ACC ThinLTO、AutoFDO 和 Bolt 等技术可以在不同程度上优化程序性能。ACC 在支持不同 CPU 架构（X86、AArch64）的基础上，进一步针对倚天 710 芯片进行优化，取得额外性能提升。

我们基于 ACC 编译器开发了一套贴近 C++ 开发者、非常易用和高性能的 C++ 基础库，包括:协程、modules、编译期反射、序列化、 rpc、http、orm 等常用的组件。

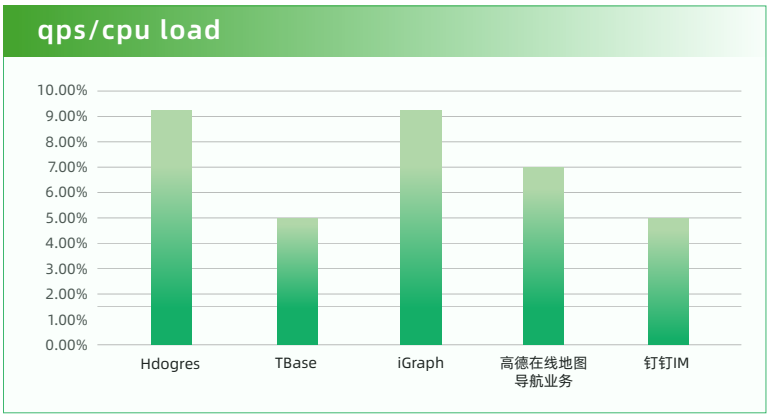
ACC 与 C++ 基础库，为龙蜥社区开发者提供了 C++ 开发的一站式解决方案，快速构建高性能的 C++ 应用:

- 通过编译器切换升级到 ACC，可以在不用大幅修改代码的情况下获得性能提升和编译速度大幅提升;
- 使用 C++ 基础库的协程子库，将一些现存业务的同步任务造成协程异步任务可以获得性能的大幅提升;
- 使用 C++ 基础库的协程子库将异步回调逻辑变成同步逻辑，让异步代码变得简单和易维护;
- 使用 C++ 基础库的 rpc、http、序列化和 ORM 等子库可以快速构建高性能 C++ 应用。

6.8.1.2 技术方案

ACC ThinLTO 性能优化

在一些业务中启用 ThinLTO 优化后，qps 和 cpu workload 提升效果较好。



ACC CoreBolt 性能优化

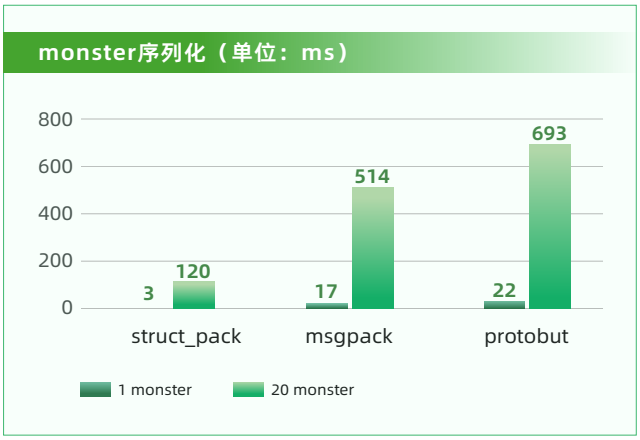
基于倚天 710 的 CoreSight 性能采集方案和 ACC Bolt 的二进制优化解决方案，在多种数据库产品中实现落地，并取得近 20% 的性能提升。

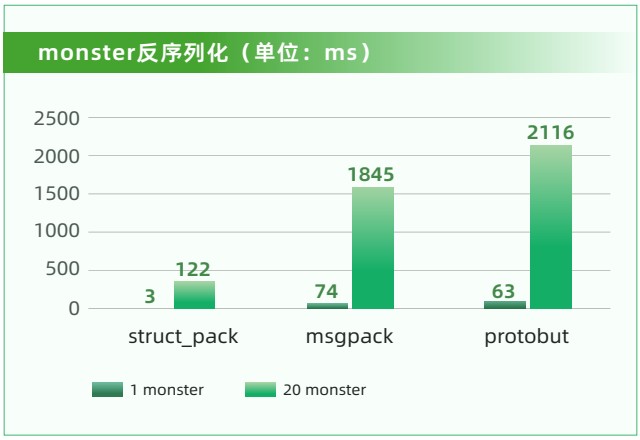
async_simple 协程库

某计算平台和搜索平台将同步改造成协程异步之后，qps 获得了数量级的提升。将异步回调改造成协程异步之后性能获得了 8% 左右的提升。

struct_pack 序列化库

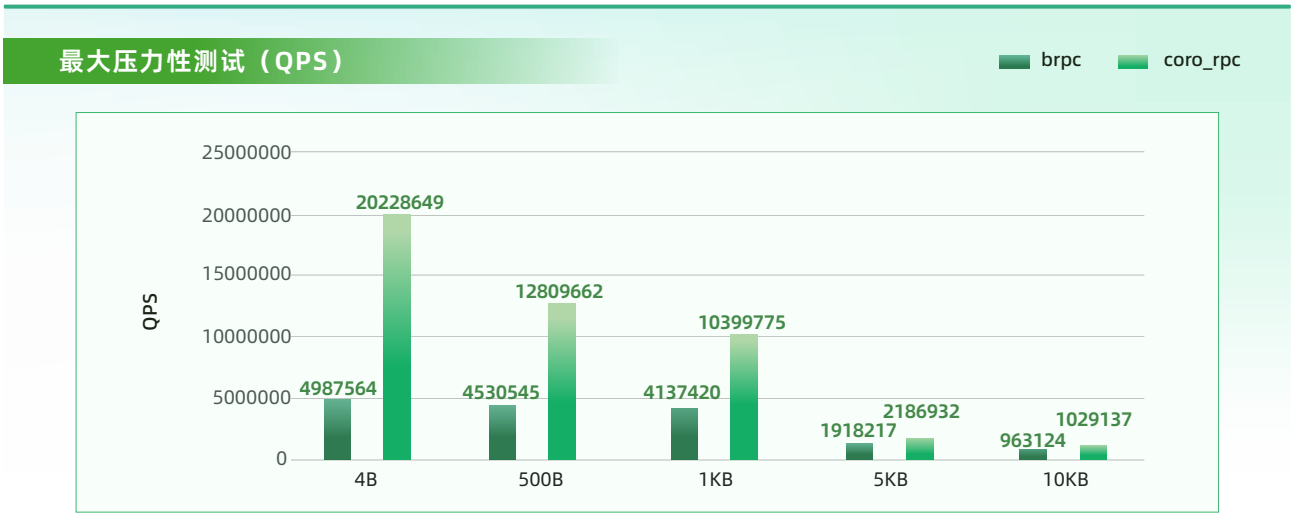
用户只需要一行代码就可以用 struct_pack 完成序列化和反序列化。相比于 protobuf 等开源库需要定义 proto 文件和生成代码的方式，大大简化了使用序列化库的使用方式。同时，相比 protobuf 性能提升 7-20 倍。





coro_rpc 库

coro_rpc 库是基于无栈协程和编译期反射的高性能易用的 rpc 库，几行代码就可以提供一个 rpc 服务，让异步 IO 变得简单，性能比 brpc 高 2-4 倍。



coro_http_client 库

coro_http_client 库是基于 C++20 协程实现的高性能易用的 http client，提供了丰富、易用的 http/https 接口，提供了常用的 get、post、chunked、ranges、multipart 和 websocket 等常用功能，适合用来快速访问 http 服务。在某计算平台中使用 coro_http_client 访问云存储，在一些高并发关键读场景 CPU 占用降低 60%，写场景 CPU 占用降低 35%。

struct_xml 库

struct_xml 是 C++17 实现的，基于编译期反射实现的 xml 序列化/反序列化库，一行代码就可以实现 xml 到对象相互转换，在典型场景下性能比 rapidxml 快 30% 以上。在某计算平台中使用 struct_xml 解析 xml 提升了性能、易用性和安全性。

6.8.2 Dragonwell

6.8.2.1 背景概述

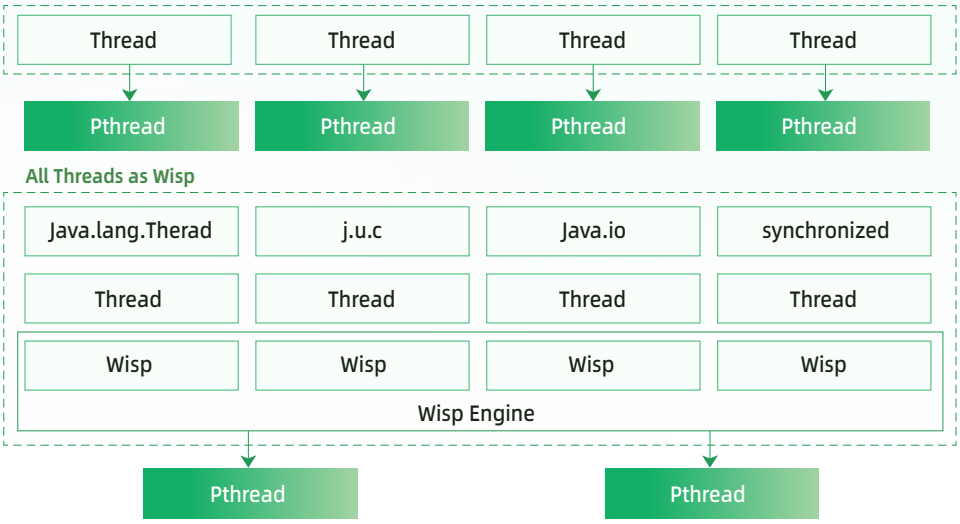
Java 诞生于 20 多年前，拥有大量优秀的企业级框架，践行 OOP 理念，更多体现的是严谨以及在长时间运行条件下的稳定性和高性 能。在如今微服务、云原生大行其道的时代，过于重量的 Java 语言面临着启动速度慢、运行消耗大等诸多挑战。

Dragonwell 是 OpenJDK 的下游发行版本，支持 X64/AArch64 平台，被广泛应用于在线电商、金融、物流等领域，百万量级部署规模。Dragonwell 依托于丰富的阿里巴巴 Java 应用场景的锤炼, 全面拥抱云原生，为 Java 注入新鲜血液。

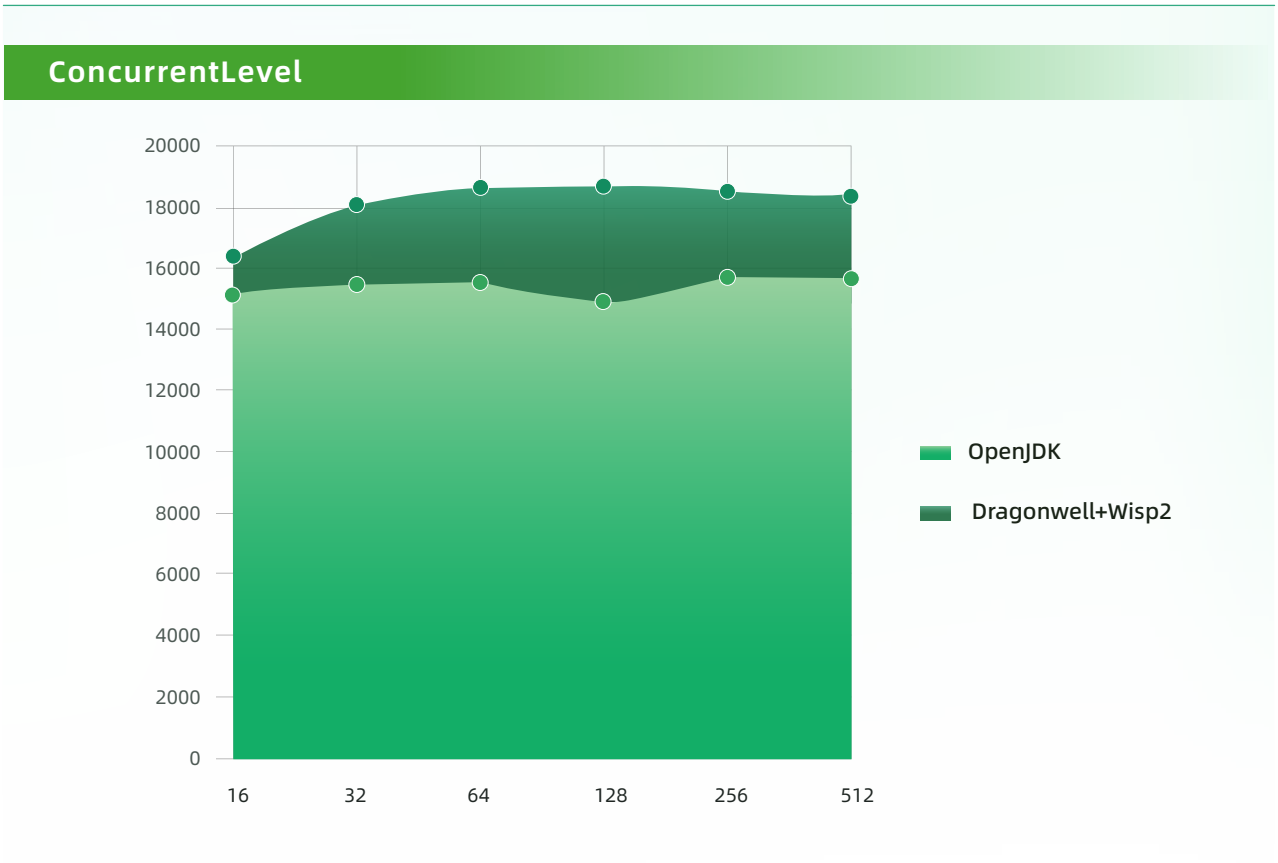
6.8.2.2 Wisp 协程

在分布式微服务应用领域，存在着大量的网络 IO，请求处理线程通常会阻塞等待后台的数据库、缓存、RPC 访问。这种开发模型导 致线程频繁阻塞，大量 CPU 资源浪费在调度和上下文切换上。协程方案可以以阻塞的方式写出异步执行的代码，极大提升密集网络 IO 型应用的性能。

Wisp 在 Dragonwell 上提供了一种用户态的线程实现。开启 Wisp 后，Java 线程不再简单地映射到内核级线程，而是对应到一个协程，JVM 在少量内核线程上调度大量协程执行，以减少内核的调度开销，提高 Java 应用性能。



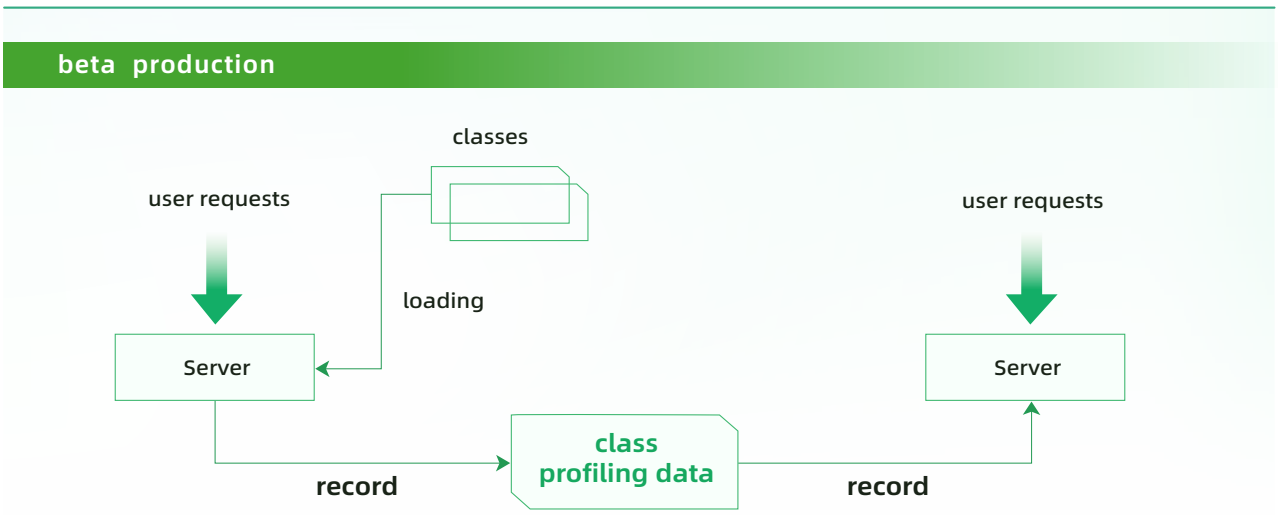
开启 Wisp 后，应用程序无需任何修改就可以获得性能提升。以下是在 Framework Benchmark 的 Spring 实现下，开启协程和关闭协程的性能比较。



6.8.2.3 JWarmup 编译预热

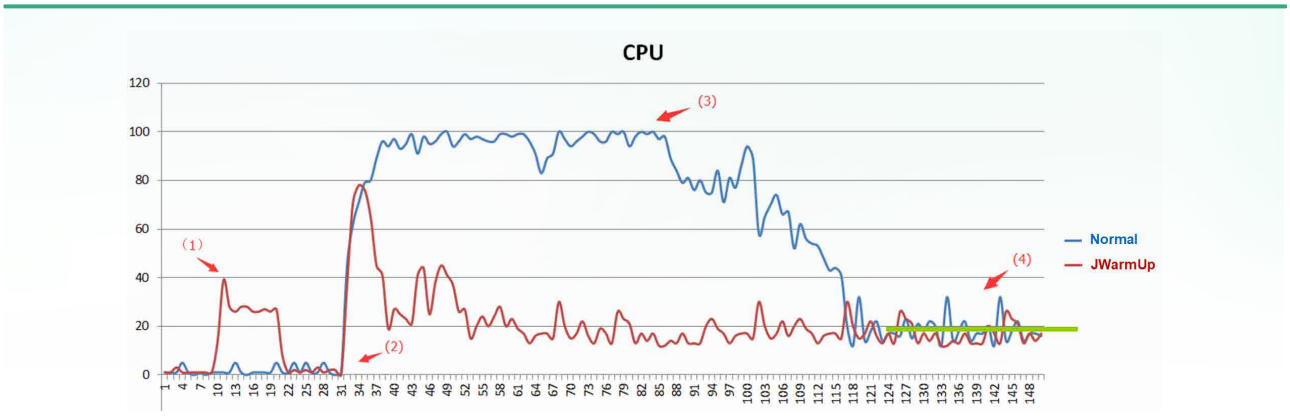
Java 拥有即时编译特性，代码以解释器执行为基础，当方法执行足够多次数成为热方法后会触发 Just In Time（JIT）编译，性能得到数十倍提升。但应用程序有一个预热的过程，通常需要运行一段时间才能达到峰值性能。

JWarmup 以可控的方式来完成预热：根据前一次程序运行的情况，记录下热点方法、类编译顺序等信息， 在应用下一次启动的时候 积极加载相关的类，并积极编译相关的方法，进而应用启动后可以直接运行编译好的 Java 代码（C2 编译）。



上图显示了 JWarmup 典型的用法:

- 在 Beta 灰度环境，进行应用压测，记录（Record）热点方法、类编译顺序等信息。
- 在 Production 环境，使用提前记录的 profiling data 提前编译（Compile）热点方法。



上图显示了生产环境下使用 JWarmup 与关闭J Warmup 的 CPU 曲线对比。可以看到代表 JWarmup 使用的红线 CPU 使用率相当稳定:

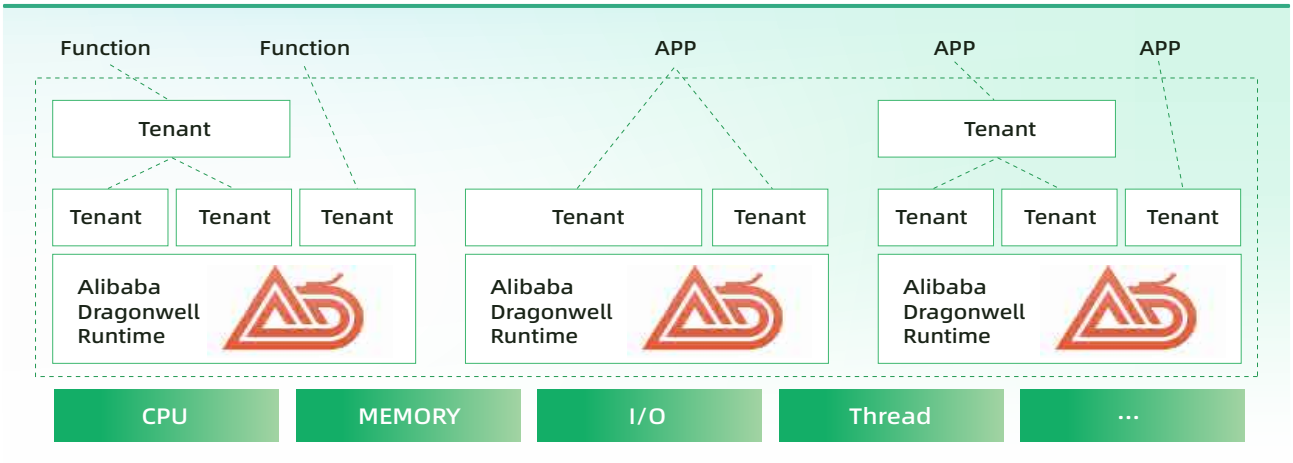
在（1）的时间点过后，JWarmup 提前编译方法，相对于正常蓝线情况，红线使用的 CPU 上升。

在（2）的时间点过后，真实流量进来，因为 JWarmup 情况下方法已经在（1）被提前编译，所有使用的 CPU 要明显低于蓝线。

6.8.2.4 多租户

在有些场景中，Java 应用并非作为单一业务用途的程序存在的，而是作为一个平台存在，在它上面运行着一些更加轻量的应用或函数。在这类平台型 Java 应用上面，一般通过动态类加载机制把遵循一定编程接口规范的，和 Java 字节码技术兼容的软件模块（可以是从 Scala、Kotlin 等语言编译而来）加载到应用进程，并运行其中的业务逻辑，一个典型的例子是 Tomca 的动态加载机制。这类架 构有一个缺点:就是平台型 Java 应用缺乏对轻量应用所使用资源的管控能力。容易出现某个应用占用过多资源——比如 CPU 时间——而 导致其他同进程应用无法响应。JDK 多租户技术的目的就是为平台型 Java 应用提供一个细粒度资源管控的能力。

Dragonwell 多租户技术通过在 JDK 中创建虚拟的进程内容器“租户”，来让 JVM 可以识别出运行时代码所持有的资源组，架构图如下:



在高密度部署场景，多租户技术允许将多个 Java 应用安全部署在同一个 JVM 中，为基于 JVM 技术栈的 PaaS、SaaS、 FaaS 应用平台提供了基于租户粒度的底层资源，包括 CPU，内存等隔离能力。

6.8.2.5 生产就绪 ZGC 及弹性堆

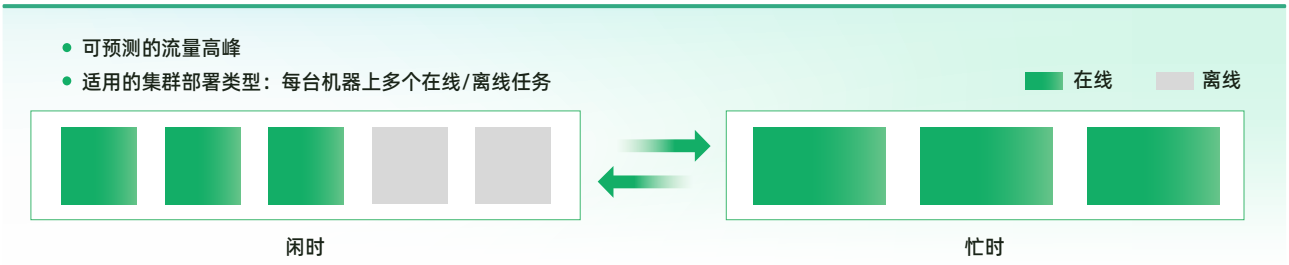
Dragonwell 11 第一个版本应业务和云上客户的需求，默认提供了 Java 11 的实验性 ZGC 特性。随着 ZGC 进入生产实践而逐 步落地，客户在享受响应时间优化的同时，也遭遇到了一些实际问题。因此我们发布了 Dragonwell11.0.11.7 ，其中的 ZGC 特性由 OpenJDK 11 中的实验特性改造为生产就绪（production-ready）特性，同时保证 Dragonwell 11 长期支持的质量稳定性：

- 系统性移植 OpenJDK15 ZGC 代码：OpenJDK15（首个支持生产就绪 ZGC 的 OpenJDK 上游正式版本）的大部分 ZGC 相关代码，这些 代码完善了 ZGC 的功能，支持更多的平台，并且修复了重大 bug。
- ZGC AArch64 平台优化：Dragonwell 向 OpenJDK 社区贡献了 AArch64 平台上 ZGC 的优化方案，减少内存屏障指令，将 Heapothesis benchmark 的 ZGC 吞吐能力提升超过 15%。

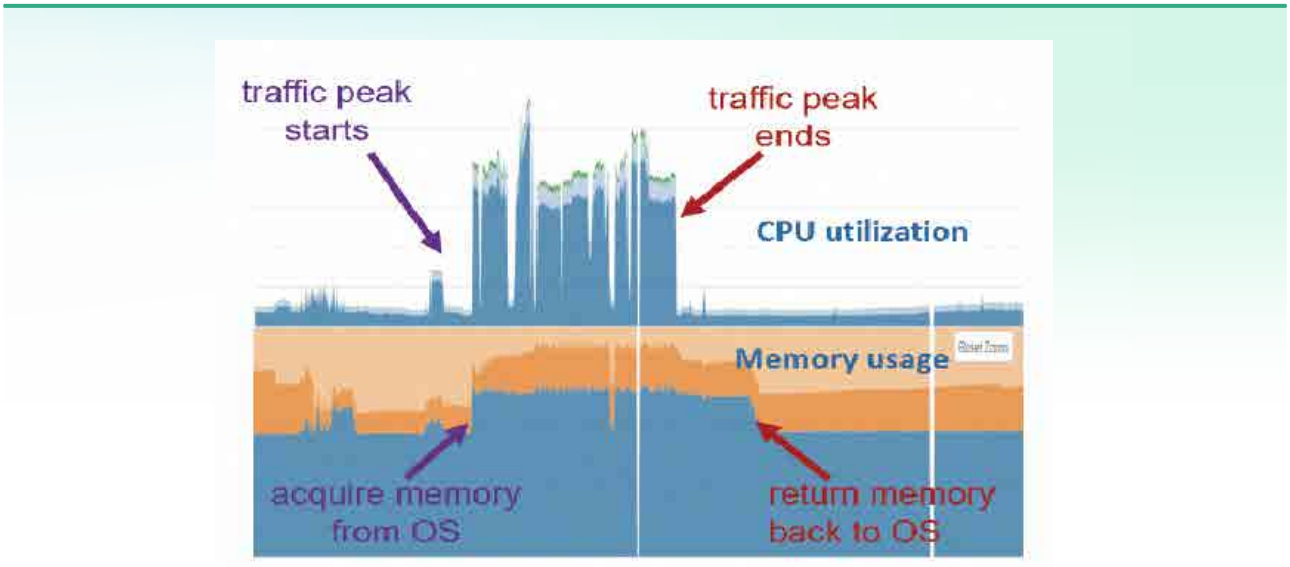
Apache RocketMQ 使用 ZGC 暂停时间从 200ms 降到 10ms 以内。

JVM 为应用程序提供了自动管理堆内存的能力，包括空闲内存在内，即使应用完全空闲，这些内存也无法被操作系统中其他进程使 用。G1ElasticHeap 特性支持在 G1 中动态归还堆物理内存并降低 Java 进程的内存占用。

在多应用混合部署的场景，G1ElasticHeap 空闲内存回收特性可以在应用低负载时降低内存使用，供其他混部的应用使用。下图绿色表示在线任务，灰色表示离线任务。在线服务空闲时通过 G1ElasticHeap 释放空闲内存，以供离线服务使用。



另外一个典型的应用场景是大促，Java 进程内存根据实际服务的压力动态调整。当大促高峰服务繁忙期间，Java 进程会占用比较多 的内存，而在服务空闲时段，Java 进程会及时归还物理内存给操作系统。如下图：



6.9 社区基础设施

6.9.1 T-One：全场景质量协作平台

6.9.1.1 背景概述

T-One（testing in one）是一站式、全场景的质量协作平台，通过它可以解决大型软件的各类测试问题；我们在利用 T-One 解决龙蜥社区测试问题的同时，也通过 T-One 建立了社区的测试标准，另外也在帮助社区的合作伙伴解决他们面临的同类问题。

基于 T-One 建立的社区众测共创模式，解决了开源项目松散的开发模式与商业化产品高质量需求之间巨大的鸿沟；统一平台、统一标准，共同打造开源模式、商业化品质的龙蜥 OS 发行版。

T-One 社区版链接：https://tone.openanolis.cn/

6.9.1.2 技术方案

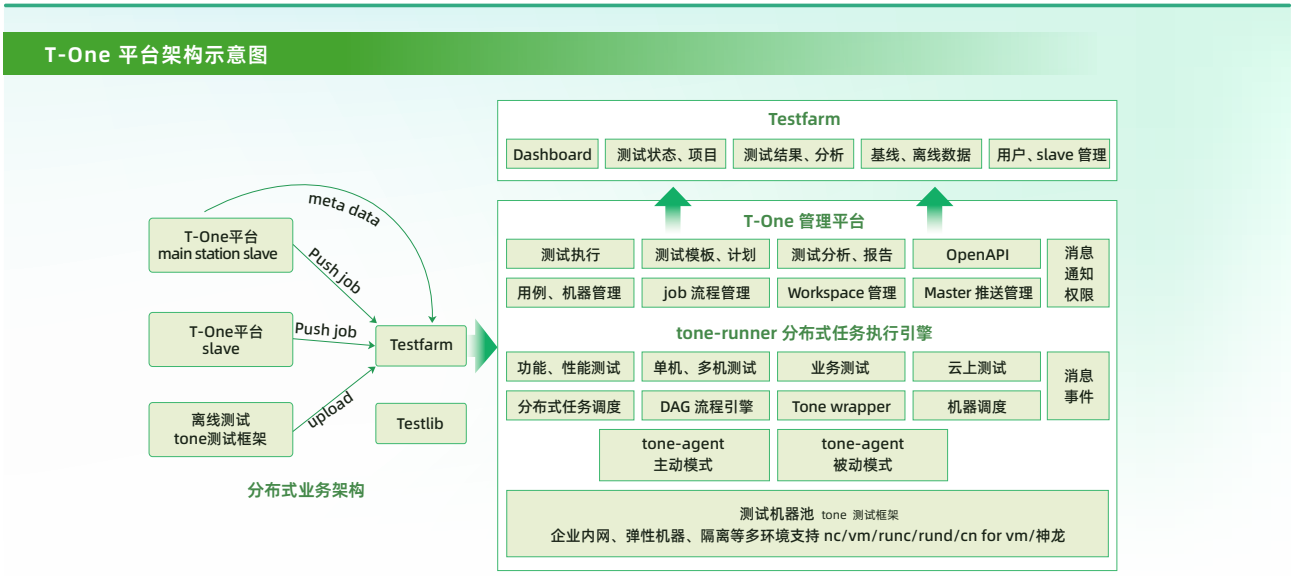
T-One 包括一整套测试解决方案，按照 1+N 模式进行演进，包含若干组件：

1. 一站式自动化测试平台 T-One，是核心的测试管理、自动化测试执行引擎。

2. Testlib 是一个测试管理系统，包括测试用例、测试方案、测试任务等管理能力。

3. Testfarm 是测试数据对外披露系统，测试数据可以自动通过 T-One 上传到 Testfarm 进行披露。为方便社区模式里多家企业的协同测试，T-One 提供分布式业务模式，可以多份部署，把测试任务分发到不同的企业、硬件环境里进行测试，最后再把数据统一上传到 Testfarm 进行管理披露。

T-One 平台架构示意图



T-One 主要有下面三方面的优势：

1. 提供全场景的测试能力

a. 支持多 CPU 混合架构（x86、arm、loongarch64、risc-v）；

b. 支持多操作系统类型（龙蜥、centos、debian、ubuntu、统信、麒麟）；

c. 支持复杂环境测试（企业内网、网络隔离环境、弹性云虚拟机/容器、应用集群及多种混合环境）。

2. 提供一站式的测试支持，打通了从环境部署，测试执行、测试分析、测试计划、测试报告等整个测试流程闭环：

a. 基线跟踪模型：聚合型基线模型、测试指标跟踪模式；

b. 分析及报告：时序分析、对比分析等分析能力；灵活定制测试报告；

c. 可快速搭建 CI 流程；自定义测试计划。

3. 高效的质量协作模式，通过独立租户空间、离线模式和独立部署，充分解决测试协作问题。

T-One 经历多年迭代开发，累计运行时长超过 930 万小时；目前集成了各领域、各类型 120 多种业界主流 benchmark、3000+ 测试套件。

6.9.1.3 应用场景

T-One 支撑了龙蜥社区的所有测试活动，包括版本测试、软件包 CI 测试、镜像 nightly 测试等等，T-One 不仅应用于产品研发过程中的质量保障，还可以作为测评项目的测试平台使用，在相关评测中提升测试效率 30 倍以上。

目前有超过 20 多家厂商设备接入社区 T-One 测试平台，支持了 40 多个项目的质量协同以及数百台测试机器并发执行；在龙蜥社区被包括统信、电子五所、浪潮信息、中科曙光、中兴通讯、新支点等 20 多家合作企业、机构采用。

6.9.2 ABS：一站式构建服务

6.9.2.1 背景概述

ABS（Anolis build service）是一站式的基础构建平台，提供免费、安全、可靠的一键构建能力，以及简单易用的编译构建环境。通过它可以完成 RPM 包、ISO 镜像 和 Docker 容器镜像等构建工作，同时它还提供发行版软件全生命周期管理能力，支撑社区开发者构建和社区产品发行构建，方便开源软件包引入，扩大龙蜥社区的产品生态。

ABS 平台链接：https://abs.openanolis.cn/all_project

容器镜像制作平台：https://cr.openanolis.cn/

Pypi 镜像仓：https://pypi.aliyun-inc.com/mirror

6.9.2.2 技术方案

平台能力

1. ABS 提供的主要功能有：

a. ABS 提供了软件包多架构的构建能力，支持多 CPU 架构（x86、arm、loongarch64、risc-V）。

b. ABS 提供生产构建及测试构建能力，方便开源软件包的引入，扩大产品生态。

c. ABS 提供一键 Anolis OS ISO 镜像 rebrand 功能，快速定制下游衍生版。

d. ABS 与 容器镜像制作平台一起提供了 Docker 容器镜像及分发的能力，支持 AI 组件全流程自动化。

e. ABS 提供 Python wheel 软件包的构建和分发能力，支持 AI 组件全流程自动化。

f. ABS 提供 OOT、内核源码等构建能力，方便开发者针对 OOT、内核项目的开发和测试。

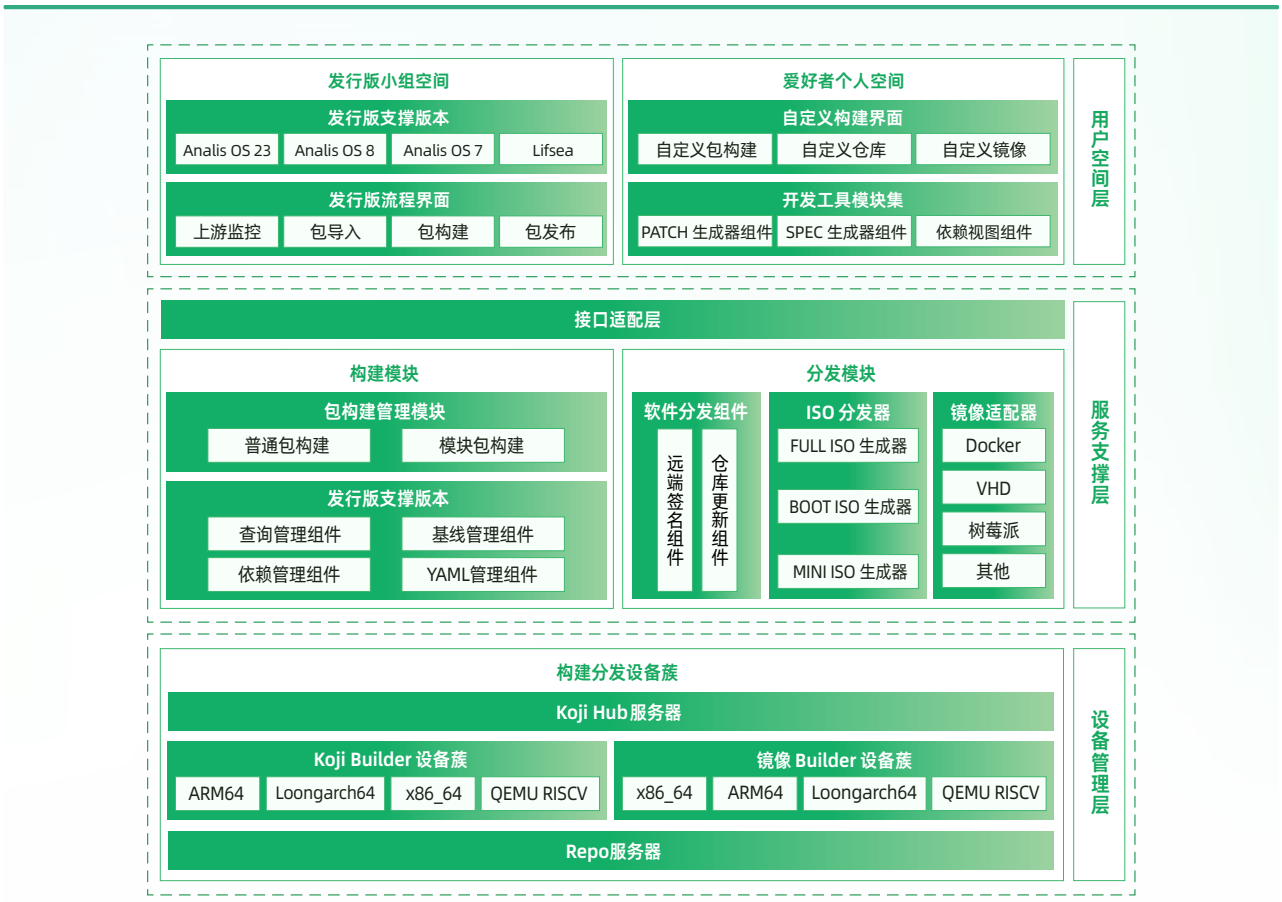
g. ABS 提供全流程的上游软件包跟踪及更新能力，保障社区软件包供应链能力。

系统架构

1. 用户空间层：为社区爱好者、合作伙伴、发行版团队提供产品构建及发行的支撑，并提供一系列开发工具集提高软件包研发效率；

2. 服务支撑层：主要包括构建支撑模块和分发模块，提供各类产品形态的构建服务，同时向上提供接口服务；

3. 设备管理层：我们 Docker 镜像构建服务基于交叉编译，其他构建服务基于 koji 开源系统进行二次开发，向上提供多架构的构建能力。



6.9.2.3 应用场景

ABS 平台支持的使用场景有：

1. 软件包研发过程的测试和构建。

2. 下游厂商可以通过 ABS 构建操作系统衍生发行版。

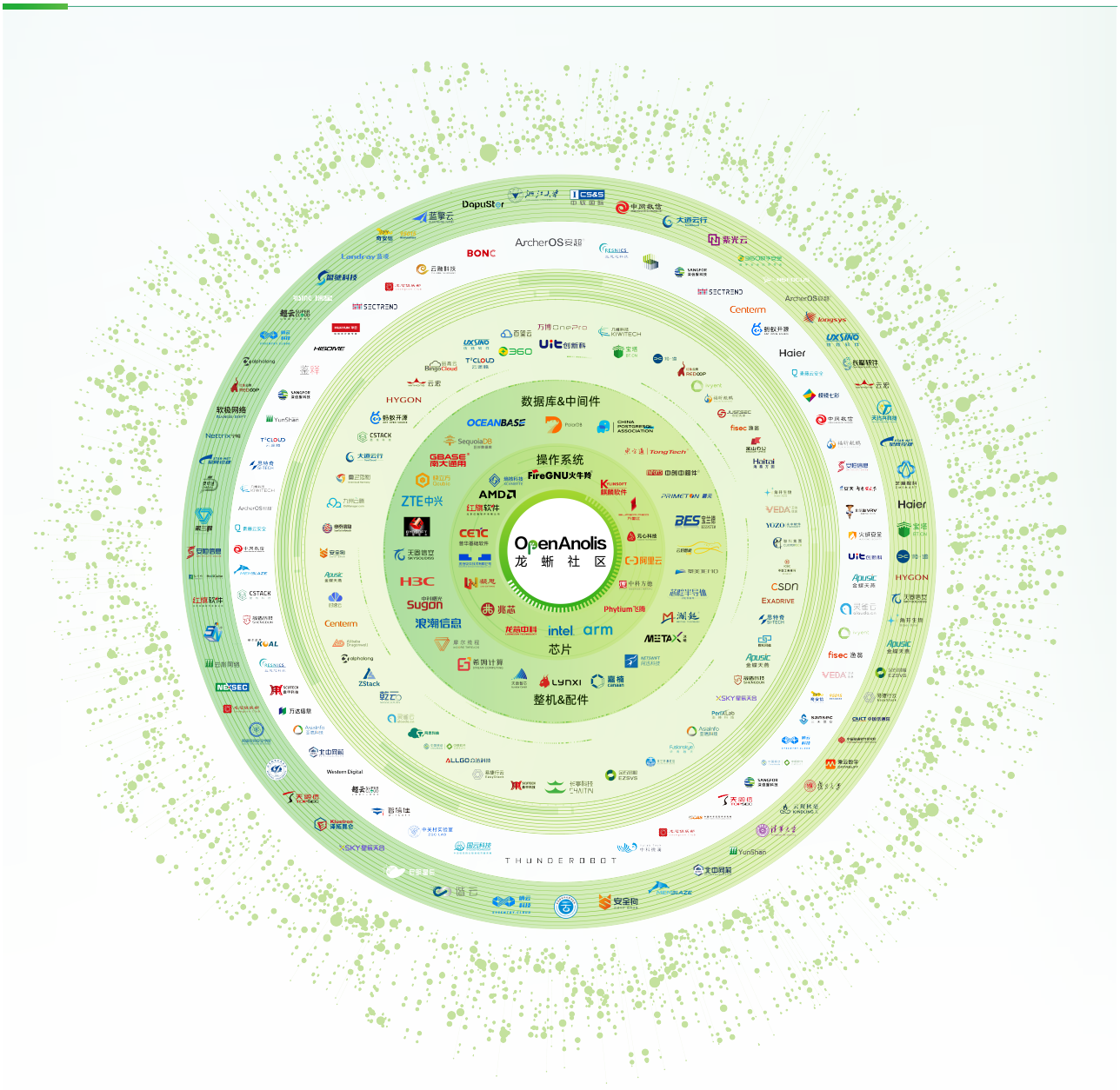
3. 社区开发者可以通过 ABS 进行 AI 容器镜像的制作工作。

4. 发行版生命周期维护。

过去一年，龙蜥社区发布了Anolis OS 8、Anolis OS 23 等重要版本，这些都是通过 ABS 构建系统支持的。ABS 上线以来，构建的软件包数量超过 40000 个，软件包构建次数超过 63000 次，镜像构建次数超过 3600 次，创建项目总量接近 3000 个。平台除服务社区外被社区爱好者以及多家企业、机构使用。

7. 社区产品与生态

7.1 龙蜥生态



2025 年， Anolis OS 23/8 与联盟厂商、GPU 厂商等超过 100 家企业完成产品兼容测试认证，涵盖数据库、中间件、芯片、整机、云平台、行业应用等 700 + 款商业产品。

7.2 龙腾计划 3.0——AI 引擎生态加速合作计划

“龙腾计划”是龙蜥社区推进生态合作的核心引擎。自 2021 年启动以来，该计划迅速凝聚产业力量，不到一年便吸引了超 250 家合作伙伴加入。2022 年云栖大会上，计划迭代升级为「生态发展计划」，完成了 50 家核心伙伴在技术、产品及商业层面的深度协同。截至目前，龙蜥社区已汇聚超 1000 家合作伙伴，共同构建繁荣开放的智算时代开源生态。

在 2025 操作系统大会上，中兴通讯副总裁、龙蜥社区副理事长刘东和中科方德高级副总裁龚文等社区代表共同站台发布了龙腾计划 3.0——**AI 引擎生态加速合作计划**，标志着龙蜥社区在推动操作系统与 AI 技术创新融合、以及利用 AI 驱动上下游产业合作变革方面迈出了重要的一步。该计划重点围绕多元算力芯片架构和 AI 基础设施推进合作，进一步加深与国产 CPU 厂商的深度合作、加大 RISC-V 生态及其上下游标准建设力度、加速与 GPU 厂商和模型厂商的合作落地，形成一个更加紧密且高效的产业合作链条。

7.3 三大联盟：智算联盟（OICA）、安全联盟（OASA）、系统运维联盟（SOMA）

龙蜥社区智算联盟（OpenAnolis Intelligent Computing Alliance，OICA）由阿里云、浪潮信息、中科方德、中兴通讯等龙蜥社区理事单位联合 Imagination 以及清华大学、上海交大、澳门大学等高校和 FlagOS 社区等多家行业领军企业及机构共同发起，于 2025 年 8 月正式成立，当前成员单位 23 家。智算联盟主要围绕智算技术共建关键的技术方案、推动技术标准化和通用化、促进技术开源以及生态和应用案例推广等，联盟设立了测试和兼容性、RAS、算子优化等 9 大 TG 技术组，**在测试标准、系统可靠性、性能分析、模型与算子优化、内存架构**等核心领域开展深度协同，并在 **AI 生态建设**中持续投入，积极贡献**上游社区**，推动国内外**头部 GPU 厂商**陆续完成与龙蜥的适配。2025 年通过举办龙蜥大会智算分论坛、龙蜥×SGLang MeetUp 等高质量技术活动，提升在国产智能算力领域的行业声量。

2025年8月，智算联盟成立

2026年1月，智算技术沙龙-圆桌讨论

2023 年 7 月，由启明星辰、绿盟、360、阿里云、统信软件、浪潮信息、中兴通讯|中兴新支点、Intel、中科院软件所等 23 家单位共同发起**龙蜥社区安全联盟**（OpenAnolis Security Alliance，OASA），该联盟主要通过安全技术合作交流，助力信息安全领域创新成果的落地。目前，联盟共有 36 家成员单位参与共建。2025 年，安全联盟新增国际安全厂商 Tenable、朗空量子、PQCX 实验室等成员单位。同年，联盟官网上线了漏洞挖掘激励计划，并积极推动安恒信息、软安科技等成员单位与 **Anolis OS 及其衍生版本完成产品适配**。基于联盟合作，阿里云、浪潮信息、江南天安、海光信息等联合发布 3 个解决方案，并获得 **2025 龙蜥年度“最佳联合解决方案”奖**。在标准化与合规建设方面，龙蜥社区先后通过了 **OpenChain ISO/IEC 5230 与 ISO/IEC 18974** 两项国际标准认证。这标志着社区在**开源合规管理能力、软件供应链安全及 CVE 管理能力**上，均获得了国际权威机构的高度认可，也进一步夯实了社区的安全技术底座。此外，联盟重点举办了 4 场专题活动，有效撬动了 26 家成员单位的资源投入，吸引了超过 200 名观众现场参会。

安全联盟年度工作会议

OpenChain 官网正式宣布

2023 年 11 月，由龙蜥社区系统运维 SIG 联合各大运维商、平台厂商及科研院校共同发起成立了**系统运维联盟**（System Operation & Maintenance Alliance，SOMA），旨在通过**构建故障演练平台与运维产品力评测系统**，搭建起厂商与客户之间的沟通桥梁；同时，依托运维技术的深度交流与合作，推动整个运维产业的繁荣发展。2025 年，系统运维联盟在标准引领方面取得了重要突破。针对异构环境下的数据兼容难题，联盟完成了**《开源数据采集组件技术要求》**团体标准的修改并提交。同时，《运维工具评估方法》行业标准已正式立项并通过工信部 CCSA 答辩，标志着联盟提出的评估体系已上升为国家行业共识，成功掌握了技术评价的话语权。此外，联盟成员单位阿里云与云杉世纪共同产出的联合解决方案，也凭借卓越的技术实力斩获了“**龙蜥社区最佳联合解决方案奖**”。

运维联盟 MeetUp

7.4 解决方案

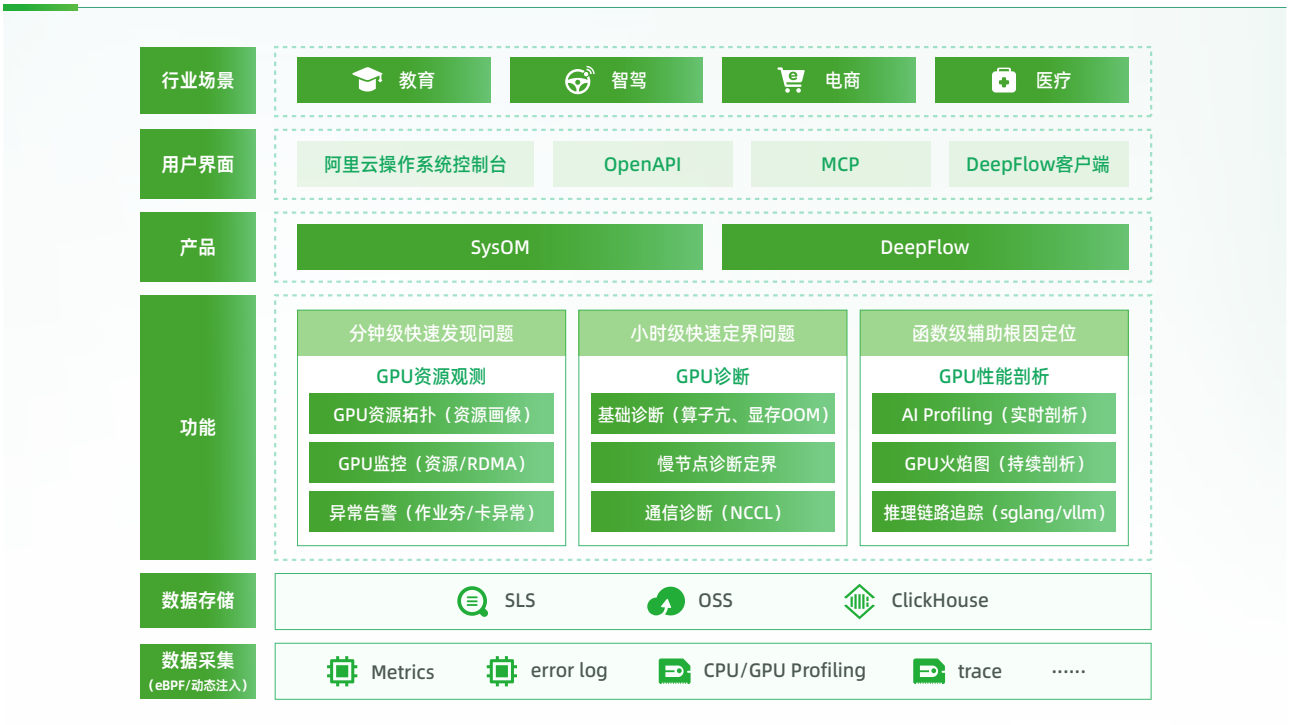
“最佳联合解决方案”是在实际业务中使用 Anolis OS 或衍生版的企业用户，内容体现业务应用的解决方案或技术创新方案。自 2023 年龙蜥社区安全联盟和系统运维联盟成立以来，不仅吸引安全、运维方面的头部厂商的加入，更在“龙腾计划 2.0”的有力推动下，持续输出高价值、可落地的联合解决方案，持续推动产业生态高质量发展。2025 年，龙蜥社区联合解决方案成果显著，在后量子安全、AI 基础设施、KMS 多加密机等领域产出多个最佳联合解决方案，部分优秀方案示例：

● 朗空量子护盾系统与龙蜥操作系统（Anolis OS）的融合方案

本方案由朗空量子牵头，联合 PQC-X 及再造云人共同构建。方案核心为朗空自主研发的“量子护盾系统”与龙蜥操作系统（Anolis OS）的深度融合。其中，朗空量子负责核心系统研发，PQC-X 实验室主导多领域应用研究，再造云提供算力底座与市场赋能。该融合方案成功攻克了在 Anolis OS 环境下，为本地化部署的 AI 大模型（如 Qwen、DeepSeek）提供全链路后量子安全防护的技术难题，实现了对 AI 应用通信与数据存储的全面量子安全防御。

● AI 基础设施可观测解决方案

针对 AI 基础设施在异构环境下面临的可观测性挑战，阿里云、云杉世纪基于龙蜥社区运维联盟联合推出了基于 eBPF 技术的 AI 全栈可观测方案。该方案有效解决了异构场景下的数据指标关联难题，具备全局维度的串联分析能力，能够精准定位 CPU 与 GPU 的性能瓶颈。目前，该方案已适用于银行、电信、教育等多个业务领域，并已在银行和电信等行业成功落地应用。



● KMS 多加密机聚合管理解决方案

在密钥管理日益复杂与安全要求提升的背景下，龙蜥社区安全联盟成员浪潮信息、三未信安及江南天安联合推出《KMS 多加密机聚合管理解决方案》。该方案融合三方优势，依托三未信安与江南天安的高性能 HSM 提供顶级密码运算能力，结合浪潮 KMS 的智能调度以应对高并发需求。最终实现通过浪潮 KMS 单一平台统一管理分散的 HSM 资源与密钥，显著简化运维操作。

7.5 用户案例

2025 年度龙蜥社区发布了《龙蜥操作系统生态用户实践精选V3》，收录包括龙蜥社区版 Anolis OS 及衍生版 Alibaba Cloud Linux、浪潮信息云峦服务器操作系统 KeyarchOS、统信服务器操作系统 V20 等案例，涉及核电、汽车、电力、电信等重点行业。「龙蜥案例」已在公众号（OpenAnolis 龙蜥）发布了系列文章，为大家带来不同行业的标杆实践样板。部分优秀案例如下：

● 龙蜥社区版 Anolis OS：核电运行研究（上海）有限公司

项目概况

中核集团作为中国核工业的领军企业，肩负保障国家能源安全与战略科技发展的重要使命。而核电运行研究(上海)有限公司定位为中核集团核电安全稳定运行降本增效创新主体。2022 年，随着 CentOS 停服，集团亟需构建安全可控的国产化数字底座，以应对国际技术环境的不确定性。在此背景下，中核集团与阿里云龙蜥社区达成合作，全面采用龙蜥操作系统（Anolis OS）作为核心基础设施平台。截至 2024 年，龙蜥操作系统已覆盖集团 60 余家成员单位，支撑超 1000 台服务器稳定运行，涵盖核能生产、供应链管理、科研仿真等关键场景，成为核工业领域首个实现全栈国产化操作系统替代的标杆案例，为国家关键行业自主可控转型树立典范。

项目挑战

- 严苛的安全合规要求核工业涉及国家安全，需满足等保三级及行业特殊安全标准，系统需抵御 APT 攻击、零日漏洞等高级威胁，同时实现漏洞分钟级响应与修复，传统开源操作系统难以满足动态安全需求。
- 复杂生态适配与稳定性保障业务系统涵盖数百个自研及第三方应用，涉及数千个软件包、内核模块及定制化 Lib 库，迁移需确保从底层内核到应用层的全栈兼容性，且需零业务中断，技术复杂度极高。
- 规模化迁移的平滑性与成本控制存量 CentOS 服务器规模庞大，涉及异构硬件及多版本系统，需设计无损迁移方案，避免业务停机风险，同时降低改造与运维成本。

项目方案

中核集团联合阿里云龙蜥社区及安全服务团队，打造“全生命周期国产化替代解决方案”，分三阶段实现安全、适配与迁移的闭环：

- **安全加固：构建全栈式可信防御体系**
  - 研发级安全嵌入：在龙蜥操作系统研发周期中集成安全左移机制，覆盖代码审计、漏洞扫描、供应链签名校验等环节，确保系统内生安全。
  - 动态防护与响应：部署阿里云安全中心，实现漏洞分钟级检测、热补丁自动修复及攻击溯源，结合龙蜥社区专属工单支持，重大漏洞修复时效提升 50%。
- **生态适配：智能迁移工具链与全栈验证**
  - 预迁移诊断：基于龙蜥迁移助手（Anolis Migration Toolkit），自动化扫描存量系统，生成兼容性评估报告，精准识别软件包冲突、路径差异等风险点。
  - 分层适配验证：联合技术团队开展“内核-系统层-应用层”三级适配测试，针对性优化内核模块兼容性、文件系统接口及依赖库版本，适配成功率超 99.8%。
- **平滑迁移：零感知切换与全链路保障**
  - 双轨并行与灰度发布：采用混合云架构实现 CentOS 与龙蜥操作系统双环境并行，通过流量灰度逐步切换核心业务，单次迁移业务中断时长 <3 分钟。
  - 一键回滚与智能运维：集成阿里云运维编排服务（OOS），支持迁移异常秒级回滚，并构建 AI 驱动的智能监控体系，实现故障自愈与性能调优。

项目成果

- 国产化替代全面落地完成 60 余家单位、1000+ 台服务器迁移，龙蜥操作系统覆盖率超 50%，业务连续性达 99.9%，成为核能行业规模最大的国产操作系统替代案例。
- 安全能力跨越式提升漏洞从发现到修复提速 50%，高危漏洞修复率 100%，抵御外部攻击次数下降 80%，通过等保三级认证及核工业专项安全审计。
- 经济效益与行业引领
  - 迁移成本降低 40%，年运维投入减少 500 万元；
  - 系统性能提升 20%，能源生产调度效率优化 15%；
  - 获评“中央企业数字化转型典型案例”，推动龙蜥操作系统在电力、军工等 10+ 行业复制推广，国产化生态建设提速 3 倍。

中核集团与龙蜥社区的深度合作，不仅实现了核工业核心系统的安全可控，更以“技术+生态”双轮驱动，打造了国产操作系统规模化替代的“中核范式”。该项目彰显了龙蜥操作系统在复杂场景下的技术领先性与服务支撑力，为核能行业提供了可复制的全栈国产化转型蓝图。

● 龙蜥社区衍生版 Alibaba Cloud Linux：浙江极氪智能科技有限公司

项目概况

极氪将云计算作为数字化发展的基石，成功赋能业务实现跨越式增长。依托云平台的极致弹性能力，实现了资源的智能调度与无缝扩展，保障了业务高峰期的高可用与稳定性；通过构建高可用架构，持续为用户提供可靠、流畅的服务体验，为此，极氪采用阿里云自研的 Alibaba Cloud Linux 操作系统，充分发挥其深度适配云环境、内核级性能优化与长期稳定支持的优势在日常运维方面，阿里云提供的一站式工具链有效支撑了多数场景需求。然而，在部分高性能、高敏感的核心业务场景中，仍需依赖操作系统级控制台进行底层系统运维分析。特别是在应对偶发性设备 Load 飙升等问题时，需通过精细化的系统层监控取样、根因定位与优化实施，才能满足极致稳定性要求。

项目挑战

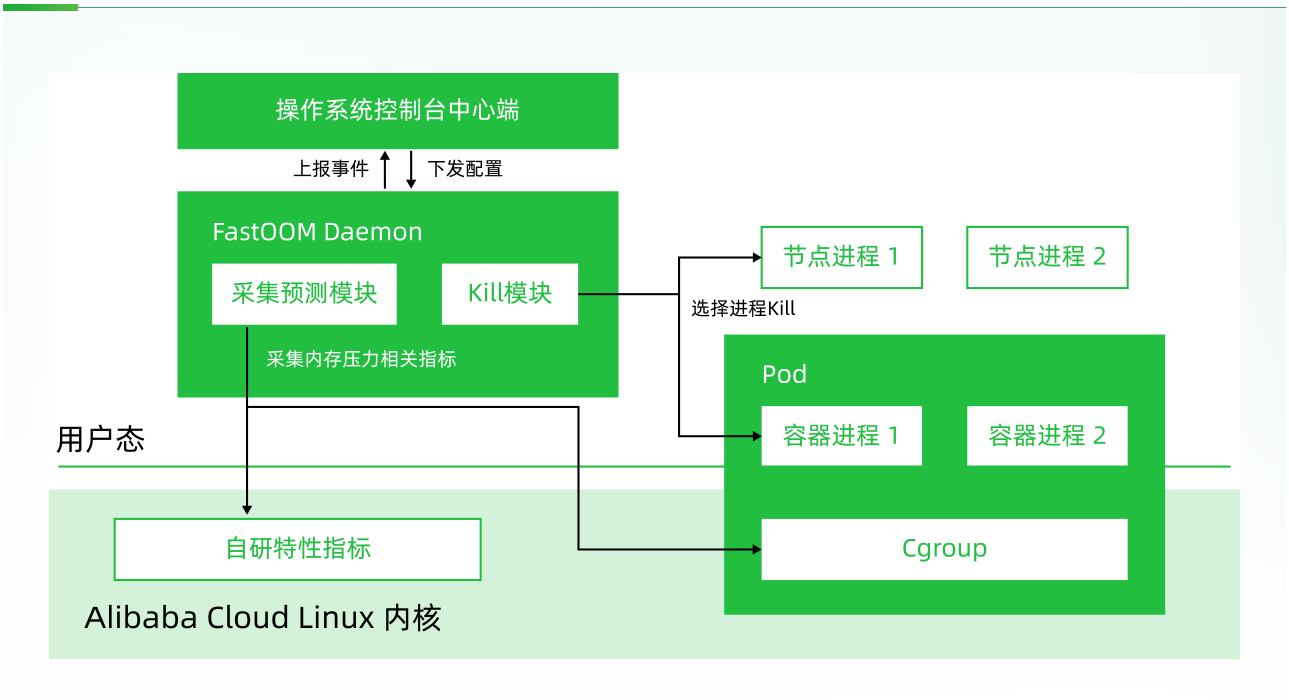
尽管极氪已在接口、日志、链路及云组件层面建立了较为完善的业务健康监控体系，实现了对上层服务状态的全面覆盖，但在面对底层系统级异常时仍面临显著挑战：

- 现有监控体系难以深入操作系统内核层面，缺乏对性能抖动、内存回收阻塞等底层问题的根因分析能力；
- 部分异常（如秒级抖动、瞬时夯机）具有偶发性、无固定复现路径与低频特性，常规监控手段无法有效捕获与回溯；
- 缺少常态化、轻量级的系统级观测工具，导致问题定位周期长、分析成本高，影响整体故障响应效率。

项目方案

为应对极氪在云环境下面临的偶发性性能抖动与内存超载导致的系统夯机问题，阿里云基础软件团队在阿里云操作系统控制台上推出了专项解决方案：

- 偶发性性能抖动治理：研发并部署“进程热点追踪”功能，支持在生产环境中常态化运行，具备毫秒级采样能力，可精准捕捉秒级内的性能波动事件。该功能覆盖由版本变更、环境差异、硬件性能偏差等引发的应用层与系统层性能异常，实现从现象到根因的快速闭环。
- 内存超载夯机防控：针对因内存超卖引发的系统长时间夯机问题，引入 Alibaba Cloud Linux 内核自研参数调优机制，并结合用户态 FastOOM 技术，提前识别内存压力、主动触发资源释放，避免系统陷入不可用状态，显著缩短夯机恢复时间，提升整体服务韧性。



项目成果

借助操作系统控制台的纳管能力，内存超卖引发的夯机问题得到有效治理，系统在高负载下的稳定性大幅提升；同时，进程热点追踪功能已常态化运行，对偶发抖动类问题形成持续监控与兜底分析能力，显著增强了故障预防与快速响应机制。

● 龙蜥社区衍生版 KeyarchOS：国网冀北电力有限公司智能配电网中心

项目概况

配电网是电力系统中连接输电网络与用户侧的关键环节，核心功能是将高压输电线路输送的电能，通过降压、分配、控制与保护，安全、可靠地送达工业、商业、居民等终端用户，是保障社会生产生活用电的“最后一公里”网络。传统配电网运营依赖人工经验与分散系统，数据未被有效利用，导致多环节效率低、风险高。

项目挑战

业务层面：需处理海量终端的实时数据、非结构化数据及历史归档数据，同时满足新能源调控、故障定位的高实时性要求，传统架构难以支撑；国产化层面：需实现硬件-操作系统-业务软件全面国产化，但面临多架构硬件适配、业务软件协同、业务稳定性等多方面挑战。

项目方案

提供浪潮信息 KeyarchOS 操作系统 +Insight HD 大数据平台，KeyarchOS 可适配多类架构具备业内领先的软硬件兼容性，内置安全模块（符合等保三级标准），通过电科院四级检测满足电力系统业务要求，Insight HD 大数据平台保障数据传输高效稳定；支持全协议数据接入，通过 AI 技术治理数据，可弹性调度算力，提升资源利用效率。

项目成果

实现全链路国产化兼容，提升硬件资源利用率，大幅降低数据接入延迟，显著加快核心业务响应速度；有效降低运维成本，提升新能源并网稳定性，提高用户满意度；为配电网从传统人工运营向智能化、精细化运营转型提供助力，也为未来智能配电网建设提供了核心技术基座。

● 龙蜥社区衍生版统信 UOS V20：中国联通软件研究院

项目概况

中国联通软件研究院成立于 2015 年，是中国联通集团 IT 数字化研发的核心支撑单位，长期负责集团集约化系统的研发与运营工作。近年来，针对 CentOS 停服等行业事件带来的供应链安全挑战，软件研究院成立操作系统停服应对专项工作组，重点开展操作系统适配、迁移及优化等核心技术攻关。同时，积极拓展对外合作，深度参与操作系统开源社区建设，不仅正式加入龙蜥社区，还成为该社区首批理事会成员单位和首批 SOMA 联盟成员单位。自 2021 年起，联通软件研究院率先开展信创操作系统在通信行业的应用试点，经过多年实践，已成功构建覆盖技术研发、迁移实施和运维保障的全周期成熟体系。目前，中国联通软件研究院正持续发挥技术优势，助力集团内部各分子公司有序推进服务器操作系统迁移工作，为全集团数字化转型筑牢安全底座。

项目挑战

中国联通集团及各分子公司在推进服务器操作系统迁移过程中，主要面临以下三方面挑战：

- 一、资源调配压力大，迁移实施空间受限
  集团业务系统普遍规模庞大、架构复杂，资源调度灵活性较低。受历史资源规划和成本控制限制，可选迁移手段有限，导致迁移周期延长、实施难度加大。
- 二、技术能力存在短板，自主支撑体系需完善
  研发与运维团队对操作系统底层技术掌握不足，传统运维模式过度依赖厂商支持，自主研发能力较弱。同时，跨团队协作机制尚未完全打通，迁移方案设计与实际环境适配存在偏差。
- 三、技术组件迁移复杂度高，关键场景适配难度大
  迁移涉及云计算、数据库、大数据等 100 余种技术组件，对数据迁移的速度和稳定性要求极高。从基础软件到自研应用，都可能因接口协议或运行环境差异出现功能异常。例如，部分计算密集型业务（如计费出账）因新操作系统调度算法适配不足，处理延迟上升约 15%-20%。

项目方案

中国联通软件研究院操作系统迁移团队采用龙蜥操作系统衍生版——统信服务器操作系统 V20，创新性地运用"嵌入式支撑"模式完成系统迁移，具体实施策略如下：

- 一、自主研发"候鸟"迁移工具，攻克核心迁移难题

  针对"冗余资源有限、变更代价高、OS 核心能力弱"三大挑战，团队自主研发"候鸟"国产操作系统迁移工具，以"原地迁移"为核心方案，开创"低影响、高效率"的迁移路径：

  1. 固化典型迁移场景与标准化流程，支持批量迁移执行，显著降低操作门槛、实施难度和迁移时间
  2. 基于业务类型实施分类处理，对有状态应用和无状态应用分别采用差异化迁移策略，充分发挥云平台调度能力
  3. 根据组件特性（主备、分布式等）制定定制化迁移方案，通过业务隔离实现操作系统替换，确保业务低感知

- 二、构建全链路保障机制，确保系统稳定运行

  针对兼容性与性能风险，团队建立"测试-调优-监控"全流程保障体系：

  1. 整合龙蜥社区系统调优工具 keentune
  2. 实施内核参数优化
  3. 运用 Perf 性能分析技术 通过实时监测系统稳定性与性能趋势，确保关键性能参数稳定，保障业务在新环境下的平稳运行

- 三、实施双重备份策略，保障数据安全

  1. 应用程序与业务数据备份：

     • 共享存储：执行断连、停止数据服务等标准化流程

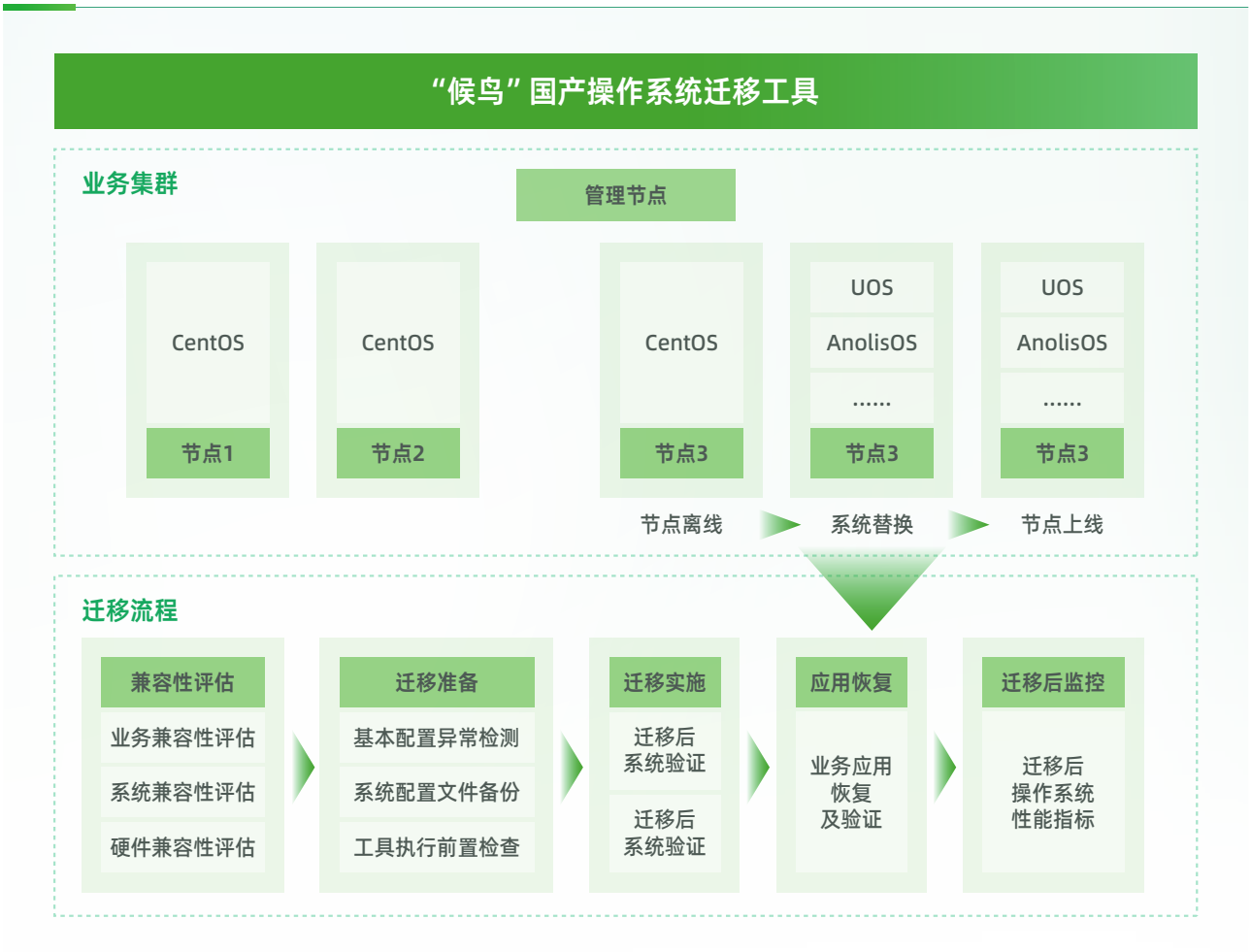
     • 本地存储：采用专业备份软件确保数据准确性

  2. 操作系统备份：

     • 使用 ReaR 备份工具将系统及配置备份至 NFS 主机

     • 通过同配置测试机验证备份可用性，确保系统可成功恢复

     这一系列创新举措有效解决了操作系统迁移过程中的技术难题，为中国联通数字化转型提供了坚实保障。



项目成果

该方案在集团内部各业务系统及客户试点中均展现出显著效能：

- 在迁移效率方面，单台服务器最低迁移时间可降低至 10 分钟左右；
- 在兼容性与性能保障层面，整体兼容性通过率超 98%，有效解决了常见业务场景下的软件适配难题。
- 自 2021 年启动规模化迁移以来，历时 4 年累计已经支撑联通集团完成 1.6 万余套的 CentOS/RHEL 迁移，迁移成功率 100%。各业务系统运行稳定，功能、性能满足业务需求。

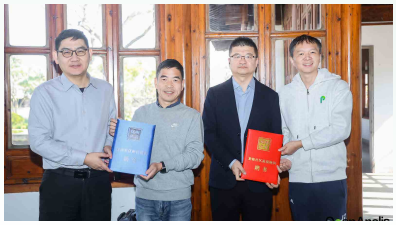
扫码阅读龙蜥案例集

8. 社区风采

8.1 龙蜥社区治理

● 理事大会

龙蜥社区理事会指导社区发展方向，制定长期发展规划和实施指导意见。截至目前，龙蜥社区已圆满召开七届理事大会，其中第七届理事大会于 2026 年 1 月在杭州成功召开，社区的理事和顾问团等代表共计 35 位嘉宾出席。会上，**浙江省经信局相关领导和龙蜥社区首届高级顾问、中国工程院院士陈纯**特别肯定了龙蜥的突出成绩及过去五年的发展成果，尤其是在云计算和 AI 基础设施方面的布局已见成效。第二届龙蜥社区顾问团正式成立（官网组织架构已更新），龙蜥社区理事长马涛和副理事长张东为现场顾问颁发聘书；新晋副理事长单位海光信息和理事单位 AMD 的代表在会上发言。同时，本次会上全票通过龙蜥社区发展倡议。

● 龙蜥社区顾问团

顾问团是龙蜥社区的关键智囊团，为龙蜥社区的发展提供前瞻性指导，对龙蜥社区的长久发展提出实质性改进和提升意见，是龙蜥社区重要的组织。第二届高级顾问团汇聚了多位成果卓著的专家学者，他们分别是：清华大学教授、青海大学校长**史元春**，西交利物浦大学数学物理学院院长、西交利物浦大学后量子迁移交叉实验室（PQC-X）主任、NIST PQC 标准核心设计者**丁津泰**，中国科学院计算技术研究所研究员、副所长**包云岗**，LVS 开源项目创始人、CCF 开源发展技术委员会副主任**章文嵩博士**。第二届特约顾问团分别为：中国开源软件推进联盟副主席兼秘书长、中国科学院软件所研究员**刘澎**，杭州指令集智能科技有限公司首席专家**潘爱民**，达闼机器人创始人**黄晓庆**，清华大学计算机科学与技术系副教授、KVCache.AI 团队负责人**章明星**四位专家担任。

顾问团对龙蜥社区过去几年取得的显著成就给予了高度肯定。其中，**史元春教授**表示，近两年，龙蜥社区致力于大模型驱动的重构，以支持更多 AI 服务器，实现云+AI 与操作系统的双向赋能；**章文嵩博士**提出，应凝聚生态合力，推动龙蜥成为 AI 时代具有影响力的开源操作系统底座；**刘澎**指出，龙蜥社区已发展成为中国最具影响力的操作系统开源社区。同时感谢首届顾问团中国工程院院士、浙江大学信息学部主任陈纯，中国工程院院士、浪潮集团首席科学家王恩东，凝思软件董事长、中国 Linux 先行者宫敏，北航网安学院院长、教授刘建伟等专家的支持。

龙蜥社区第二届高级顾问团

史元春

清华大学教授
青海大学校长

丁津泰

西交利物浦大学数学物理学院院长、西交利物浦大学后量子迁移交叉实验室（PQC-X）主任、NIST PQC 标准核心设计者

包云岗

中国科学院计算技术研究所
研究员、副所长

章文嵩

LVS开源项目创始人、CCF开源发展技术委员会副主任

龙蜥社区第二届特约顾问团

刘澎

中国开源软件推进联盟副主席兼秘书长、中国科学院软件所研究员

潘爱民

杭州指令集智能科技有限公司
首席专家

黄晓庆

达闼机器人创始人

章明星

清华大学计算机科学与技术系副教授、KVCache.AI 团队负责人

委员会会议

● 技术委员会

2025 年，龙蜥社区技术委员会会议召开 5 次，来自 25 家理事单位 40 位委员及委员代表出席。会上确定了龙蜥社区重点技术项目，Anolis OS 版本规划及维护策略，RISC-V、AI 方向规划等，并前瞻性地布局下一代技术生态。在版本发布方面，发布全面支持 RVA23 RISC-V 架构的 Anolis OS 23.4 版本；在核心技术研发上，ANCK 6.6 内核版本引入了全新的研发机制，通过构建内核补丁自动化同步平台，实现对上游 Linux 6.6 LTS 分支的持续监控、智能合入与质量闭环，显著提升了研发效率与系统稳定性。

2025 年 10 月，龙蜥社区走进阿里云召开技术会议

● 运营委员会

2025 年，龙蜥社区运营委员会全年召开 10 次会议，来自 25 家理事单位的 30 位委员及代表共同参与。会议明确了年度三大运营目标与社区规划，优化了顾问管理机制，强化“贡献多、收益大”的正向激励，夯实共建共享的治理基础。通过目标牵引，全年厂商贡献、用户案例、活动及贡献度增长 25%～60%。此外，会议就第三届龙蜥大会的举办情况、三大联盟的生态运作成果，以及后续生态合作计划与治理重点等进行了同步，推动社区在技术协同、生态落地、品牌影响力等方面实现全面突破。

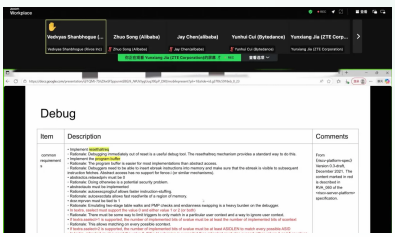
2025 年 7 月，龙蜥走进中兴通讯召开运营会议

● 联盟组织会议

龙蜥社区三大联盟智算联盟（OICA）、安全联盟（OASA）和系统运维联盟（SOMA）闭门会及月会举办超 20 次。在联盟会上，投票表决新成员单位加入、推进联盟　2026　年的目标和规划、联盟的运作规范及各单位后续的投入计划实施。其中，2025 年安全联盟年度工作会议确认了 2026 年总体发展规划，围绕 AI 与安全、供应链安全、PQC 密码等方面持续布局，明确以“标准规范为引领、技术产品为核心、生态运营为纽带”推进下一步工作。

● RISC-V 等重点项目会议

2025 年，社区活跃 SIG 召开了超 60 次会议，积极贡献上游社区和版本研发，推进 RISC-V 操作系统的发展与成熟。其中，龙蜥社区 RISC-V SIG 2.0 主导了 RISC-V 国际基金会 Data Center SIG 月度 10+ 场会议，围绕 RISC-V 在云数据中心和 AI 智能领域展开深入探讨，并联合达摩院、中兴通讯、如意社区等伙伴共建 RISC-V 繁荣生态。



8.2 龙蜥活动

● 峰会活动

龙蜥操作系统大会

2025 年 11 月 17 日，以“生态共融·智驱未来”为主题的 2025 龙蜥操作系统大会（OpenAnolis Conference）在北京圆满召开。500+家企业，1300 多名技术领袖、业界精英和行业开发者线下参会，北京市网信办、北京市经信局等多位领导莅临指导，中国工程院院士**倪光南**，中国工程院院士、清华大学计算机系教授**郑纬民**，开放原子开源基金理事长**程晓明**，阿里巴巴集团合伙人、阿里云智能集团基础设施事业部负责人**蒋江伟**，清华大学教授、清华大学人工智能研究院视觉智能研究中心主任**邓志东**，龙蜥社区高级顾问、凝思软件首席科学家**宫敏**等一众大咖倾情加盟。

中国工程院院士倪光南与龙蜥社区理事长、阿里云智能集团研发副总裁马涛、海光信息副总裁李亚东等共同出席发布了《国产服务器操作系统发展报告（2025）》，并邀请**全国政协委员、中国科学院计算技术研究所研究员张云泉**做了详细解读。同时，中兴通讯副总裁、龙蜥社区副理事长刘东和中科方德高级副总裁龚文等社区代表共同站台发布了龙腾计划 3.0——**AI 引擎生态加速合作计划。**

龙蜥大会现场嘉宾合照

《国产服务器操作系统发展报告（2025）》发布现场

龙腾计划 3.0——AI 引擎生态加速合作计划发布仪式现场

本届龙蜥大会举办了以自动驾驶、AI 创新与系统安全、RISC-V、智算基础设施、龙蜥技术生态为主题的五场技术分论坛，由**龙蜥社区安全联盟、智算基础设施联盟、RISC-V SIG 组联合阿里云、海光信息、中兴通讯、英特尔、浪潮信息、AMD** 等 10 多家单位组织、24 家单位共同承办，围绕智驾解数据闭环面临的通用技术挑战、智能化安全防御、软硬协同、云+智能计算、AI 与操作系统融合新范式等热门技术方向展开深度探讨。

本次分论坛出品人，由**金美琴、杨勇、宋卓、龙勤**等作为社区代表，与**阿里云李三红、海光信息杨继国、中兴通讯徐立锋、英特尔王庆、AMD 曲大健、如意 RISC-V 社区丁欣、阿里巴巴达摩院朱祯贞**等联合出品，超 40 场前沿技术分享，汇聚 400+ 业界专家、技术负责人、开发者等共同深度探讨 AI 与操作系统融合的新范式。


智算新基础设施分论坛


AI创新与系统安全分论坛


龙蜥技术生态分论坛


RISC-V 分论坛

数据X模型X软件分论坛

除此之外，现场充分展示了阿里云、海光信息、英特尔、中兴通讯、AMD、浪潮信息、如意社区、大普微等多家伙伴企业的最新动态，与会的开发者和技术爱好者们踊跃交流前沿技术、体验创新产品。

龙蜥大会活动现场

展区现场

龙蜥实验室体验

参加 2025 云栖大会、第 20 届中国 Linux 内核开发者大会等，分享社区前沿技术

2025 年 9 月 24-26 日，阿里云联合龙蜥社区举办了“操作系统开源与 AI 进化”分论坛，来自阿里云、清华大学、AMD、货拉拉、OPPO、英特尔、中兴通讯、安谋科技等资深专家，围绕智驾领域 AI 性能增强、原生安全、智能运维等维度上的突破性实践，创新技术的探索实战，以及多元算力基础设施协同新范式展开深度探讨。此外，龙蜥社区也设立了专属展区。在为期 3 天的活动中，展区人潮涌动，吸引了超千名观众参与互动，分论坛现场更是座无虚席，反响热烈。

2025云栖大会分论坛嘉宾合照

云栖大会龙蜥展区现场

云栖大会活动现场

此外，龙蜥还重点参加了第 20 届中国 Linux 内核开发者大会（简称 CLK 大会）、2025 世界人工智能大会、2025 RISC-V 生态大会等活动，分享龙蜥在操作系统领域的创新实践。其中，龙蜥社区委员会成员陈绪、金美琴作为 CLK 内核委员会成员组织并参加第 20 届 CLK 大会，会上也向马涛、陈绪等 17 位长期参与并作出重要贡献的专家学者颁发了 **CLK 20 周年特殊贡献奖**；在 2025 世界人工智能大会 AI 开源论坛上，龙蜥作为首批成员加入 **GDPS 全球开发者先锋社区平台**。

第20届CLK大会

龙蜥作为首批成员加入GDPS全球开发者先锋社区平台

● 龙蜥技术认证

2023 开放原子全球开源峰会上，开放原子开源基金会副秘书长辛晓华等 6 位产学研嘉宾代表共同出席《龙蜥社区发布人才培养计划》发布仪式，联合阿里云、统信软件、浪潮信息 3 家理事单位推出“龙蜥+”合作模式的人才认证体系和操作系统课程。

2025 年龙蜥学习中心上线中级认证课程，课程内容不仅涵盖基础理论知识，更注重结合实际案例，培养学员实战技能。龙蜥技术认证已联合浪潮信息等单位举办多场培训专场，覆盖学生、个人开发者及企业人员超 3 万人，持证人数超 3000 人。



●开源赛事和高校行

近一年，龙蜥社区参与共建了首届 CIE 全国 RISC-V 高水平创新和应用大赛、全国大学生操作系统设计赛、开源之夏等多个赛事和活动，提供了 30+ 赛题，AI 推理训练、RISC-V 等议题颇受欢迎；通过龙蜥高校行、开放原子校源行，社区多位专家走进了浙江大学、中南大学、中国农业大学、山东大学、吉林大学等高校分享龙蜥前沿技术，覆盖高校师生近 1500 人次，推进龙蜥技术布道、实操培训及技术认证。此外。龙蜥参与共建的第二届开放原子大赛赛项也完成线上决赛。开源赛事有效推动开源人才的培育与发展，为开源生态的可持续发展注入了蓬勃新生力量。

2025 年 11 月，走进中南大学


2025 年 4 月，走进中国农业大学

●龙蜥 MeetUp

「龙蜥社区“走进系列”MeetUp」是由龙蜥社区与生态合作伙伴联合主办的系列月度活动，每期走进一家企业，聚焦龙蜥社区和合作伙伴的技术、产品和创新动态，展示硬核技术，共建繁荣生态。每期活动邀请 10+ 技术大咖分享，目前共吸引了 300+ 企业，50 万+ 线下/线上开发者参与。

2025 年，龙蜥已走进理事单位兆芯、阿里云、浪潮信息、中兴通讯|新支点、Arm 等举办了系列 MeetUp，围绕 **AI 时代开源 OS 新生态、Java 与 AI 融合创新、面向泛在智算场景的操作系统、操作系统运维等技术**，采用线上+线下的活动形式，现场座无虚席，嘉宾们畅所欲言聊技术；此外，龙蜥也联合众多生态合作伙伴，共同举办了智算技术沙龙等多场技术 MeetUp，围绕 AI、操作系统、大模型、智能运维的最新技术展开分享。

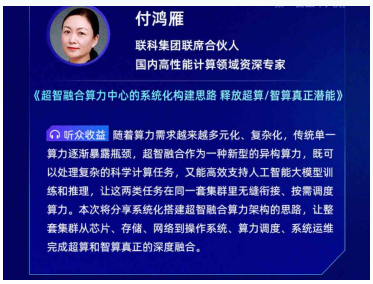
2025 年 4 月，龙蜥社区走进阿里云 MeetUp

2025 年 6 月，龙蜥社区走进中兴通讯 MeetUp

2025 年 12 月，龙蜥社区走进 Arm MeetUp

2026 年 1 月，龙蜥 X SGLang 智算技术沙龙

作为龙蜥社区的明星项目之一的「龙蜥大讲堂」和「龙蜥开发者说」，已成为连接社区合作伙伴与广大开发者的核心桥梁。其中，2025 年「龙蜥大讲堂」共举办 10 场，包括龙蜥社区智算技术专场、Anolis OS 23 专场、浪潮信息专场等，来自阿里云、AMD、浪潮信息、沐曦、联科等企业超 10 位技术专家分享，覆盖 16 万+开发者，与超千位开源爱好者互动；「龙蜥开发者说」共发布 4 期，邀请了来自等申威、海光信息、浪潮信息等企业的龙蜥合作伙伴技术成员或以个人形式贡献社区的开发者分享其在社区的所见所闻。



龙蜥大讲堂

龙蜥开发者说

●「AI 进化论：智算时代操作系统的破局之路」系列直播

在 AI 浪潮重塑算力基础设施的背景下，服务器操作系统正经历从"被动适配"向"主动原生驱动"的技术范式跃迁。由龙蜥社区、阿里云联合 InfoQ 打造的直播 IP 栏目《AI 进化论：智算时代操作系统的破局之路》，以云、AI、安全等技术与服务器操作系统如何融合演进为主线，聚焦服务器操作系统在智算时代的进化之路，围绕原生智能、原生安全、软硬协同等热点议题展开深度对话。目前已陆续推出**智算时代操作系统的破局之路、重构专有云 OS 安全的信任基石、CMaaS 重塑 AI 时代的信任边界**等系列总共 8 期内容，覆盖机密计算、智驾场景、可信计算、大模型推理等多领域热门话题展开探讨，邀请了阿里云、英特尔、OPPO、云杉网络、中兴新支点、中国科学院软件研究所，以及北京大学、 清华大学的技术专家参与探讨，观看人次覆盖 66 万+。

附录：

本白皮书里提到的龙蜥社区系列白皮书、技术白皮书等可在龙蜥官网-用户栏目获取：

https://openanolis.cn/

9. 社区年鉴

2019 十年自研

2009 年至 2019 年，龙蜥操作系统源自阿里云十余年操作系统的自研积累。

- 坚定将十年自研成果开源

2020 龙蜥诞生

阿里云、统信软件、海光信息等国内外头部企业共同发起龙蜥操作系统开源社区。

- 龙蜥社区理事会正式成立
- 近百家企业、千余开发者

2021 产业协同

汇聚 16 家操作系统产业生态链头部企业全票通过龙蜥首届理事大会章程和运营规划。

- 第一个龙蜥发行版正式发布
- 龙蜥生态发展计划正式启动
- 龙蜥社区安全联盟和系统运维联盟相继成立

2022 行业落地

中兴通讯、浪潮信息、Arm 等国内外知名厂商加入龙蜥理事会，共同推动操作系统根社区演进。

- 支持全国首个政府采购云平台顺利完成 CentOS 迁移
- 阿里云、浪潮信息、中兴通讯、移动、联通等13 家厂商发布龙蜥衍生版
- 多位行业专家加盟龙蜥特约顾问团，为龙蜥发展提供指导

2023 占比第一

据中国信通院用户调研显示，53% 用户操作系统迁移首选龙蜥，占比第一，位列首位。

- 陈左宁、梅宏、王怀民、陈纯、王恩东等多位院士肯定龙蜥发展成果，通过加入龙蜥高级顾问团等方式指导龙蜥未来发展
- 首款全面拥抱智算的国产操作系统 Anolis OS 23 正式推出
- 《龙蜥社区人才培养计划》正式推出，近 3000 人持证

2024 引领创新

龙蜥坚定“云+AI”的可持续发展路线，引领国产操作系统创新发展。

- **超 1000 家**社区伙伴，生态上下游产业链类型全覆盖
- **超 800 万**装机量，龙蜥生态装机量平均增速 300%
- **超 17000 名**开发者，SIG 组超 60 个，社区活跃度爆发式增长
- **超 100 万**用户，龙蜥落地政务、金融、电信、互联网等行业

2025 AI 领航

从生态共建到国际标准主导，龙蜥正将创新势能转化为产业动能。

- 丁津泰、包云岗、章文嵩博士、潘爱民、黄晓庆、章明星等 8 位成果卓著的专家学者加入龙蜥社区第二届顾问团，对龙蜥的长久发展提出实质性改进和提升意见
- 产品首发，推出 Anolis OS 23 系列版本和 RISC-V 正式版
- 龙蜥操作系统装机量**突破 1000 万**，存量市场份额**近 50%**，增量市场份额将**突破 50%**
- 《国产服务器操作系统发展报告（2025）》显示，国民喜爱度**超 54%**，龙蜥蝉联“用户意愿迁移操作系统榜单”**第一位**

10. 他们说

对长期深耕开源生态、推动技术创新的龙蜥社区全体成员致以诚挚敬意，也高度认可了龙蜥社区对杭州数字经济的贡献。

杭州市相关单位代表

我国操作系统产业取得显著进步，涌现出了一批像龙蜥社区这样充满活力的创新力量，开源项目活跃创新、生态合作不断深化，产业创新生态繁荣发展。我们需要持续发扬协同创新的合作精神，推动开源生态向更高层次、更广领域迈进。

潘 锋

北京市网信办副主任

筑牢数字底座，操作系统作为自主可控生态的核心支柱，正成为提升信息产业创新能级的关键引擎。

姜洪朝

北京市经济和信息化局副局长

当前在 AI 浪潮的推动下，全球都面临重塑产业格局的新局面。今天我们看到，五年来龙蜥社区广大开发者们始终践行“开源、开放、共建、共享”的开源发展理念，取得了令人瞩目的成就。面向未来，希望龙蜥社区不断发扬开源精神、传递开源文化，培育开源沃土，让更多的开发者成为创新发展的生力军。

倪光南

中国工程院院士、中电标协 RVEI 战略指导委员会主任

成立五年，龙蜥社区在‘云+AI’的可持续技术路线指引下，对此展开了富有成效的探索，通过凝聚操作系统厂商、芯片厂商、科研机构与行业用户的生态合力，优化新一代智能内核架构，充分释放硬件潜力；更在工业制造等重点领域，助力企业实现智能化升级。期待龙蜥社区继续勇担使命：加快构建智能算力调度网络，深化国产芯片生态协同，培育高水平人才队伍，为中国算力基础设施建设贡献更大力量。

郑纬民

中国工程院院士、清华大学计算机学院教授

未来五年特别是在“云+AI”融合趋势下，龙蜥社区还需持续夯实技术底座、推动生态协同。

陈　纯

中国工程院院士、浙江大学信息学部主任
龙蜥社区第一届高级顾问

龙蜥社区的成长令人瞩目。自社区成立以来，阿里云在社区建设中持续贡献，与多家理事单位共同推动产品持续优化、深耕社区生态建设。目前社区已汇聚开发者 2 万余人，超 1000 家合作伙伴，243 万多用户，在政务、金融、能源、通信、交通等多个领域内开展应用，构建起软硬件协同发展的良性生态。

程晓明
开放原子开源基金会理事长

作为基础软件从业者，我们深感责任重大。同时作为龙蜥社区的亲历者和见证者，五年的耕耘和收获也让我们充满信心。阿里云将持续加大在龙蜥社区的技术和生态投入，深化“一云多芯”战略，联合国产芯片伙伴共建软硬协同方案，筑牢供应链安全底线，大力推动龙蜥与通义大模型、魔搭社区等阿里 AI 开源体系深度融合，让龙蜥成为中国操作系统走向世界的名片。

蒋江伟
阿里巴巴集团合伙人
阿里云智能集团基础设施事业部负责人

感谢您认真阅读来到这里（或直奔我们而来）

我们坚信，未来十年，操作系统的大发展一定势不可挡。

加入龙蜥，一起打造面向智算时代的开源操作系统！

欢迎您随时和我们联系。

龙蜥视频号

龙蜥视频号
微信扫码查看更多精彩瞬间

龙蜥公众号

龙蜥微信公众号
微信扫码获取更多一线资讯

龙蜥社群

钉钉交流群
钉钉扫码加入交流前沿技术

龙蜥官网

https://openanolis.cn/

联系邮箱

secretary@openanolis.org

龙蜥助手：小龙

微信号：openanolis_assis

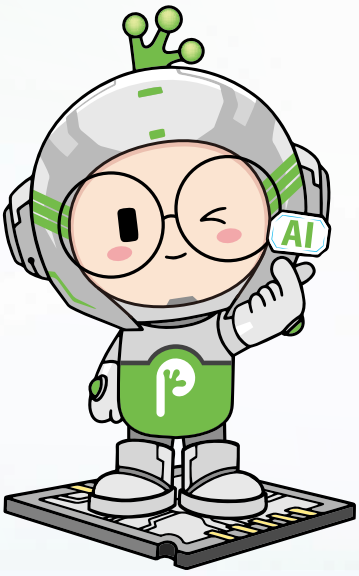
龙蜥交流群

钉钉群号：134730002067

白皮书作者

蔡佳丽　陈健康　陈盛德　陈宗耀　程书意　段勇帅　高　畅　高　翔　贺　迪
胡玉溪　赖　堃　李会佳　李靖轩　李三红　刘　峥　龙　勤　马　丁　马　涛
马　腾　庞训磊　彭媛洪　钱　君　曲大健　沈　培　舒圆鹤　宋　卓　孙维东
谭伯龙　谭钦云　田瑞冬　王　强　王　庆　王　卓　杨　勇　袁艳桃　张家乐
张　佳　张金利　张天佳　张旭芳　张永超　郑臣明　郑　耿　周　鹏　周伟涛

*按照首字母排序



OpenAnolis
龙 蜥 社 区