OpenAnolis
龙蜥社区

2022
龙蜥社区全景白皮书

2022 OpenAnolis White Paper

目录 Contents

06 “龙蜥+”精选方案与案例

07 社区风采

01 寄语

龙蜥社区拥有庞大的商业应用客户群，龙蜥社区是开发和维护操作系统关键技术的上游社区，它的进步将在国产基础软件支持产业迭代创新进程中发挥领导作用。

龙蜥社区特约顾问

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**

龙蜥社区今年在社区与技术方面稳步前进，在云原生、eBPF、安全等方面都孵化出了新的成果，并惠及了更多的社区爱好者，吸引了越来越多的伙伴加入。数字经济的大潮汹涌向前，龙蜥在中国的操作系统开源社区中无疑站在了潮头，我们相信秉持中立、开放、平等的龙蜥开源社区必将在未来继续勇敢前进，持续创新，为中国与世界的基础软件开源产业做出更大的贡献。

龙蜥社区副理事长 统信软件CTO | **张磊**

发展信息产业基础层核心技术，为世界数字经济的发展创新贡献力量，是我们中国计算机人长久以来的壮志理想，也是龙蜥社区的光荣使命。共同的愿景和信念将国内最优秀的人才和力量汇聚到了这个平台，众擎易举，未来可期。

龙蜥社区理事 龙芯中科副总裁 | **高翔**

龙蜥操作系统自发布以来，在过去两年的时间里一直保持持续创新和技术演进，龙蜥社区也得到了长足的发展，汇聚了众多开发者和社区用户。新一代的龙蜥OS基于分层分类系统架构，全面支持下一代数据中心芯片，尤其是对Intel最新可伸缩至强平台以及最新特性的支持，更是走在全球操作系统社区的前列。本书对于龙蜥OS的关键技术做了很好的介绍和总结，希望有志于参与龙蜥社区以及开源项目的开发者都能从中获益。

龙蜥社区理事 Intel技术总监 | **杨继国**

“立志欲坚不欲锐，成功在久不在速”，无论是软件还是硬件，从无到有成长起来都是一个不断试错、不断完善的过程，只有从实践中不断磨砺才能进入良性循环。如今，我们看到了国产操作系统从无到有，从基本能用到完全可用，再到实实在在支撑起一个个应用场景，解决一个个问题，相信本次《2022龙蜥社区全景白皮书》的发布可以给更多企业、开发者以实践参考，吸引更多开发者加入，为社区建设、操作系统的未来发展提供真知灼见。

龙蜥社区特约顾问 极客邦科技创始人兼 CEO | **霍太稳**

在当前中国操作系统的新江湖中，龙蜥操作系统以其稳健、可靠的风格赢得了一席之地。在阿里云、统信等诸多社区成员的协力之下，除了不断迭代的技术创新之外，龙蜥还提供了满足企业级需求的基础设施支撑，以及长期的技术支持承诺，这使得企业级需求得到了可靠保障。

龙蜥社区特约顾问 Linux中国创始人 | **王兴宇**

02 序

一年前，我们发布了《OpenAnolis龙蜥操作系统开源社区技术创新白皮书》，对龙蜥社区在技术发展和创新上的思路以及围绕思路的初步探索进行了系统性的展示。白皮书发布之后，收到了很多社区用户的反馈，既有诚挚的感谢，也有中肯的意见，当然最多的还是对于龙蜥操作系统开源社区发展的充分肯定。这里先感谢所有社区的所有参与者，我们的理事单位、合作伙伴以及千千万万的开发者和用户。

2022年即将过去，回顾这两年龙蜥社区的发展，如果说2021年，龙蜥社区在技术创新和开源发展上明确了方向，那么2022年龙蜥社区的发展是全方位的。

2022年是龙蜥伙伴快速发展壮大的一年。目前我们有二十一家理事单位和近三百家合作伙伴，涵盖了主流的操作系统厂商、芯片厂商、整机厂商、云计算厂商和应用厂商，大家一起为社区献策献力，共同推动操作系统的发展，实现了龙蜥社区的开放、平等、协作和创新。去年我们号召“集合全社会的力量一起来推动社区的发展“，今年已经初见成效。

2022年是龙蜥社区在产业扎实落地的一年。通过理事单位的努力，我们帮助一批重点企业完成了CentOS的迁移和替换，彻底解决了CentOS停服的隐忧。同时通过进一步的沉淀和积累，形成了覆盖政务、电信、金融、交通，能源，制造等多个领域的迁移产品和解决方案，真正做到了“手中有粮，心中不慌“。

2022年是龙蜥社区技术持续进步的一年。技术是社区的核心竞争力，创新是社区的源动力。过去的一年，社区在原生技术方面不断突破，在软硬件协同、新型操作系统内核技术、云原生、系统安全、大规模运维等方面均实现了完整的、自主的技术革新，既解决了社区用户的痛点，也做到了面向未来的布局。

2022年是龙蜥社区不断获得认可的一年。我们获得了OSCAR尖峰开源社区及项目奖项、CSDN年度技术影响力「年度开源项目」奖、中国开源云联盟年度优秀开源项目”奖、2021“科创中国”开源创新榜入榜、年度OSCHINA优秀开源技术团队”奖、工信部电子标准院首批开源项目成熟度评估，唯一获“卓越级”（最高等级）的开源项目等殊荣。

龙蜥社区的使命是共创数字化发展开源新基建，并正努力成为全球数字创新基石。我们深知“不积跬步，无以至千里；不积小流，无以成江海“。所以每一年，我们的突破都是扎扎实实的，也是充满希望的。

我们坚信，未来十年，操作系统的大发展一定势不可挡。欢迎所有有志于参与操作系统研发的同仁一起加入龙蜥社区，打造面向未来的下一代操作系统。

马涛

龙蜥社区理事长、阿里云研究员、阿里巴巴集团内核团队的创始人，阿里云基础软件操作系统负责人

03
社区伙伴风采展

“双龙组合”打造中国操作系统核心力量

龙芯中科作为龙蜥社区的理事会成员，主要负责LoongArch SIG工作小组，致力于提供Anolis OS对LoongArch架构的支持，并围绕LoongArch构建软件生态。2022年1月，在龙芯中科以及龙蜥开源社区LoongArch SIG组成员的共同努力下，龙蜥Anolis OS LoongArch正式版对外发布。**Anolis OS 8.4 LoongArch是全球首个支持龙芯LoongArch架构的同源异构服务器操作系统。**

- **内核更新到4.19.190-4，经过功能测试、性能测试以及压力测试多项测试，质量可靠；**
- **修复了libxcrypt的以及相应的依赖问题；**
- **支持图形界面和多种安装场景；**
- **使用docker-ce 20.10.3为默认的容器管理工具。**

龙芯中科作为龙蜥社区LoongArch SIG工作小组Maintainer，和各项目成员一起合作完成了基于LoongArch架构的服务器OS研发工作。其中，龙芯团队完成了基于龙芯Server8.3系统为基础的OpenAnolis 8.4最小开发环境的构建，贡献了约200个软件包，参与构建软件包400多个，参与解决社区软件移植BUG 50多个，联合OSV共完成2767个src.rpm，共构建6673个软件包。

龙芯中科技术股份有限公司（简称：龙芯中科）面向国家信息化建设需求，面向国际信息技术前沿，以创新发展为主题、以产业发展为主线、以体系建设为目标，坚持自主创新，全面掌握CPU指令系统、处理器IP核、操作系统等计算机核心技术，打造自主开放的软硬件生态和信息产业体系。

LoongArch基于龙芯二十年的CPU研制和生态建设积累，从顶层架构，到指令功能和ABI标准等，全部自主设计，不需国外授权。LoongArch已得到国际开源软件界广泛认可与支持，正成为与X86/ARM并列的顶层开源生态系统。

Intel是龙蜥社区首批理事成员单位，全面参与理事会，技术委员会， 运营委员会的日常运营工作与社区治理决策。创建、参与、维护Intel架构、内核、容器镜像、机密计算等多个兴趣小组。将Intel架构的新平台、新特性、新优化贡献并集成到Anolis OS发行版中，使得广大社区上下游生态用户能第一时间在Anolis OS上获得最佳的性能体验，进而使得Anolis OS成为在全球范围内，最先支持最新Intel平台的操作系统之一。

“芯”“蜥”相通，共建“芯”生态

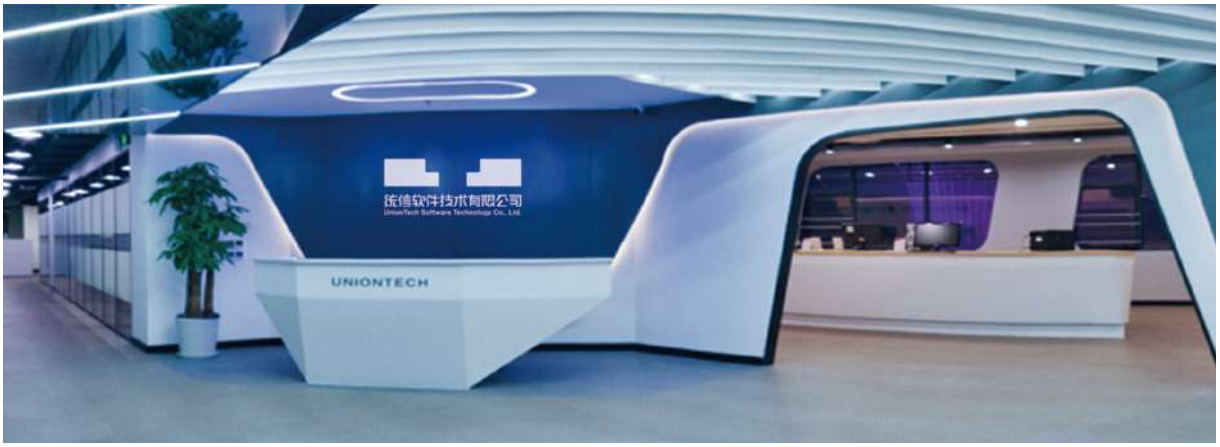
关键贡献

1. 参与**20+ SIG的工作和讨论；**
2. 提交merge request，65个；
3. feature request & bug fix，50个；
4. 维护**4个代码仓库；**
5. 提交**16万+行代码；**
6. 组织了7场SIG会议以及技术分享；
7. 举办了一次在线Meetup，**14位技术专家的13场技术分享**，现场达到了**15万**的访问量和**230万次的媒体曝光。**

项目支持

- 维护Intel ARCH SIG内核代码仓库，支持Intel新平台新特性在 ANCK 上的迭代工作；
- 维护AI SIG的 TensorFlow、Pytcrch、IPEX代码仓库，赋能龙蜥在AI领域的场景探索；
- 推进CRI-RM在云原生SIG的落地；
- 机密计算SIG，集成基于Intel机密计算特性代码。

Intel公司介绍：英特尔（NASDAQ：INTC）作为行业引领者，创造改变世界的技术，推动全球进步并让生活丰富多彩。在摩尔定律的启迪下，我们不断致力于推进半导体设计与制造，帮助我们的客户应对最重大的挑战。通过将智能融入云、网络、边缘和各种计算设备，我们释放数据潜能，助力商业和社会变得更美好。www.intel.cn



统信软件作为龙蜥社区副理事长单位，积极参与到社区的建设工作，深入支持多个社区版本的发布、组织并参与了多个SIG，并将自主研发的DDE桌面软件移植到了龙蜥操作系统，为龙蜥的用户提供了更好地用户体验。

17
参与SIG组

9000
代码仓库同步

8378
koji提交

32
社区问题支撑

472
BugFix

187
文档贡献

2100
Merge Requests

124
SIG Meetings

重点贡献

- 主导创建DDE SIG和Anolis Course SIG
- Anolis OS 7.7、8.2、8.4、8.6的正式发布
- Anolis OS 7.9 RC1、QU1的发布
- Anolis OS 8.7、23版本开发
- 贡献独立自主研发的桌面环境-DDE
- 加入到17个社区SIG组中参与贡献，在其中8个SIG组内担任maintainer，并且在kernel、Keentune、DDE、版本发布、安全、OPS、Course等SIG组作为主要维护者和贡献者

社区贡献

- 在安全sig组，共同推动确立系统安全基线，输出漏洞扫描
- 工具和加固脚本，并且安全基线被OpenScap社区所采纳
- 完成了OpenAnolis oval漏洞库建设，并向第三方漏扫工
- 具开放漏洞库
- 在keentune项目开发中，实现了keentune异步分布式架
- 构，充分兼容上游tuned
  为社区OPS sig组贡献了安全中心特性，并优化提升sysom DFX
- 在kernel sig组中，主导完成了对文件拷贝、并发性能进行优化提升，以及对关键bug/CVE进行修复
- 为满足不同条件下的系统迁移需求，实现了自定义仓库下的在线、离线系统迁移
- 在高性能存储sig组中，完成了ceph、集中存储、本地磁盘等存储方案在mariadb集群场景下测试方案和脚本编写
- 在容器sig组完成了ostree集成工具部分接口重写
- 为提升产品质量，在QA sig组中完成了新增开发测试用例500+，脚本60+

统信软件技术有限公司（简称：统信软件）是由国内领先的操作系统厂家于2019年联合成立，总部设立在北京，同时在武汉、上海、广州、南京、成都、重庆、西安、太原、深圳等地设立了地方技术支持机构、研发中心和通用软硬件适配中心。专注于操作系统的研发与服务，发展和建设以中国技术为核心的创新生态，致力于为不同行业提供安全稳定、智能易用的产品与解决方案。

04
社区技术演进

云计算为信息产业带来了颠覆性变化，也开始快速渗透到千行百业。大量企业用户开始关注如何实现自身数字基础设施的云化以及应用架构的现代化。同时，大数据、人工智能等新兴场景对算力快速增长的需求对进入后摩尔定律时代的基础设施提出了更高的挑战，充分释放基础设施的算力也成为企业用户关注的焦点。龙蜥社区成立之初即秉持与社区伙伴共创数字化开源新基建的愿景，希望通过建设面向云计算为终态的开源软件创新生态，扫清用户迁云障碍，帮助开发者研发和使用基于云计算场景的开源软件。

操作系统作为连接用户应用与硬件资源间的桥梁，是充分利用基础设施算力的纽带，也为新的应用场景提供运行环境，是龙蜥社区技术投入的重点方向。为此社区打造了以龙蜥操作系统Anolis OS为核心的操作系统发行版及软硬件生态，希望将Anolis OS建设成为数字基础设施互联互通的基石，让用户搭上云计算时代的快车。

图 操作系统是联系用户应用与基础设施的纽带

4.1 社区技术布局

用户在建设云计算时代数字基础设施时，需要克服诸多技术挑战。Anolis OS在过去两年中也着重在此方面进行能力建设，全面支持了各类芯片架构；完成了一批软硬件协同技术的标准化和规模化；发布了一系列在通用场景、云原生场景与安全可信场景下经过大规模使用的技术；为用户业务的平滑迁移与稳定运行构建了经过实践检验的迁移与运维工具。以期帮助用户更好的实现云计算时代数字基础设施的建设。

图 龙蜥社区整体技术布局

异构算力支持

在社区各芯片体系结构SIG与编程语言SIG的指导下，Anolis OS实现了对各体系结构芯片的完善支持，提供了包括x86、ARM、LoongArch、RISC-V在内最新芯片的支持。在社区伙伴的贡献下，Anolis OS成为最早支持Intel Sapphire Rapids (SPR)、ARMv9等芯片的操作系统发行版，为充分发挥用户基础设施算力提供了有效支撑。

处理器体系结构	支持版本
x86	Anolis OS 8.4及以上
ARM	Anolis OS 8.4及以上
RISC-V	Anolis OS 8.6及以上
LoongArch	Anolis OS 8.4及以上

表 主要体系结构支持版本

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。社区也在此领域推动了VIRTIO 1.2、XQUIC等一批软硬件协同标准的制定，实现了AF_XDP、SMC-R等技术在社区伙伴业务上的规模化部署使用，构建了针对DSA架构的SDK和面向开源芯片软硬件协同设计的操作系统。

面向不同场景的开源技术

用户应用需要的常见运行环境包括通用计算场景、云原生场景和安全可信场景。社区也分别在这三类场景中开源了一批项目，帮助用户的应用更好的在这些场景中运行。

在通用计算场景中，用户经常需要运行数据库、Web服务器等应用。为了优化此类应用的性能，社区开发者提供了诸如io_uring等技术和相关优化方案，帮助用户充分发挥基础设施的能力。

社区为云原生场景孵化了龙蜥云原生套件Anolis Cloud-Native Suite（ACNS）。此套件中包含完整的云原生技术栈，包括：容器

引擎RunD、容器镜像分发系统Nydus 、容器操作系统发行版LifseaOS和容器编排调度系统Koordinator。ACNS可以提供开箱即用的体验，让用户快速体验云原生技术的魅力。

随着数据安全与隐私保护成为用户关注的重点领域，运行在安全可信场景上的用户应用在不断增多。社区在这一场景中不仅结合各类芯片硬件特性构建了完整的机密计算能力，同时也构建了基于容器技术的机密计算容器开源项目Inclavare Containers，在不侵入应用的条件下实现用户对机密计算的需求。

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。

保证应用稳定运行

完成业务迁移等工作后，用户业务的稳定运行就成为基础设施要完成的最重要工作。这就需要经过大规模实践检验的运维技术。在社区伙伴的贡献下，龙蜥社区先后成立了系统运维SIG和eBPF SIG，孵化了SysOM、SysAK、Coolbpf等运维项目。SysOM (SYStem Operation & Maintenance) 是龙蜥社区一站式系统运维平台，通过统一的前端Web将所有运维服务的分析数据展示给用户，用户可以在同一个平台上进行主机管理、系统监控、异常诊断、日志审计、安全管控等复杂操作系统管理。SysOM的前端则使用了深度诊断解决方案SysAK，该解决方案沉淀了在百万级别服务器上使用运维工具的经验。Coolebpf极大降低了eBPF应用开发编译的门槛，创新的提出远程编译思想，具有资源占用低、可移植性强等优点，适合在生产环境批量部署所开发的应用，可以使同一个eBPF应用无需修改就能在3.x/4.x/5.x新老内核版本安全运行，解决了eBPF应用开发的痛点。

当前社区已经形成了以Anolis OS为核心的操作系统发行版矩阵，形成了包括LifseaOS、Alibaba Cloud Linux、统信服务器操作系统V20、银河麒麟服务器操作系统、BC-Linux、凝思安全操作系统在内的上下游发行版。同时，社区已启动Anolis OS 23的研发工作。Anolis OS 23基于分层分类理论构建，基于该理论对发行版软件包进行筛选，制定维护策略，为用户提供安全、稳定、可靠的操作系统发行版。

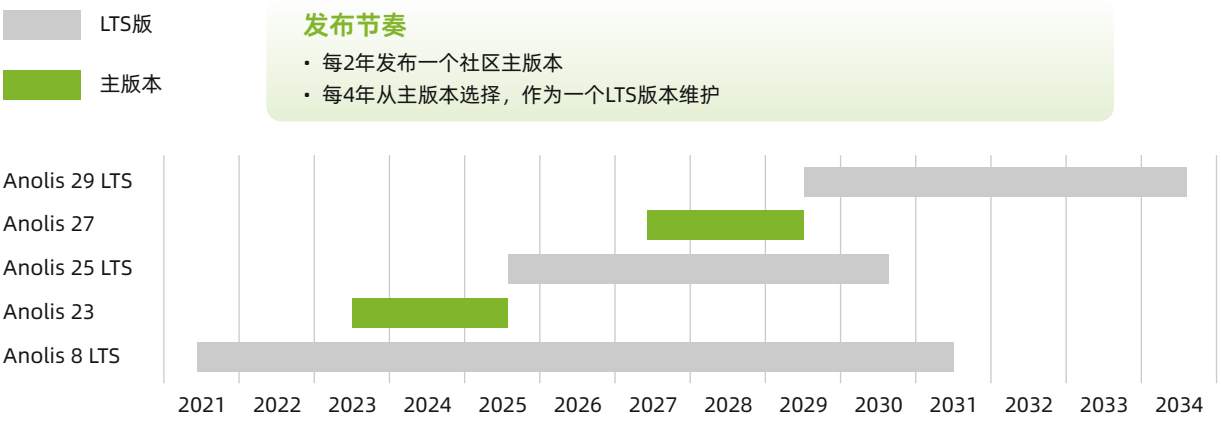


图 Anolis OS发行版版本规划

4.2 技术生态与产业协作

开源社区往往采用传统开源软件的研发模式，即采用集市模式进行开发，各发行版在软件架构层面缺少共识，操作系统开发和运维人员、操作系统服务提供商在进行发行版研发，核心软件选择，安全漏洞修复，软件版本维护时缺少对相关问题的决策依据和评判标准，给操作系统厂商和终端用户在软件版本选择、维护等方面造成困扰。龙蜥社区对操作系统发行版涉及的整体架构和软件版本探索建立了分层分类理论，用以在Anolis OS及其衍生版本研发过程中指导发行版研发的技术规划，协调研发过程中合作伙伴的职责分工，帮助筛选关键核心软件。在发行版使用维护过程中，用以评估缺陷漏洞的影响范围和严重程度，衡量发行版软件升级的策略，为Anolis OS及其衍生发行版提供全生命周期的理论指导。当前社区重点研发的Anolis OS 23发行版在软件包选型上已经采用分层分类策略，软件包选型过程优先考虑重要软件包的版本和相关依赖，确定重要软件包（如Linux Kernel）的版本后，再逐批引入更高层的软件包。这些软件版本确定后，社区也会对这些重要的软件包制定维护和更新策略，以便简化操作系统厂商发行版的制作，简化终端用户软件版本选择和维护工作。

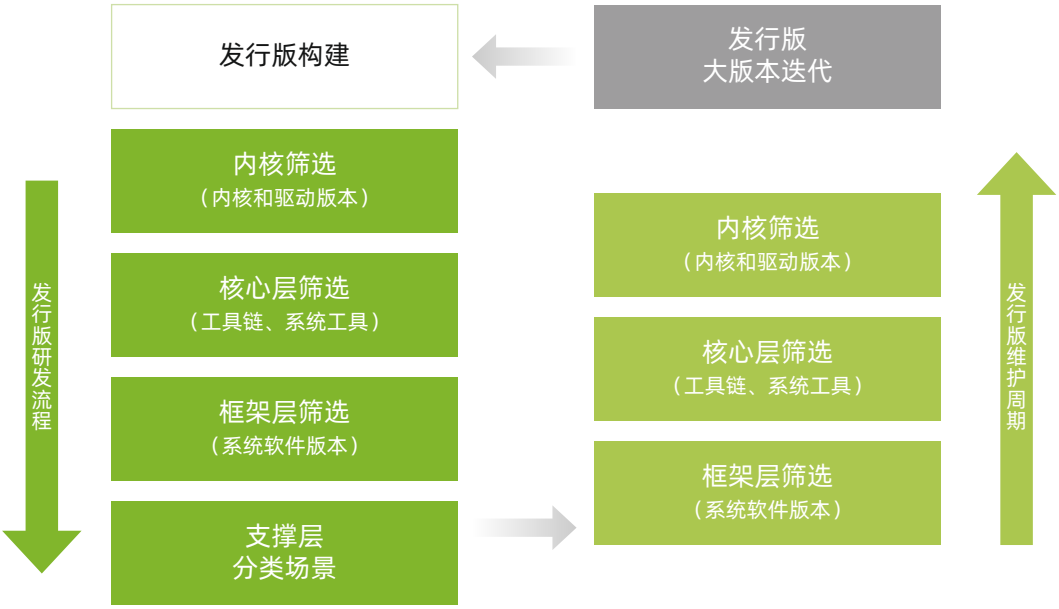


图 分层分类理论指导 Anolis OS 23 软件包选型

分层分类思想不仅指导社区操作系统发行版的软件包选型策略，也在社区技术生态布局方面发挥着积极的作用。龙蜥社区SIG的建立也会优先考虑更为重要和核心的基础SIG，如社区基础设施SIG、合规SIG、CI/CD SIG、文档SIG等都是第一批被建立的。此外，与操作系统技术生态直接相关的芯片类SIG等也是首批建立的SIG。当前龙蜥社区已经完整构建了与社区运作相关的SIG以及硬件芯片相关的SIG，并开始逐步完善上层应用和解决方案相关的SIG。

在构建社区生态的同时，为了更好的发挥产业链上下游合作伙伴的的能力，社区还依托分层分类理论和技术生态形成自己的产业协同，在操作系统全生命周期中形成明确的分工。

图片由龙蜥社区提供和授权

4.3 未来技术演进与展望

当前信息产业既面临着诸多技术挑战，也存在许多潜在的技术创新空间，而云计算也成为应对挑战和突破创新的中心。

摩尔定律失效造成算力增长缓慢

算力增长缓慢是目前信息产业面临的重要挑战。在后摩尔定律时代，产业界也提出了领域专用架构（Domain Specific Architecture，DSA）来应对大数据、人工智能领域对算力增长的需要。云计算的业务形态使得其可以通过将底层异构芯片的算力进行抽象，让用户更方便的使用异构芯片和DSA架构。

操作系统是支持芯片最为基础的系统软件，因此对异构芯片和DSA架构的支持就成为未来龙蜥社区的重要演进方向。未来社区和Anolis OS会持续投入和完善对异构芯片的支持。当前Anolis OS已经完成对主流处理器芯片的支持，未来社区在继续完善操作系统、编译器和工具链组件对异构处理器芯片支持的同时，会持续支持更多DSA架构，让Anolis OS提供云-边-端多场景异构处理器的支持。

云计算催生全新计算范型

计算产业的形态已经产生了极大的变化，从机器抽象的IaaS态，到容器抽象的CaaS形态，再到服务器无感的Serverless形态，云计算推动了计算产业形态和相关技术的快速发展。而随着计算服务的边界抬升上移，一方面可以让底层异构芯片的算力更容易被抽象，形成对用户透明的统一算力。另一方面也为实现从运行时（Runtime）到硬件（Hardware）充分的软硬件协同提供了必要基础。通过软硬件协同配合DSA架构充分发挥异构硬件算力已经成为当下重要的技术发展趋势。

龙蜥社区也将持续孵化云计算相关开源项目。云原生相关技术既是云计算未来技术演进的主要方向，更是社区重点投入的领域。云原生与Serverless以及软硬件协同技术结合，在持续释放底层算力的同时，能够让用户更好的享受到云计算在算力、资源弹性等方面的优秀体验。同时，社区也将持续改善用户迁移上云方案和工具，让Anolis OS成为用户平滑迁云的基石。

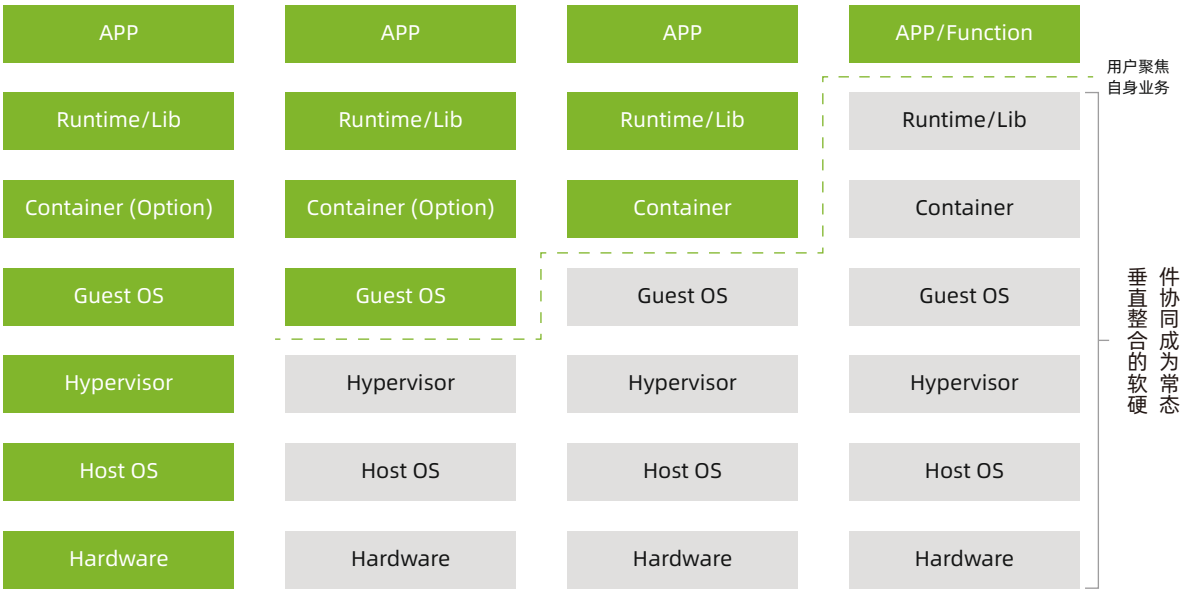


图 云计算扩大软硬件垂直整合空间

软硬件全栈协同设计与优化

软硬件协同是充分发挥各类硬件算力的关键。在“硬件 + 操作系统 + 编译器”相结合的基础上，将软硬件协同的边界拓展到运行时组件，结合Serverless等新型计算技术，实现用户业务逻辑与底层基础设施的彻底解耦与高效运行，帮助用户业务快速便捷的实现架构现代化。

数据爆炸需要突破内存墙限制

从智能网卡SmartNIC到存内计算PIM，软硬件协同趋势促使硬件不断创新。其中以CXL为代表的新型总线技术对未来基础软件架构设计影响最为重要。CXL总线提供了缓存一致性能力，在处理器与外设之间提供了更细粒度的数据访问能力，为减少数据移动提供了必要的基础，为提高大型软件的运行效率提供了可能。同时CXL总线的服务界面让操作系统和编译器更容易对其进行封装，减小对用户应用程序的侵入，避免用户应用程序的大规模改造。

数据安全和隐私保护推动新的计算技术

随着国内外数据安全和隐私保护法律规范的相继出台，用户会更加关注自身数据的安全保护。硬件芯片厂商近年来不断加强自身芯片中与隐私计算相关的技术，如TrustZone、SGX、SEV等芯片特性相继发布。而与之配套的计算技术发展仍然相对滞后，这也推动了相关计算技术的快速发展。

信息安全作为信息产业重要的领域，是龙蜥社区未来重点持续发力的方向之一。未来社区会持续通过软硬件协同等方式实现系统安全与机密计算技术，形成更加成熟的数据保护与隐私技术栈，让用户在享受云计算所带来的红利时，也能够更好保障自身的数据安全。

云计算在过去十年间推动了信息产业的变革，也促进了以操作系统为代表的基础软件行业的发展。龙蜥社区也会继续秉持平等、开放、协作、创新的信念，在基础软件领域持续创新开源技术、拓展开源生态、扩大开源影响力，助力社区伙伴与用户便捷平顺的过渡到以云计算为基础的数字基础设施上。

05
原生技术概览

5.1 芯片技术

5.1.1 Intel下一代芯片支持

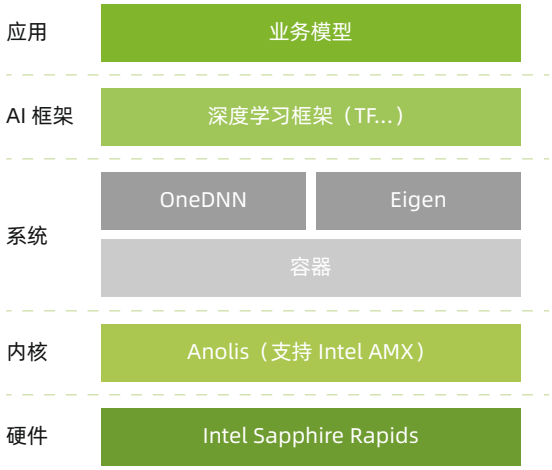
Intel Sapphire Rapids（简称SPR）即第四代至强可扩展处理器，是Intel承上启下的一代产品。新引入增强指令、硬件加速器、AMX矩阵计算、SGX机密计算、Scalable IOV、PCIe 5.0、CXL1.1协议，涵盖了计算、安全、I/O及虚拟化方面的众多技术升级和增强。

龙蜥社区对Intel平台的支持一直领先于业界，紧随Intel研发节奏，基于Intel Arch SIG第一时间完成对SPR产品全面的支持，助力龙蜥客户以开箱即用的方式，享用新技术特性，整体支持情况如下图：

图 操作系统是联系用户应用与基础设施的纽带

以上特性均在龙蜥社区得到全面支持：如加速器层面，Anolis 5.10内核已经支持DSA，IAA，配合龙蜥社区提供的accel-config用户态工具，用户可以对这两个加速器进行快速配置并使能。对于QAT和DLB，其驱动已经以OOT的形式发布于龙蜥社区，用户只需下载安装对应的rpm包即可享用新硬件加速带来的性能提升。安全层面，龙蜥社区已经集成SGX的SDK/PSW/DCAP软件栈，并以rpm包形式发布，用户可以通过Anolis系统便捷部署自己的机密计算方案。虚拟化层面，Anolis内核率先支持SIOV特性，用户不论是使用SPR平台自带的硬件加速器，还是支持SIOV的第三方硬件，抑或是DWQ/SWQ的部署形式，都可以无感使用，真正做到了One for All。

以下以对AMX指令的支持为例，介绍一下基于Anolis内核构建业务方案所带来的巨大性能提升：



Anolis通过支持AMX指令并对其进行深层次的优化，搭建如图所示基于Anolis内核的深度学习方案，实测表明：

- 相较于AVX512 INT8，Anolis-AMX-INT8带来8倍整型性能提升。
- 相较于AVX512 BF16，Anolis-AMX-BF16可以带来16倍浮点性能提升。

5.1.2 龙蜥ARM生态支持

生态背景

随着ARM架构技术的快速迭代发展，以及云服务厂商多元化的算力需求，ARM架构在数据中心服务器市场的渗透率正在逐年提升，对ARM生态的支持也成为各大厂商近些年来的重点发展趋势。在这种背景下龙蜥社区也同步开展了对ARM软硬件生态的构建，为了满足多样化的使用需求，龙蜥操作系统围绕一云多芯，对各个主流的ARM架构芯片同时展开了支持，其中包括阿里巴巴平头哥自研的倚天芯片和其他厂商的ARM架构芯片。在对芯片支持的同时，更进一步在ARM架构丰富的芯片技术结合软硬件不断完善技术体系和方案，同时对ARM应用生态的业务支持与探索也是龙蜥OS的主要发力点。

整体架构

芯片研发： 龙蜥社区围绕一云多芯，以及丰富ARM生态的策略，展开了对ARM芯片的全方位研发支持。龙蜥社区不仅深度参与倚天710芯片的研发，从芯片设计，软硬件协同研发，到芯片验证，流片，以及结合业务的产品化领域深入合作。同时，龙蜥社区对鲲鹏，飞腾，安培等其他ARM生态产品各种硬件功能也提供了完整的支持。用户可以使用龙蜥操作系统，针对在业内主流的几大ARM芯片进行丰富的业务开发和应用。

内核技术：龙蜥操作系统内核基于ARM生态硬件，围绕软硬件结合技术，深度展开全面的协同研发和优化，并将这些技术服务于应用场景。例如：在性能加速方面，结合倚天的加解密软硬件加速器的全栈支持；稳定性方面，自研的RAS内核以及内存隔离技术，能够有效降低服务器宕机率；隔离和混部技术，内核对MPAM进行了完善和优化，使之成为业务易用的混部资源隔离技术；以及全新的硬件IP支持，如PCIE5，DDR5； 等。

应用场景

龙蜥社区围绕ARM生态持续展开研发和探索，同时深度结合ARM架构进行性能优化，完善ARM生态的工具化和CICD支撑体系。目前龙蜥操作系统ARM多芯片的支持，围绕云计算在公有云，电商，大数据，云原生，混部，存储，数据库，混合云等业务领域获得了广泛应用。

5.1.3 龙芯自主指令级的支持

硬件

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等已经研发成功并量产。AnolisOS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。

关键技术

内核支持

基于4.19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处理器的支持和相关配套桥片7A1000、7A2000的支持。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是目前最流行的虚拟化技术，它基于内核提供的kvm模块，结构精简，性能损失小。AnolisOS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QEMU6龙芯平台的优化、支持代码已经合入AnolisOS主线分支。此次改动除专注通用优化外，还支持7A2000桥片iommu功能；支持加解密sec模块直通虚拟机技术。

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AnolisOS社区主线分支。这些改动除专注通用性优化，如GC NUMA和编译策略优化外，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。

产业链

AnolisOS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%。

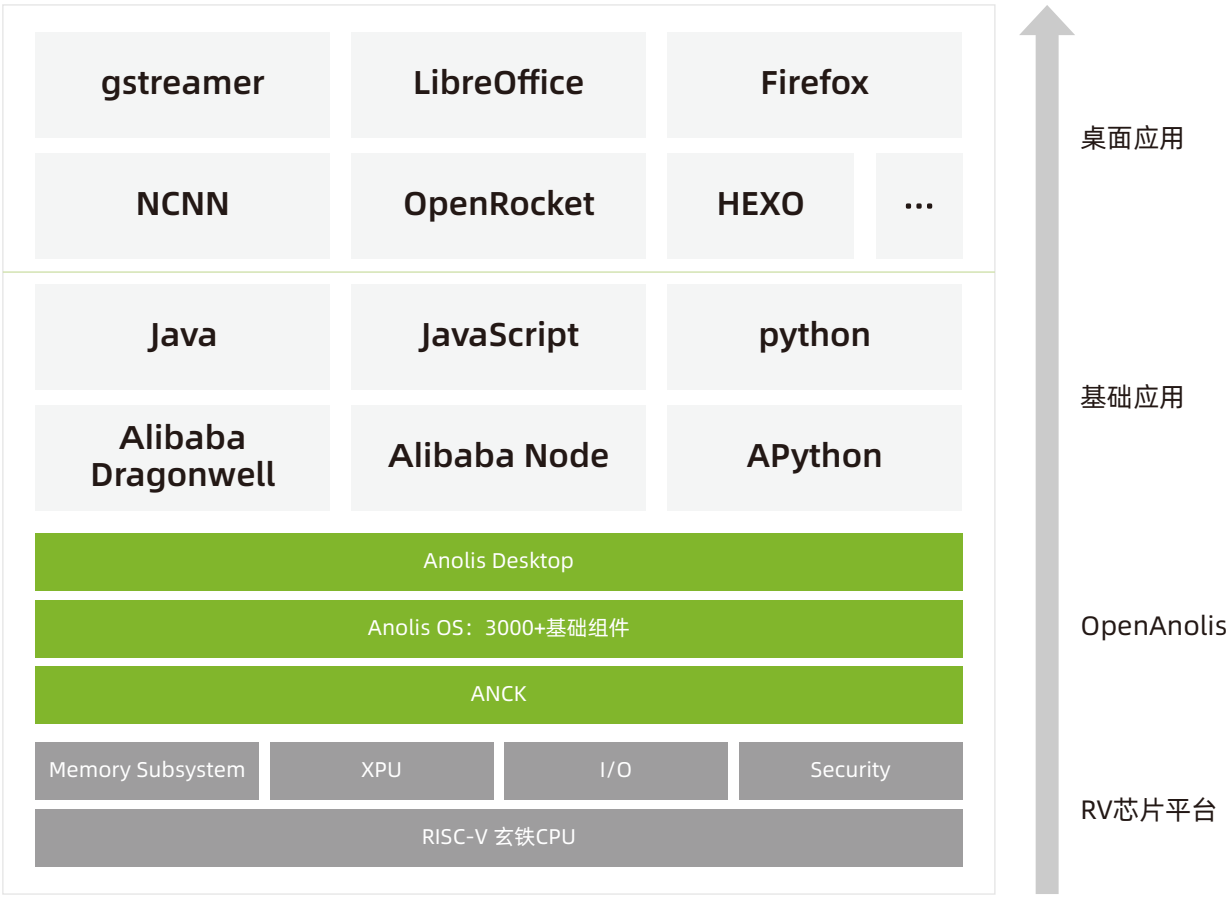
5.1.4 开源硬件RISC-V支持

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，于Arm和x86有很大不同，因此也受到了业内人士的普遍关注，再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。

关键技术

龙蜥社区RISC-V SIG秉承RISC-V共享共治的模式，与PTG、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。



龙蜥操作系统5.10内核全面支持RISC-V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。再生产力应用方面，支持了JAVA、Python、NodeJS等主流语言。

除此之外，龙蜥社区还完成了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，龙蜥社区已经联合平头哥、统信软件、中科院软件所PLCT实验室共同打造了围绕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。

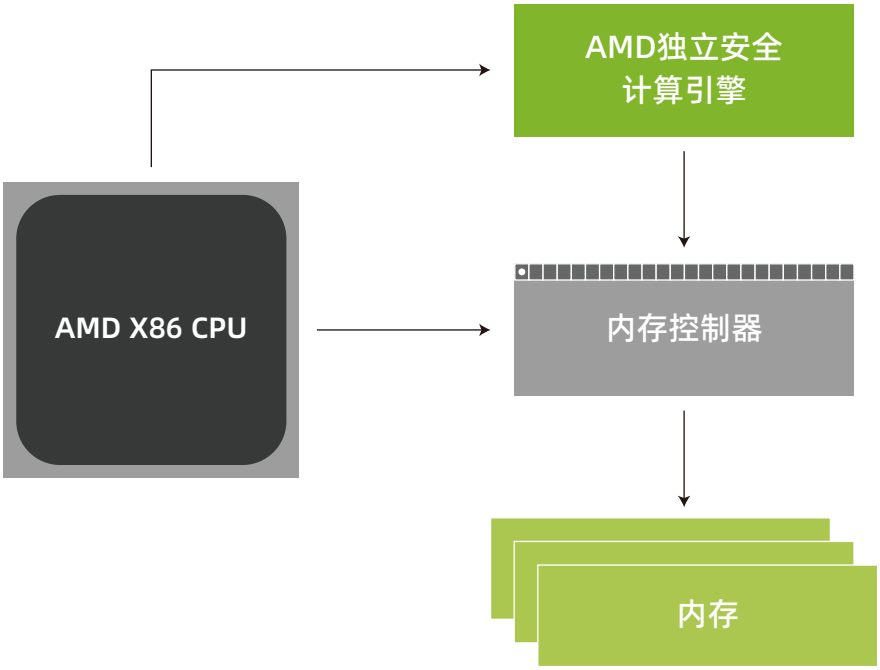
应用场景

目前，龙蜥社区提供的RISC-V Anolis已经支持平头哥高性能RISC-V平台：曳影1520，能够在曳影1520上流畅运行Anolis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。

5.1.5 AMD安全虚拟化的增强

AMD SEV提供了硬件级的内存加密方案，用以实现安全加密的虚拟化：

- 内存控制器中集成了AES-128硬件加速引擎：客户操作系统通过页表选择要加密的页，对终端用户的应用程序没有任何改变。
- AMD安全内存加密（SME）：所有内存由单一的密钥进行加密，仅仅在BIOS中开启就可以实现（TSME）。
- AMD安全加密虚拟化（SEV）：每台虚拟机都会被分配自己的独立加密密钥，宿主机和客户虚拟机之间相互加密隔离。



AMD的TEE硬件方案最新的SEV-SNP以现有的SEV和SEV-ES功能为基础，在提供对用户VM内存和CPU寄存器加密的同时增加了新的基于硬件的内存完整性保护，防止如数据重放、内存重映射等恶意攻击. 此外,SEV-SNP还引入了额外的可选安全增强功能，旨在支持更多的虚拟机使用模式，提供针对中断行为的增强保护，并对最近越来越多的侧通道攻击提供更强的防御能力。
AMD的SEV和SEV-ES功能已经完整支持Anolis OS 8.6。用户不需对自己的业务进行任何改动，只需要根据云原生机密计算SIG的文档配置好支持SEV和SEV-ES的虚拟机就可以实现安全加密的虚拟化, 同时预计年底前基于机密容器的方案也会支持Anolis OS。

5.2 软硬件协同

5.2.1 面向DPU场景的软硬协同协议栈

背景概述

随着高性能计算、机器学习和大数据等技术的广泛使用，对于云VPC、数据中心内的网络提出了更加苛刻的要求。此时传统的以太网卡和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。与此同时云、硬件厂商提供了高性能DPU解决方案，因此需要一个高性能的软硬协同网络协议栈，对下适配DPU并充分发挥硬件性能，对上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。

技术方案

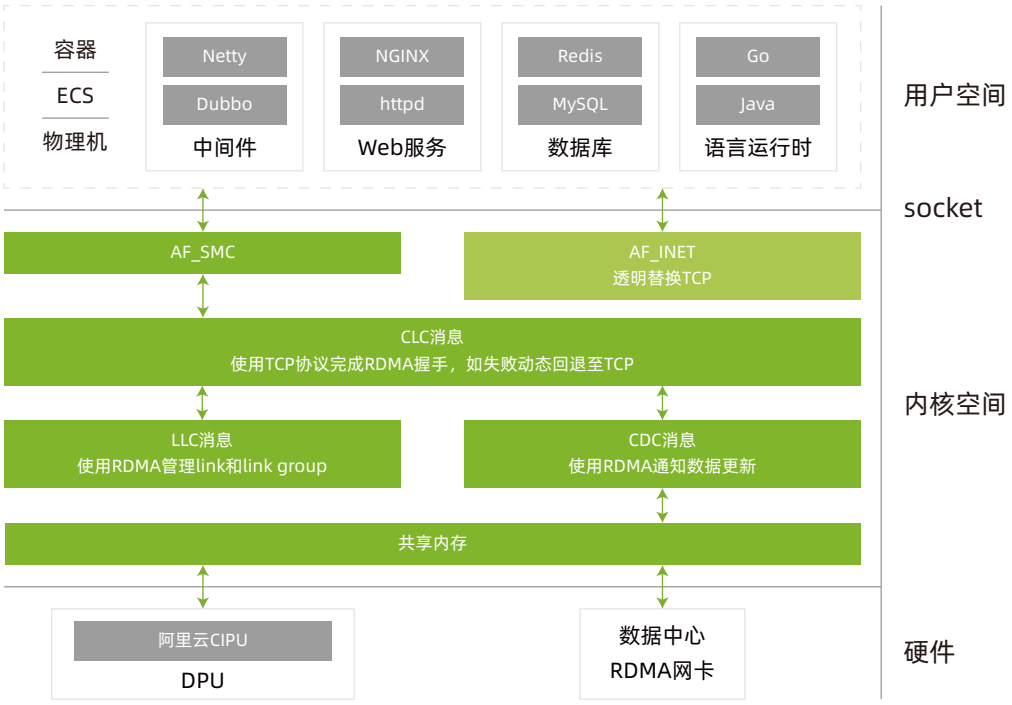
共享内存通信SMC是由IBM首次贡献至Linux社区，并由龙蜥增强和维护的软硬协同的高性能协议栈。针对不同的规模场景、硬件和应用模型，SMC提供多位一体的方案以解决当前传统协议栈的问题：

（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；



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；

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%；

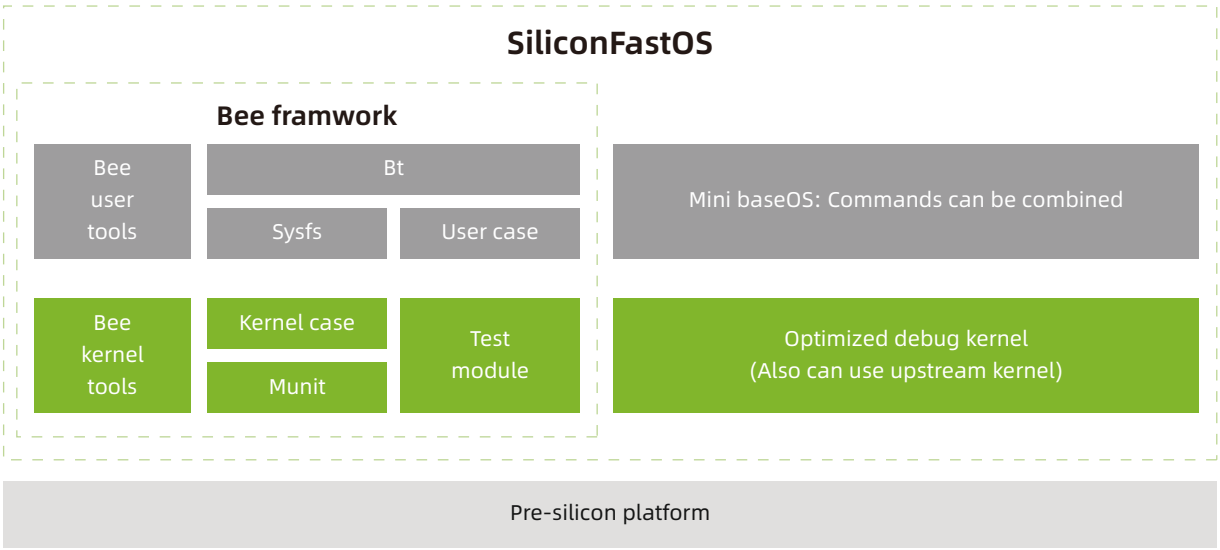
5.2.2 面向芯片研发和验证的操作系统SiliconFastOS

背景介绍

随着ARM和RISC-V芯片研发的大爆发，越来越多的芯片设计、制造厂商甚至是云厂商都开始参与到芯片领域中来。面对芯片使用者尤其是处理器芯片与日俱增的需求，且芯片研发研发和流片成本高，芯片制造厂商必须要提高芯片质量，并缩短芯片研发周期，迅速对市场作出反应，实现产品的快速迭代。借着ARM和RISC-V架构更加开放的生态，业界陆续提出了了芯片敏捷开发方案。

技术方案

芯片验证时，软件运行在FPGA平台上。具有运行速度慢、硬件变化快且没有ROM等特点。造成了在基于OS的芯片验证过程中OS启动速度慢、命令行工具不够丰富、却少芯片验证环境、灵活性差和使用成本高等困难。为了解决这些问题，龙蜥社区开发了芯片验证操作系统SiliconFastOS。SiliconFastOS具有OS启动加速、丰富的命令行工具、原生验证环境、灵活的配置手段和开箱即用等特点，可以帮助提高芯片验证效率，缩短芯片研发周期。



1. 最小化rootfs，SiliconFastOS尽可能去掉在开发过程中不会使用到的命令，从而减少内核解压rootfs和创建页表的时间。此外，SiliconFastOS内建build system，支持使用者选择单个软件包中的单个命令来构建最终的rootfs，可以在尽可能减小rootfs大小的前提下，提供丰富的命令行工具，兼顾启动速度和rootfs镜像大小。

2. 裁剪内核。SiliconFastOS去掉了在芯片验证阶段不会使用的内核驱动，裁剪内核镜像的大小。SiliconFastOS还使用了PCIe delay等方式进一步缩短OS。

3. 原生芯片验证环境。SiliconFastOS内部提供了轻量级内核异步单元测试框架Munit，帮助开发，管理、运行大量的内核单元测试用例。还提供了芯片验证测试框架Bee，可以和Munit搭配使用，同时管理内核和用户态的测试用例。此外，还包含了丰富的芯片验证工具集，帮助开发者快速定位和修复软硬件bug。

4. 一键编译，开箱即用。SiliconFastOS内部使用Kbuild搭建，和linux内核一脉相承，开发者可以快速上手，利用Kbuild灵活配置rootfs软件包和内核，一键生成最终OS镜像。

技术优势

基于OS的端到端芯片验证系统

传统的基于固件和硬件的芯片验证方案在芯片级仿真、调试、硬件固件交互方面有其特有的优势，但随着软硬件融合程度加深，需要更多的端到端验证方案来验证硬件设计对上层应用的影响。相比于固件和硬件方案，基于OS的芯片验证系统在灵活性、软硬件协同、对端到端真实场景的模拟、还有工具生态丰富度上面都有其得天独厚的优势。在此基础上，SiliconFastOS还提供了Bee + Munit组成的原生芯片验证环境，帮助使用者开发，管理和运行大量的端到端测试用例。使用SiliconFastOS + Bee + Munit的组合方案，可以帮助芯片在流片前评估硬件设计对上层应用的影响，大大降低了各种软硬件协同设计、硬件加速器和异构硬件设计的流片风险。

应用场景

Sliconfastos（开源链接：alibaba/SiliconFastOS (github.com)，在倚天710上得到了很好的应用，作为端到端的验证系统与硬件验证协同配合，从而进一步提高了倚天710的研发效率。Sliconfastos作为通用的芯片验证系统解决方案，可以有效帮助提高芯片验证的研发效率，目前也支持了RISCV架构，可以在不同的平台和架构的芯片验证平台（如：仿真，FPGA）中使用。

5.3 内核技术

5.3.1 跨云-边-端的只读文件系统EROFS

背景概述

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。

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的Nydus镜像服务深度融合，打造了RAFS v6、FS-Cache等技术，服务容器runC、Kata等场景，未来还将发力page cache内存去重进一步提供内存超卖能力。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：

（1）Linux内核原生，通过压缩，块去重，字节滚动压缩去重节省镜像存储空间；

（2）原地解压等技术进一步优化运行态内存占用，提升性能；

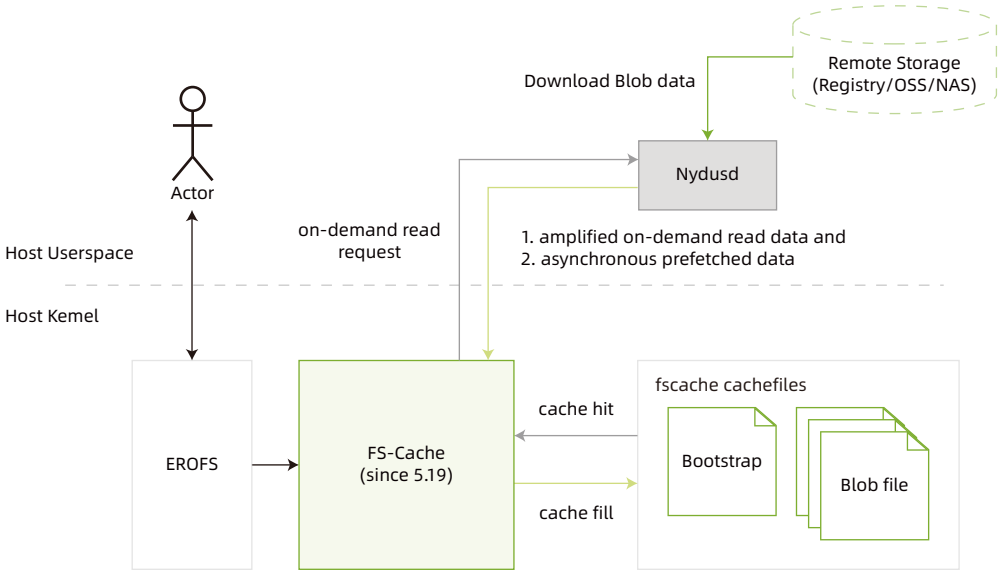
（3）提供内核原生按需加载能力，从源头解决FUSE额外拷贝和上下文切换开销。

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。

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：

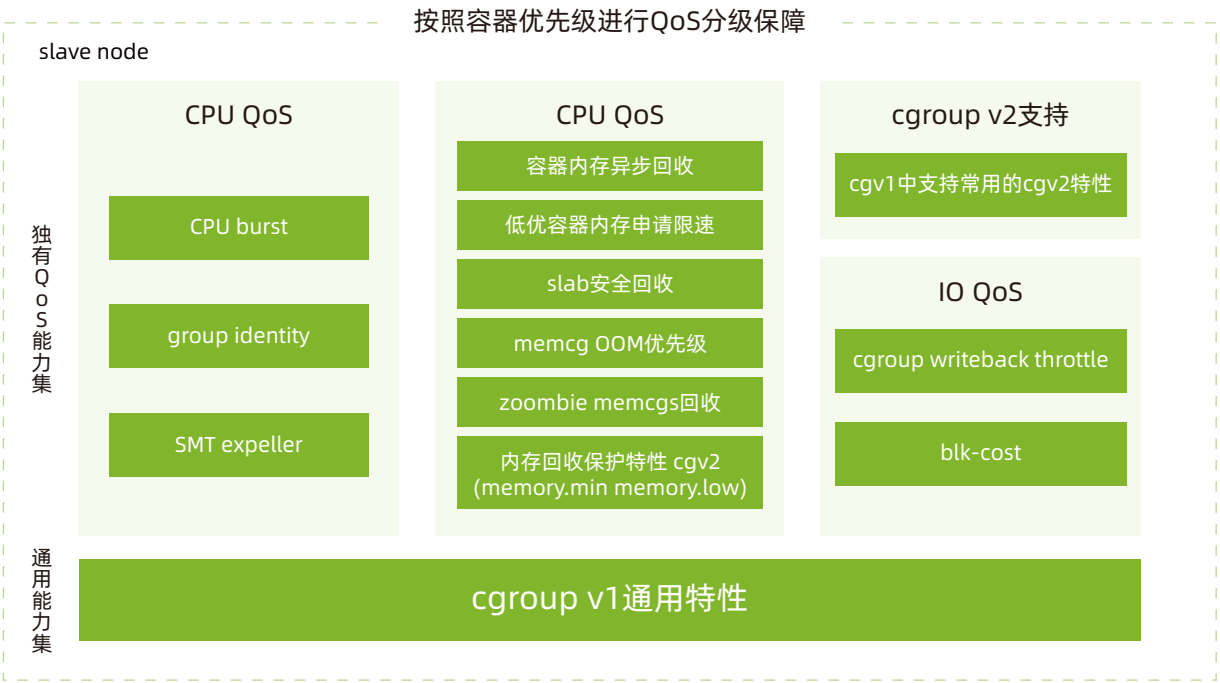
	OCI	EROFS + FUSE	EROFS + FS-Cache
本地cache 4K顺序读	387068KB/s	211767 KB/s	366291KB/s
本地cache 4K随机读	6153KB/s	5450KB/s	6170KB/s

5.3.2 资源隔离技术

技术方案简介

混部就是将不同类型的业务在同一台机器上混合部署起来，让它们共享机器上的CPU、内存、IO等资源，目的就是最大限度地提高资源利用率，从而降低采购和运营等成本。混部通常是将不同优先级的任务混合在一起，例如高优先的实时任务(对时延敏感，资源消耗低；称为在线)和低优先级批处理任务(对时延不敏感，资源消耗高；称为离线)，当高优先级业务需要资源时，低优先级任务需要立即归还，并且低优先级任务的运行不能对高优先级任务造成明显干扰。

为了满足混部的需求，在单机维度的内核资源隔离技术是最为关键的一项技术，龙蜥云内核在资源隔离的技术上深耕多年，并且在行业中处于领先地位，这些内核资源隔离技术主要涉及内核中的调度、内存和IO这三大子系统，并且在各个子系统领域根据云原生的混部场景进行了深入的改造和优化，关键优化包括但不限于：cpu group identity技术，SMT expeller技术，基于cgroup的内存异步回收技术等。这些关键的技术使客户有能力在云原生混部场景中根据业务特点给出最优解决方案，可有效提高用户的资源使用率并最终降低用户资源的使用成本，非常适用于容器云混部场景，同时也是大规模化混合部署方案所强依赖的关键技术。



规模化部署和收益

目前龙蜥OS的资源隔离技术已经在手机制造企业、互联网企业、大型国企，以及阿里云、蚂蚁集团内部规模化应用。从以往众多用户的部署经验来看，在保障客户的QoS的前提下CPU的使用率可以达到30%~50%，比如龙蜥和 Koordinator 开源混部技术带来的红利，帮助阿里巴巴实现规模超千万核的云原生混部，混部CPU利用率超50%，帮助2021年“双11”计算成本下降50%，为客户降本增效提供了良好的技术保障和售后服务。

技术竞争力

目前龙蜥社区的资源隔离方案，不仅仅在互联网、云计算场景得到规模化的应用，近年也在一些知名企业的私有云场景得到广泛应用。并且，这一开源方案会持续在社区演进，形成最佳云原生实践，作为企业降本增效的重要手段之一，继续服务企业私有云建设，以及阿里云的客户。龙蜥在资源隔离这块不管从源码透明度，还是从技术的深度，以及场景的广度都是用户第一选择。

5.3.3 数据库/JAVA等高性能场景中的内存优化

背景概述

在处理器内存缓存层级结构中，iTLB miss性能指标对访存优化至关重要，并且在ARM平台上优化效果更为明显。

在数据库/JAVA 等高性能场景中，iTLB miss可以成为影响性能的主要因素，我们通过实验观察到iTLB miss引入的CPU停顿时间最高占任务运行时间的~13%。优化iTLB miss的手段很多，主要分为两类。一类是优化代码段布局，例如hfsort/gold linker、BOLT、PGO，缺点是不通用；一类是使用大页映射代码段，例如静态大页（hugetlbfs)、共享内存大页（shmem)，缺点是调试信息缺失，需要额外运维等。

通用透明的方案需要基于文件透明大页来实现。社区Linux内核从5.4合入READ_ONLY_THP_FOR_FS特性，支持普通二进制文件的代码段部分映射文件透明大页；并通过写文件时清空文件缓存来规避写文件透明大页的问题。但仍有如下两个缺点。

- 应用程序需要主动通过madvise系统调用来使能代码段映射文件透明大页；
- 共享库、PIC/PIE（位置无关二进制文件）代码段的映射地址通常不能2M对齐，导致不能映射文件透明大页。

技术方案：透明代码大页（Hugetext）

我们给出透明代码大页的方案（Hugetext），提出四点优化和改进。如图5.3.3-1所示。

1. 检测可执行文件加载/映射，分配地址2M对齐，自动使能普通二进制、共享库、PIC/PIE（位置无关二进制文件）的代码段映射文件透明大页；
2. 检测匿名可执行代码 (例如JAVA code cache)，提供开关自动映射匿名透明大页；
3. 相比普通透明大页，内核khugepaged线程优先整理可执行文件透明大页，达到加速效果；
4. 对于大小不足2M的代码段，通过补齐映射地址空间，增加文件透明大页的覆盖率。

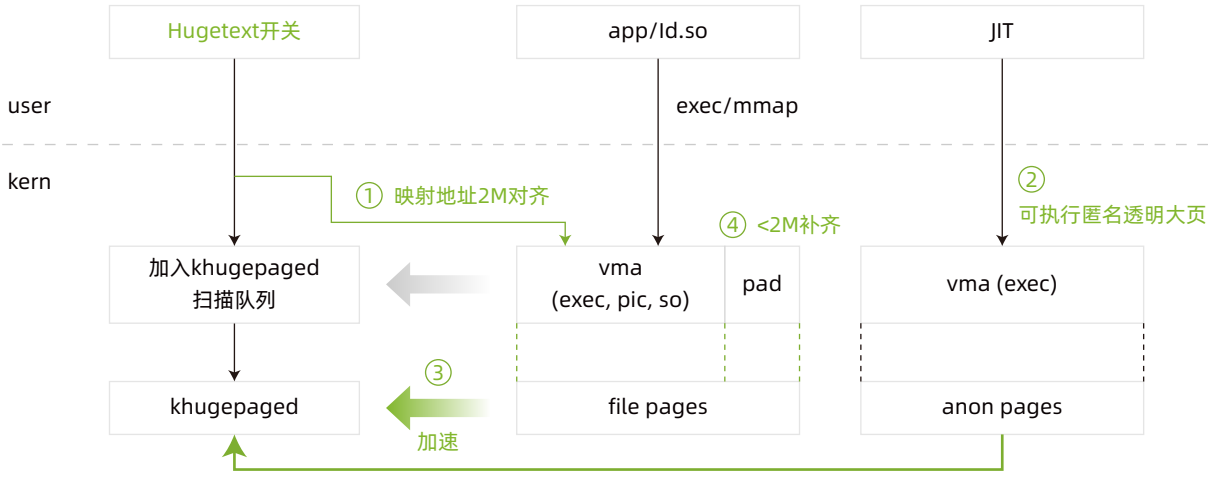


图 5.3.3-1：Hugetext 的整体架构和主要优化（绿色标记）

应用场景及性能收益

本地测试中，某ARM平台上数据库类业务（例如MySQL），Hugetext可以提升性能5-12%；某ARM平台上JAVA类业务（例如Spring），Hugetext可以提升性能4-13%。

真实业务场景中，例如某ARM平台上MySQL/PostgreSQL业务，Hugetext可以带来6+%的端到端性能提升。

Hugetext特性也在Linux内核社区和Glibc社区回馈了开源补丁，增强了稳定性，修正了代码段地址映射问题。

5.3.4 跨处理器节点内存访问优化

背景概述

在新平台多节点大内存的趋势背景下，打开NUMA是必要的性能手段。随之而来的问题是，跨NUMA访问会引入性能开销。业务一般配合用户态任务调度，利用绑核等手段规避跨NUMA访问。但文件页跨节点访问不能很好解决。其中，代码段文件页跨节点访问性能影响比较明显，对于数据库/存储等业务来说，甚至成为性能瓶颈；该性能影响在ARM平台上更为明显。

现有的内核接口（例如NUMA Balancing）、用户态工具都不能很好地解决代码段的跨节点访问。

技术方案：代码多副本（Duptext）

我们给出代码多副本方案（Duptext），执行流程如图5.3.4-1所示。

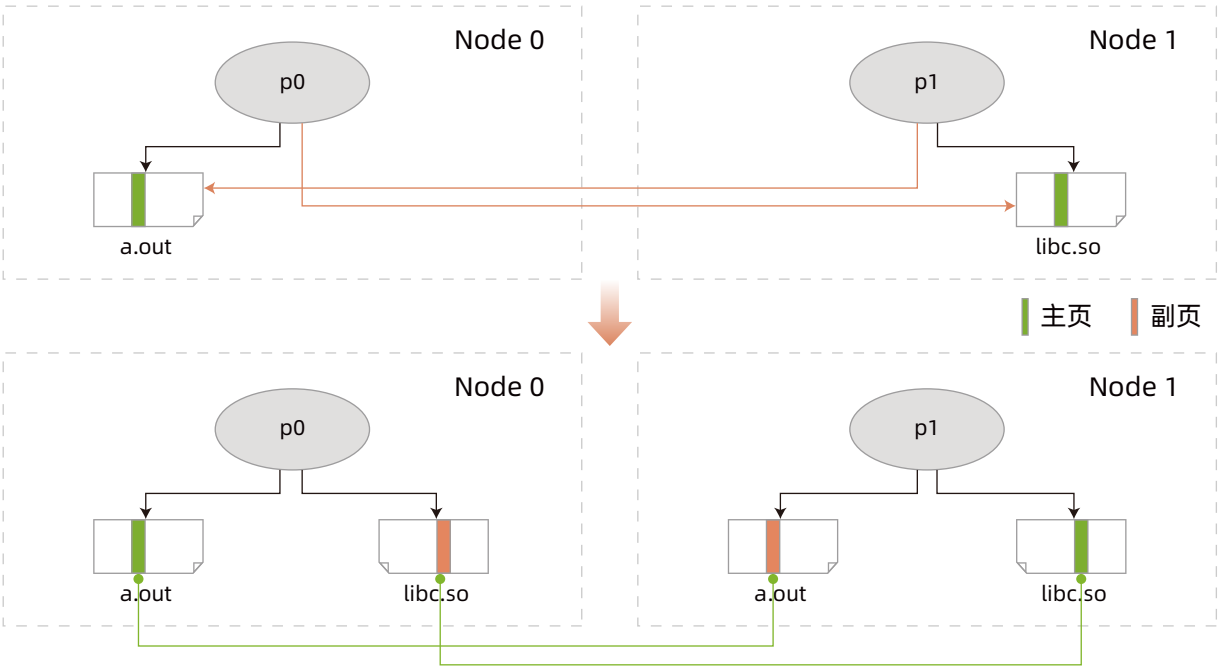


图 5.3.4-1：Duptext整体流程示意图

Duptext主动检测代码段跨节点访问。在文件缺页以及主动映射流程中，检查当前需要映射的可执行文件页所属节点和当前进程运行节点是否一致。如果不一致，则在本地同步创建副本，并用副本建立此次映射。

代码副本按需创建，在每个节点上通过基数树管理。考虑到代码段通常体积较小，Duptext引入的内存开销可控，利用空间换取时间。同时，Duptext提供整机粒度和Memcg粒度的开关，支持重点应用使能代码副本，支持整机回退，稳定性得到保障。

应用场景及性能收益

本地测试中，例如某ARM平台上MySQL代码段跨节点访问带来的性能下降可以达到-3%（无背景压力）~ -22%（有背景压力），应Duptext之后，MySQL端到端性能都能达到本地访问的性能基线。

真实业务场景中，例如某ARM平台上分布式块存储系统业务，Duptext可以带来最高16%端到端性能优化（性能基线为默认状态下代码段跨节点的性能）。

5.3.5 敏捷开发场景下的调度器热升级SDK

背景概述

不同的应用程序，通常需要不同的调度策略来优化性能。而内核发布周期很长，升级内核的成本通常较高，优化无法快速规模化部署。并且针对特定应用的调度器优化，常常造成其他场景的性能回退，发生问题也难以回滚。通过传统热修复技术，可以在不升级内核的情况下，实现内核局部更新和优化，针对性提升一些应用的性能。但传统技术无法实现整个子系统升级，不支持大型调度特性，停机时间长。而调度器热升级技术解决了以上问题。

技术方案

调度器热升级SDK通过模块化、数据重建、热替换等技术，实现调度器研发、测试、上线、维护的敏捷化和定制化。其中，模块化技术自动地从内核中解耦出调度器模块代码，面向内核开发者提供敏捷开发的SDK；热替换技术使得管理员可在毫秒级downtime内部署；数据重建技术将数据状态从升级前的调度器迁移到升级后的调度器。通过这些技术，可以实现定制化调度器，从而解决不同应用和负载需要不同调度器的问题，并实现生产可用。软件架构如下图所示：

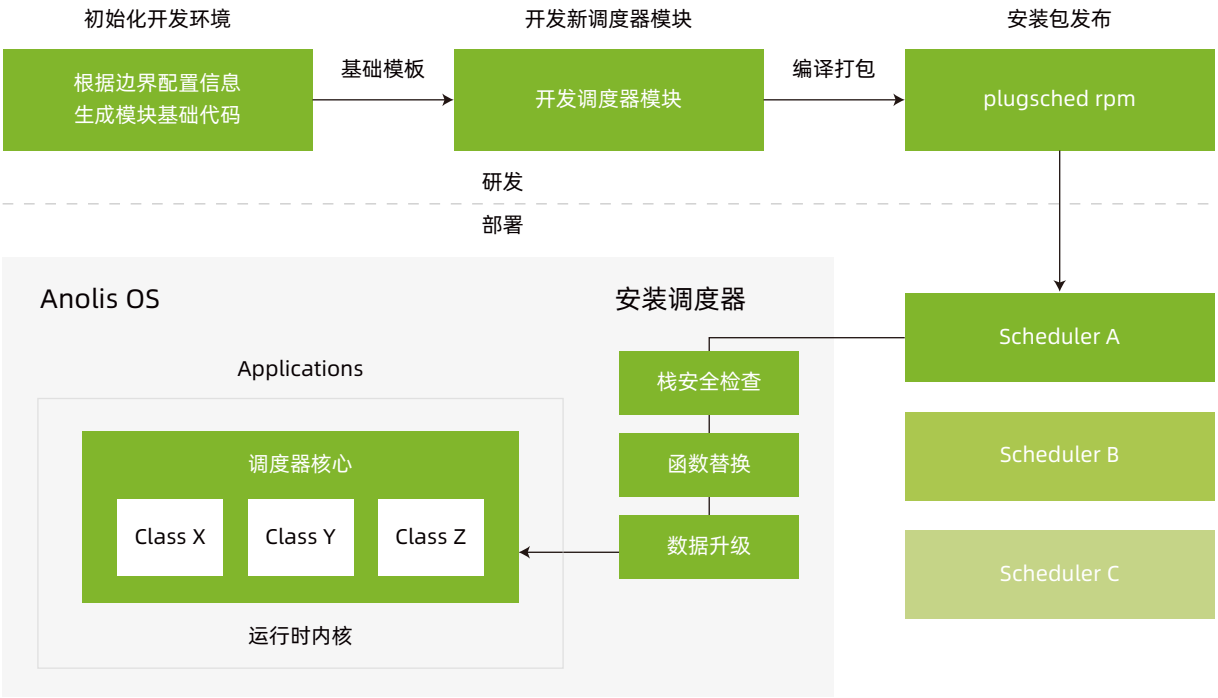


图 调度器热升级技术架构

方案兼容多架构、内核版本。已测试通过的包括aarch64、x86-64架构；4.19、5.10系列内核。其余版本正在测试支持中。方案同时支持各种调度器特性，经测试验证通过的包括以下：微型调度器、物理核调度器（Core Scheduling）、删除限流功能（CFS bandwidth control）、龙蜥CPU混部特性等、上游Linux社区的各种bugfix。

应用场景及性能收益

调度器热升级SDK适合于下面几种场景，这些场景均已得到验证。

- 架构上有针对特殊硬件、应用、负载来定制化调度器的需求。
- 管理员不能切换内核，或周期很长；或由于社区运作停止内核版发布和维护。
- 内核开发者需要敏捷且可回滚地开发测试上线调度器。

某云Serverless服务，通过调度器热升级SDK，将Linux上游物理核调度器特性，以及基于之上自研的算力稳定技术，安装到较老的ANCK 4.19内核的系统上。最后通过这一优化，他们的客户实例减少了约10%的P99延迟，并降低了性能的抖动，还大幅降低了启动时间。验证了调度器热升级方便支持大型特性以及扩展研发的能力。

某云容器服务，基于龙蜥操作系统为客户提供混部技术，但部分客户同时也使用CentOS系统，他们发现使用CentOS的客户无法享受到龙蜥操作系统同等的混部能力。因此通过调度器热升级SDK，将龙蜥内核的CPU混部技术安装到CentOS内核，让使用CentOS的客户也能使用龙蜥操作系统的CPU混部技术。

某互联网金融服务公司，利用调度器热升级SDK，敏捷地将他们自研的调度器优化安装到他们的核心业务中。包括了Linux CFS调度器和ANCK的CPU资源隔离技术的优化。稳定减少了5%的CPU资源浪费，同时降低业务rt。上线过程中，在40,000线程的环境中，停机时间小于12ms。最终优化效果得到运维人员的认可，调度器热升级SDK的易用性也得到研发人员的认可，希望继续使用调度器热升级SDK进行系统优化。

5.4 编程语言

5.4.1 C++编译器和基础库

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）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。

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 Bolt性能优化

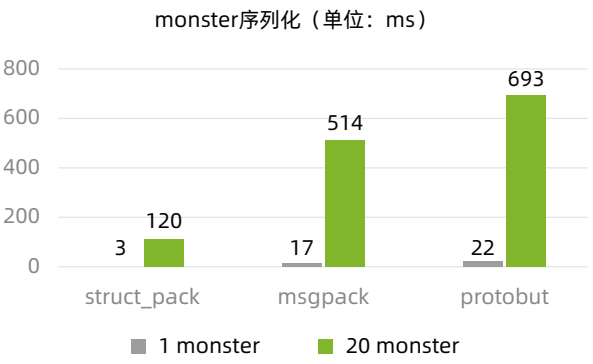
在某存储平台中启用Bolt性能优化后，4K顺序写场景IOPS可提升8%，延时降低7.4%，优化效果明显。

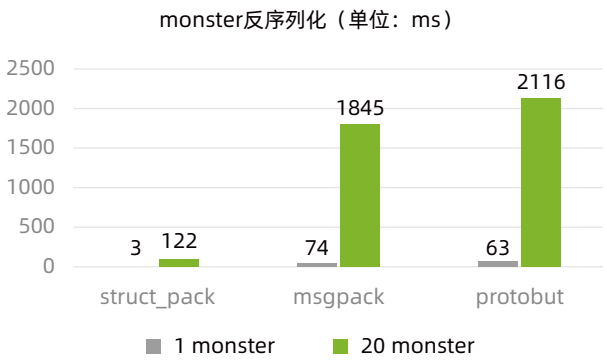
• async_simple协程库

某计算平台和搜索平台将同步改造成协程异步之，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

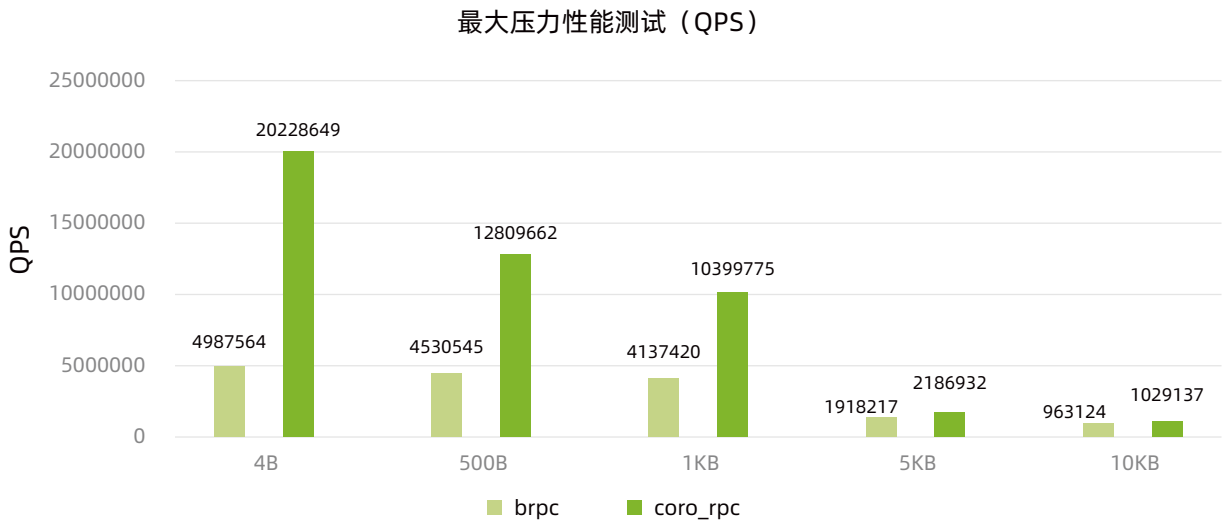
用户只需要一行代码就可以用struct_pack完成序列化和反序列化。相比于protobuf等开源库需要定义proto文件和生成代码的方式，大大简化了使用序列化库的使用方式。同时，相比protobuf性能提升7-20倍。





• 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。



5.4.2 Alibaba Dragonwell

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。

Alibaba 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。 Alibaba 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。

社区维护	质量体系	平台支持
• 稳定的LTS发布 • 安全补丁	• Adoptium CI • 线上验证 • 兼容性测试 • SVT	• Linux, Linux-Apline • Windows (experimental) • X64, AArch64 • RISC-V (coming soon)

Alibaba Dragonwell 8/11/(17)发行版

OpenJDK | Dragonwell增强（RAS、协程...）

Adoptium CI
Anolis OS

构建 | 测试 | 线上验证 | 发布

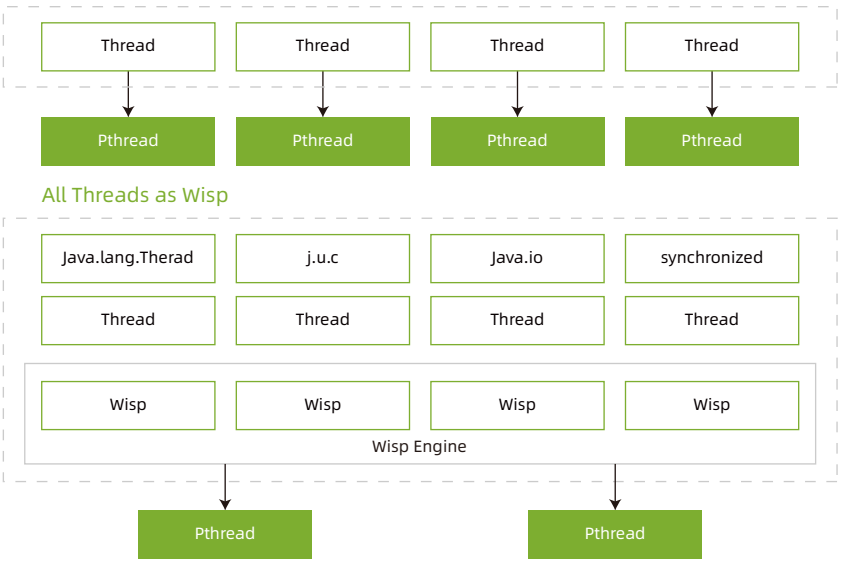
开源社区共建

GraalVM | OpenJDK | OpenAnolis 龙蜥社区 | ADOPTIUM

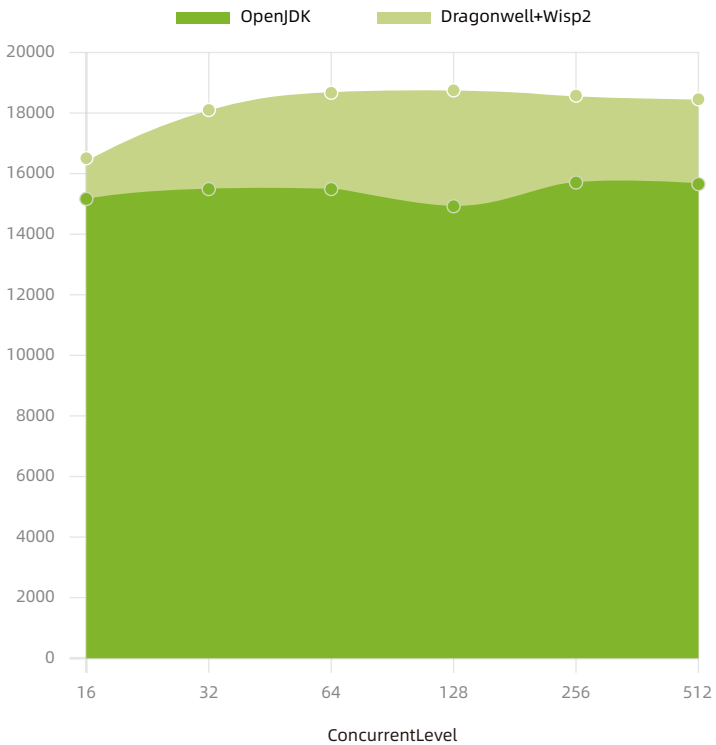
Wisp协程

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Alibaba 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。



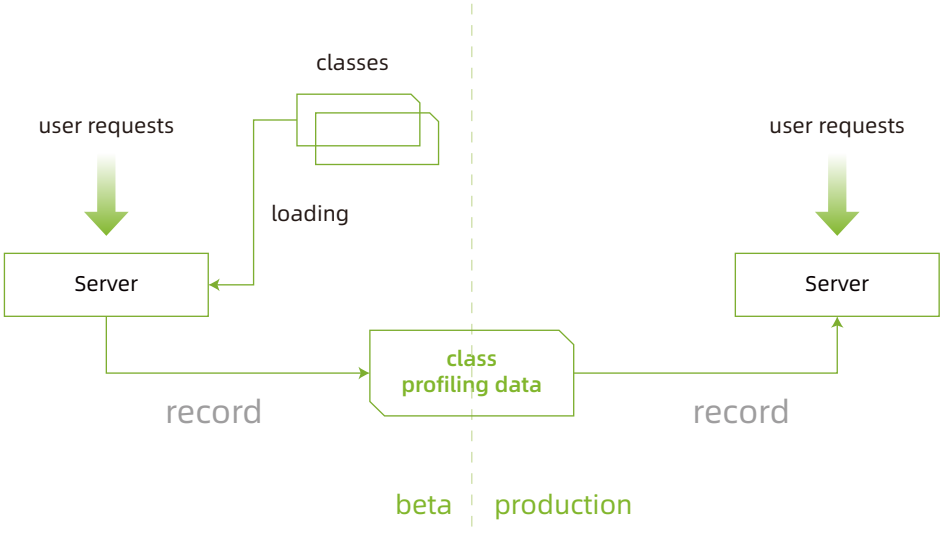
开启Wisp后，应用程序无需任何修改就可以获得性能提升。以下是在Framework Benchmark的Spring实现下，开启协程和关闭协程的性能比较。



JWarmup编译预热

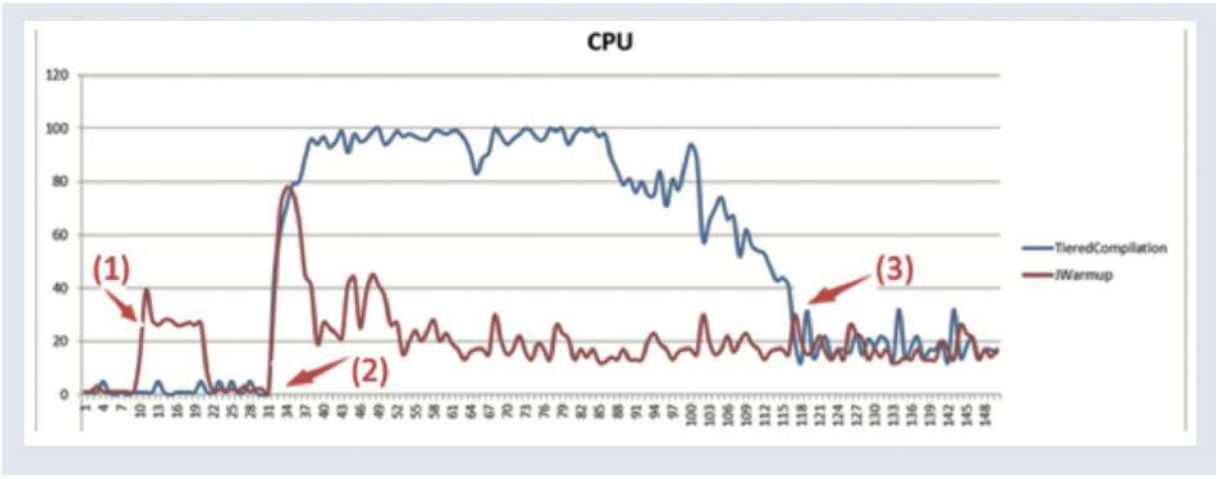
Java拥有即时编译特性，代码以解释器执行为基础，当方法执行足够多次数成为热方法后会触发Just In Time（JIT）编译，性能得到数十倍提升。但应用程序有一个预热的过程，通常需要运行一段时间才能达到峰值性能。

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JWarmup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：

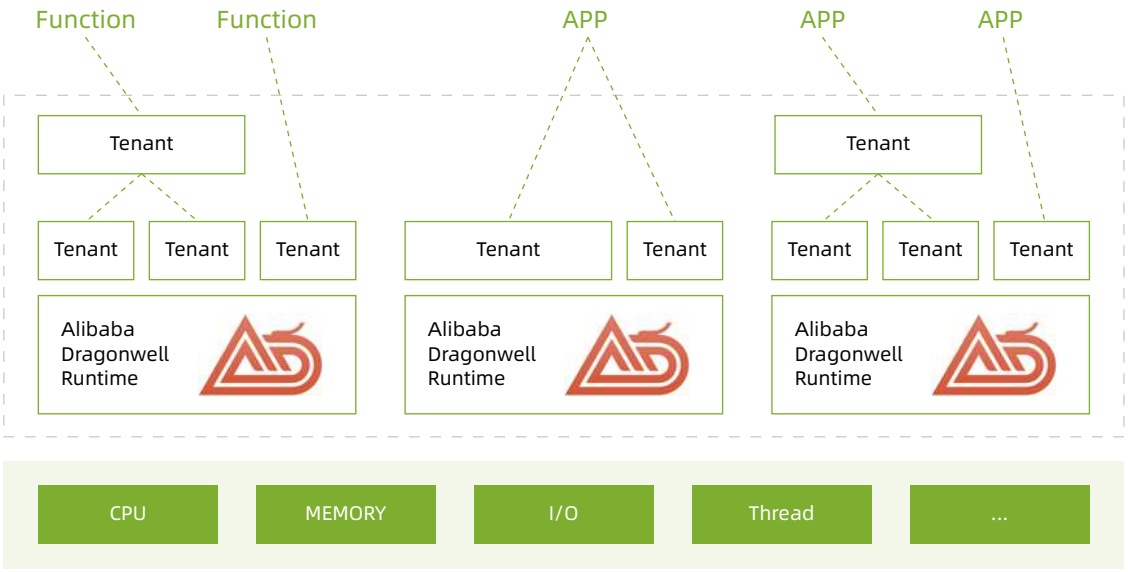
在(1) 的时间点过后，JWarmup提前编译方法，相对于正常蓝线情况，红线使用的CPU上升。

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。

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，在他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。

Alibaba 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。

生产就绪ZGC及弹性堆

Alibaba 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Alibaba 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Alibaba Dragonwell 11长期支持的质量稳定性：

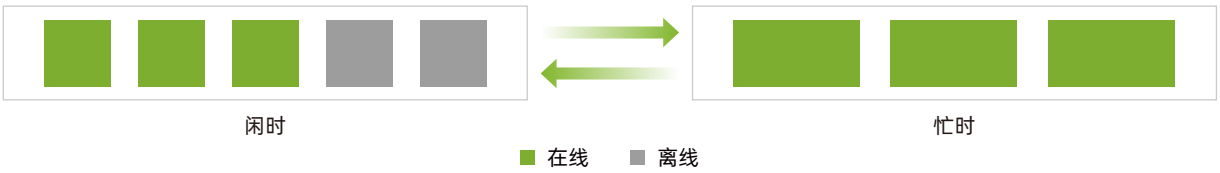
- 系统性移植OpenJDK15 ZGC代码：OpenJDK15（首个支持生产就绪ZGC的OpenJDK上游正式版本）的大部分ZGC相关代码，这些代码完善了ZGC的功能，支持更多的平台，并且修复了重大bug。
- ZGC AArch64平台优化：Alibaba Dragonwell向OpenJDK社区贡献了AArch64平台上ZGC的优化方案，减少内存屏障指令，将Heapothesis benchmark的ZGC吞吐能力提升超过15%。

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

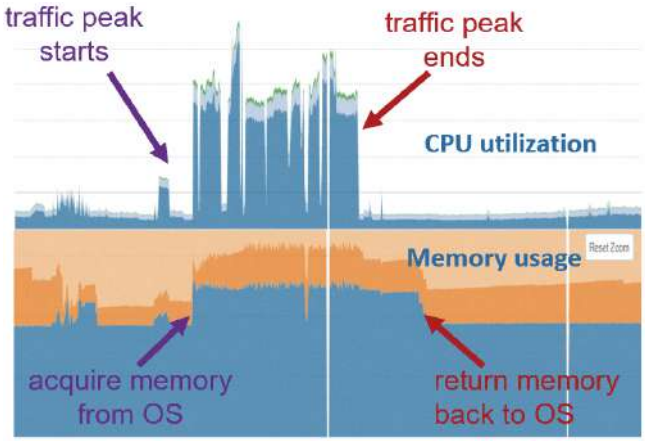
JVM为应用程序提供了自动管理堆内存的能力，包括空闲内存在内。即使应用完全空闲，这些内存也无法被操作系统中其他进程使用。G1ElasticHeap特性支持在G1中动态归还堆物理内存并降低Java进程的内存占用。

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。下图蓝色表示在线任务，绿色表示离线任务。在线服务空闲时通过G1ElasticHeap释放空闲内存，以供离线服务使用。

- 可预测的流量高峰
- 适用的集群部署类型：每台机器上多个在线/离线任务



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



5.5 通用计算

5.5.1 利用io_uring提升数据库系统性能

背景概述

传统的IO软件栈已经无法完全释放出高性能存储设备的性能，高性能IO栈是当前存储领域重点研究的课题之一，代表性的如用户态方案SPDK，以及标准的内核态方案io_uring。

技术方案

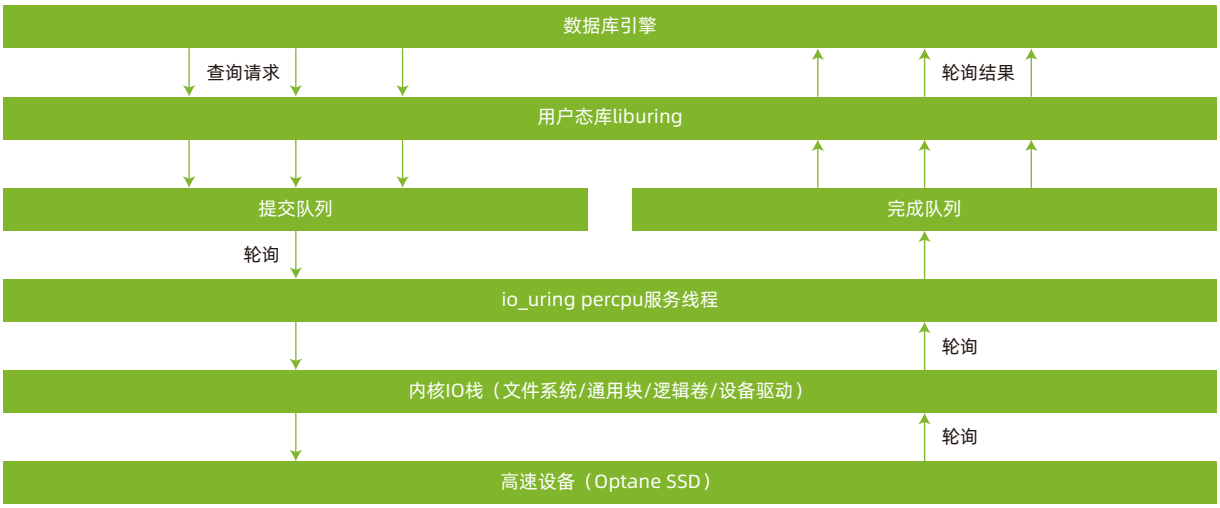
Linux社区从零开始设计一种全新的异步IO框架io_uring。io_uring为了避免在提交和完成事件中的内存拷贝，设计了一对共享的ring buffer用于应用程序和内核之间的通信。该设计带来的好处有：1）提交、完成请求时无需应用和内核之间的内存拷贝；2）使用SQPOLL高级特性时，应用程序无需系统调用；3）无锁操作，用Memory Ordering实现同步等。龙蜥社区自2020年上半年开始参与io_uring社区开发，贡献了多个特性和优化，并在图数据库场景探索容器化部署和针对性优化。

技术优势：应用程序通过统一的标准系统调用来使用io_uring。相比传统的Linux Native AIO，io_uring消除了仅支持Direct IO的限制以及额外的内存拷贝开销；相比用户态框架SPDK，io_uring可复用Linux内核的标准驱动，无需额外的用户态驱动开发，应用场景更通用，编程接口更友好。

该技术特点包括：1）简单易用，方便应用集成；2）可扩展，不仅仅为存储IO使用，同样可以用于网络IO；3）特性丰富，满足所有应用，如支持Buffer IO；4）高效，尤其是针对大部分512字节或4K IO场景；5）可伸缩，满足峰值场景的性能需要等。
应用场景：io_uring可适用于绝大多数对异步IO有诉求的业务和应用。目前，io_uring已在多个主流开源应用中集成，如RocksDB，Netty，QEMU，SPDK，PostgreSQL，MariaDB等。

图数据库引擎iGraph优化实践

图计算服务Graph Compute是阿里云自主研发的高性能分布式图计算产品，支持复杂图关系数据的存储、查询和计算，高效对接图算法与模型，在搜索推荐广告、实时风控、知识图谱、社交网络等场景有着广泛的应用。其内核引擎iGraph在基于磁盘的查询访问场景下引入了io_uring，支持高IOPS下稳定运行。



实践效果：图数据库引擎iGraph经过io_uring适配优化后，线上运行环境在CPU开销不高于原始使用Linux Native AIO版本的前提下，业务端到端时延优化达20%。

5.5.2 面向HTTP 3.0时代的高性能网络协议栈

背景概述

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。

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。

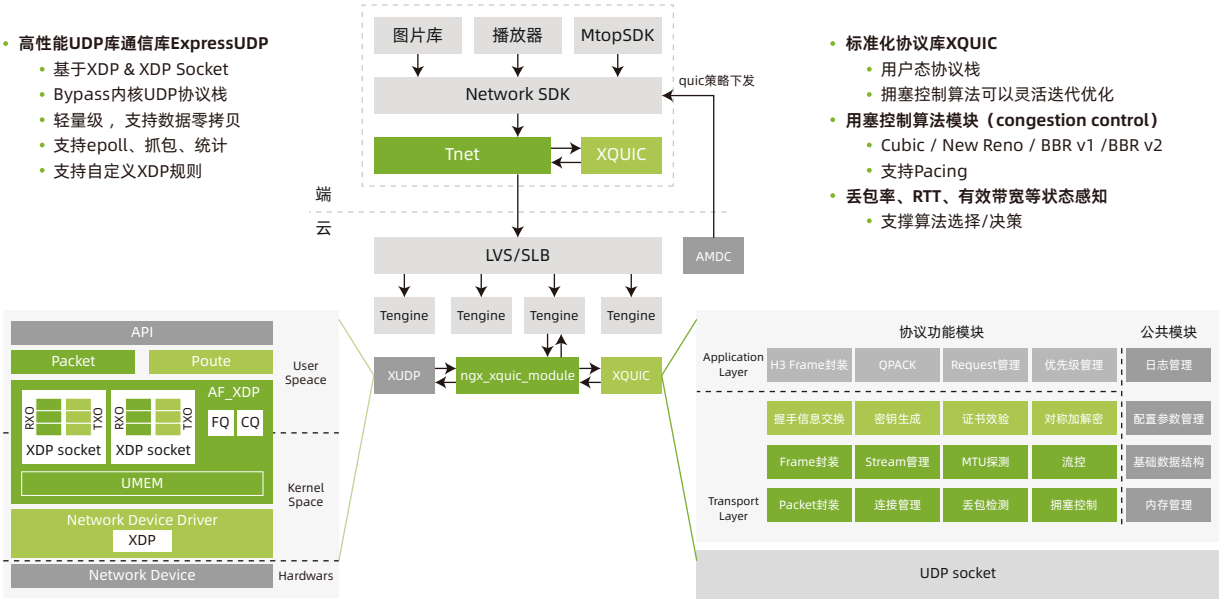


图 XQUIC+ ExpressUDP整体架构和传输框架

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。

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%。

5.5.3 面向异构计算的加速器SDK

背景概述

来自Entrust的调查数据显示，目前至少有50%的公司采用了加解密技术来保护公司信息，这个比例还在逐年上升。然而因为安全漏洞引发的数据泄漏事件还是时有发生，为此网络协议也在逐渐演进，强制使用更安全的算法。比如TLS 1.3中废弃了很多存在安全风险的算法。但是这些更安全的算法一般密钥长度更长，或者计算过程更加复杂，意味着对计算资源的消耗会更加巨大，对CPU的处理能力提出了挑战。追求极致的安全拉低自身业务的处理能力又显得得不偿失，为了解决这个难题，龙蜥社区开源了面向异构计算的加解密加速器SDK。

YCC（Yitian Cryptography Complex）是阿里巴巴针对网络安全领域推出的硬件加速解决方案，以IP的形式集成在倚天710的SoC中，支持IPSEC/TLS等主流网络安全协议，支持对称数据加密（如AES，SM4）、非对称公钥加密（如RSA、ECC、SM2等）和数据完整性（SHA1/2/3，SM3等），加速数据的加解密和数字签名等操作。

技术方案

YCC 全栈SDK包括内核驱动、用户态驱动、硬件抽象层、驱动抽象框架、Engine组成，如下图所示：

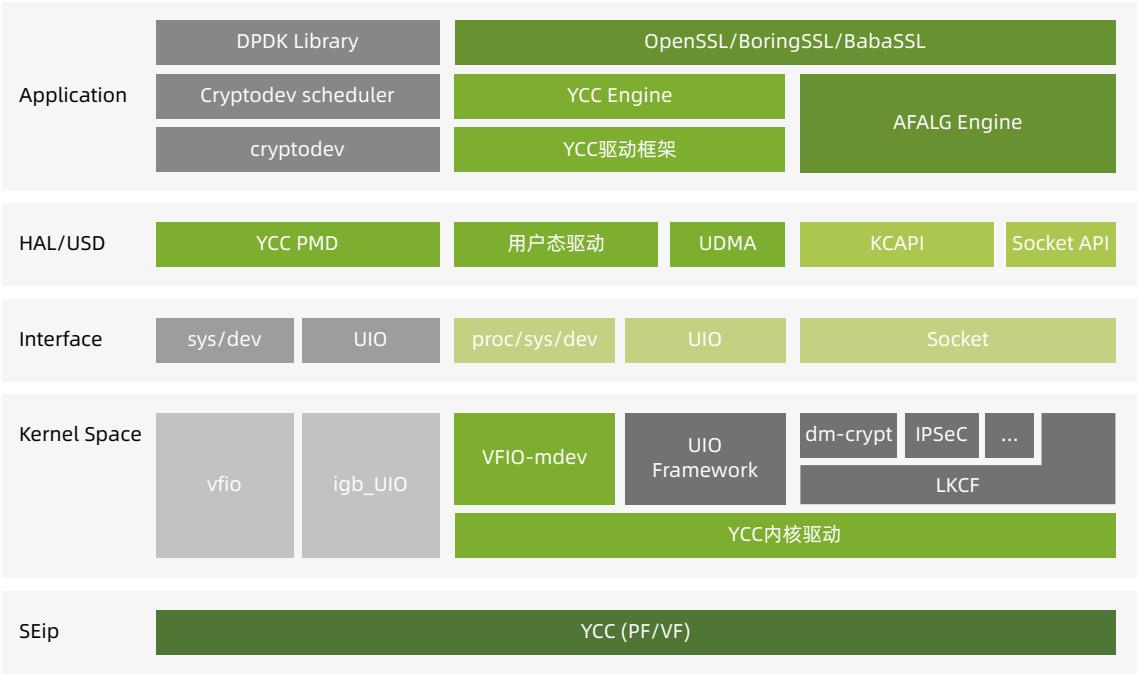


图 倚天加解密加速器YCC SDK分层设计图

下面针对每一个组件，自底而上分别做一些简单的描述：

1. YCC内核驱动。承接来自通用Linux内核LKCF的加解密流量、PF管理、算法注册、虚拟化支持等；
2. YCC用户态驱动。管理UIO框架暴露的硬件环形缓冲区、算法注册、UDMA用户态DMA内存管理；
3. YCC PMD。硬件抽象层，将上层的算法请求转换为硬件定义的命令格式，负责将命令传递给硬件；
4. YCC驱动框架。提供统一的用户态加解密API，对称、非对称、密钥协商、授权加密四种不同的算法场景，不同场景下提供各自的API；
5. YCC Engine。对接OpenSSL、BaBaSSL、BoringSSL等加解密库，承接用户态加解密流量

技术优势

结合YCC加速器和YCC SDK，加解密算法性能得到极大的提升，以国密算法SM4为例，单个YCC性能相对于单颗CPU有18~78倍性能提升。

应用场景

面向异构计算的YCC SDK可以有效发挥YCC硬件加速器的性能，可应用于密码学计算密集型应用，例如Nginx、Tengine等，有效提升Web应用网关https握手和数据加解密性能。针对E2E场景，基于nginx进行性能数据测试，并且和其他产品做了对比。客户端测试采用wrk测试软件测试nginx https短连接的握手能力，加密套件采用ECDHE-RSA-AES256-GCM-SHA384。有了YCC加速器和YCC SDK的加持，整体nginx网关的能力有成倍的性能提升，这可以极大提升nginx/tengine业务的实际收益，帮助业务提高收益降低成本。

5.6 云原生技术

5.6.1 面向云原生生态的操作系统发行版LifseaOS

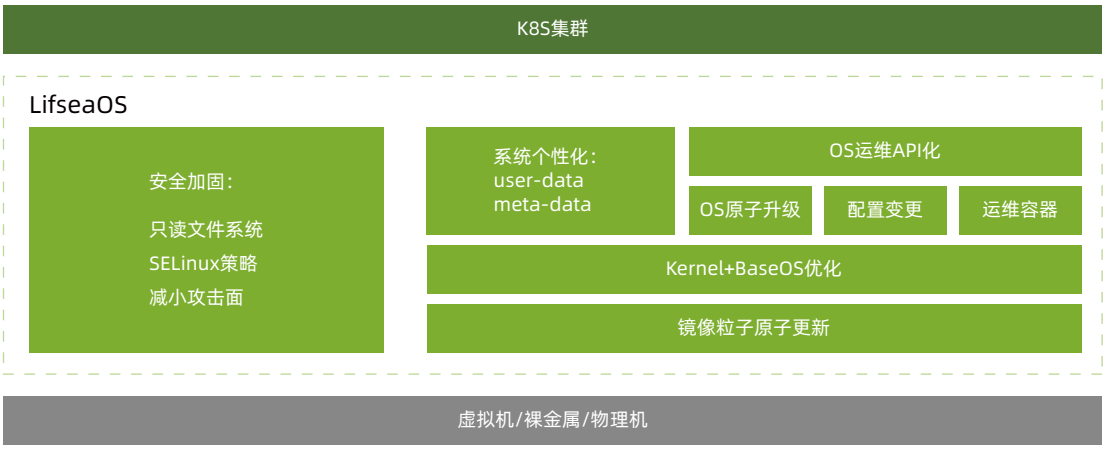
背景概述

近几年，云原生的概念迅猛发展，业务被直接部署到容器中，容器镜像包含了业务运行所需的全部运行时依赖，这使传统操作系统与云原生逐步脱节，并带来了如下问题：系统臃肿、启动速度慢、组件繁多、运维难度大。此外，以RPM包为粒度的管理方式和登录机器进行操作的运维方式极易导致大规模集群下各个节点状态千姿百态，给统一运维带来困难。

技术方案

LifseaOS基于Anolis OS并针对容器场景进行垂直优化，系统中仅保留容器运行所需的必要软件包与系统服务，系统以镜像为粒度进行升级回滚，提供只读根文件系统避免系统被恶意篡改。相比传统操作系统，LifseaOS具有以下关键特性：

- 针对云原生场景优化的内核
- 操作系统镜像版本的原子化升级与回滚，最大程度保证集群内节点状态的一致性
- 安全加固，默认开启SELinux
- 组件与系统服务精简，缩减镜像体积，减小攻击面，加快系统启动速度
- 内置云原生场景常用组件，开箱即用
- 系统提供运维容器，方便用户对系统进行运维操作



应用场景

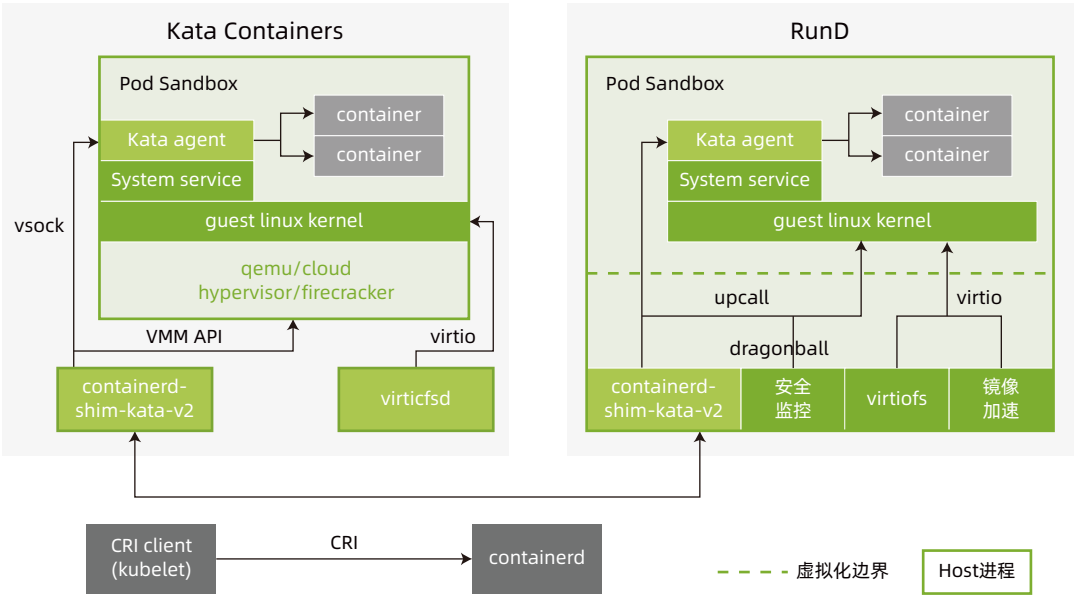
适于作为云上云下容器场景的宿主机操作系统（HostOS）与诸如Kata Containers等安全容器的客户操作系统（GuestOS）。LifseaOS已经在云服务商云原生服务业务上大规模部署，集群千节点扩容时间小于一分钟。

5.6.2 云原生场景下的计算核心RunD

技术介绍

1. 传统QEMU单实例的内存开销超过100MB，启动时间也达到了数秒；
2. 完整操作系统需要加载大量驱动来支持各类设备，同时systemd等重量级服务带来的资源和启动开销也很大。

RunD为了解决上述问题进行了大量优化和改进（相较Kata Containers的改进如下图所示）：



用Rust实现VMM Dragonball：让原有VMM的内存资源开销从100+MB下降到3MB。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。

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。

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上。当前，RunD部分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。

5.6.3 容器镜像大规模分发技术Nydus

背景概述

在容器的生产实践中，当应用的镜像达到几个GB以上的时候，在节点上下载镜像通常会消耗大量的时间。Dragonfly通过引入P2P网络有效提升了容器镜像大规模分发的效率。然而，用户还是必须等待镜像数据完整下载到本地才能创建自己的容器。为此，我们为Dragonfly项目引入了一个容器镜像加速服务Nydus。Nydus能够极大缩短镜像下载时间，并提供端到端的镜像数据一致性校验，让用户能够更安全快捷地管理容器应用。容器镜像加速服务Nydus的项目地址如下：

https://github.com/dragonflyoss/image-service

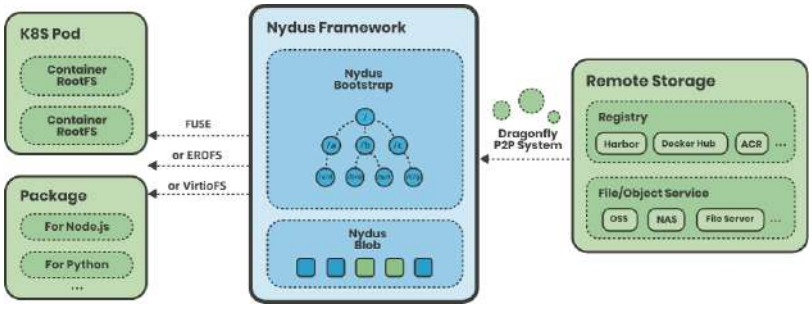
https://nydus.dev

技术方案

Nydus项目优化了现有的OCI镜像标准格式，并以此设计了一个用户态的文件系统。通过这些优化，Nydus能够提供如下特性：

- 容器镜像按需下载，用户不再需要下载完整镜像就能启动容器；
- 块级别的镜像数据去重，最大限度为用户节省存储资源；
- 镜像只有最终可用的数据，不需要保存和下载过期数据；
- 端到端的数据一致性校验，为用户提供更好的数据保护；
- 兼容OCI分发标准和artifacts标准，开箱即可用；
- 支持不同的镜像存储后端，镜像数据不只可以存放在镜像仓库，还可以放到NAS或类似S3的对象存储上；
- 良好集成到Dragonfly。

架构上，Nydus主要提供了Nydus镜像格式以及解析Nydus格式容器镜像的文件系统（可选FUSE用户态文件系统或者Linux内核文件系统）。



Nydus能够通过解析FUSE/virtiofs协议或Linux内核文件系统的方式来支持传统的runc容器或者Kata容器。容器仓库、OSS对象存储、NAS以及Dragonfly的超级节点和peer节点都可以作为Nydus的镜像数据源。同时，Nydus还可以配置一个本地缓存，避免每次启动都从远端数据源拉取数据。

应用案例

阿里巴巴集团，蚂蚁集团通过部署Nydus满足了云原生系统对于弹性能力和容器镜像供应安全保障的要求。

5.7 机密计算

5.7.1 机密计算平台技术

机密计算是一种依赖于硬件的使用中数据保护技术，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受保护的可信区域。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。OpenAnolis社区为推动机密计算技术的应用，衍生出若干基于机密计算的平台技术项目，其目的是不断降低机密计算的高使用门槛。

技术方案

JavaEnclave

JavaEnclave（Teaclave Java TEE SDK）是一个面向Java生态的Host-Enclave机密计算编程框架，与Intel SGX SDK和OpenEnclave具有相同的编程模型。它提供了一个Pure Java的机密计算开发界面和构建工具链；创新性的采用Java静态编译技术，将Enclave模块Java敏感代码编译成native包并在SGX环境下运行。在保证机密计算极致安全的同时，将机密计算开发生态从C/C++ 扩展到Java，极大降低了机密计算应用的开发与编译构建门槛，提升了开发效率与用户体验。

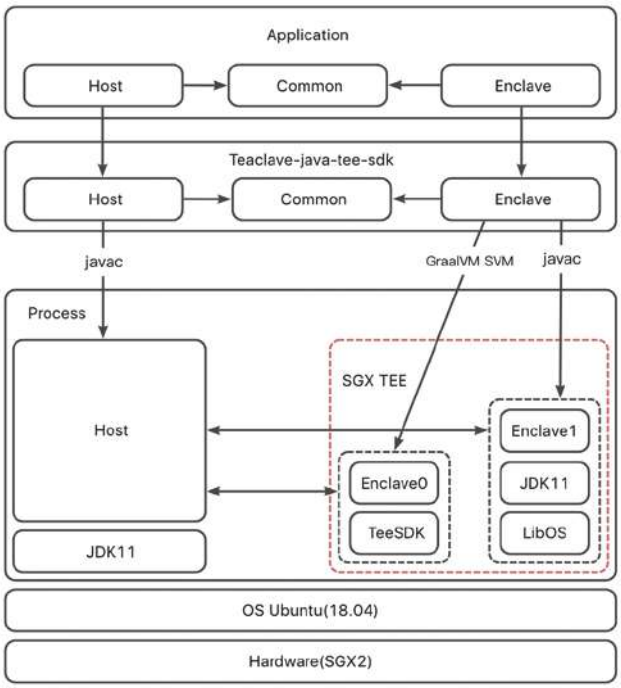


图 JavaEnclave项目架构方案

目前JavaEnclave已经在Apache开源社区开源并更名为Teaclave Java TEE SDK，作为ApacheTeaclave孵化项目的一个子项目继续发展。JavaEnclave将立足龙蜥社区云原生机密计算SIG生态，支持更多操作系统和TEE硬件平台，吸收社区广大开发者的反馈意见和贡献，持续改进并不断完善功能特性。

Occlum

Occlum是基于Intel SGX实现的一套轻量级的具有内存安全的LibOS，大大简化了SGX应用开发的难度。使用Occlum后，用户的工作负载只需要修改极少量（甚至无需修改）源代码即可在Intel SGX上运行，以高度透明的方式保护了用户数据的机密性和完整性：

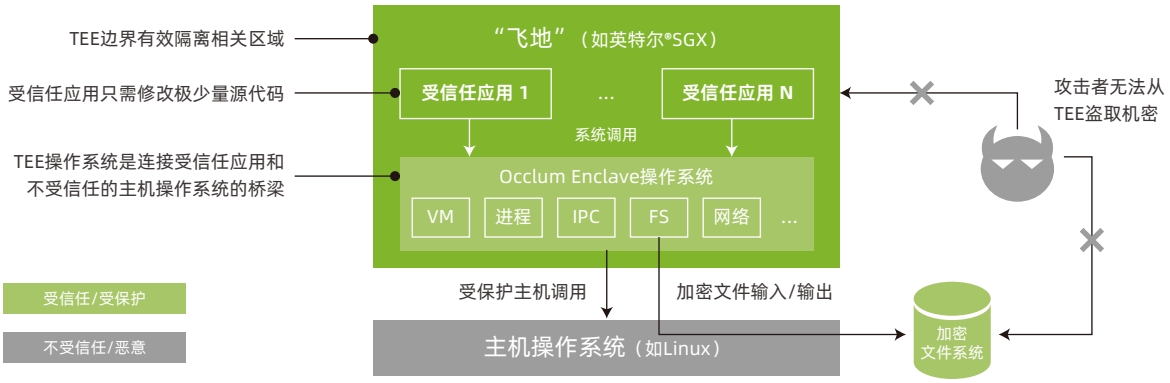


图 Occlum LibOS技术方案

今年年底即将发布的Occlum的v1.0版本采用了协程和异步架构用于提升Occlum在线程创建/切换、时钟获取和IO操作等方面的性能。

RATS-TLS

RATS-TLS设计了一种支持异构硬件机密计算技术的双向传输层安全协议，它在TLS的基础上增加了将TLS中的公钥与TEE远程证明Evidence绑定的能力，解决了不同TEE之间难以通过安全可信的方式传输数据的问题：

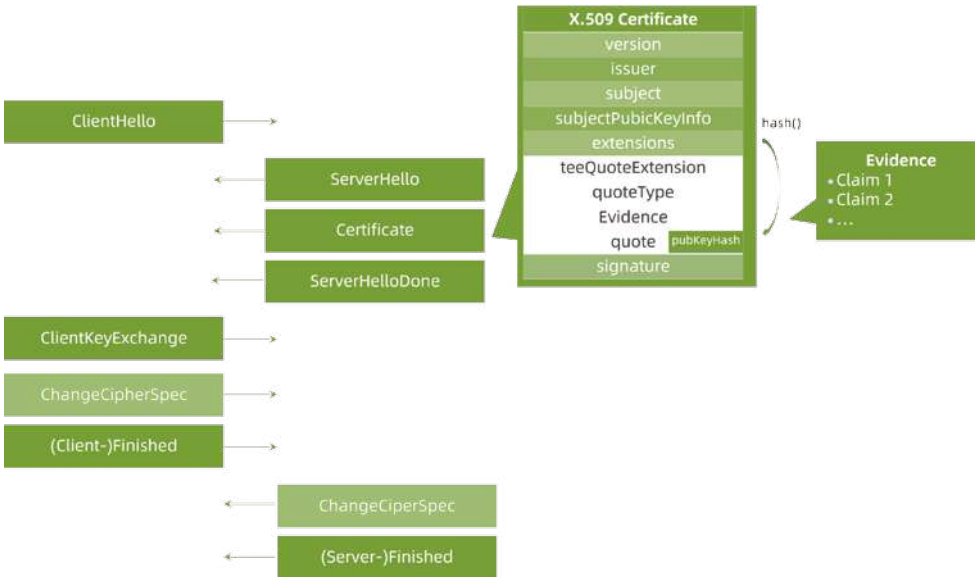


图 RATS-TLS实现原理

“从RATS-TLS项目衍生出的新项目librats已经支持多家主流芯片厂商的远程证明认证机制，并允许异构TEE之间进行双向远程证明认证。librats将支持最新定义的TCG DICE Evidence格式，并计划捐赠给CCC社区。

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。

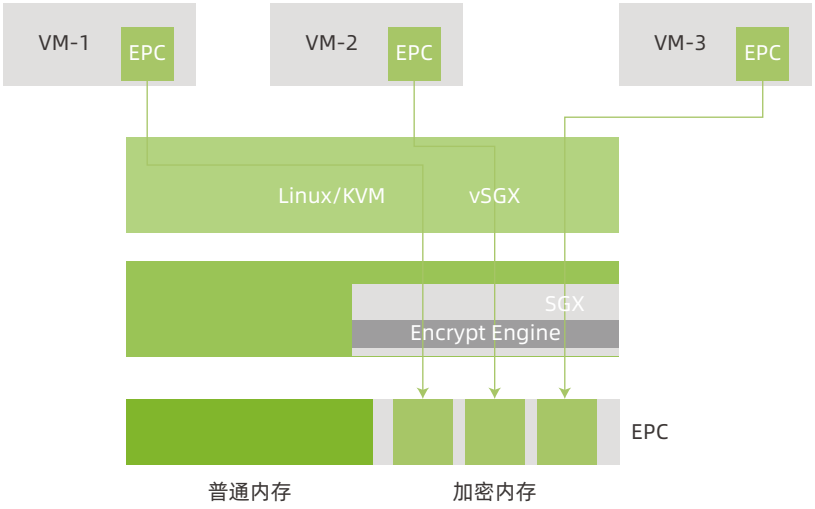


图 SGX虚拟化技术实现方案

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：

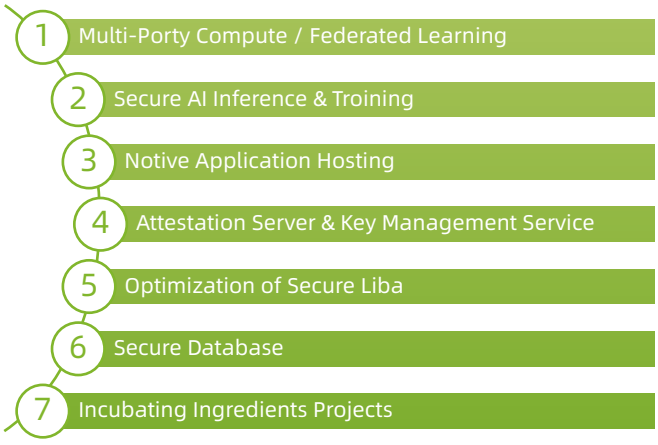


图 CCZoo项目内容

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。

Intel HE

Intel提供了对于同态加密技术的全栈式支持，包括一系列工具套件和加速库，如Intel HE Toolkit、Intel HE Acceleration Library (Intel HEXL) 、Intel Paillier Cryptosystem Library (IPCL)，以及公允且标准的性能测试基准Homomorphic Encryption Benchmarking Framework (HEBench)：

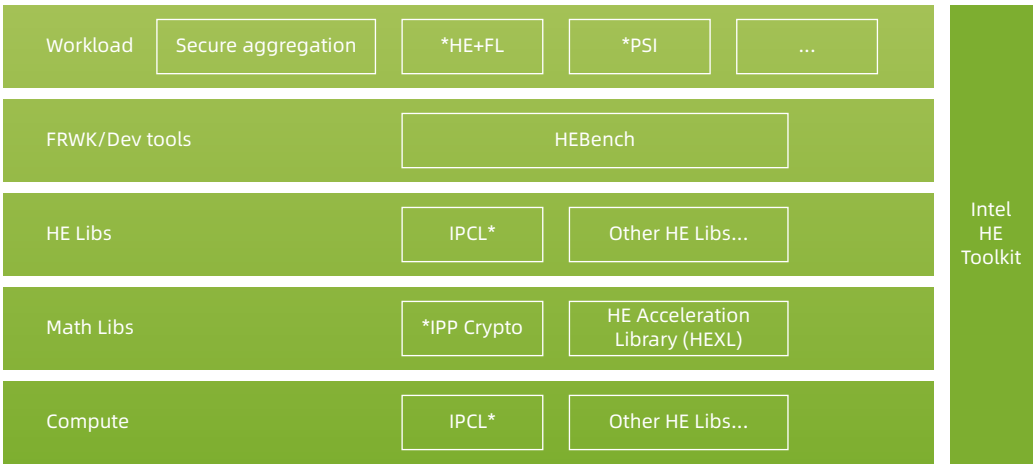


图 Intel HE同态加密支持方案

5.7.2 机密容器

机密容器是机密计算技术和云原生技术结合的产物。机密容器基于硬件（HW-TEE）实现资源隔离、数据加密以及远程证明，能够防止云服务提供商和任何高权限的第三方对执行环境中的数据进行窃取和篡改，从而有效地保护用户数据和资产安全。

机密容器具有如下特点：能够对数据，尤其是运行态的数据，提供机密性、完整性和安全性保护；能够为敏感工作负载的部署和运行提供全链路的安全保障；能够提供方法证明机密工作负载当前运行的环境是真实可信的；用户保持和普通容器几乎相同的使用体验，免运维成本部署敏感应用负载。

技术方案

龙蜥机密容器解决方案支持多种HW-TEE，提供Pod级机密容器和进程级机密容器两种解决方案，满足用户不同场景的使用需求。

在Pod级机密容器领域，近年来，多家硬件厂商陆续发布针对虚拟化场景的机密计算解决方案，给创建基于轻量级虚拟机的机密容器提供了硬件基础。基于CoCo（Confidential Container），龙蜥社区理事单位阿里巴巴和Intel等硬件厂商合作构造了Pod级机密容器解决方案，用户可以零修改地将他们的工作负载运行在该解决方案中。目前社区已经实现了对Intel TDX等硬件平台的支持，后续还计划增加更多的HW-TEE解决方案的支持。

在进程级机密容器领域，阿里巴巴和Intel把SGX技术和容器技术相结合，创新性地开发出了Inclavare Containers机密容器项目，该项目同时也成为Cloud Native Computing Foundation（CNCF）的第一个机密容器项目。Enclave-cc是基于Inclavare Containers实现的进程级机密容器解决方案，并且满足了CoCo社区设定的安全模型。后续enclave-cc将成为CoCo社区中的进程级机密容器的参考实现，丰富enclave-cc的上下游软件生态链，并持续提升方案的产品化能力。

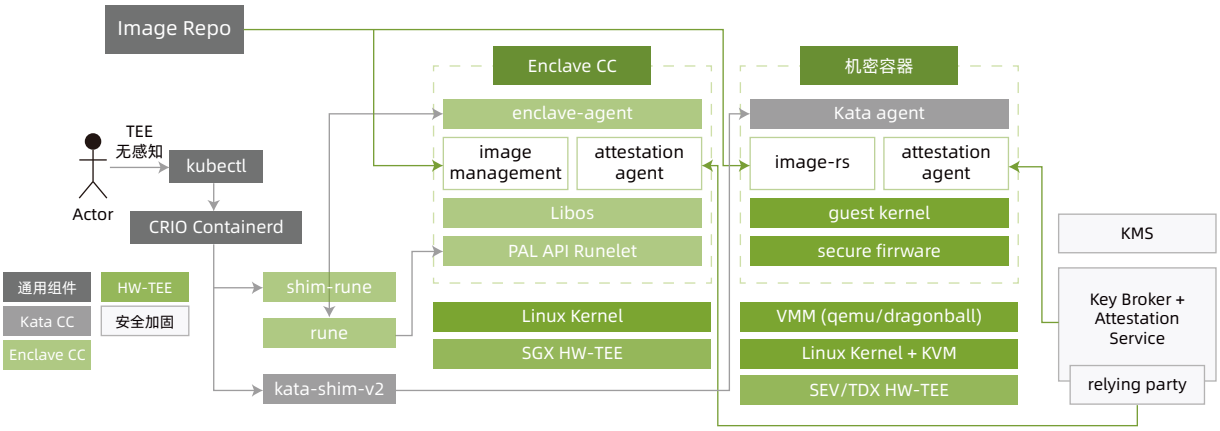


图 机密容器解决方案

应用案例

大数据分析和人工智能的应用日益广泛，这也带来了隐私安全风险以及监管要求。分布式隐私保护机器学习（Privacy Preserving Machine Learning, PPML）解决方案基于Intel BigDL开源人工智能解决方案平台，能够在不透出原始数据的前提下，实现数据的有效流动和高效处理。龙蜥社区理事单位阿里巴巴和Intel结合机密容器解决方案和Intel BigDL PPML开源方案, 构建了一个分布式的隐私保护机器学习模型，能够实现端到端（包括数据输入， 数据分析，机器学习，深度学习的各个阶段）的数据保护。

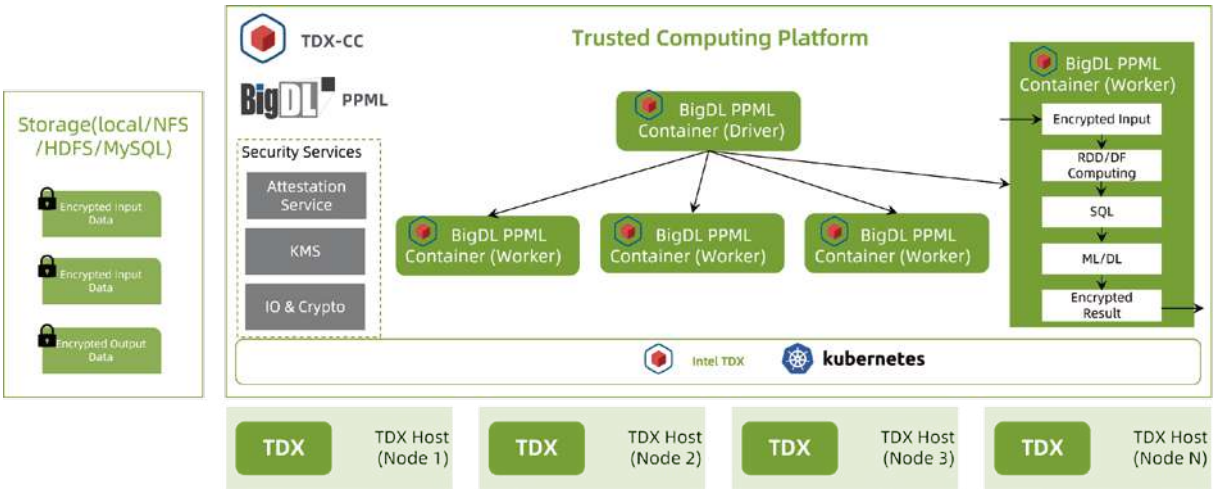


图 BigDL PPML on TDX-CC

5.8 系统安全

5.8.1 龙蜥操作系统漏洞管理

龙蜥社区已经建立起从漏洞感知、评估，到漏洞修复、披露的全生命周期闭环的安全漏洞管理流程体系。社区针对最新安全威胁进行及时的跟踪与风险评估，制定完备的漏洞修复策略。并依据威胁等级发布安全更新，帮助龙蜥用户及时修复安全漏洞，全面提升系统安全。

- **漏洞感知**

社区通过主动跟踪业界知名漏洞库、安全论坛、邮件列表、安全会议，以及与知名第三方漏洞情报感知服务合作等多种渠道，以保证第一时间感知到Linux操作系统相关漏洞信息。并通过建立完备的，覆盖龙蜥全线产品的漏洞库，以保证有效的记录、追踪和闭环每个安全漏洞。

- **漏洞评估**

社区基于业界通用的CVSS评分框架，融合龙蜥操作系统成熟的软件包分层分类理论，建立起基于风险的安全漏洞管理体系，多维度对漏洞进行威胁风险及严重级别的评估。依据漏洞在实际业务环境中龙蜥产品被利用的风险，来制定具体的漏洞修复策略与应对策略。

- **漏洞修复**

社区完善的漏洞修复与验证流程，安全可信的构建体系和质量管理体系，能够有效地保障安全漏洞在预定的服务时间内完成响应与修复。

- 漏洞披露

社区秉承对用户负责的漏洞修复与披露原则，在符合业界规范和政策法规的前提下，通过社区官方安全门户发布龙蜥安全公告（ANSA），以及邮件订阅等多种途经，及时向龙蜥用户及下游生态推送漏洞缓解和修复方案与建议，帮助用户及时修复漏洞，减少安全风险。截止目前，社区已经收录的漏洞信息和发布的安全更新对社区开发者和用户全网公开，并形成社区共建能力。

- 龙蜥社区CNA

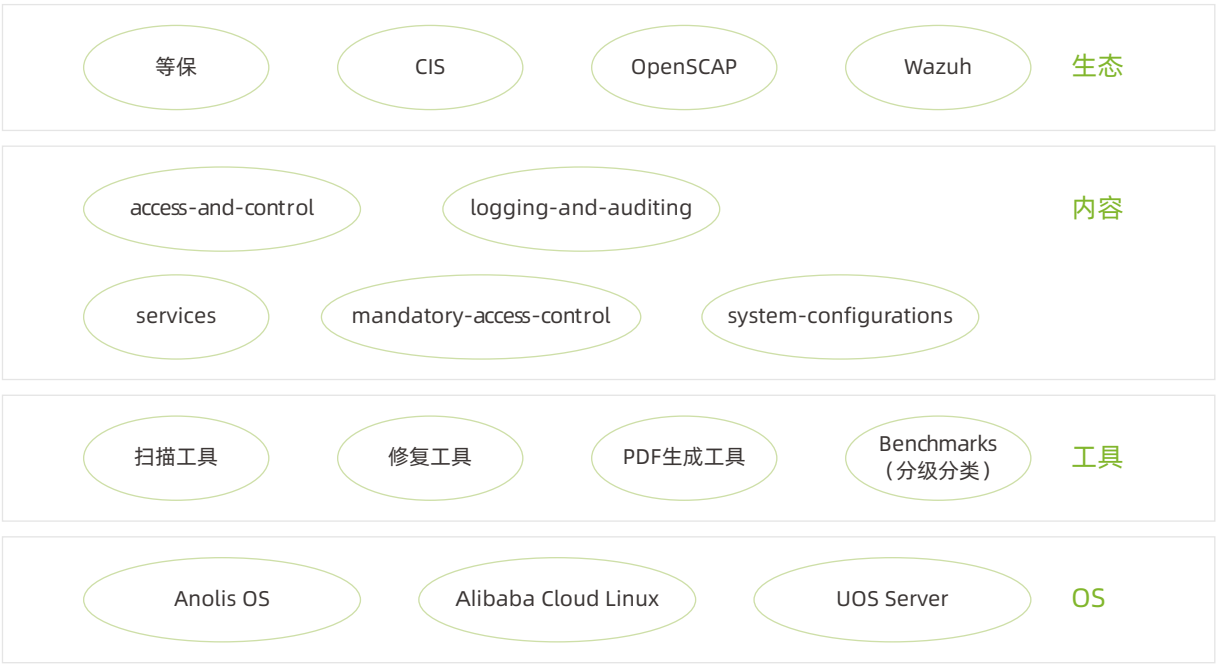
社区已经成功加入CVE.org组织，成为CVE编号授权机构。社区积极与合作伙伴展开安全合作，先后发现并申报了多个Linux系统相关的CVE，并为相关漏洞赋予编号。社区也欢迎广大社区爱好者、系统安全领域相关的个人和组织，通过龙蜥社区上报安全漏洞(security@openanolis.org)，共同维护社区安全。

5.8.2 安全合规

龙蜥社区联合阿里云、统信软件等社区成员，基于最新Anolis OS发行版发布了《Anolis OS 8安全最佳实践v1.0.0》，该最佳实践基于Anolis OS 8下游发行版UOS Server v20和Alibaba Cloud Linux 3大规模产品落地的安全经验和实践进行打造，兼容CIS（Center for Internet Security）、等保2.0等安全标准的安全保护基本要求，包括五个类别共150多条内容。

最佳实践的每一项安全指南涵盖对应的安全等级、描述、修复建议与扫描检测等内容，同时采用分级分类的定义方式，能够支持用户结合实际的安全需求选择实施不同安全等级的加固，以更好地满足对不同用户不同场景的配置安全基准要求。龙蜥安全最佳实践还提供对应可配置的扫描验证工具和修复工具，结合安全工具集进行发布以支持系统安全配置的自动化核查与加固，帮助Anolis OS及其下游厂商更好地满足安全合规的要求，降低安全风险。

结合Anolis OS的体系化安全合规建设，下游发行版UOS Server v20及Alibaba Cloud Linux 2/3近期也先后完成与国际知名安全社区OpenSCAP的产品支持整合，并成为OpenSCAP官方首批支持的国内OS产品。Anolis 8安全最佳实践的发布，能够持续与OpenSCAP安全策略进行结合，进一步繁荣龙蜥社区的安全生态。



龙蜥安全加固最佳实践

关键应用场景

1. 安全合规镜像：基于龙蜥下游发行版打造的Alibaba Cloud Linux 2/3等保2.0三级版镜像等。
2. 安全镜像扫描/监控工具：能够查看镜像的安全合规情况。
3. 普通镜像到合规镜像的转换/加固工具：可以选择适合用户的合规项将用户使用的普通镜像转换成合规镜像而不是直接迁移到已有的合规镜像。

龙蜥社区安全最佳实践是基于UOS和Alibaba Cloud Linux发行版在等保/CIS等领域的大规模产品落地经验打造，以Alibaba Cloud Linux为例，其等保/CIS加固镜像目前已被银行、保险、证券等金融客户广泛使用。

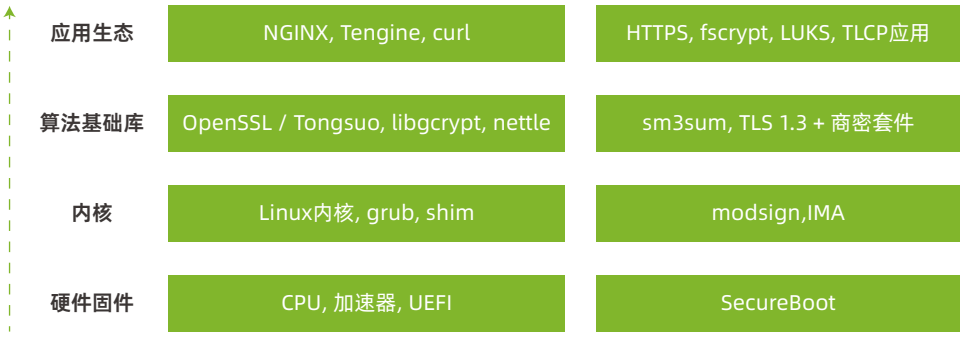
5.8.3 商密软件栈

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。

商密全栈架构

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实现了商密算法以及性能优化，包括Linux内核，OpenSSL，libgcrypt，gnulib，nettle等在内的基础组件，支持了商密算法以及大量的性能优化，并且得到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一些短板，在兼容现有API的情况下，提供给普通开发者平滑的使用体验。

商密基础设施架构



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，可以平滑的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

- LUKS磁盘加密支持使用SM4商密算法；
- fscrypt支持使用SM4算法，以及SM4-XTS，SM4-CBC模式优化加速；
- TLS 1.3 支持使用商密算法套件（TLS_SM4_GCM_SM3）；
- IMA，modsign支持使用SM2/SM3算法组合的签名验签，涉及Linux内核，sign-file工具和ima-evm-utils的支持。

5.9 运维与性能

5.9.1 SysAK：大规模复杂场景的系统运维利器

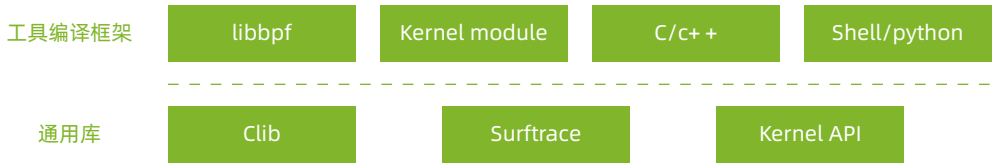
概述

SysAK（System Analyse Kit）是龙蜥社区系统运维SIG，通过对过往百万服务器运维经验进行抽象总结，而提供的一个全方位的系统运维工具集，可以覆盖系统的日常监控、线上问题诊断和系统故障修复等常见运维场景。工具的整体设计上，力图让运维工作回归简单，让系统运维人员不需要深入了解内核就能找出问题的所在。

技术描述

SysAK在功能集上会进行全方位覆盖，垂直打通整个应用的生命周期。当前工具支持监控和诊断两种模式。其中监控模式下SysAK常驻后台，为运维人员提供系统的各项指标。而诊断模式随用随启，主要用于分析不同运维场景下的系统现象诊断与程序控制等。其整体功能如下图所示：

SysAK不仅限于一个工具集，除了提供系统运维工具本身外，还设计实现了一套工具开发框架。并通过松散耦合、依赖管理、多架构多版本的构建支持等方式，保障了工具开发者，一次开发，无需额外工作，就能在主流的架构和操作系统版本上集成。其整体结构如下图所示，



应用场景

SysAK提供的诊断工具可满足不同应用场景的运维需求：

- 日常监控：针对各种系统资源更精细化的资源监控，帮助业务运维实现细粒度的运维调度和资源控制。除此之外，还实现了许多增强的系统指标，实时监控系统的干扰和抖动等情况。
- 问题诊断：针对负载异常、网络抖动、内存泄漏、IO夯、性能异常等情况提供线上诊断功能。 同时减少工具的专业性，可操作性强。
- 故障修复：对于非整机异常的问题（例如死锁、夯机等）， 该工具提供介入能力对系统进行恢复或故障隔离。

5.9.2 SysOM： 一站式运维管理平台

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。

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打通了系统监控、告警、诊断以及安全运维的全流程。基于SysAK入的内核行为分析，服务端的大数据和器学习分析，不仅能让运维人员发现问题，还能非常精确的定位到问题的故障点，从而实现“傻瓜式”的运维体验。整体平台具备以下特点：

- 统一平台：一个平台解决操作系统运维过程中所需要的多种关键能力。SysOM将主机管理、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。
- 简单易用：降低运维的门槛，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。让用户清楚引发问题的自身代码缺陷点或系统具体问题配置。
- 深度分析：集成SysAK工具集深度诊断解决方案，沉淀百万级的运维经验，进行内核源码级别的问题剖析。让每一个应用的行为都“知其所以然”。
- 安全可靠：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。

应用场景

- SysOM采用微服务、前后端分析的技术架构， 针对不同集群规模的应用场景，提供灵活的集成方案。
- 小规模集群应用：一站式简易集中部署，覆盖主机管理、堡垒机、监控、诊断以及安全运维等全流程方案。
- 大规模集群应用：各服务模块使用独立docker部署，弹性扩容。接入标准的Kafka/Flink大数据框架。满足大规模的流量冲击。
- 集成到现有系统：对于已有运维系统的用户，用户可以抛弃SysOM前端，使用规范的后端接口， 接入SysOM功能。 SysOM已经为这类用户专项考虑，降低对接成本。

5.9.3 Coolbpf：一站式eBPF开发编译平台

概述

eBPF是一项新的动态跟踪技术。基于eBPF技术，衍生出的BCC、bpftrace等开源工具，极大降低了eBPF的应用代码开发的门槛，成为当前eBPF各类应用开发的主要方式，但此类工具存在如下缺点：

- 生产环境需要部署依赖库较多，如Clang/LLVM等库，部署不便；
- 每次运行都要执行Clang/LLVM编译，严重消耗CPU、内存等资源；
- 需要安装对应内核的头文件；
- 不支持3.10等低内核版本

Coolbpf是一个便捷高效的一站式eBPF开发编译平台，创新的提出远程编译（云编译）思想，以CO-RE（Compile Once-Run Everywhere）为基础实现，保留了资源占用低、可移植性强等优点，适合在生产环境批量部署所开发的应用，加上新实现的带eBPF verifier的内核模块，使得同一个eBPF应用无需修改就能在 3.x/4.x/5.x内核版本安全运行。基于Coolbpf的用户只需专注自己的功能开发，不用关心底层库安装和环境搭建，给广大eBPF爱好者提供了一种新的探索和实践。

技术描述

Coolbpf的功能架构如下图，它当前主要包含六大功能：远程编译；本地编译和基础库封装；低版本内核支持；BTF自动生成和发布；eBPF应用自动化测试；支持python/go/rust等高级语言进行应用开发。

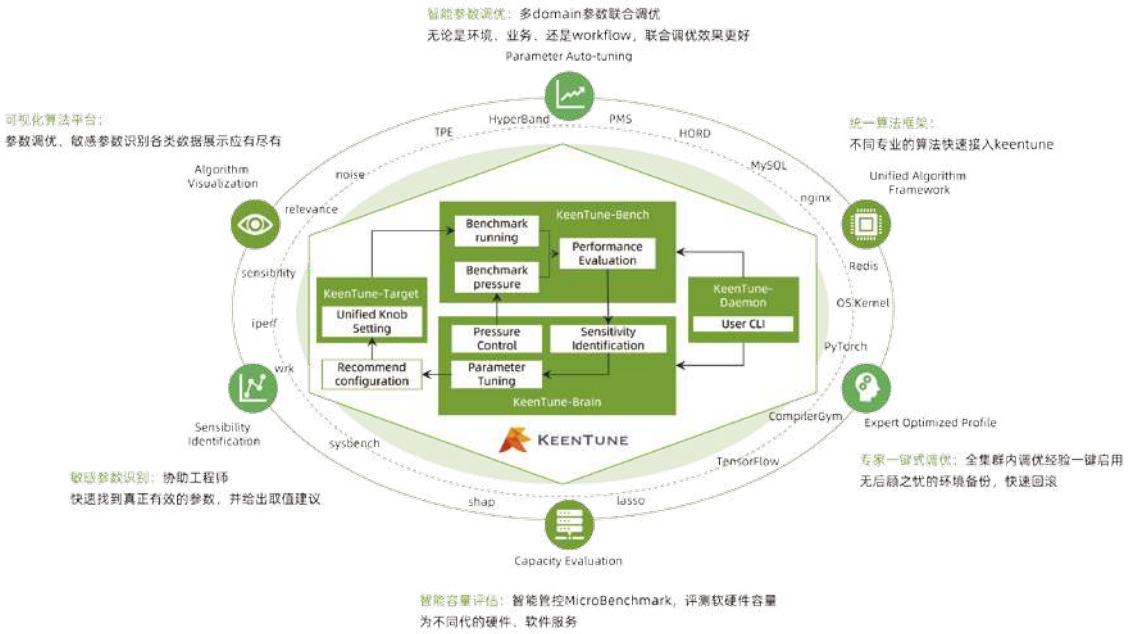
应用场景

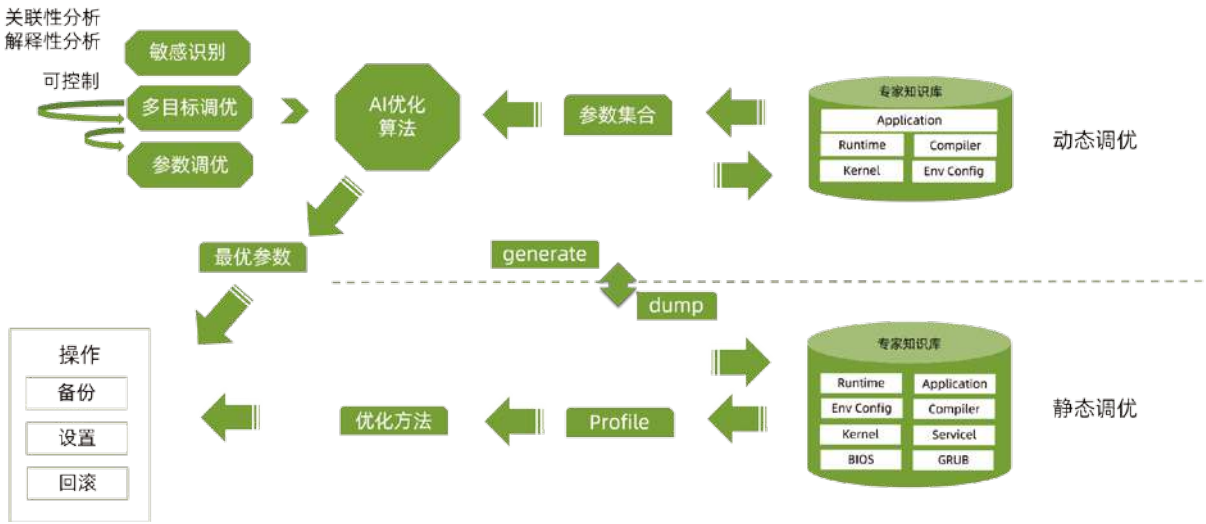
Coolbpf如其名一样非常酷的提升了eBPF的开发和编译效率，应用场景包括：系统故障诊断，网络优化，系统安全，性能监控。未来，Coolbpf还将探索新技术、新特性：提升字节码翻译效率、内核运行时安全等，应用场景将更加多样。

5.9.4 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。

如图中心框图所示，KeenTune有五个模块：分别是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。





智能化性能调优

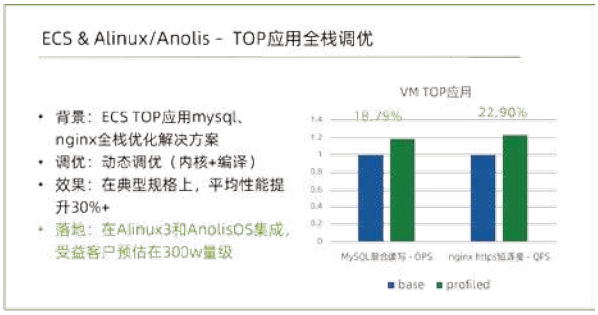
KeenTune通过AI算法与专家知识库的配合使用，通过智能参数调优（动态）与专家一键式调优（静态）两项能力来实现OS上应用的高效调优。

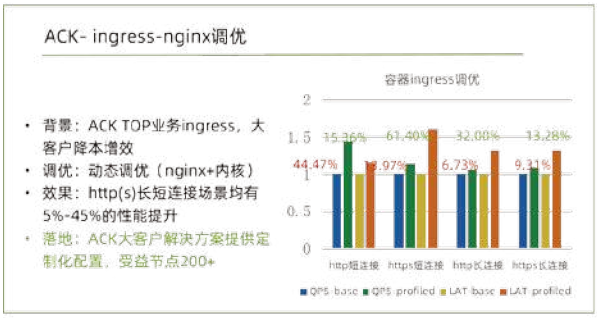
- 智能参数调优：集成自研及主流开源的高效算法，提供内核、编译器、运行时、应用的全栈的智能参数调优，全方位提高业务性能；同时，高效可信的参数可解释性算法辅助人工决策线上业务的最佳配置。
- 一键专家调优：针对基础业务负载和云上TOP应用，KeenTune都提供了典型场景的调优专家库，一键设置即可提升该场景运行环境的性能。
- 动、静态的联合调优：动态调优的结果固化为静态调优配置，一套环境调优可以有效扩展到多套；静态调优配置成为动态调优的初始参数集合，有效保证应用运行在定制化的最佳设置环境。

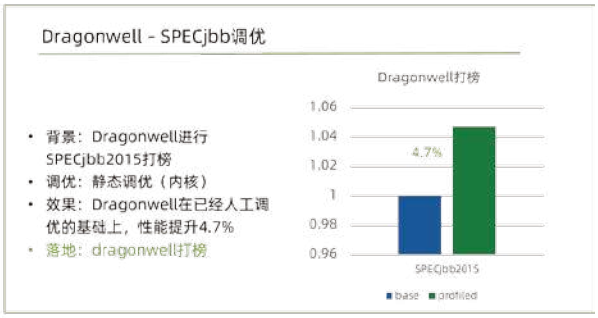
智能化容量评估

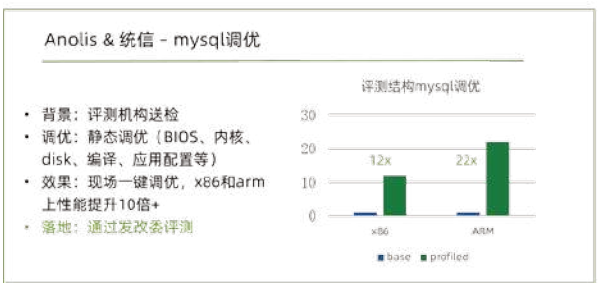
KeenTune借助参数调优算法的能力，实现了对于benchmark参数的管控，从而达成快速进行系统容量评估的目的。目前已经形成了硬件容量评估的体系，覆盖CPU、内存、IO、网络。后续也会持续在系统、业务容量评估持续发力。

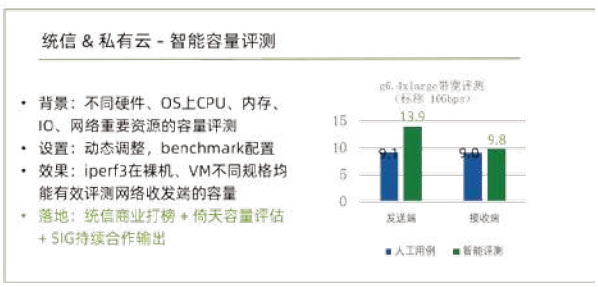
KeenTune在公有云、私有云、物理机上的多种业务场景都有比较好的效果，部分典型场景的效果如下所示：











5.10 社区基础设施

5.10.1 T-One：全场景质量协作平台

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。

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

技术方案

T-One主要有下面三方面的优势：

（1）提供全场景的测试能力

1. 支持多CPU混合架构（x86、arm、loongarch64、risc-v）；
2. 支持多操作系统类型（龙蜥、centos、debian、ubuntu、统信、麒麟）；
3. 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：

1. 基线跟踪模型：聚合型基线模型、测试指标跟踪模式；
2. 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；
3. 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。

图 T-One平台架构示意图

T-One经历5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。

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。在相关评测中提升测试效率，提升测试效率30倍以上。

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。同时支持了40多个项目的质量协同以及数百台测试机器并发执行。服务龙蜥社区，被包括统信、电子五所、云原生、浪潮信息、高性能网络、中科曙光等多家企业、机构的10多个项目采用。

5.10.2 一站式构建服务ABS

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，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镜像自定义等构建功能，同时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：https://abs.openanolis.cn/all_project

技术方案

ABS主要能力有：

- ABS提供了多架构的构建能力，支持多CPU架构（x86、arm、loongarch64、risc-V）；
- 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；
- ABS提供一键 Anolis OS ISO镜像rebrand功能，快速定制下游衍生版；
- ABS提供一键化构建Docker镜像并分发的功能；
- 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

- 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；
- 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；
- 设备管理层：我们基于koji开源系统进行二次开发，向上提供多架构的构建能力。

应用场景

ABS平台支持的使用场景有：

- 软件包研发过程的测试和构建；
- 下游厂商可以通过 ABS构建操作系统衍生发行版；
- 社区爱好者、合作伙伴可以作为 docker 自定义构建平台。
- 发行版生命周期维护。

过去一年，龙蜥社区发布了Anolis OS 8.2、8.4、8.6、龙芯版等重要版本，这些都是通过ABS构建系统完成的。ABS上线半年时间，构建的软件包数量超过10000个，构建次数超过20000次，创建项目总量超800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，包括西软、统信等等。

5.10.3 龙蜥实验室，基础设施资源底座

背景概述

为了拉近社区的技术与用户的距离，方便用户了解龙蜥的产品生态，我们建立了龙蜥实验室。龙蜥实验室通过课程学习，解决方案体验和免费的机器资源三种方式让用户直接触达到龙蜥社区的产品生态，大大提升社区技术的辐射速度。

系统链接：https://lab.openanolis.cn/#/apply/home

技术方案

为了提升并发服务能力，我们通过k8s建立了一个可伸缩的分布式资源服务框架，能支持超过500个用户的并发使用。同时我们搭建了一个通用的课程框架并基于此建立了课程创作中心，方便开发者贡献课程内容。

基于强大的资源服务框架，龙蜥实验室为用户提供了三大服务能力：

1. 机器资源服务：利用弹性云资源为社区的用户提供免费的、预装了Anolis OS的机器资源，用户可以随时申请来体验Anolis OS；
2. 课程学习服务：建立在线课程学习体验平台，目前已经接入10多个课程，涵盖云原生、调优、问题诊断、DDE、测试、构建、迁移等多个方向，方便用户了解龙蜥的产品生态，用户也可以自助接入更多的课程；
3. 解决方案体验：提供龙蜥产品解决方案在线体验的模式（建设中）。

龙蜥实验室架构示意图

平台特色

通过龙蜥实验室三大能力，可以方便用户了解社区产品生态，龙蜥实验室的主要优势有：

1. 免费资源：用户的体验完全免费，用户可以随时进行使用；
2. 秒极体验：不管是机器资源还是课程学习，都能为用户提供秒极的体验支持；
3. 自助接入：为课程框架提供在线接入能力，通过创作中心，用户可以自助接入新的课程。

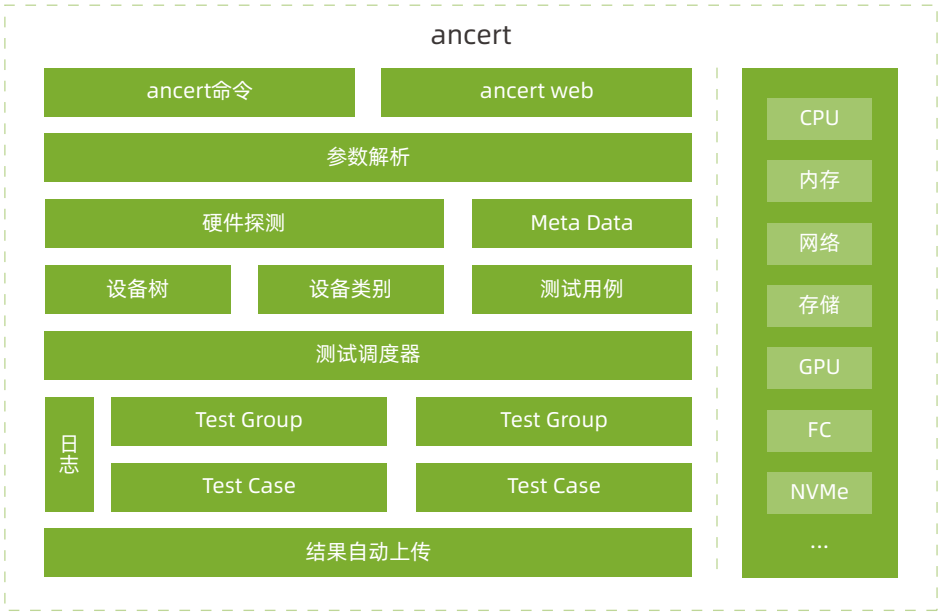
5.10.4 ancert：硬件兼容性验证与守护

ancert是龙蜥操作系统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。

核心技术特点

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；

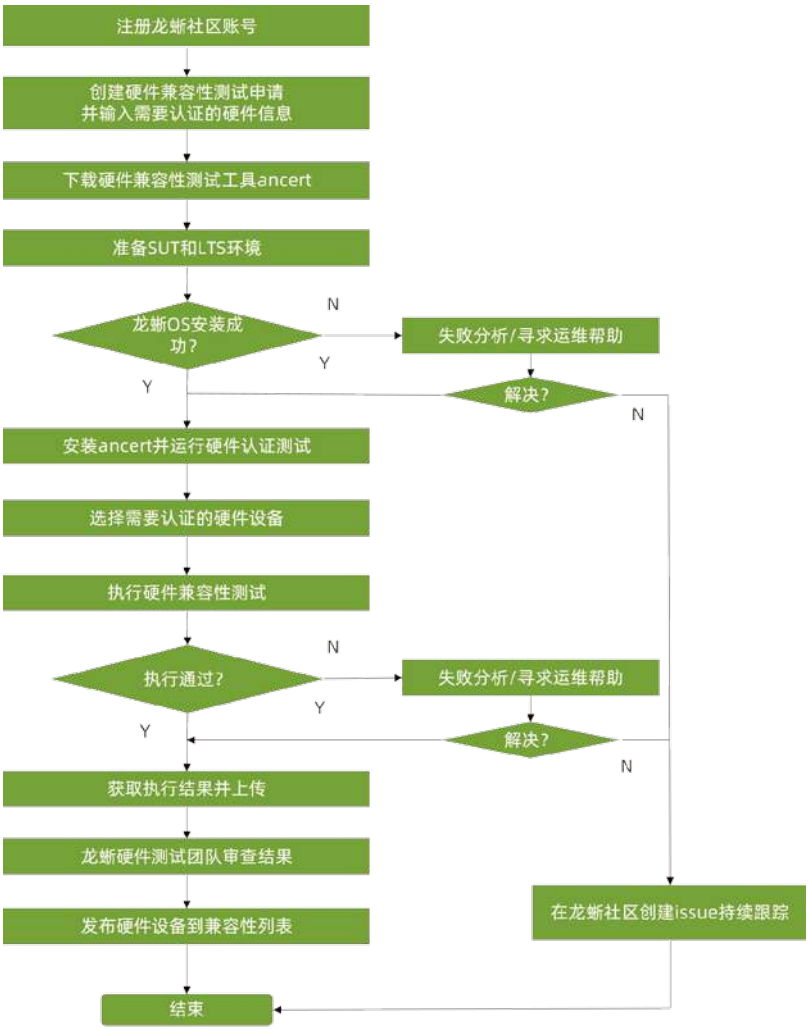


硬件兼容性测试流程

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，IHVs或者硬件设备集成商等通过安装龙蜥OS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OS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OS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OS硬件生态建设。



06 “龙蜥+”精选方案与案例

6.1 精选典型方案

龙蜥社区重视操作系统相关技术在产业和商业化上的落地，以社区创新引领能力建设为基础，形成创新技术赋能社区商业合作伙伴，伙伴反馈社区并服务社区用户的良性循环。随着龙蜥操作系统应用在越来越广泛的使用场景中，以操作系统和上下游软硬件生态为核心，龙蜥社区在典型场景的挑战和价值驱动下形成众多“龙蜥+X”解决方案，本文基于过去一年的社区实践精选出CentOS停服替代场景的平滑迁移、系统安全场景的加解密加速、资源混部场景的内核隔离实现、云原生集群部署场景的镜像分发加速的算力优化方案，希望为社区用户在技术应用和创新过程中提供有力支撑。

6.1.1 CentOS停服替代场景的平滑迁移方案

概述

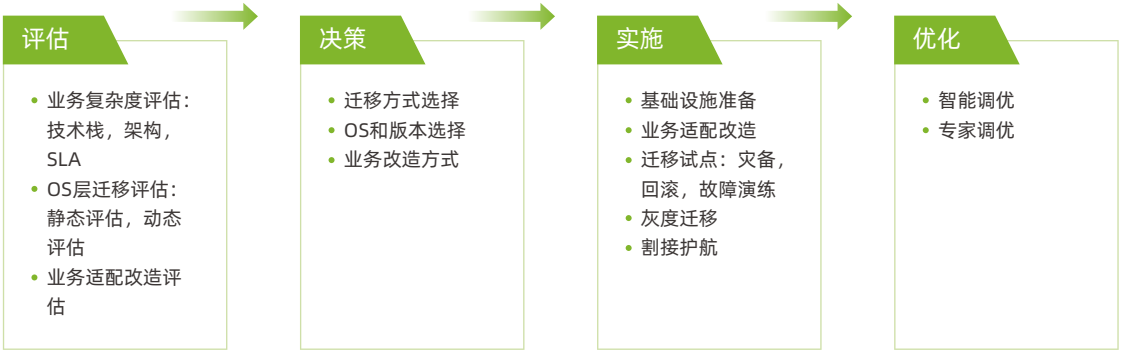
操作系统迁移是一个复杂的工程，而在云原生时代，IaaS与PaaS的迁移复杂度更高，且相互影响。因而操作系统迁移不再是一个单机维度的OS切换，而是系统性的迁移工程。针对这一痛点，龙蜥社区在支持用户进行操作系统迁移的过程中，逐步狠点了一套行之有效的迁移方法论，并为CentOS用户提供了迁移到Anolis OS的迁移系统AOMS（Anolis OS Migration System）。

场景挑战

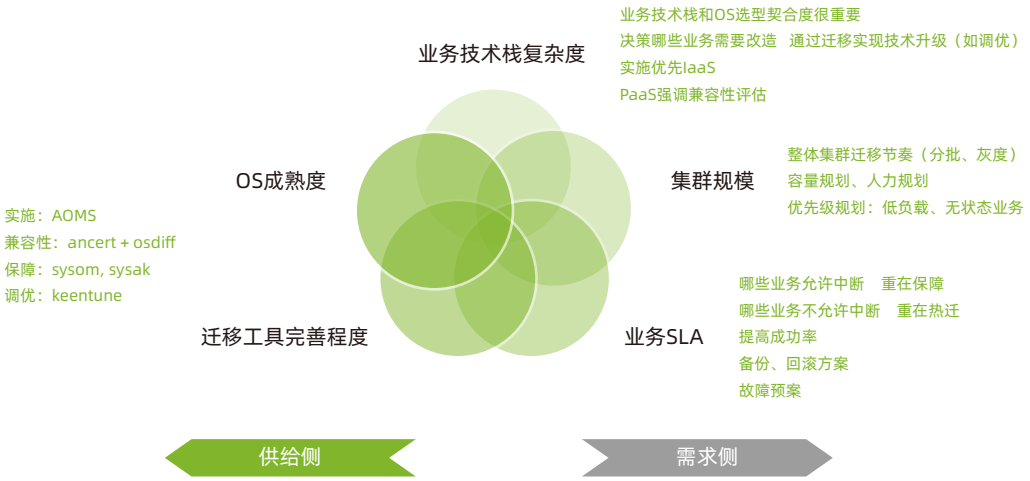
随着各种虚拟化技术，开发语言的繁荣发展，在进行操作系统迁移时会出现多种开发语言，中间件，数据库，虚拟化手段混杂在一起的情况，而平台，业务，产品等不同纬度的诉求也会产生叠加。在这些场景下，操作系统迁移不再是一个单机维度的OS切换，而是需要从集群迁移视角来看待，做好全局评估与实施方案，做好灾备，灰度，回滚方案，并结合上层业务调度来进行迁移的系统工程。

方案特色

迁移方法论：评估，决策，实施，优化四步迁移法。



迁移评估的5个维度及其关键的决策信息：



迁移实施也是业务迁移实现平稳交付的关键环节，其阶段详细的流程要经过实施方案制定、基础设施准备、业务适配改造、迁移试点、迁移批量实施、割接护航6大步骤，确保迁移的交付环节可靠和高效。

实践验证

AOMS迁移方案包含如下三个场景：

- CentOS 8迁移Anolis OS 8
- CentOS 7迁移Anolis OS 7
- CentOS 7迁移Anolis OS 8

CentOS 8迁移Anolis OS 8及CentOS 7迁移Anolis OS 7场景：

Anolis OS 8在做出差异性开发的同时，在生态上和依赖管理上保持与CentOS 8的兼容，AOMS充分利用了兼容的特性，提供了一键式迁移工具：centos2anolis.py。

CentOS 8迁移使用Anolis release相关的包替代CentOS release ，通过yum distro-sync重装当前系统中所有的系统软件包。

软件重装的过程并不会修改当前系统基础配置，所以系统配置，业务配置，业务数据都不会被清除，迁移完成后这些数据无需重新设置。

使用迁移脚本前需要注意如下事项：

- 迁移过程涉及到访问Anolis OS的官方repo ，需要确保待迁移环境网络能够正常访问Anolis OS repo。
- 需要使用root用户执行，当前只支持CentOS 8系统的迁移，不支持CentOS stream系统迁移。
- 迁移过程依赖于yum/dnf ，需要确保组件能够正常运行。
- 迁移脚本提供了Anolis OS repo访问加速的功能，如果访问Anolis OS官方repo速度较慢，可以通过-s选项进行加速访问。
- 迁移日志保存在/var/log/centos2anolis.log。

CentOS 7迁移Anolis OS 8场景：

CentOS 7到Anolis OS 8，无论是内核，基础软件包，工具链都发生了较大的变化。迁移工具需要考虑这些变化带来的兼容性问题。

AOMS提供的迁移工具leapp包含了迁移评估，迁移实施，配置还原等步骤，用于实现CentOS 7到Anolis OS 8的就地迁移

第一步：迁移评估

leapp扫描待迁移系统，搜集内核，软件包，系统配置基础信息，同时与目标系统（ Anolis OS 8 ）进行对比分析，对于不兼容项给出影响分析和解决方案。

- 内核角度：给出Anolis OS 8中不再支持的内核特性，硬件驱动；
- 软件角度：给出系统命令的变更项，提示用户适配业务程序。

迁移评估报告会给出当前系统中所有可能影响到迁移的影响项目，当这些影响项目都被解决后，用户才能够继续做迁移实施。同时业务程序可根据评估报告中的兼容性提示来适配迁移业务程序。

第二步：迁移实施

leapp首先搜集当前的系统信息，记录需要在重启后恢复的配置（如selinux状态）。迁移实施过程中，工具首先按照当前系统安装的软件包列表，并根据CentOS 7到Anolis OS 8的软件包映射关系，从Anolis OS repo上提前下载迁移所需要的软件包，并基于Anolis OS 8的软件包制作upgrade-initramfs ，在下一次重启后，系统自动进入upgrade-initramfs ，并触发所有软件包的就地升级。在所有的软件包就地升级完成后，自动重启进入系统配置还原阶段，待所有信息完成配置，系统重启进入新的OS ，完成OS的就地迁移。

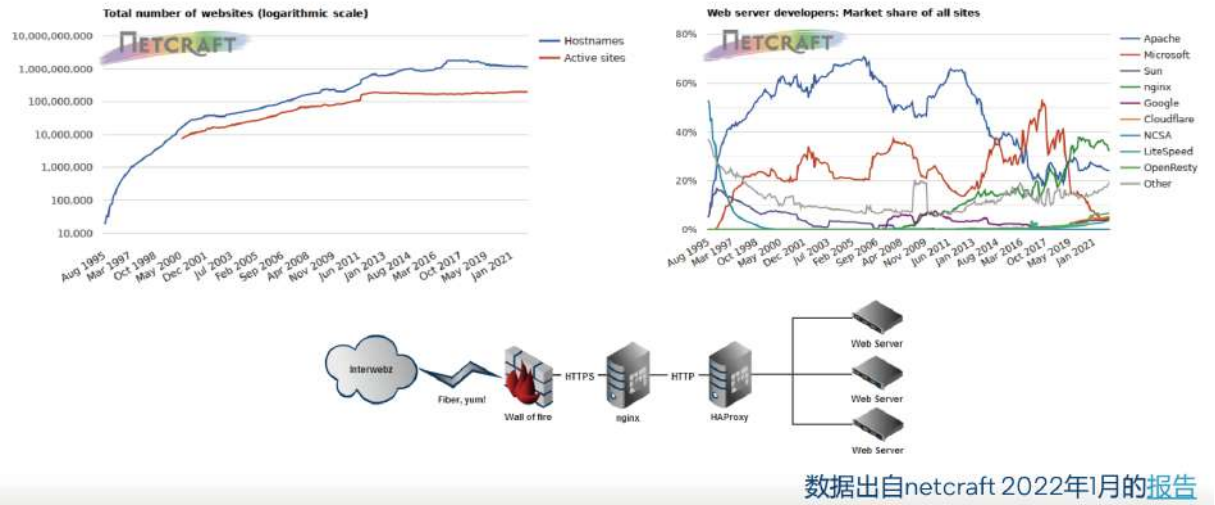
总结

基于龙蜥社区AOMS迁移工具，用户可以解决由于CentOS停服带来的软件供应链风险，同时大大降低由于操作系统迁移带来的高技术要求、高复杂操作的工程难度，帮助用户快速完成操作系统迁移。

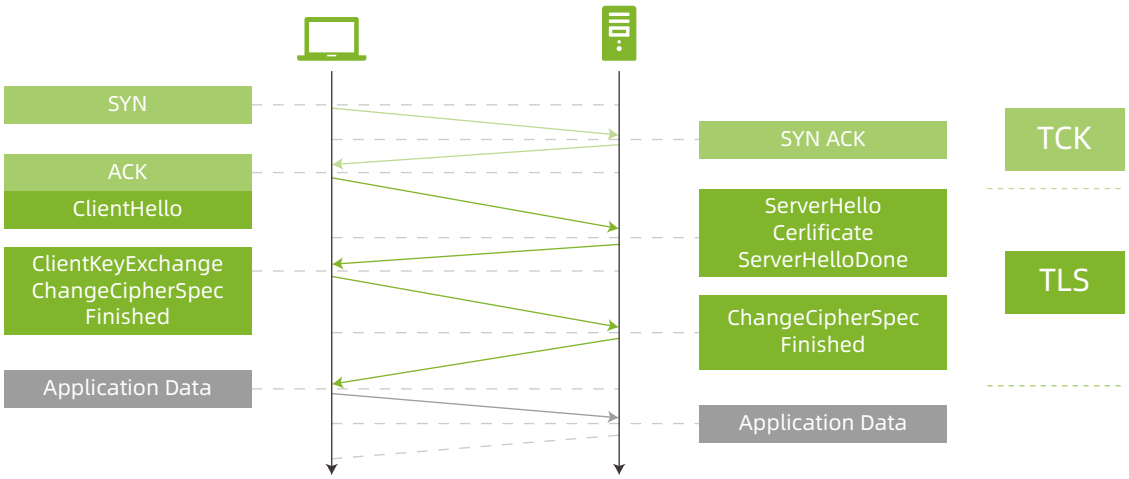
6.1.2 系统安全场景的加解密加速方案

概述

如今，科技社会日益发展，互联网连接万物，生活中的方方面面都离不开网络。尤其是近年来发展迅猛的移动互联网，让人们的生产生活方式发生了翻天覆地的变化，也从侧面证明网络在给我们日常生活提供便利的同时，需要更关注安全。介于安全与性能需兼顾的需求，龙蜥操作系统基于芯片新特性制定了加解密加速方案。



在网站访问过程中，HTTPS使用到SSL/TLS加解密过程。



场景挑战

我们知道密码学是为了保证信息的可靠性、机密性和完整性。密码学计算包括加解密、校验、签名等计算，消耗较多CPU资源。因此各硬件厂商推出过不少加速卸载方案：

- Intel在Leiwsburg PCH推出QAT，用于密码学和解压缩offload；
- ARM KP920推出的KAE加速卡，封装在CPU die，作为PCIe设备交互；
- ARMv8.0加入了crypto extension通过SIMD指令加速密码学计算；
- 云厂商Cloud HSM服务使用的SSl/TLS专用offload方案。

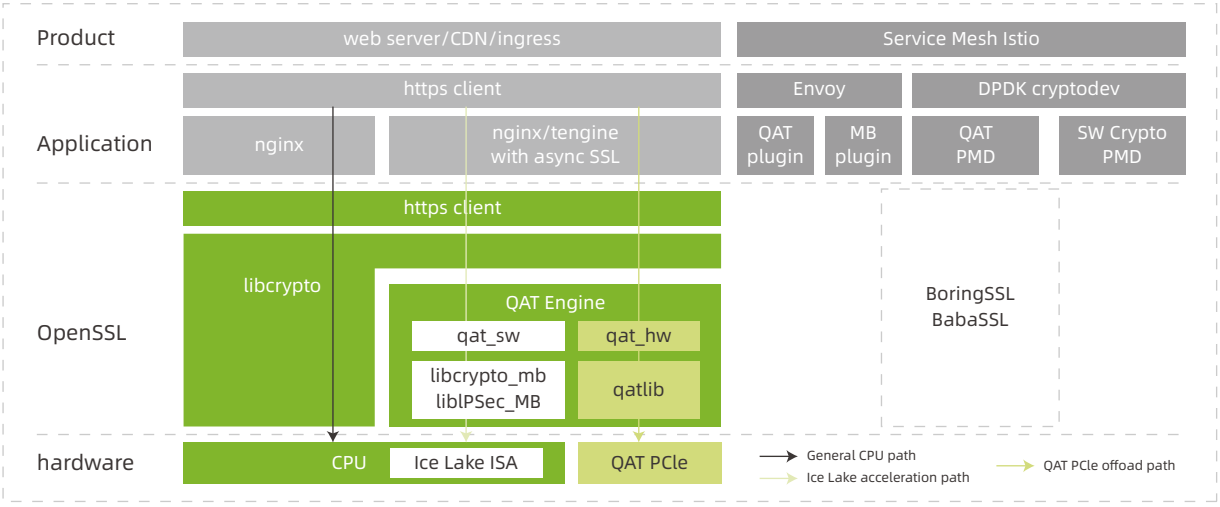
方案特色

不久前Intel发布了第三代英特尔® 至强® 可扩展处理器（代号Ice Lake），单核性能提升30%，整机算力提升50%以上。龙蜥操作系统（Anolis OS）和Intel深入合作，输出加速特性，进一步提升性能和效率。

新增指令集：在传统的AES-NI加速指令基础上，Ice Lake新增了基于Intel® Advanced Vector Extensions 512 (Intel® AVX-512)的Intel® Crypto Acceleration特性，包括Vector AES (VAES)、Integer Fused Multiply Add (IFMA大数计算)、Galois Field New Instructions (GFNI) 等。通过multi-buffer lib配合，加速AES, RSA, EC等密码学计算，可用于OpenSSL (BabaSSL), Nginx (Tengine), DPDK Cryptodev, CDN, dm-crypt, ISA-L等。

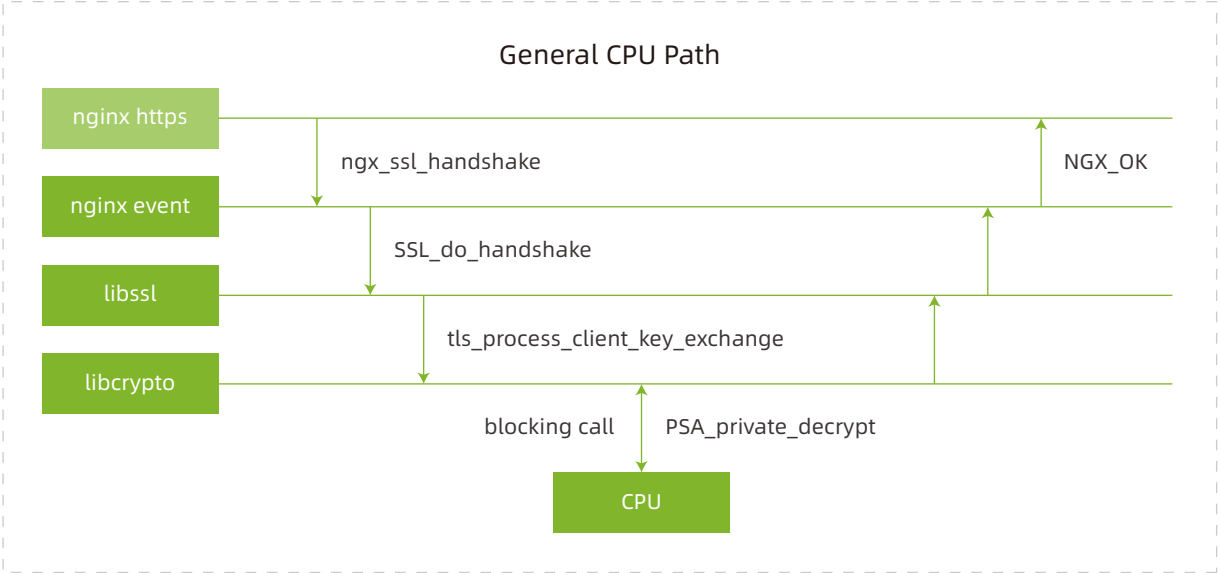
此外，Ice Lake补充支持了SHA extension (SHA-NI)，补足了前代Cascade Lake和Skylake的SHA256性能短板。结合Intel AVX512的vpternlog三元逻辑运算指令，进一步减少SHA256中一半的vpxor/XOR指令，可以达到和AMD Rome持平的单核SHA256性能。

加速软件栈：受益于以往的QAT方案，OpenSSL（BabaSSL，BoringSSL），Nginx（Tengine），DPDK，Envoy等主流软件已经支持offload SSL/TLS加速。Intel® Crypto Acceleration方案沿用和扩展了QAT的软件框架QAT Engine，使之从专用的硬件offload卡场景扩展到了通用的CPU指令加速场景，极大扩展了适用范围。



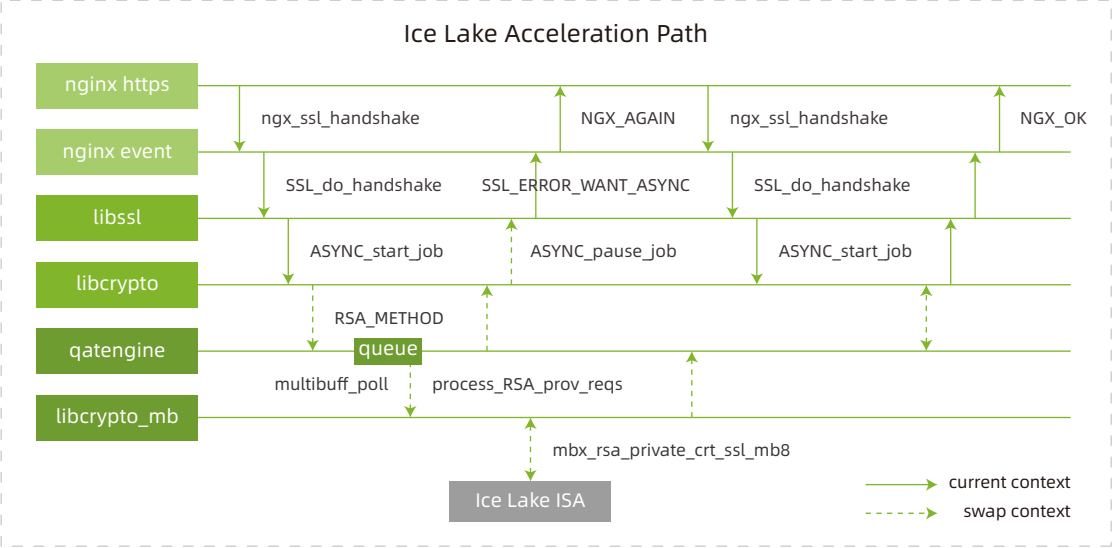
接下来我们详细对比各个方案的加速软件栈。

General CPU path：通用的CPU计算是最常见的SSL/TLS handshake软件栈，性能取决于libcrypto里对应的密码学算法在各型号CPU上的计算性能和效率。



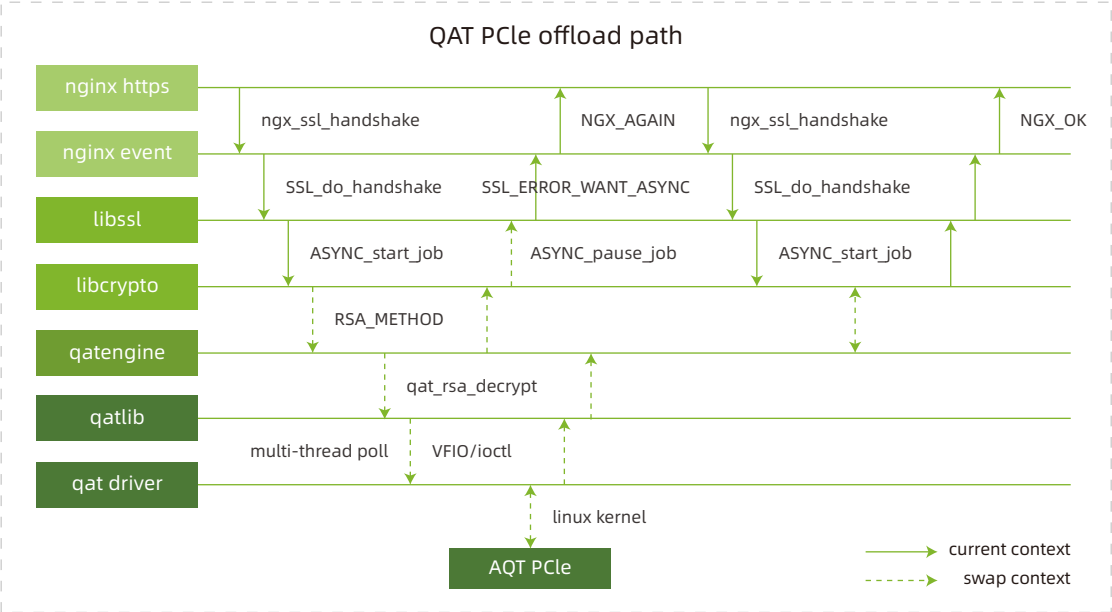
Ice Lake acceleration path：Ice Lake的Intel® Crypto Acceleration方案，从QAT软件栈演进而来，区别是底层改用multi-buffer lib中的Ice Lake加速指令进行密码学计算，不需要QAT PCIe设备，对应的驱动中断、系统调用、CPU内存消耗、硬件维护也可以省略。

相对于默认的同步模式软件栈，Ice Lake加速软件栈和QAT一样需要异步SSL集中批量请求，通过multi-buffer lib配合Intel AVX512指令的并行计算实现加速。因此Ice Lake加速软件栈会更长，但相对QAT软件栈更短。RSA大数计算消耗大量CPU，超过了SSL/TLS handshake全栈耗时的一半，异步SSL加速很有必要，也是典型的异步优化场景。



QAT PCIe offload path：QAT软件栈是Skylake上推出的SSL/TLS加速方案，经过演进派生出了上面的Ice Lake CPU加速方案。两种方案都基于QAT Engine，区别在底层是用CPU加速还是PCIe设备。例如Tengine已经支持QAT，则也支持Ice Lake加速，类似的还有Envoy, DPDK, ZFS，QZFS只用了QAT的解压缩加速）等。

QAT硬件集成在PCH主板（Lewisburg和Lewisburg refresh），或者作为独立的PCIe AIC卡。支持Ice Lake, Cascade Lake, Skylake三代服务器平台，和部分Intel Atom® C Series, Intel Xeon® D Series的专用SoC平台。下一代产品按照目前的规划会作为加速器集成封装到CPU die，从系统看还是PCIe设备。



同样可以用bpftrace分析QAT软件栈的处理流程和各阶段的latency分布。由于需要专用的QAT硬件平台，此处略过。

总的来说，QAT是专用的PCIe offload设备，可以协助CPU处理密码学和解压缩的大量计算消耗，释放CPU资源。但和所有PCIe设备一样需要专用的系统驱动、CPU内存消耗和硬件维护。QAT软件栈和延迟较长，计算能力上限也受系统和PCIe限制。Ice Lake强化了密码学计算能力，且同时支持QAT。CPU资源也比QAT更易获取和可扩展。

实践验证

我们关注通用CPU计算和Intel® Crypto Acceleration（qat_sw）方案的加速对比，同时参考QAT（qat_hw）方案的数据。

系统环境：基于阿里云ECS云服务器

实例规格		实例规格	qat_sw	qat_hw	OS
ecs.g7.xlarge	4核16GB	Intel Xeon (Ice Lake)	有	无配置 Intel QAT	Anolis OS 8.4
ecs.g5.xlarge		Intel Xeon (Skylake)	无效		
ecs.g6a.xlarge		AMD EPYC 7H12			
ecs.g6r.xlarge		Ampere Altra			

软件包：基于龙蜥操作系统Anolis OS 8.4

Package name	Version	Anolis OS 8.4	Notes
tengine	2.3.3	已发布	推荐应用
asynch_mode_nginx	1.18.0	已发布	参考demo
openssl	1.1.1g	系统自带	系统自带
babassl	8.2.1	已发布	兼容性验证，待正式发布支持
qatengine	0.6.6	已发布	openssl qat engine加速lib
intel-ipp-crypto-mb	0.5.3	已发布	Ice Lake密码学计算加速lib
intel-ipsec-mb	1.0.0	已发布	Ice Lake IPSec加速lib

兼容性：Anolis BabaSSL文档，BabaSSL兼容openssl-1.1.1的全部能力。下面在AnolisOS 8.4上验证BabaSSL兼容qatengine，支持Ice Lake加速和QAT offload。

OpenSSL性能：CPU指令加速属于底层特性，我们按照自下而上的方法逐层验证，首先对比OpenSSL底层的基础加解密算力。

对称密码：Ice Lake单核平均比Skylake高42%，比Rome高12%，比Ampere高20%：

1、符合Intel发布的Ice Lake单核1.4倍于Cascade Lake；

2、符合ECS发布的实时音视频通信加密的场景性能最大提升40%。

非对称密码：Ice Lake单核平均比Skylake高35%，比Rome高15%，是Ampere的3.5倍：

1、符合RSA特点：sign签名计算的消耗高运行慢，verify验证计算的消耗低运行快；

2、符合EC特性：sign签名计算比RSA快一个量级以上，verify验证计算比RSA慢；

3、Ampere的公钥密码计算性能较低（ARMv8 crypto extension）。

加速性能

```
# openssl speed -engine qatengine [algo] #包含qat_sw加速效果，无qat_hw offload
```

对称密码：Ice Lake加速效果符合预期的最高3.1倍：

1、block size大于256Byte时，AES-GCM能达到加速3~4倍。

2、block size小于64Byte时，AES-GCM有少量20%回退 。小于64Byte场景较少， Nginx默认ssl_buffer_size为16KB，可选4KB加速time to first byte响应。

非对称密码：部分场景可以达到预期的优化3~6倍：

1、同步模式没有指令加速效果，有30%左右单核通用计算的提升；

2、异步模式jobs大于8时有优化，RSA 2K sign 6倍，verify 3倍，ECDSA P256 sign 2倍，ECDH 3~4倍；

3、异步模式jobs小于8时有回退。参考2.3.2章节分析，低负载场景没有加速需求，应该关闭加速。

Nginx性能：在OpenSSL底层加解密算力基础上，继续验证典型的端到端场景Nginx SSL/TLS性能：

1、复用QAT加速Nginx方案（需要Nginx支持SSL/TLS异步加速）；

2、Nginx https SSL/TLS handshake的标准性能测试方法文档。

加速效果：Nginx SSL/TLS性能在单核Ice Lake加速后可提升2~3倍，是单核Skylake的3~4倍：

1、RSA有以上加速效果，ECDSA有60%通用计算提升。经分析是由于4核的client压力不足，在core通用提升基础上，nginx还有17%的加速效果。

2、Client connect数少于10时有回退，和OpenSSL异步低负载回退一致，低负载场景没有加速需求，应该关闭加速；

3、TLSv1.2的加速效果比TLSv1.3更好；

nginx conf				wrk		Ice Lake								Skylake			Ice Lake vs Skylake	
						normal			qat_sw			Q vs N		normal				
key	protocol	ssl_ciphers	ecdh_curve	thread	connection	req/s	KB/s	Latency (ms)	req/s	KB/s	Latency (ms)	req/s	Latency	req/s	KB/s	Latency (ms)	req/s	Latency
RSA 2048	TLSv1.2	AES128-GCM-SHA256	n/a	1	1	1086	259	0.0	121	20	0.0	-89%	+53%	748	221	0.1	-84%	-15%
					10	1580	376	1.1	997	238	0.4	-37%	-64%	1126	333	1.6	-11%	-75%
					100	1595	380	2.2	4697	1090	3.6	+194%	+60%	1129	334	3.5	+316%	+2%
		AES256-GCM-SHA384	n/a	1	1	1073	256	0.0	125	30	0.0	-88%	+66%	755	223	0.1	-83%	-8%
					10	1563	372	1.1	1003	239	0.4	-36%	-62%	1138	337	1.6	-12%	-73%
					100	1580	376	2.4	4532	1050	3.7	+187%	+53%	1142	338	3.6	+297%	+1%
		ECDHE-RSA-AES256-GCM-SHA384	X25519	1	1	928	221	0.0	126	30	0.0	-86%	+37%	651	193	0.1	-81%	-24%
					10	1413	337	1.6	499	119	0.3	-65%	-83%	1027	304	2.2	-51%	-88%
					100	1433	342	3.0	3523	839	4.0	+146%	+36%	1040	308	4.1	+230%	-3%
	TLSv1.3	TLS_AES_256_GCM_SHA384	n/a	1	1	885	211	0.1	129	31	0.1	-85%	+52%	621	184	0.1	-79%	-18%
					10	1348	321	2.3	574	137	0.3	-57%	-88%	979	290	3.2	-41%	-91%
					100	1360	324	4.0	3006	716	6.7	+121%	+68%	976	289	6.0	+208%	+12%
ECDSA P256	TLSv1.2	ECDHE-ECDSA-AES256-GCM-SHA384	X25519	1	1	1836	438	0.0	144	34	0.0	-92%	+33%	1150	230	0.1	-87%	-27%
					10	2932	699	0.6	722	172	0.5	-75%	-21%	1933	572	1.0	-63%	-53%
					100	2972	708	4.0	2984	706	4.3	+0%	+8%	1874	555	7.4	+58%	-42%
			P-256	1	1	1721	410	0.0	144	34	0.0	-92%	+25%	1072	317	0.1	-87%	-30%
					10	2781	663	0.6	877	209	0.3	-68%	-55%	1812	536	1.1	-52%	-74%
					100	2817	671	2.7	2817	671	4.2	+0%	+53%	1739	515	6.4	+62%	-35%
	TLSv1.3	TLS_AES_256_GCM_SHA384	n/a	1	1	1676	399	0.1	141	34	0.1	-92%	+45%	1045	309	0.1	-86%	-26%
					10	2563	608	0.8	997	238	0.5	-61%	-39%	1685	499	1.3	-41%	-61%
					100	2616	623	9.2	2575	614	8.2	-2%	-10%	1682	498	14.2	+53%	-42%

Tengine性能：Tengine从2.2.2版本开始支持async SSL，使用QAT offload SSL/TLS，支持的底层lib版本为：OpenSSL-1.1.0f和QAT_Engine-0.5.30。我们引入CPU加速后升级到：OpenSSL-1.1.1g和QAT_Engine-0.6.6，同时不需要QAT驱动和qatlib。

加速效果：Tengine SSL/TLS性能在单核Ice Lake加速后可提升2.7~3.2倍，是单核Skylake的3.5~4.2倍：

1、RSA有以上加速效果，ECDSA有60%通用计算提升。经分析是由于4核的client压力不足，在core通用提升基础上，tengine还有36%的加速效果；

2、tengine效果和asynch_mode_nginx类似；

3、仅对比TLSv1.2，TLSv1.3 tengine有专门的实践方案，此处略过。

tengine.conf				wrk		Ice Lake								Skylake			Ice Lake vs Skylake	
						normal			qat_sw			Q vs N		normal				
key	protocol	ssl_ciphers	ecdh_curve	thread	connection	req/s	KB/s	Latency (ms)	req/s	KB/s	Latency (ms)	req/s	Latency	req/s	KB/s	Latency (ms)	req/s	Latency
RSA 2048	TLSv1.2	AES128-GCM-SHA256	n/a	1	100	1529	1085	2.1	4884	3471	1.8	+219%	-15%	1146	813	3.2	+326%	-45%
				2	200	1498	1065	2.9	4783	3400	2.1	+219%	-28%	1139	809	6.5	+320%	-67%
					400	1528	366	5.0	4773	1147	3.6	+212%	-27%	1119	894	14.9	+327%	-76%
		AES256-GCM-SHA384		1	100	1484	1055	2.1	4708	3338	2.3	+217%	+6%	1147	814	3.2	+311%	-30%
				2	200	1464	1044	3.4	4613	3277	3.0	+215%	-13%	1139	809	6.2	+305%	-52%
					400	1464	350	9.4	4527	1085	4.6	+209%	-51%	1122	796	15.6	+304%	-70%
		ECDHE-RSA-AES256-GCM-SHA384	X25519	1	100	1358	963	3.0	2730	1935	2.4	+101%	-20%	1048	744	4.1	+161%	-40%
				2	200	1352	324	3.2	3482	833	4.8	+157%	+50%	1040	738	4.6	+235%	+5%
					400	1336	320	4.2	3576	856	9.2	+168%	+120%	1037	736	6.1	+245%	+51%
			P-256	1	100	1359	325	3.0	2786	667	1.9	+105%	-36%	1030	731	4.2	+170%	-54%
				2	200	1353	324	3.3	3670	881	6.5	+171%	+100%	1019	723	4.6	+260%	+40%
					400	1338	320	4.0	3722	891	12.4	+178%	+210%	999	709	6.9	+272%	+81%
ECDSA P256		ECDHE-ECDSA-AES256-GCM-SHA384	X25519	1	100	2699	646	3.0	2516	609	2.2	-6%	-28%	1606	1137	4.9	+59%	-56%
				2	200	3130	749	10.4	3023	723	4.0	-3%	-62%	2042	1454	12.4	+48%	-68%
					400	3075	736	20.6	2988	715	9.2	-3%	-55%	1957	1393	20.7	+53%	-56%
			P-256	1	100	3084	738	4.3	2725	652	2.3	-12%	-48%	1789	1270	5.2	+52%	-57%
				2	200	3701	881	8.0	3446	2447	5.0	-7%	-37%	2276	1618	11.0	+52%	-54%
					400	3620	866	15.2	3259	780	8.5	-10%	-44%	2213	1567	23.6	+47%	-64%

总结

经过实践分析，Intel® Crypto Acceleration在Anolis OS 8.4可以达到理想的SSL/TLS加速效果。

传统主流的AES和RSA加速效果最好，Ice Lake的底层OpenSSL计算AES和RSA达到上一代Cascade Lake的3.4和5.1倍。应用层Nginx SSL/TLS性能相应地加速到Cascade Lake的3.2倍，Tengine是3.3倍。整机多核性能随核数并发扩展，OpenSSL的AES和RSA加速效果增加到3.8和5.7倍，Nginx增加到4.3倍，Tengine是3.8倍。

新兴的ECDSA加速效果相对较小，底层的OpenSSL计算ECDSA和ECDH加速了2倍和4.3倍，但应用层Nginx SSL/TLS性能小幅加速1.5倍，Tengine是1.6倍，和上一代Intel® QAT的1.26倍效果类似。EC椭圆密码算法大幅优化了以往server端RSA的计算消耗，同等安全性保证下的OpenSSL计算ECDSA性能已经达到RSA的10倍了。因此SSL/TLS性能中的密码学计算瓶颈不再凸显，端到端效果小于底层加速是合理的结果。

兼容性方面，BabaSSL全兼容OpenSSL支持Intel® Crypto Acceleration。适配过QAT的应用也支持Ice Lake的CPU指令加速SSL/TLS，例如Tengine，Envoy，DPDK等。其他应用可以参考asynch_mode_nginx进行异步SSL改造。tengine承载大量应用稳定运行多年，我们推荐tengine作为Anolis OS 8.4的最佳SSL/TLS性能实践方案。

6.1.3 资源混部场景的内核隔离实现方案

概述

2014年，阿里巴巴开始了第一次探索混部，经过七年磨练，这把资源利用率大幅提升的利剑正式开始商用。通过从计算资源、内存资源、存储资源、网络资源等全链路的隔离以及毫秒级的自适应调度能力，通过智能化的决策与运维能力，支撑着内部百万级的Pod混部，不管是CPU与GPU资源，普通容器与安全容器，包括国产化环境各种异构基础设施，都能实现高效混部，这让核心电商业务生产集群成本下降了50%以上，同时核心业务受到的干扰小于5%。在商用化输出的版本里面，资源混部能力完全基于云原生社区标准，以插件化的方式无缝的安装在k8s集群作为输出交付形态。其中核心的OS操作系统层隔离能力，已经发布到支持多架构的开源、中立、开放的Linux操作系统发行版-龙蜥（Anolis OS）中。

场景挑战

资源混部就是将不同类型的业务在同一台机器上混合部署起来，让它们共享机器上的CPU、内存、IO等资源，目的就是最大限度地提高资源利用率，从而降低采购和运营等成本。混部通常是将不同优先级的任务混合在一起，例如高优先的实时任务(对时延敏感，资源消耗低；称为在线)和低优先级批处理任务(对时延不敏感，资源消耗高；称为离线)，当高优先级业务需要资源时，低优先级任务需要立即归还，并且低优先级任务的运行不能对高优先级任务造成明显干扰。

假设我们现在有一台服务器，上面运行了高优的在线业务，以及离线任务也在上面运行运行。在线任务对响应时间（Response Time，RT）的需求是很明确的，要求尽可能低的RT，故被称之为延迟敏感型（Latency-Sensitive，LS）负载；离线任务永远是有多少资源吃多少资源的，故此类负载被称之为Best Effort（BE），如果我们对在线和离线任务不加干涉，那么离线任务很有可能会频繁、长期占用各种资源，从而让在线任务没有机会调度，或者调度不及时，或者获取不到带宽等等，从而出现在线业务RT急剧升高的情况。所以在这种场景下我们一定需要必要的手段来对在线和离线容器进行资源使用上的隔离，来确保在线高优容器在使用资源时可以及时的获取，最终能够在提升整体资源使用率的情况下保障高优容器的QoS。

通过一个例子，说明在线和离线混着跑的时候，可能出现的情况：

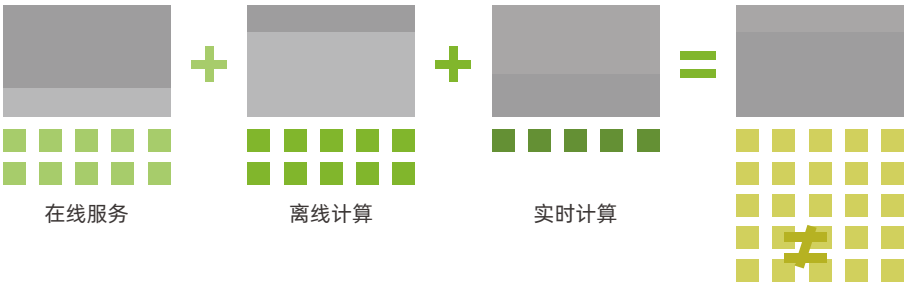
首先最有可能发生在离线竞争的，可能是CPU，因为CPU调度是核心，在线和离线任务可能分别会调度到一个核上，相互抢执行时间；
当然任务也可能会分别跑到相互对应的一对HT上，相互竞争指令发射带宽和其他流水线资源；
接下来CPU的各级缓存必然是会被消耗掉的，而缓存资源是有限的，所以这里涉及到了缓存资源划分的问题；
再接下来，假设我们已经完美解决了各级缓存的资源划分，后面还是有问题。首先是内存是CPU缓存的下一级，内存本身也类似，会发生争抢，对于在线和离线任务分别来说都是需要像CPU缓存一样进行资源划分的；

另外当CPU最后一级缓存（Last Level Cache，LLC）没有命中的时候，内存的带宽（我们称之为运行时容量，以有别于内存大小划分这种静态容量）会变高，所以内存和CPU缓存之间的资源消耗，是相互影响的；
然后我们假设CPU和内存资源都没问题，对于本机来说现在隔离已经做得很好了，但是在线高优的业务和离线任务的运行过程中都是和网络有密切的关系，那么很容易理解，网络也可能是需要隔离的；
最后，线上部分机型对IO的使用可能会发生抢占，我们需要有效的IO隔离策略。

以上就是一个很简单的资源隔离流程的思路，可以看到每一环都有可能会出现干扰或者竞争。

方案特色

把集群混合起来，将不同类型的任务调度到相同的物理资源上，通过调度，资源隔离等控制手段，在保障SLO的基础上，充分使用资源能力，极大降低成本，我们称这样的技术为混部（Co-loaction）。



混部在一起的任务有两个比较重要的特征：

1、可以划分优先级：一定需要优先级比较低的任务，它们能像水和沙子一样，随时能被赶走，而不会受到不可承受的影响，让优先级高的任务不受干扰。在线的特点是：峰值压力时间不长，对延时比较敏感，业务的压力抖动比较厉害，典型的如早上10点的聚划算活动，就会在非常短的时间内，造成交易集群的压力瞬间上升10几倍，对于稳定的要求非常高，在混部的时候，必须要保证在线的通畅，需要有极强的抗干扰能力。而计算任务的特点是：平时的压力比较高，相对来说计算量可控，并且延迟不敏感，失败后也可以重跑。至少需要几分钟跑完的计算任务，相对于几秒甚至几十秒的延迟，并不会产生严重的问题，正好可以承提起水和沙子的角色。

2、资源占用互补性：两种任务在不同的时间点对水位的占用不一样。如在线服务是，平时比较低，大促时比较高；凌晨比较低，白天比较高。而计算任务则反过来，平时比较高，大促时可以降级；凌晨非常高，白天却要低一些。

这种方式带来的成本节省是非常巨大的：假设数据中心有N台服务器，利用率从R1提高到R2，不考虑其他实际制约因素的情况下，节约X台，那么理想的公式是：

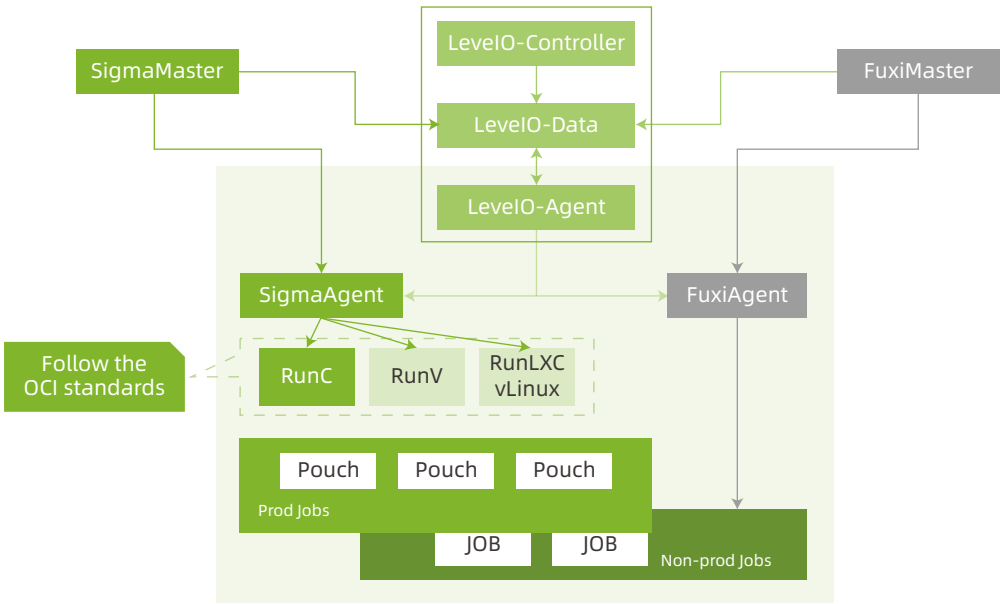
N*R1 = (N-X)*R2

=> X*R2 = N*R2 - N*R1

=> X = N*(R2-R1)/R2

也就是说如果企业有10万台服务器，利用率从28% 提升到40%，代入上述公式，就能节省出3万台机器。假设一台机器的成本为2万元，那么节约成本就有6个亿。

混部调度架构：



混部解决的核心问题：

在保证部署应用的服务等级目标SLO的前提下，充分利用集群中的空闲资源，来提升集群整体的利用率。

（a）在线服务类型应用（延时敏感型，Latency Sensitive）：

容器混合部署时的互相干扰（noisy neighbor）

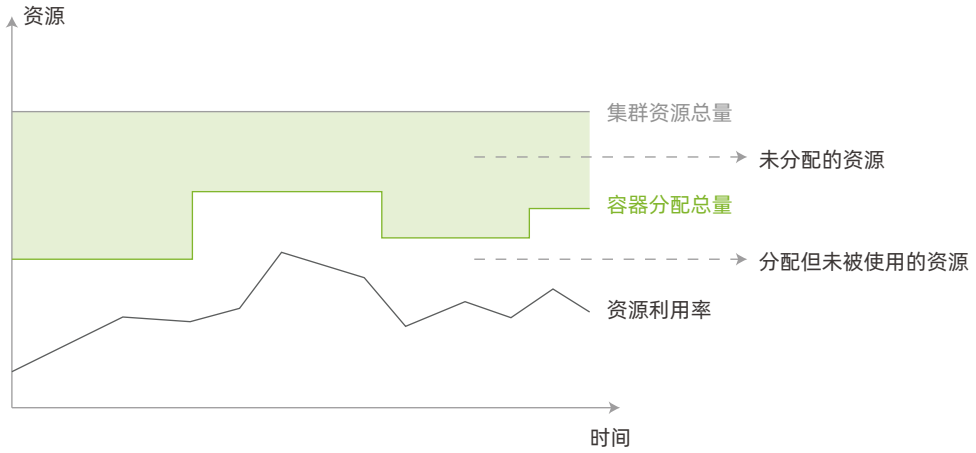
单机的资源竞应用对资源的敏感程度不同，在资源调度和单机服务质量上都需要精细化的管理策略

资源竞争引发应用响应时间出现抖动毛刺的现象，产生长尾问题（tail latency）

（b）作业类型批处理任务（计算密集型）：

分级可靠的资源超卖，满足差异化的资源质量需求

及时识别干扰源，避免影响LS应用



混部的资源隔离：

在混部中，放在首位的就是资源隔离问题，如果隔离问题没做好，竞争问题没解决，那就很容易引发线上的问题。轻一点，让用户的感官体验变差；重一点，引发线上故障，造成无法服务的恶劣影响。

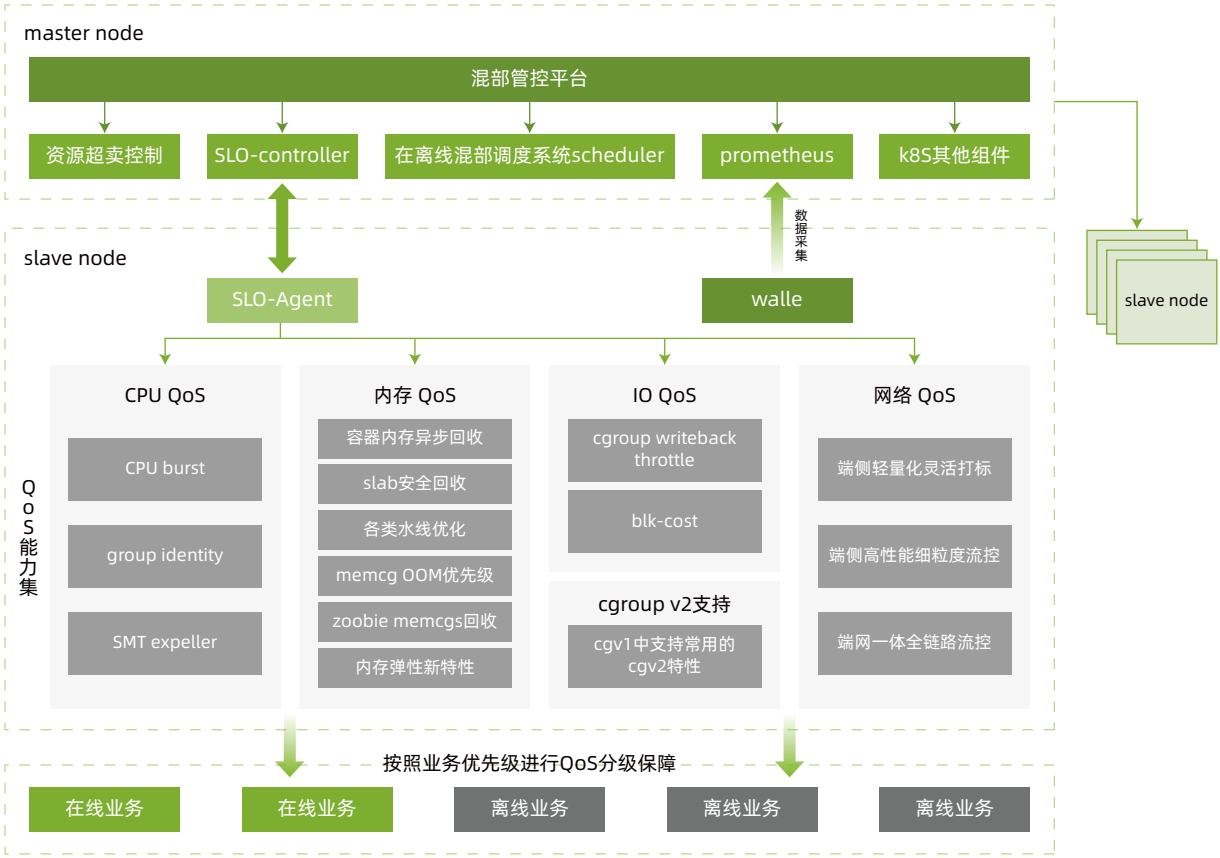
而解决资源竞争的问题，主要从两个方面出发：

1.调度：通过资源画像的技术，在资源真正发生竞争之前，就预测规划好，尽量减少这种情况发生的可能性。它是主动触发，可以不断优化，但延时较高。

2.内核：在资源真正发生竞争的极端情况下，根据任务的优先级，如何做到既能保障高优先级任务不受影响，又能控制影响到的低优先级任务伤害最低。它是被动触发，保底的必须手段，生效快。

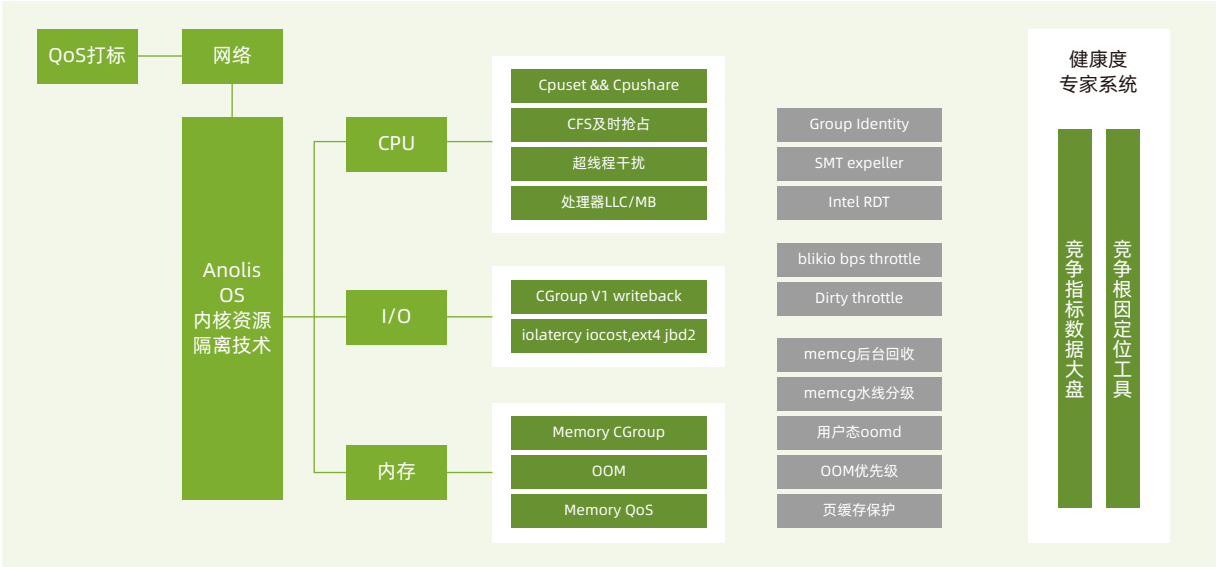
为了满足混部的需求，在服务器单机维度的内核资源隔离技术是最为关键的一项技术。龙蜥云内核（ANCK）在资源隔离的技术上深耕多年，并且在行业中处于领先地位，这些内核资源隔离技术主要涉及内核中的调度、内存和IO这三大子系统，并且在各个子系统领域根据云原生的混部场景进行了深入的改造和优化，关键优化包括但不限于：cpu group identity技术，SMT expeller技术，基于cgroup的内存异步回收技术等。

下图是资源隔离能力在整个混部方案中的位置：



内核资源隔离技术主要涉及内核中的调度、内存和IO这三大子系统，这些技术基于Linux CGroup V1提供资源的基本隔离划分以及QoS保障，适用于容器云场景，同时也是大规模化混合部署方案所强依赖的关键技 术。

除了基本的CPU、内存和IO资源隔离技术外，龙蜥内核还实现有资源隔离视图、资源监控指标SLI（ServiceLevelIndicator）以及资源竞争分析工具等，提供一整套集监控、告警、运维、诊断等整套的资源隔离和混部解决方案，如下图所示：



弹性容器场景的调度器优化

如何保证计算服务质量的同时尽可能提高计算资源利用率，是容器调度的经典问题。随着CPU利用率不断 提升，CPU带宽控制器暴露出弹性不足的问题日趋严重，面对容器的短时间CPU突发需求，带宽控制器需要对容器的CPU使用进行限流，避免影响负载延迟和吞吐。

Anolis OS中的CPU Burst技术是一种弹性容器带宽控制技术，在满足平均CPU使用率低于一定限制的条件下，CPU Burst允许短时间的CPU突发使用，实现服务质量提升和容器负载加速。

在容器场景中使用CPU Burst之后，测试容器的服务质量显著提升，如下图所示，在实测中可以发现使用该特性技术以后，RT长尾问题几乎消失。

总结

龙蜥云内核在资源隔离的关键技术使社区用户有能力在云原生混部场景中根据业务特点设计最优解决方案，可有效提高用户的资源使用率并最终降低用户资源的使用成本，非常适用于容器云混部场景，同时也是大规模化混合部署方案所强依赖的关键技术。

6.1.4 云原生应用场景下的镜像分发加速方案

概述

容器镜像是容器技术的基础设施之一，是容器运行时文件系统视图基础。而在云原生应用场景中，应用的构建、分发、部署和运行都需要与容器镜像的生命周期紧密地绑定在一起，可以说，没有容器镜像，就没有当代的云原生应用架构。

虽然容器镜像在云原生领域中至关重要，但自它诞生以来，镜像设计本身并没有多少改进，在在云原生技术被广泛、稳定地应用之后，传统容器镜像也逐渐表现出了一些缺陷：

- 启动速度慢，在镜像体积较大的情况下，容器的启动会很长；
- 较高的本地存储成本和镜像仓库存储成本；
- 集中式的镜像存储仓库，对镜像仓库的资源有了很高的要求。

而以上的问题在生产集群规模庞大的时候，会带来以下问题：

- 集群弹性效果差：在一些突发流量的场景下，用户往往需要在短时间内拉起大量的容器用来应对流量高峰，典型的业务场景如电商双11大促、社交平台热点等等。而由于容器启动时间长，往往应对高峰的时候准备时间需要很长，进而影响集群业务表现；
- 集群运行性能差：在大数据、AI场景下，用户往往需要同时运行多个容器，不同容器计算结果聚合形成最终的计算结果。而由于容器启动时间长，在单一时间内完成的任务数量就会有限，影响大数据、AI集群的吞吐能力，集群运行性能不加；
- 成本高抬：上述的弹性效果差、运行性能差，往往需要通过扩大集群规模以满足业务的诉求，简介抬升了集群成本，另外，由于存储成本、仓库成本的太高也会影响整体的运行时成本。

龙蜥社区（OpenAnolis）的云原生特殊兴趣小组对容器镜像技术进行了深层次的研究、探索和开发，与上游社区深度配合，一同推出了镜像分发加速解决方案 --- Nydus + Dragonfly，通过先进的技术，解决云原生应用场景下传统容器镜像带来的启动慢、资源浪费等问题，基于龙蜥社区的镜像分发加速方案，用户可以在应用服务弹性场景、大数据AI场景等主流的云原生应用场景中获得集群弹性性能、运行时性能的极大提升，在获得集群表现竞争力的同时，降低集群的运行成本。

场景挑战

容器在使用之前，需要容器运行所在的物理机上执行下面3个步骤

- download镜像，
- unpack镜像，
- 使用overlayfs将容器可写层和镜像中的只读层聚合起来提供容器运行环境。

其中，download镜像时需要download整个镜像，再加上download镜像本身受限于网络带宽的影响，当容器镜像size在到几个G时，下载时间会较长，而unpack大体积的包往往也要耗费不少的时间。而由于容器镜像的版本机制和分层机制，生产中镜像的规模会越来越大，比如某电商平台内部的容器镜像体积都在1G以上，因此集群内的弹性能力受到了很大的限制，在应对大促时，往往需要提前部署好大量的容器，这些容器所耗费的资源都带来较大的成本抬升。

因此，如果不解决容器镜像的使用方式、镜像格式，很难在根本上解决弹性性能问题。

另外，在download容器镜像的服务器成为容器镜像仓库，一般是集中式的仓库，既集群里所有的机器都会从这个（这批）服务器上拉取镜像。在集群规模很大（万级node），弹性并发情况下，数万节点同时从仓库拉取镜像，往往会造成镜像仓库DOS，进而引发故障，所以在大规模集群内提升镜像分发效率也有很大的挑战。

方案特色

2016年的usenix的论文Slacker: Fast Distribution with Lazy Docker Containers中曾发表数据，在容器启动的过程中，平均只需要读取镜像数据中的7%不到的数据，因此在实际应用过程中，通常不需要全量拉取数据我们就可以完成业务的发布过程。

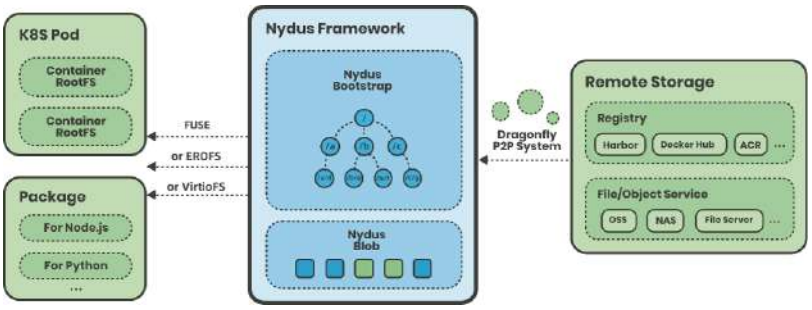
基于这个研究，Nydus镜像格式提出了按需加载的方案，既仅下载运行过程中需要的文件，对于冗余的镜像内容不予下载，进而减少容器运行过程中的数据下载量，减少镜像下载时间，提升容器启动性能，进而提升集群弹性性能。

为了达到这个效果，Nydus提出了一种新的镜像格式，做了如下创新：

- 镜像元数据/数据分离，用户态按需加载与解压；
- 更细粒度的块级别数据切割与去重；
- 扁平化元数据层（移除中间层），直接呈现Filesystem视图；
- 端到端的文件系统元数据树与数据校验。

基于以上的创新，用户在启动容器时，仅需要下载少量的metadata数据和manifest数据就可以实现容器的启动，然后在真正读取文件的时候从远端拉取数据，大幅度减少了启动过程中的数据拉取量，提升了启动性能。

同时，引入了Dragonfly作为集群间镜像P2P分发方案，将镜像数据由南北向传输转变为东西向传输，避开了容器镜像仓库瓶颈，大幅度提升集群镜像分发的效率，也提升了镜像分发的稳定性。

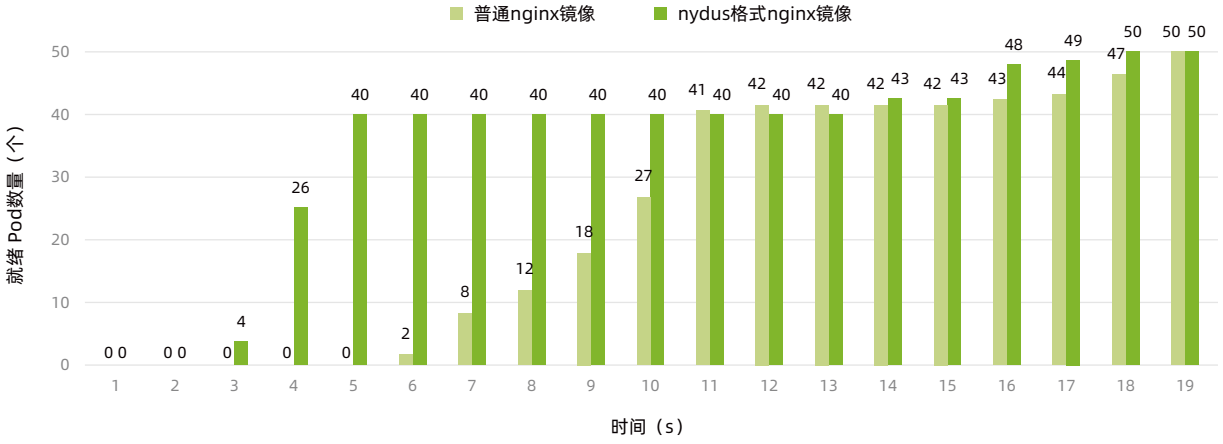


实践验证

使用Nydus + Dragonfly在快速弹性场景下的模拟效果对比。

在电商应用或者社交软件中，经常会遇到突发流量的场景，如果用户将业务托管在云厂商上，通常的做法是在云厂商上快速地创建新的VM节点，并在节点里面部署容器，这时，往往节点内并没有缓存镜像，镜像拉取会成为扩容速度的关键影响因素之一。本实践通过在一个具备50个全新node的k8s集群里面，并发启动50个nginx容器，用于模拟突发流量场景下的集群极速扩容场景。

这里我们用deployment的方式部署50个nginx容器，nginx镜像的大小约为70M。我们创建deployment之后，不断探测ready的POD数量直到在线容器数量为50，用来检测启动完成的容器的数量。本实践中，我们将普通容器镜像和Nydus容器镜像的每秒启动完成PDO时间做一个对比。



可以看到，使用nydus镜像达到50 ready的时间比使用normal镜像少1s，同时，在deployment创建4s之后，nydus的ready数量就已经达到了40个，而使用normal镜像的场景，还没有容器ready，到了接近11s才达到了40个ready的pod，以80%的扩容水位作为安全水位的话，使用nydus相比使用normal镜像可以减少一半以上，集群的弹性性能可以提升100%。

使用Nydus + Dragonfly在大数据/AI Job场景下的模拟效果对比。

在AI或者大数据计算中，一次计算往往需要切分成多个任务独立运算，在计算完成之后再进行数据聚合，因此需要每一个单独的计算结果都完成了，才能聚合出最终的结果。在云原生应用中，既每一个独立的任务用一个POD来承载，只有每一个POD都完成了才算最终完成，K8S将这种计算模型用Job类型来抽象。而AI或者大数据计算通常会有新的计算模型出现，每一个计算模型的出现会在镜像中集成新的数据，因此在计算启动的时候，往往需要进行镜像的下载，因此镜像也成为了Job类型运算的关键影响因素之一。

本实践我们模拟一个经常会出现的大镜像的场景，镜像中存放了用来做AI计算或者大数据计算的数据，用来展示使用了Nydus以及Dragonfly之后，镜像加速带来的集群镜像性能的提升效果。

首先，我们构建一个容器镜像：

```
FROM registry.anolis.com/library/anolisos:8.6
RUN base64 /dev/urandom | head -c 1073741824 > /root/origin.txt
RUN dnf install -y diffutils
COPY test.sh /root

CMD ["/root/test.sh"]
```

在这个镜像中，以anolisos:8.6为基础镜像，在镜像中生成了一个1G大小的随机文件origin.txt，同时安装了diff工具。然后在容器启动的时候，运行/root下面的test.sh脚本，脚本内容如下：

```
#!/bin/bash

base64 /dev/urandom | head -c 1073741824 > /root/new.txt
diff /root/origin.txt /root/new.txt >diff.txt
exit 0
```

脚本的内容也很简单，再生成一个1G的随机文件new.txt，然后比较origin.txt和new.txt，将两个文件对比，对比的结果再放置到diff.txt中。

这个testcase包含了几个部分：

- 在镜像中存储了大量的，需要用于计算的数据 ---> 镜像加载和镜像数据读取
- 在容器运行过程中生成新的数据，并与镜像数据进行比较 ---> 计算
- 再把数据存回到容器汇总 ---> 存储

本次实践，以1个并发、5个并发、50个并发为例，对比不同并发能力下，普通镜像跟nydus+dragonfly的性能差异，通过普通镜像和Nydus + Dragonfly镜像的Job运行时长对比得出不同方案的性能差异。

普通镜像的Job类型的计算时间如下：

```
[root@iZ2zei2xgj5fqblburoxwwZ ~]# kubectl get jobs
NAME                 COMPLETIONS   DURATION   AGE
job-test-normal      1/1           63s        9m53s
job-test-normal-5    5/5           63s        7m53s
job-test-normal-50   50/50         2m25s      5m23s
```

单并发场景下，63s完成job
5并发场景下，63s完成job，说明仓库还没有被打爆
50并发场景下，145s完成job，说明仓库已经成为瓶颈

随机挑选3个不同job的pod，可以印证我们的猜测，50并发下，原本20s可以拉下来的镜像花了接100s：

```
[root@iZ2zei2xgj5fqblburoxwwZ ~]# kubectl describe pods job-test-normal--1-ps8xr | grep pulled
  Normal   Pulled     13m                kubelet            Successfully pulled image "registry.anolis.com/library/nydus-dragonfly-test:latest" in 19.707755435s
[root@iZ2zei2xgj5fqblburoxwwZ ~]# kubectl describe pods job-test-normal-5--1-r2sqz | grep pulled
  Normal   Pulled     11m                kubelet            Successfully pulled image "registry.anolis.com/library/nydus-dragonfly-test:latest" in 18.896308145s
[root@iZ2zei2xgj5fqblburoxwwZ ~]# kubectl describe pods job-test-normal-50--1-dlmbp | grep pulled
  Normal   Pulled     8m4s               kubelet            Successfully pulled image "registry.anolis.com/library/nydus-dragonfly-test:latest" in 1m41.91277142s
```

接下来我们测试nydus + dragonfly的情况，方案升级之后Job的计算时间如下：

```
[root@iZ2zei2xgj5fqblburoxwwZ ~]# kubectl get jobs
NAME                 COMPLETIONS   DURATION   AGE
job-test-nydus-1     1/1           41s        5m36s
job-test-nydus-5     5/5           51s        3m36s
job-test-nydus-50    50/50         65s        65s
```

单并发场景耗时41s
5并发场景耗时51s
50并发场景耗时65s

在这个use case中，需要访问到大量的镜像内的文件，在1并发、5并发场景下，虽然没有打到Harbor的瓶颈，但是由于使用nydus减少了镜像数据的传输，所以整体的job运行时间也有了较大的提升，提升比例基本与镜像中base image的大小相当。

而在50并发场景下，并发pull已经打爆了Harbor带宽，这时候Nydus的按需加载 + dragonfly的P2P就很好的体现了效果，整体的Job运行时长缩减了一半以上，意味着集群的吞吐能力提升了接近100%。

并发数	普通镜像Job完成时间	Nydus+dragonfly镜像 Job完成时间	性能提升比例
1	63s	41s	53%
5	63s	51s	23%
50	145s	61s	123%

总结

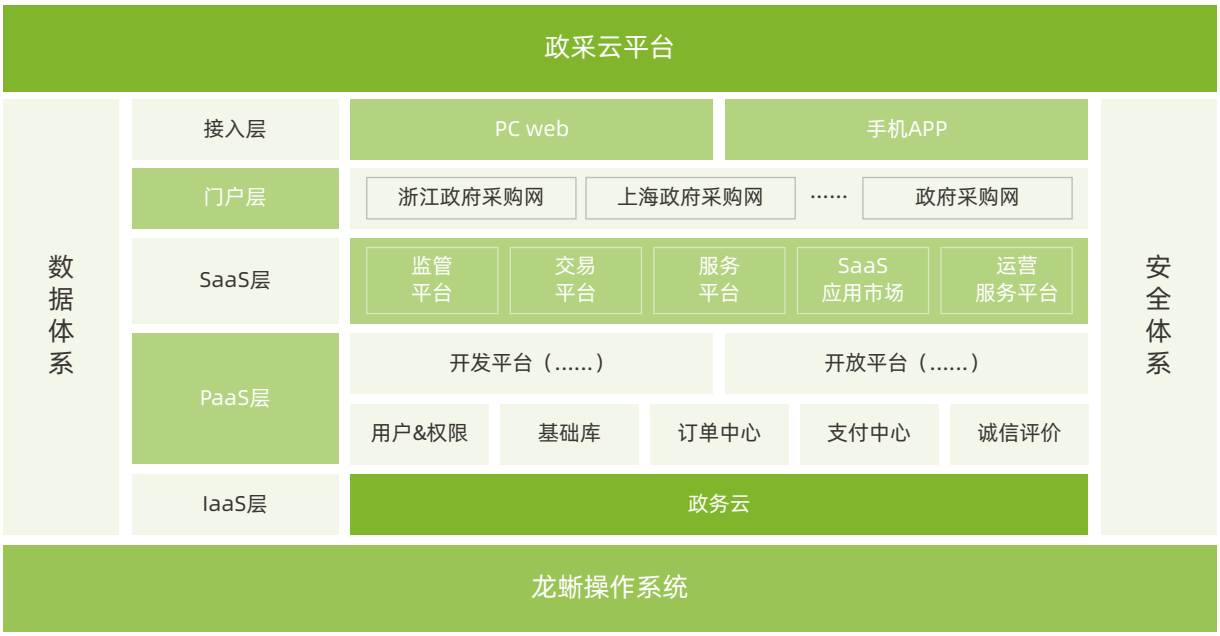
基于龙蜥提供的Nydus + Dragonfly解决方案，用户在云原生场景中可以大幅度提升镜像的分发效率，不仅可以帮助用户提升自己的集群业务表现，还能够降低集群的运行时成本。此方案已经在云厂商、互联网企业中有了广泛、稳定地应用，持续地为龙蜥用户贡献业务价值。

6.2 优秀案例

6.2.1 龙蜥社区助力全国首个政府采购云平台完成CentOS迁移 - 政采云

政采云有限公司成立于2016年，专注服务于「互联网+政企采购」，助力政企采购数字化，搭建了全国首个政府采购云服务平台。此次参与国产化操作系统适配的是该平台的核心业务系统，包括电子招投标系统、电子卖场系统以及监管平台系统，其中，电子招投标系统获财政部推荐，列入“政府采购项目电子化交易系统介绍”名单；电子卖场系统经财政部《浙江省政府采购电子卖场试点工作方案》批复同意，肩负着全国电子卖场试点的重任。

综合国产化需求和CentOS停服的考虑，政采云平台决定尽快启用龙蜥操作系统，将原有环境CentOS 7.6升级至Anolis 8.2，且保证完全通过兼容测试。期间，龙蜥社区提供龙蜥操作系统（Anolis OS）、迁移工具和技术服务，成立技术专家小组，制定出多种备份方案，确保迁移工作正常进行，采用业务扩容方式，按照预定操作计划和回滚方案，开展迁移工作。值得一提的是，政采云平台原应用系统所需软件包和目标操作系统中软件包的兼容性和依赖包存在差异，经过对系统所需的软件及中间件进行兼容性测试，包括JDK、Nginx、ElasticSearch、RocketMQ、Zookeeper、MySql等，以及对基准性能的测试，磁盘IO，网络IO以及中间件等性能压测的表现均于同配置下CentOS保持一致，系统内核版本的升级对于应用层影响实现可控。最终，项目采购系统试点替换完成，运行状态稳定，性能无明显差异，替换前后几乎无影响。这次顺滑迁移验证了基于龙蜥操作系统的解决方案，可以在政府项目采购应用场景下替代国外主流产品，为面临同样问题的各政企/行业单位提供了最佳实践样板。



政采云有限公司运维总监苏木表示：“我们的系统主要服务于政务业务，对安全、稳定、智能的要求都比较高，从运维的角度来看，我们关心的是交付标准化和生态的活跃度问题。很高兴看到龙蜥操作系统在国产化生态建设上的前瞻性贡献，特别是对于现有中间件和业务系统的高度兼容性，极大的降低了我们的迁移和适配成本，坚定了我们共建国产化生态的信心，我们也希望能在国产化生态上共享自己的力量。”

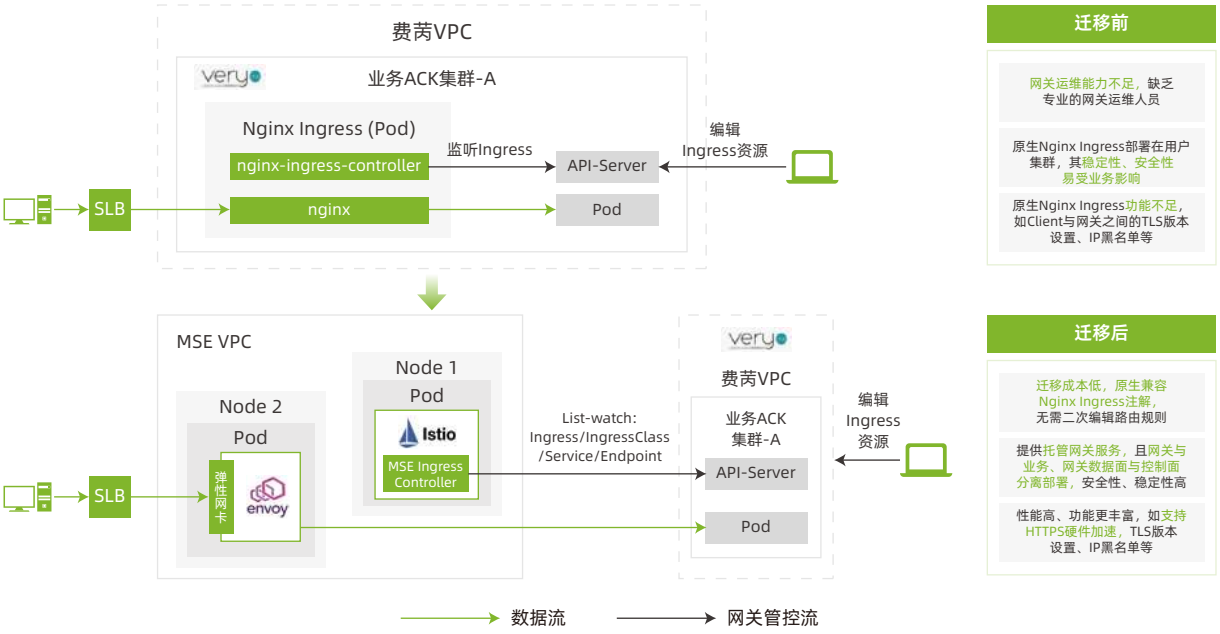
“实现国产化操作系统的替换只是第一步，“政采云CTO铁燕表示，“相信在产业各方积极努力下，未来国产化生态将日益丰富和健全，政采云将更多的引入国产化中间件和数据库，沉淀相关基础工具链，以打造国产化交付和运维标准为方向，将信息系统国产化坚定的走下去。“国产化适配是一个系统性工程，越早确定规划路径和方向，技术上的沉淀就越深，业务上的灵活性就越强，也能更好的适应行业的变化。龙蜥社区在版本发布、生态完善和技术创新的路线上持续演进，助力用户完美平滑地迁移至龙蜥操作系统(Anolis OS)，融入国产化生态，满足CentOS停服后的各领域、各行业用户的使用习惯和需求。

6.2.2 龙蜥社区助力云原生网关实现TLS硬件加速 - 上海费芮网络科技

网络信息传输的可靠性、机密性和完整性要求日渐提升，HTTPS协议已经广泛应用。HTTPS的SSL/TLS协议涉及加解密、校验、签名等密码学计算，消耗较多CPU计算资源。因此CPU硬件厂商推出过多种加速卸载方案，如AES-NI, QAT, KAE, ARMv8安全扩展等。

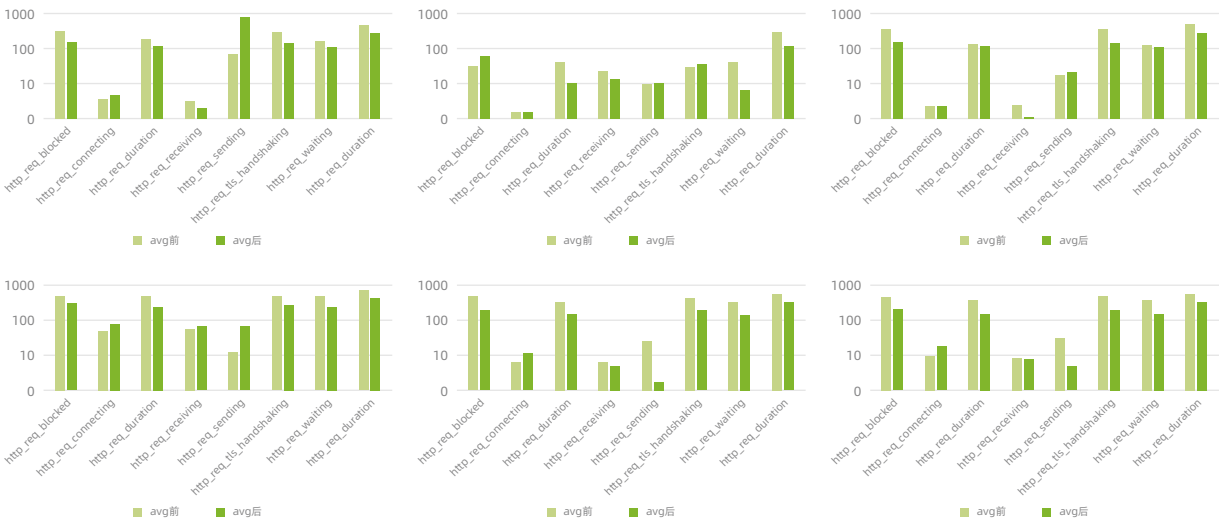
业界软件生态在优化HTTPS的性能上也做了诸多探索，传统的软件优化方案有Session复用、OCSP Stapling、False Start、dynamic record size、TLS1.3、HSTS等, 但软件层面的优化无法满足流量日益增长的速度，CPU硬件加速成为业界一个通用的解决方案。

上海费芮网络科技有限公司之前一直使用Nginx Ingress，使用过程中遇到运维成本高、安全差、原生功能弱等痛点，期望能够找到一款替代产品；在接触MSE云原生网关后，在上线前的测试过程中对于HTTPS安全加速功能非常认可，测试验证开启后的加速效果非常明显；结合网关提供的Nginx Ingress注解兼容功能 + HTTPS安全加速两个差异功能，用户最终选择使用MSE云原生网关来替代Nginx Ingress网关。



加速效果

注：测试采用HTTPS短连接且关闭session ticket复用。



加速后：

- 1C2G压测HTTPS QPS从1004提升到1873，提升约86%。
- TLS握手RT从313.84ms降到145.81ms，下降一倍。

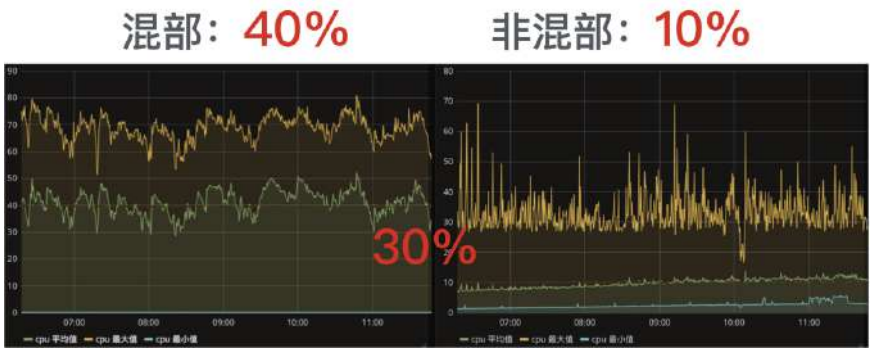
方案优点

- 无需独立专用的硬件支持，运维成本低且易于弹性扩缩容。
- 通用CPU加速特性的适用场景更广泛。

6.2.3 龙蜥社区助力阿里数据中心大幅降低成本

每年双十一创造奇迹的背后，是巨大的成本投入。为了完成对流量峰值的支撑，我们需要大量的计算资源，而在平时，这些资源往往又是空闲的。另一方面，为了在极端情况下，如机房整体断电等还能保障阿里巴巴的业务不受损失，也需要在全国各地建立冗余资源。而且就算是一天当中，在线服务的负载也是不一样的，白天一般情况下要比凌晨高得多。根据盖特纳和麦肯锡前几年的调研数据，全球的服务器的CPU利用率只有6%到12%。即使通过虚拟化技术优化，利用率还是只有7% - 17%，而阿里巴巴的在线服务整体日均利用率也在10%左右。

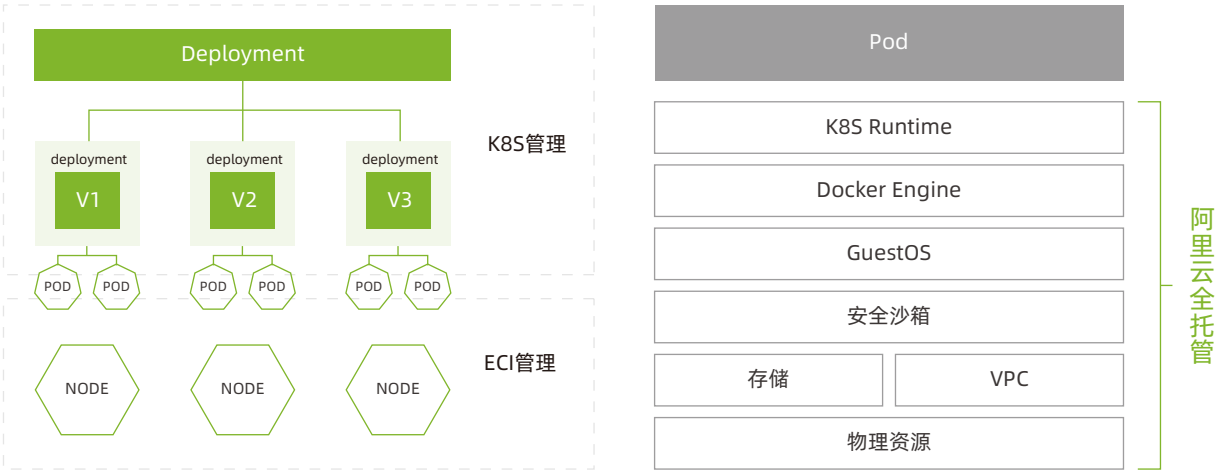
另一方面，全球从IT时代全面走向了DT时代，现在又在向更深入的AI时代迈进。各各样的大数据处理框架不断涌现，从Hadoop到Spark，从Jstorm到Flink，甚至包括深度学习框架 Tensorflow 的出现，成千上万的数据分析背后是大量的计算任务，占用了大量的计算资源。由于计算任务占用的计算量很高，CPU水位通常在50%-60%以上，不同于在线服务，计算任务的峰值通常出现在凌晨，水位甚至能达到70%以上。所以我们往往就会建立独立的计算任务集群。



混部能产生这么大的帮助，可是业界能使用在生产的没有几家公司，其原因也非常简单，第一个是规模，第二个是技术门槛。当你机器规模不够大的时候，显然意义不大。而在技术上，计算型任务通常都可以把利用率跑到很高，如果计算型任务和在线型业务运行在同一台机器上，怎么避免计算型任务的运行不会对在线型业务的响应时间等关键指标不产生太大的影响呢，这个需要在技术上有全方位的突破，而阿里巴巴从无到有，花了4年多的时间才让这项技术在电商域得以大规模落地。

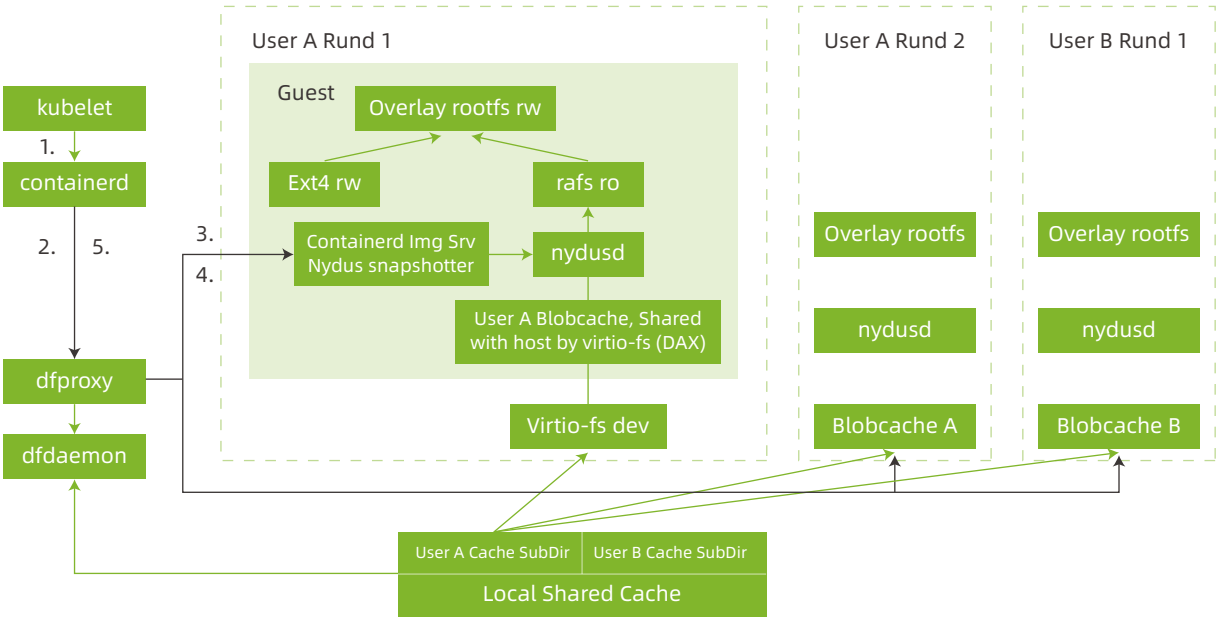
6.2.4 龙蜥社区助力阿里云Severless容器产品获得出色的弹性产品能力

阿里云弹性容器实例（简称ECI，Elastic Container Instance）是阿里云结合容器和Serverless技术提供的容器运行服务，是一款Serverless容器产品。通过使用ECI，在阿里云上部署容器时，无需购买和管理云服务器ECS，可以直接在阿里云上运行Pod和容器，省去了底层服务器的运维和管理工作。简单来说，一个ECI就是一个Pod，可以被K8s编排和调度。阿里云弹性容器实例特别适用于突发的业务流量，或者短周期的任务运行。那么ECI和客户自己去购买ECS，在ECS里运行Docker最大的区别在于如果使用ECI，整个容器的运行时会将由阿里云来运维。

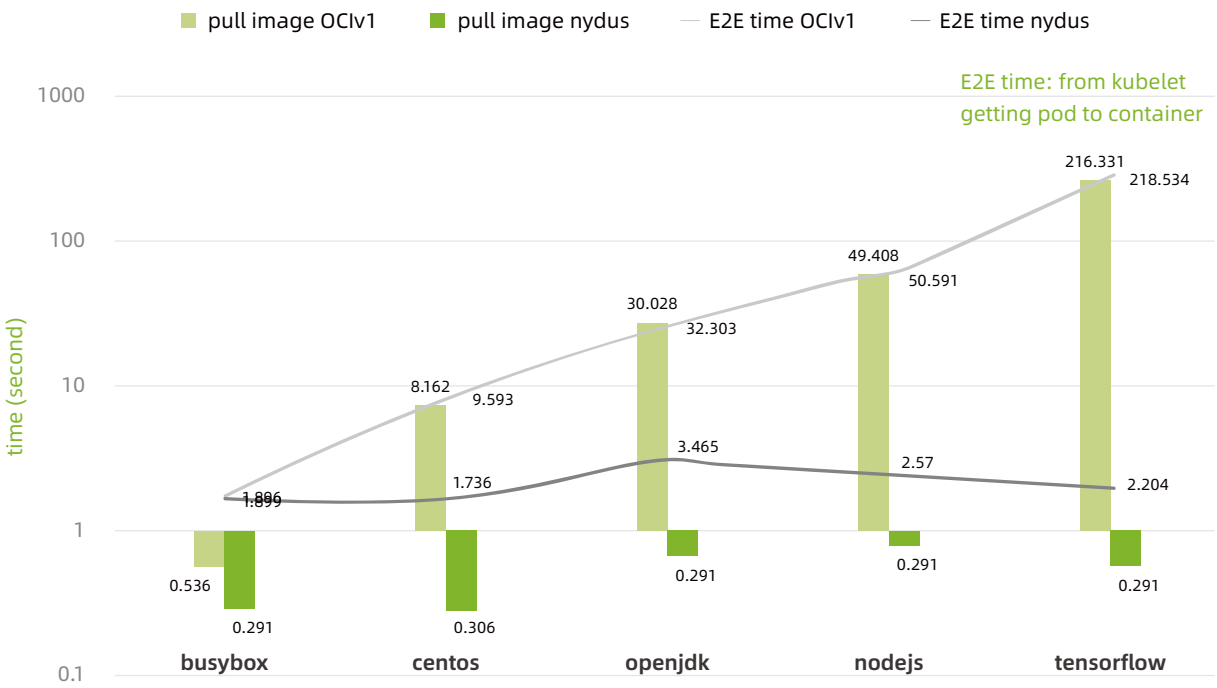


由于ECI一款云计算产品，天然需要在一台物理机上支持不同用户的不同容器，而且某一台机器上有可能启动什么容器是完全无法预测的，因此每一个容器在启动的时候都需要进行镜像的拉取，而ECI被广泛使用在突发业务流量，突发弹性的场景，因此经常出现在同一时间内多个容器从同一个镜像仓库拉取镜像的现象，因此非常容易引发由于网络带宽不足或者仓库DOS导致容器启动时间长或者启动失败的问题。

基于ECI遇到的挑战，ECI开发团队采用了龙蜥社区提供的Nydus + Dragonfly方案，使用镜像的按需加载以及集群内的镜像内容P2P分发。



通过减少不必要的镜像内容的传输以及P2P能力，ECI容器实例在镜像拉取的时间上有了大幅度的降低。



根据实测，某集群内的镜像拉取时间由p99 6s下降到了p99 1s，更是实现了ECI Job类实例在6s内扩容3000个实例的产品能力，在业界均处于领先位置。

07
社区风采

7.1 特色活动

7.1.1 面向生态伙伴-“走进合作伙伴”系列

「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，吸引来自70+企业的1000+开发者参与。

其中，走进Intel MeetUp首次联动媒体线上直播，线上观看人次上万，收回调研问卷近500份，直播间上万人参与互动，调研显示，此活动具备“技术干货”、“内容丰富”、“易懂有用”的标签，活动满意度达87%。龙蜥社区当前已有21家理事单位、250+家合作伙伴，通过一次次的无间合作，龙蜥社区生态伙伴的链接更加紧密，共同努力去打造一个面向未来的开源操作系统生态。

龙蜥社区走进合作伙伴图

龙蜥社区走进合作伙伴图

7.1.2 面向高校师生

协同育人项目

龙蜥社区积极参与各大赛事，为大赛设计赛题、提供导师、培训技术、参与评审等，将产业发展思考融入赛事，加强操作系统领域人才的培养。上半年，龙蜥社区走进了北大开源实践课，为多所高校生介绍培训龙蜥操作系统，参与了中科院开源之夏、阿里巴巴编程之夏、教育部产学合作协同育人项目、大学生系统能力大赛操作系统设计赛、“互联网”大学生创新创业大赛等等，在学术研究、课程合作、赛事赞助等多方面保持与高校的密切合作。

2022年初，龙蜥社区走进北京大学的课堂，参与讲授“开源软件开发基础及实践”课程。第一堂课上，龙蜥社区的技术专家为北京大学软件与微电子学院的研究生同学们，介绍了社区概况和围绕龙蜥操作系统（Anolis OS）的实践课程大纲。在整个春季学期期间，龙蜥社区的9位技术专家为同学们介绍开源实践通识知识、分享龙蜥操作系统的技术亮点、开设四个方向的精品小课堂，并组织同学们进行龙蜥操作系统开源开发实践。

走进国家重点实验室

龙蜥社区走进高效能服务器和存储技术国家重点实验室，举办“异构计算赋能操作系统”学术交流活动。此次交流，是实验室和龙蜥OS对双方原有战略合作的研究创新和探索，致力于完善优化科技创新生态，后续将继续开展学术研讨和交流活动，与各产学研机构共同培养产业人才，共建开源社区、学术推广和应用示范。

龙蜥社区走进
重点实验室活动现场图

7.1.3 面向广大开发者

人人都可以参与开源活动

「人人都可以参与开源」，这是一款面向开源新人、技术常客的开发者活动，数千款基础、进阶型任务可供选择，让开发者从入门到精通玩转龙蜥社区。完成活动任务，不仅能够拥有社区定制礼品/实习证明，还有机会获得实习机会、参与年度评选，可关注龙蜥官网参与活动。

在2022开放原子全球开源峰会上，龙蜥论坛和展区吸引了近1500位参会者参与互动，亮点满满，精彩不断。

2022 开放原子全球开源峰会
线下展区活动图

该活动也为高校学生打造了一个体验开源、参与社区贡献的平台，通过任务来熟悉开源流程、提升实践能力，因此龙蜥社区针对「人人都可以参与开源」活动，将长期招募校园推广大使，请关注龙蜥社区官方网站进一步了解。

龙蜥大讲堂

龙蜥大讲堂是龙蜥社区推出的系列技术直播活动，邀请龙蜥社区的开发者们分享龙蜥技术、龙蜥SIG双周会、开源贡献入门，内容包括但不限于内核、编译器、机密计算、容器、存储等相关技术领域。龙蜥大讲堂系长期活动，每周1-2场，欢迎开发者们积极参与，共享技术盛宴。

龙蜥大讲堂技术系列直播现已举办50场，单场直播间累计观看高达6000+，覆盖35000+开发者，与近百位开发爱着互动。

为了让更多的开发者们认识、了解龙蜥技术，龙蜥社区联合媒体共同推出了《龙蜥大讲堂》。社区长期招募龙蜥大讲堂演讲讲师，共同探讨社区热点技术。

龙蜥大讲堂技术大咖分享部分图

7.1.4 展览集锦

2021云栖大会龙蜥专场

2021云栖大会上龙蜥社区重磅亮相，龙蜥专场论坛线上直播+线下参会人次1500+；展区曝光2万+，展台参观流量1000+，动手体验200+；除了精彩论坛外，龙蜥社区分别在C馆云起实验室、D馆阿里云十大核心技术设有龙蜥展区，通过集徽章换好礼、现场抽奖互动、沉浸式体验龙蜥OS等互动环节，近万人次进行现场互动。

云栖大会论坛和展区现场图

2022 开放原子全球开源峰会龙蜥专场

2022年7月27-29日开放原子全球开源峰会期间，龙蜥社区携技术委员会、运营委员会、新一届理事成员以及18位业界大咖重磅亮相，吸引了近1400位参会者参与互动。更有阿里巴巴集团CTO程立为龙蜥操作系统站台，表达了阿里持续投入建设龙蜥的决心。

开放原子全球开源峰会论坛和展区现场图

CLK大会

随着RedHat停更CentOS稳定版，国内的各大CSP开始不断尝试新的发行版技术，希望以此为国内的服务器环境提供一个更稳定、更安全的通用Linux发行版。在此龙蜥社区技术委员会主席，阿里云操作系统团队技术总监杨勇分享了龙蜥社区：新计算场景驱动下的开源创新。

CLK大会活动现场图

龙蜥社区理事长马在CLK大会主论坛演讲图

1024程序员节

CSDN「人物志」栏目采访了龙蜥社区理事长、阿里云研究员、基础软件部操作系统团队负责人马涛，请关注龙蜥社区公众号【OpenAnolis】一起看看在5G、云计算等技术的冲击下，云操作系统面临的挑战和机遇。

龙蜥社区成立运营委员会主席陈绪博士参加1024程序员节闭门研讨会：操作系统生态与共识。

龙蜥社区理事长马涛参加1024程序员节

龙蜥社区成立运营委员会主席陈绪博士闭门研讨会现场图

阿里云峰会龙蜥专场

在阿里云开发者峰会上，龙蜥社区（OpenAnolis）主办的“开源操作系统社区与生态”分论坛圆满结束。9大主题，10位技术大咖，242位开发者，现场座无虚席。龙蜥社区（OpenAnolis）合作伙伴和开发者们济济一堂，来自阿里云、统信软件、Intel、ARM中国、联通等公司的各位大咖相聚，第一次全面分享社区理念、工作进展和开源技术，倡导和业界共建一个开放、合作、共赢的开源社区。

峰会论坛合照及现场图

龙蜥社区亮相阿里巴巴开源开放周

第一届阿里巴巴开源开放周是阿里巴巴联合开源办公室、主流媒体共同举办的系列线上分享会，重点围绕五大关键领域（操作系统、数据库、云原生、大数据、终端）的生态系统与阶段性成果，龙蜥社区作为合作伙伴之一，打造了操作系统专场分享，与开发者、开源爱好者、关注开源领域趋势等众多生态伙伴共同践行开放共享的好科技。

本次活动在全平台直播观看量累计突破700万+。本次活动联合内部28+技术线BU钉群、15+阿里技术联盟自媒体矩阵，包含阿里技术、大淘宝技术、阿里巴巴云原生微信视频号等平台，以及InfoQ、开源中国、CSDN、思否等内外部头部技术社区和媒体共同传播。

7.2 优秀开发者

7.2.1 开发者故事

开发者与开源社区相辅相成，相互成就。这些个人在龙蜥社区的使用心得、实践总结和技术成长经历都是宝贵的，我们希望让更多人看见技术的力量。所以龙蜥社区正式推出「龙蜥开发者说」，开发者说系长期活动，诚邀开发者们分享真实体验，以文会友、共同学习、一起进步。

7.2.2 年度优秀开发者

为了鼓励龙蜥社区的开发者同学积极参与社区共建，龙蜥社区每年都会推出年度优秀开发者评选活动，所有社区的注册用户均可参与。首届评选活动已于2021年圆满落幕，评选出10 位年度突出贡献奖及10位开源参与贡献奖。

08 社区年鉴

2020.9

龙蜥社区正式成立

- 由阿里云、统信软件等业内头部企业共同发起

2020.11

龙蜥社区决定将全部捐赠给开放原子开源基金会

- 社区理事会一致决议捐赠到开放原子开源基金会
- Anolis OS荣获“中国开源云联盟年度优秀开源项目”奖
- Arm加入龙蜥社区并成为理事单位
- 杭州西软、红象云腾、网思科技加入龙蜥社区
- 龙蜥大讲堂正式对外开播

2021.3

龙蜥社区第一个发行版诞生

- 龙蜥操作系统（以下简称Anolis OS）8.2 RC1 发布

2021.5

龙蜥操作系统社区版和衍生版发布

- Anolis OS 8.2 正式发布
- 统信服务器操作系统V20正式发布

2021.6

社区首届运营会议召开

- Anolis OS 8.4 正式发布
- 首次运营委员会议召开
- 红旗软件加入并成为理事单位

2021.7

社区首届理事大会召开

- 社区首届理事大会召开
- 成立LoongArch SIG，发布龙芯版路线图
- 中国系统加入龙蜥社区

2021.8

移动云发布基于Anolis OS的衍生版

- 移动云发布基于龙蜥Anolis OS的BC-Linux V8.2通用版操作系统
- 大连红旗加入龙蜥社区
- Anolis OS正式登陆阿里云公共云

2021.9

社区首次走进合作伙伴

- 龙蜥社区“走进龙芯“MeetUp启动
- 龙蜥荣获“开源社区及开源项目”OSCAR 开源尖峰案例奖
- 浪潮云加入龙蜥社区

2021.10

龙蜥重磅亮相云栖大会

- 助力中国联通完成核心业务CentOS试点替换
- 工商银行浙江分行、时速云、海尔智能加入龙蜥社区
- 龙蜥社区重磅亮相云栖大会，阿里云宣布投入20亿扶持
- 发布《OpenAnolis龙蜥社区技术创新白皮书》

2021.12

社区上线CentOS停服解决方案专区，启动「龙腾计划」

- 上线CentOS停服解决方案专区，落地行业案例
- 「龙腾计划」启动，截至目前近250家伙伴加入社区
- 普华基础软件加入龙蜥社区并成为理事单位
- Anolis OS通过工信部电子标准院首批开源项目成熟度评估，评为唯一“卓越级”

LoongArch GA 版发布

- 龙蜥LoongArch GA 版正式发布
- “龙蜥实验室”在社区官网正式上线
- 堡塔等60+企业加入龙蜥社区
- 龙蜥社区走进高效能服务器和存储技术国家重点实验室
- 2021年度龙蜥社区优秀开发者评奖活动开启

2022.1

龙蜥社区启动“开发者说”栏目

- 电科申泰加入龙蜥社区并成为理事单位
- 龙蜥社区正式启动“开发者说”栏目
- 一站式T-One测试平台上线

2022.3

「人人都可以参与开源」活动正式上线

- 更多龙蜥自研、生产可用的Anolis OS 8.6 正式发布
- 龙蜥社区技术委员会召开，并发布新一代操作系统研发路线图
- 龙蜥社区走进 Intel MeetUp

2022.6

龙蜥支持全国首个政府采购平台-政采云完成CentOS迁移

- 龙蜥社区支持全国首个政府采购平台-政采云完成CentOS迁移
- 龙蜥社区亮相阿里巴巴开源开放周
- 开源人说-龙蜥故事正式上线

2022.8

2022.2

龙蜥社区走进北大课堂

- 龙蜥社区走进北大课堂
- 首届社区年度突出贡献奖公布
- 龙蜥操作系统&龙蜥社区双双荣登2021“科创中国”开源创新榜
- 万达信息等13家企业加入龙蜥社区

2022.5

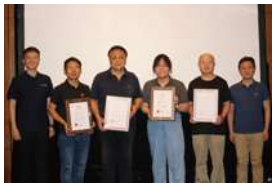
首个CSV机密容器解决方案上线

- 龙蜥社区上线首个CSV机密容器解决方案
- 龙蜥社区参与2022开源之夏、编程之夏等活动
- 中兴通讯加入龙蜥社区

2022.7

新一届理事大会隆重召开

- 龙蜥社区第二届理事大会召开，4位特约顾问加入
- 社区技术委员会一致决议发布龙蜥技术治理路线等规划
- 龙蜥社区参加2022开放原子全球开源峰会



2022.9

麒麟软件、浪潮信息、中科曙光、新华三4家重磅理事成员加入龙蜥

- 麒麟软件、浪潮信息、中科曙光、新华三加入龙蜥社区并成为理事单位
- 龙蜥及其理事分获“2022 OSCAR尖峰开源社区及项目、尖峰开源人物”奖项

09
伙伴寄语

张东 浪潮信息副总裁

操作系统是连接底层硬件和上层应用的桥梁，处于行业生态体系的中坚位置，作为服务器行业的头部企业，浪潮信息高度重视操作系统生态建设，期待与龙蜥社区合作伙伴和开发者一起建设社区，通过开源推动操作系统技术创新，共同打造操作系统新生态。

王彦瑞 朗思科技CEO

朗思科技将积极参与龙蜥社区的生态建设与合作，依托自身积累的技术和经验，我们希望朗思科技'数字机器人'产品及开发平台作为IDE平台工具软件植入龙蜥操作系统，为社区贡献技术力量，并携手社区伙伴共同促进社区建设以及行业生态持续发展，为企业数智化转型及业务数据化建设赋能。

管健 涌现科技创始人&CE

未来，涌现科技将积极参与龙蜥社区（OpenAnolis）的各项技术交流活动及产业探索，与社区伙伴们一起，促进操作系统的持续健康发展和广泛应用，为全面数字化赋能。

王成 中盈优创云网系统服务事业群负责人

中盈优创将积极参与龙蜥社区建设，基于龙蜥操作系统推进中盈优创产品与服务的适配验证，并携手更多社区合作伙伴完善操作系统技术体系，为龙蜥开源生态繁荣及广泛应用贡献力量，为数字经济的发展保驾护航。

杨成伟 爱奇艺高级技术总监

爱奇艺很高兴加入龙蜥社区，在进一步实现爱奇艺操作系统迭代升级的同时，爱奇艺也将积极投入参与到社区建设中，为基础软件的落地和推广出一份力。

闫磊 云脉芯联软件研发副总裁

未来，云脉芯联将积极参与龙蜥社区的深度合作，提供云脉芯联DPU产品针对可编程网络、高性能网络等新技术方向，探讨新的软硬件一体化方案在龙蜥操作系统（Anolis OS）的落地。

侯前明 贝联珠贯联合创始人

贝联珠贯和龙蜥社区颇有渊源。贝联珠贯提升资源利用率的方法中一个核心部分是单机的QoS隔离，龙蜥社区在这方面有不错的积累，和社区的合作非常有助于进一步共同提升这方面的能力，同时客户大量使用龙蜥操作系统也将大幅度降低贝联珠贯在客户侧的实施难度和成本。

陈荣 安全狗高级副总裁

希望通过在安全技术探索、解决方案研究、关键产品开发等工作方面的深入交流及探讨中，发掘自身优势将产品推向新的台阶，同时也希望为基础软硬件安全协同防护能力贡献自身的一份力量。安全狗将积极参与龙蜥社区合作，输出自身安全能力与经验，为开源操作系统的安全侧贡献力量，助力操作系统产业安全、平稳、快速的发展。

敖钢 芯动科技副总裁

加入龙蜥社区是风华GPU走进千家万户的重要布局，也是芯动赋能智慧生态的战略部署。未来，我们愿与众多产业伙伴合作共享，持续用先进的芯片级创新和软硬件支持能力，推动生态繁荣发展，助力合作伙伴产品成功，赋能数字经济和智慧生活。

莫庆良 中兴新支点操作系统产品总监

中国操作系统开源社区正在高速发展，百花争艳，推动我国的基础软件往更高的技术水平发展。龙蜥社区作为一个开源协作及创新平台，聚焦技术创新。中兴新支点作为国内作为技术创新型的操作系统厂商，我们相信积累的经验能给社区注入力量。未来，中兴新支点操作系统将积极参与龙蜥社区的合作，并与社区的伙伴们共同携手共建社区的繁荣，全力推进我国信息产业安全和数字经济的发展。

孙冀军 昆仑芯科技副总裁

AI算法和应用开发者在构建AI应用和业务的过程中，需要一套成熟的编程语言以及完善的软件工具集来快速迭代开发任务，昆仑芯SDK可以提供从底层驱动环境到上层模型转换等全栈的软件工具。加入龙蜥社区后，我们希望通过这个开放、创新的平台携手各方，推动开源操作系统的持续健康发展，并构建软硬一体的AI芯片生态，共同打造从芯片到系统、应用、云端、服务的生态闭环。

胡捷 中兴通讯开源战略总监

中兴通讯作为全球领先的ICT综合方案提供商，一直致力于芯片、操作系统、数据库等底层技术的研究和自主创新。龙蜥社区是国内领先的操作系统开源社区，中兴通讯将本着开放、共享、创新的原则积极参与开源社区的建设，携手产业合作伙伴共同为核心技术的自主创新做出贡献。也祝愿龙蜥社区越来越兴旺，为社会和行业带来价值。

沈翔 普华基础软件事业部总经理

龙蜥是国内领先的操作系统开源社区，普华是一家具有开源基因的国产操作系统公司。

孙尔俊 沐曦CMO

很高兴能够加入龙蜥社区，并期待未来沐曦可以与龙蜥社区进一步合作，参与推动软硬件及应用生态的发展与合作，聚焦技术创新，繁荣开源生态。

刘文华 海泰方圆副总经理

未来，海泰方圆将在国密算法、隐私计算领域与龙蜥社区开展深度合作，以可信数据治理和密码全能力积极为开源操作系统赋能。

李利军 东方通集团执行副总裁、东方通软件总经理

东方通作为中间件领域的领导者，将积极参与龙蜥社区合作，也将凭借东方通在中间件领域的技术积累加速推动开源操作系统生态的发展。

邓克武 宝德计算IA BU总经理

产品成功的关键离不开生态的繁荣，龙蜥社区是国内领先的操作系统开源社区，加入龙蜥'朋友圈'，对于宝德具有重要的战略意义。未来，宝德将立足自身的产品技术，发挥关键领域的竞争优势，积极参与龙蜥社区建设，实现共赢发展。

刘杭军 宝兰德生态合作部总经理

宝兰德以积极开放的态度加入龙蜥社区，希望以此为契机，发挥宝兰德的技术经验和人才优势，与社区伙伴共同促进开源社区发展，为数智化经济服务。

白皮书作者

毕黛琛	蔡佳丽	陈　刚	陈　佳	陈健康	崔　开	丁　宁
杜　凡	杜　炜	段　然	冯富秋	冯世舫	高　翔	葛立伟
韩伟东	郝世荣	胡玉溪	黄韶宇	黄晓军	金美琴	李　崇
李靖轩	李三红	李艺林	林　演	刘　博	刘　峥	龙　勤
陆　扬	吕荟晶	马　涛	毛文安	彭媛洪	齐　江	祁　宇
钱　君	尚旭春	石晓海	宋仲儒	宋　卓	孙林林	田瑞冬
王晨光	王洪虎	王劲男	王立刚	王　强	汪少军	王　帅
王轶飞	吴朝峰	吴一昊	夏敏琪	徐　宇	严力科	杨　亮
杨　勇	叶　华	袁艳桃	张　佳	张家乐	张金利	张　菁
张梦瑶	张鹏程	张世乐	张顺达	张天佳	张　毅	张永超
张　元	赵冠军	郑　琦	周伟涛	朱江云	朱运阁	

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。
加入龙蜥社区，一起打造一个面向下一代的开源操作系统！欢迎您随时和我们联系。

微信公众号

扫码关注微信公众号
“OpenAnolis龙蜥”

全景白皮书链接合集

扫码获取
更多详情内容

龙蜥社群

钉钉扫码入群
交流社区热门技术

龙蜥官网
https://openanolis.cn

龙蜥助手： 小龙
微信号： openanolis_assis

龙蜥邮箱
secretary@openanolis.org
partner@openanolis.org


OpenAnolis
龙 蜥 社 区