龙蜥社区
新计算场景驱动下的开源创新

杨勇

龙蜥社区技术委员会主席

目　录

CONTENTS

云时代操作系统面临的挑战

多架构的支持

操作系统碎片化

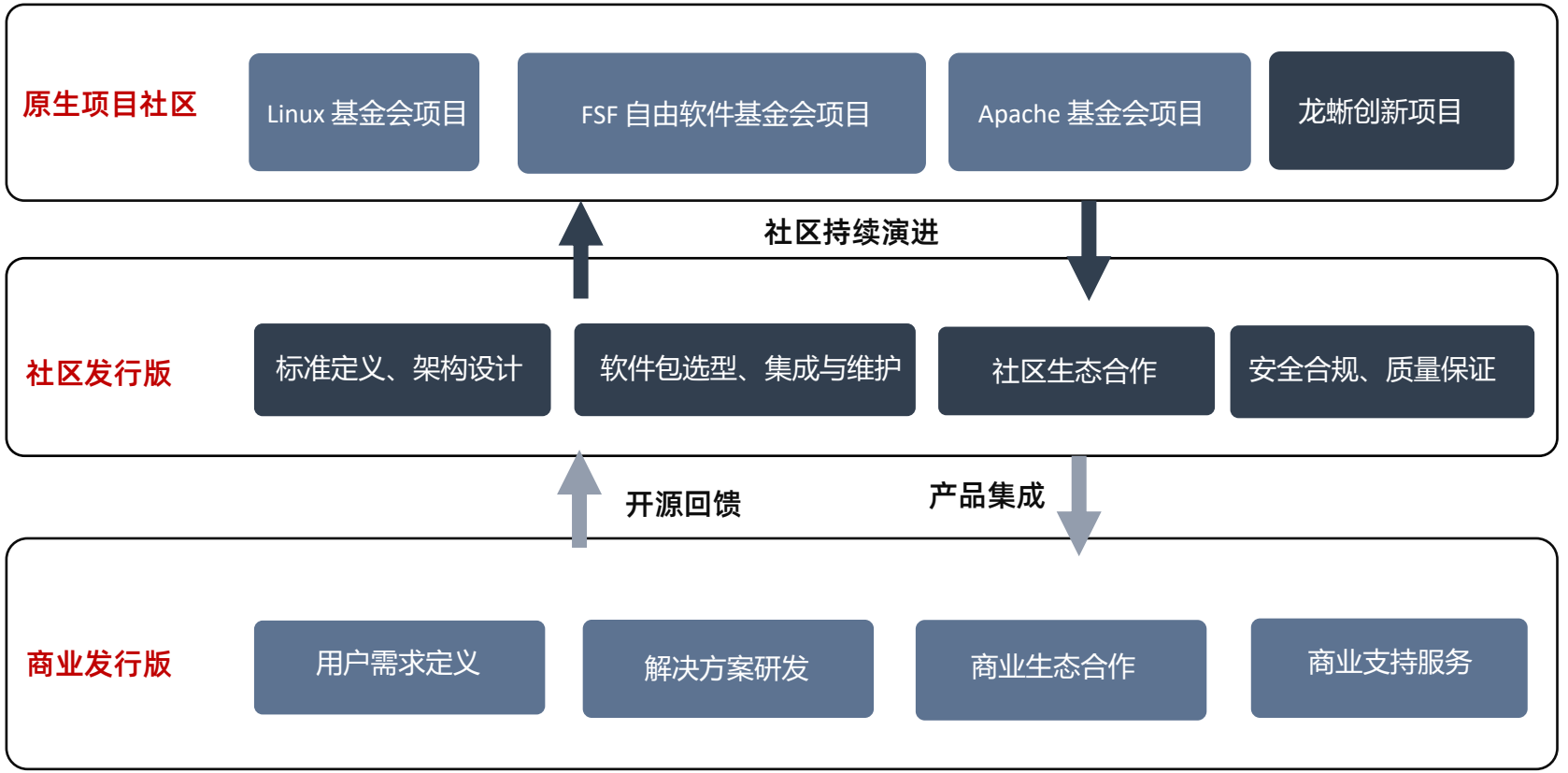
全栈系统的割裂

- 用户体验差、运维成本高
- 产业协作难、研发成本攀升
- 算力难以充分释放

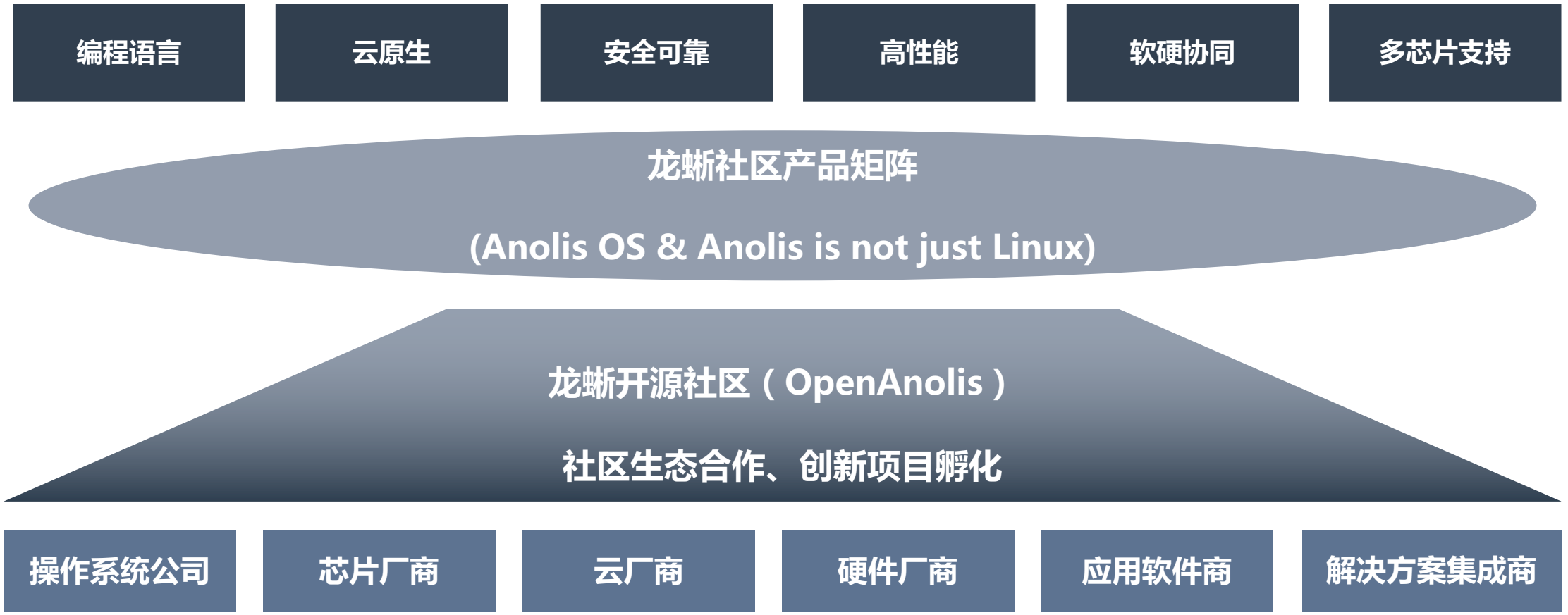
龙蜥社区定位 – 不仅仅是 Linux 发行版社区

- 技术创新
- 开源协同
- 商业服务

Linux 操作系统 = 海量上游项目 + 社区发行版 + 商业衍生版



社区技术方向


编程语言
云原生
安全可靠
高性能
软硬协同
多芯片支持
龙蜥社区产品矩阵
(Anolis OS & Anolis is not just Linux)
龙蜥开源社区（OpenAnolis）
社区生态合作、创新项目孵化
操作系统公司
芯片厂商
云厂商
硬件厂商
应用软件商
解决方案集成商

Anolis OS 8 发行版介绍

Anolis OS 8 是龙蜥社区发行的长期稳定 Linux 发行版，提供了低成本的 CentOS 迁移解决方案及工具

多芯片支持

支持国内外主流芯片厂商

多内核支持

- ANCK（Anolis Cloud Kernel）

  社区 4.19 和 5.10 LTS 内核，30+ 特性增强，经过阿里云百万以上部署实践、双十一的极限压力测试

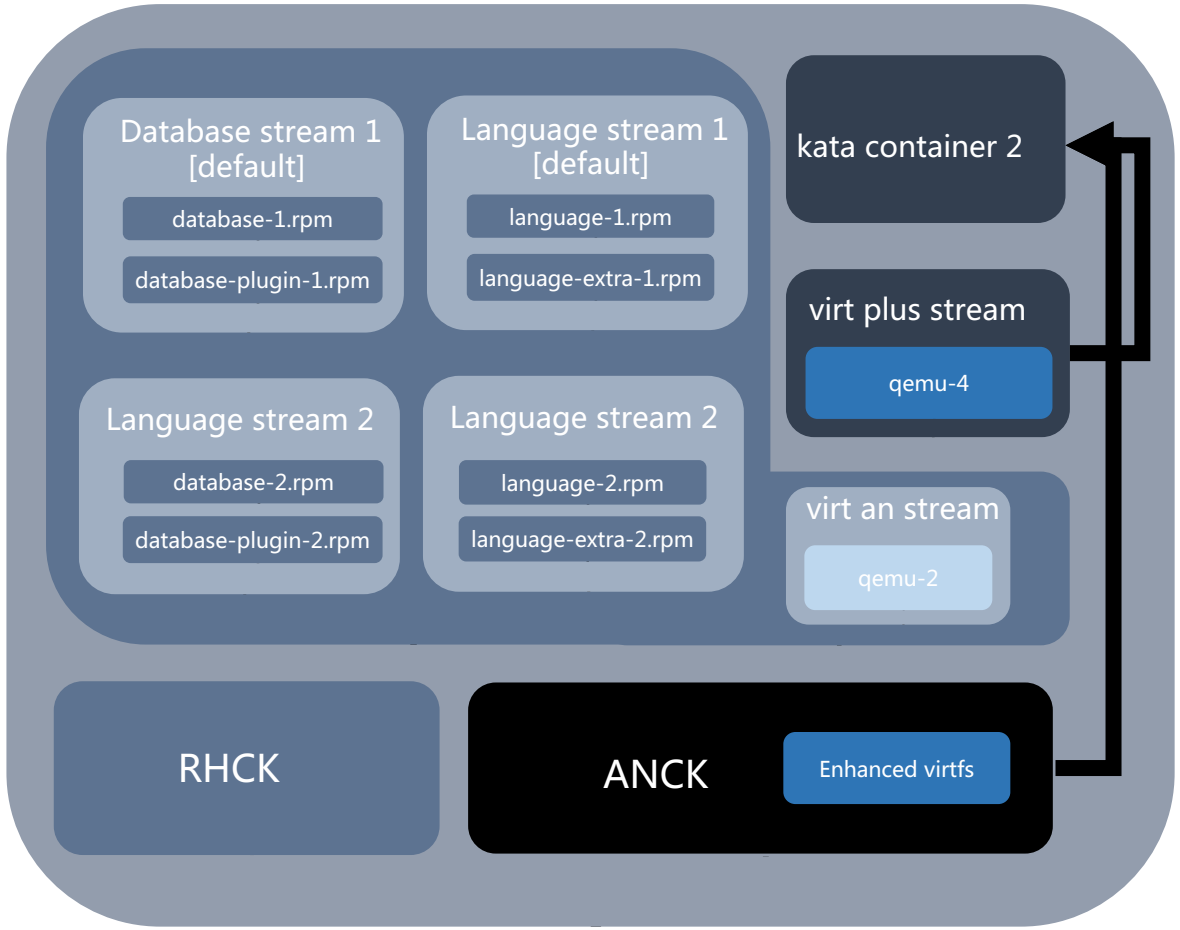
- RHCK（RHEL Compatible Kernel）

  基于源代码重新构建，满足部分用户诉求

下游衍生版本

- OSV 厂商：统信服务器操作系统V20（10120a）
- 云厂商：移动 BC Linux，联通、Alibaba Cloud Linux

Anolis OS 8 特性 - 应用流（AppStream)



特性介绍

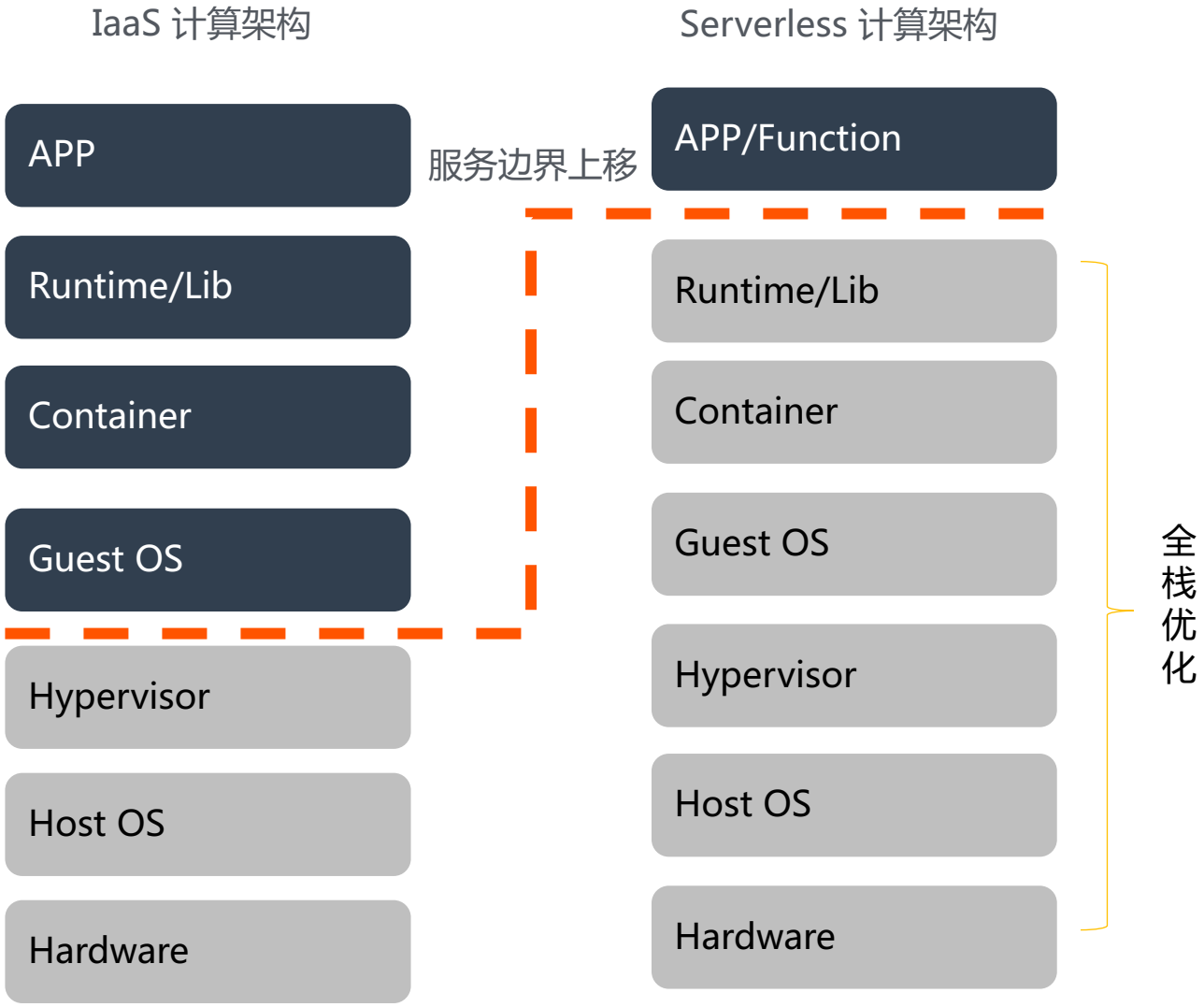
- OS 自带软件包版本单一，无法满足应用频繁更新的需求。而频繁更新应用软件，可能影响系统稳定性
- 通过核心基础组件与应用组件解耦合，平衡了可扩展性和稳定性

模块化应用

- 对多个应用流采取简化的统一管理方式
- 更新的版本，提供稳定的更新版本的开源软件
- 效果用例: 在龙蜥操作系统8.2上，在保障兼容性情况下，基于应用流机制引入高性能的 kata container 新版本

机制比较	优势	劣势
应用流	保障稳定性的前提下，提供新应用版本，实现应用解耦	开发门槛高，应用更新由 OS 厂商完成，难以接入业务 DevOPS 流程
容器	应用更新接入 DevOPS，与应用持续集成，快速部署	滥用容器混淆运维边界，带来版本管理、稳定性、安全问题。

新计算场景下的持续创新



分类	特性
编程语言	Dragonwell – WISP 协程
云原生	资源隔离解决方案
	CPU burst
	TCP-RT
	runD
安全可靠	全栈国密
	机密计算
高性能	io_uring
	Express UDP
软硬协同	SMC-R
多芯片支持	云边端一体，一云多芯

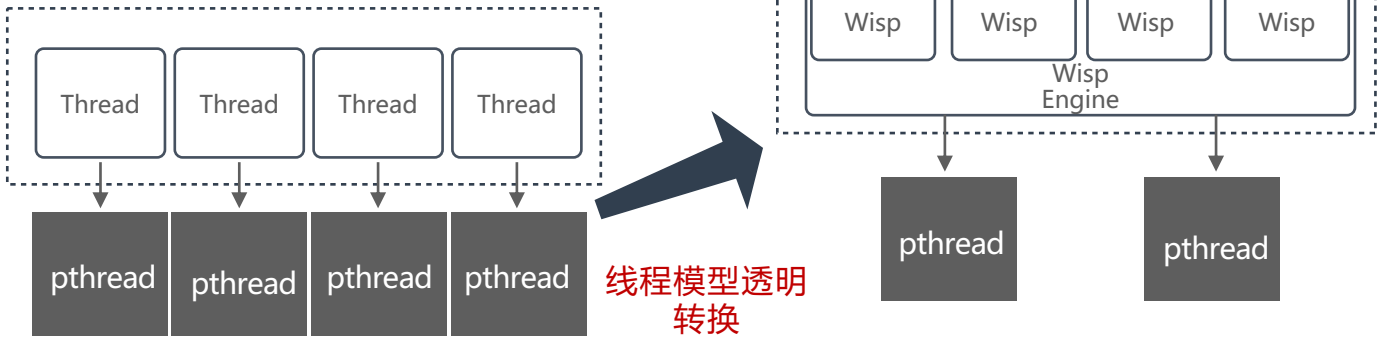
编程语言： Dragonwell - WISP 协程

10 ~ 30%

Throughput 提升

5 ~ 10%

latency 下降



场景和挑战

- 对标 Goroutine 的 Java 协程机制缺失
- 使用协程提升性能，但需要应用移植

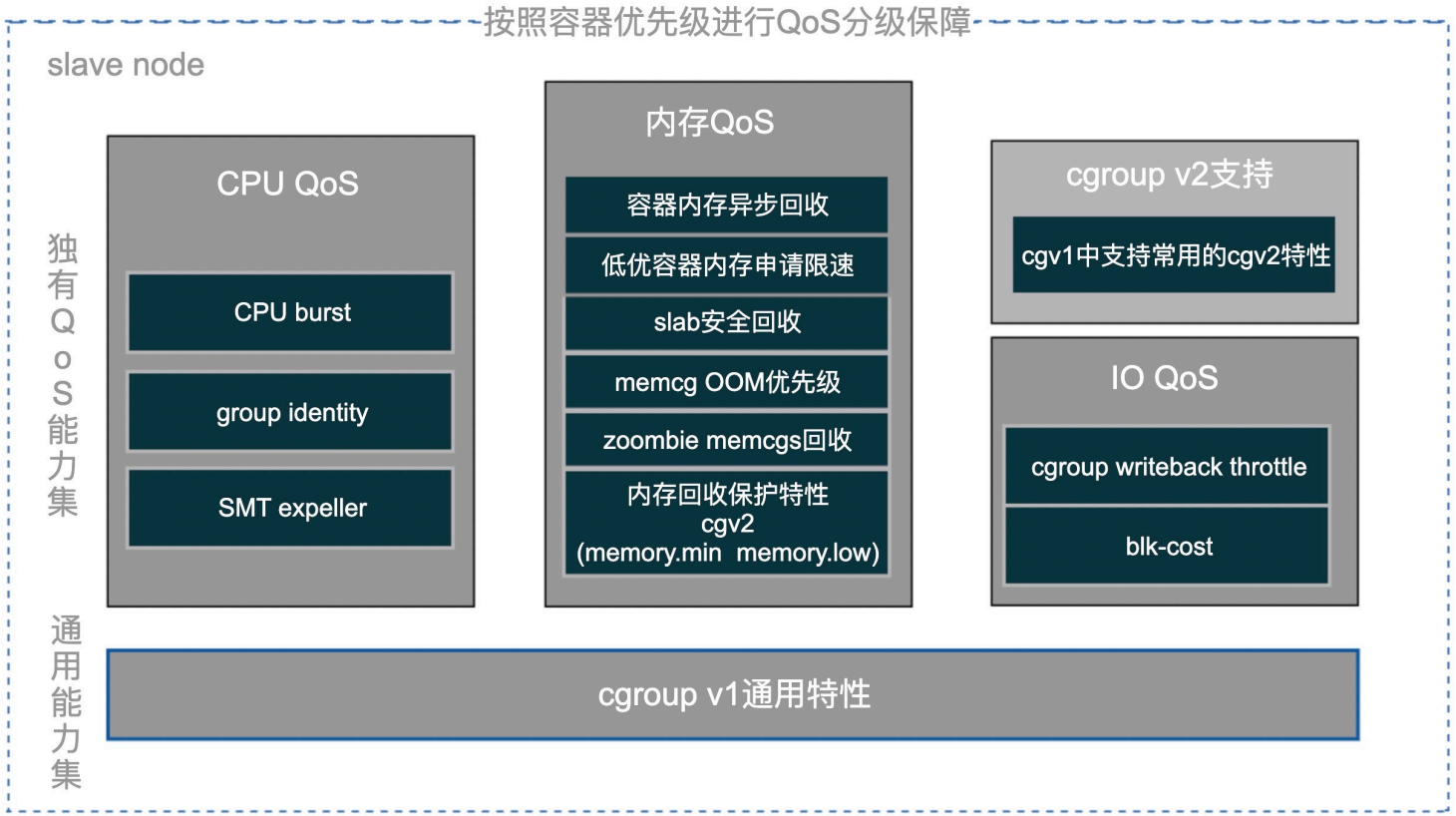
解决方案

- 实现协程切换能力和调度算法
- 处理好 JDK 同步和阻塞代码

价值和效果

- 基于多线程的 IO 密集的 Java 应用，性能提升、延迟下降
- 开箱即用，对应用透明，只需修改 JVM 启动参数

云原生：资源隔离解决方案



场景和挑战

- 离在线混部过程中，在确保在线业务QoS的前提下，通过容器混部提升整机资源使用率，降低业务成本。
- 如图所示传统的内核只有cgroup v1自带的通用隔离能力，无法真正适应云原生环境下的混部场景需求，无法确保离在线容器混部过程中在线业务不受离线容器的干扰。

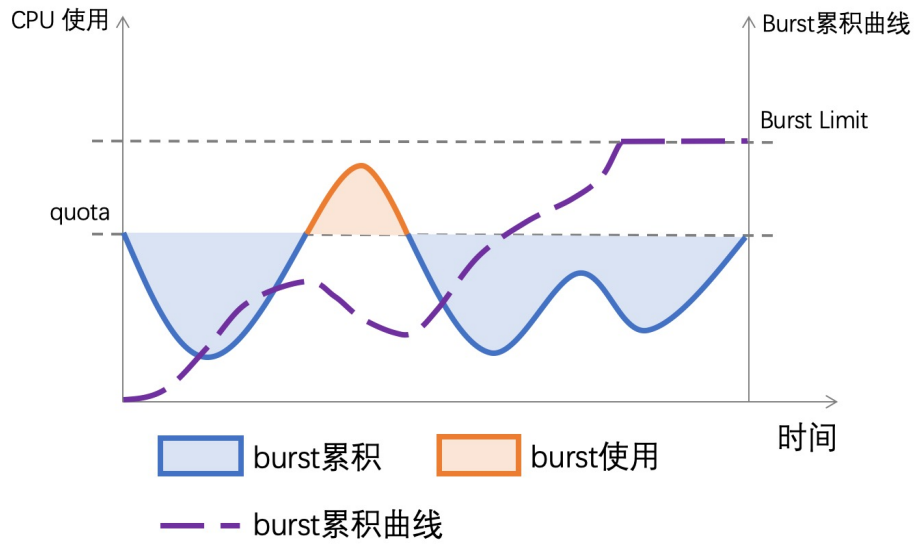
解决方案

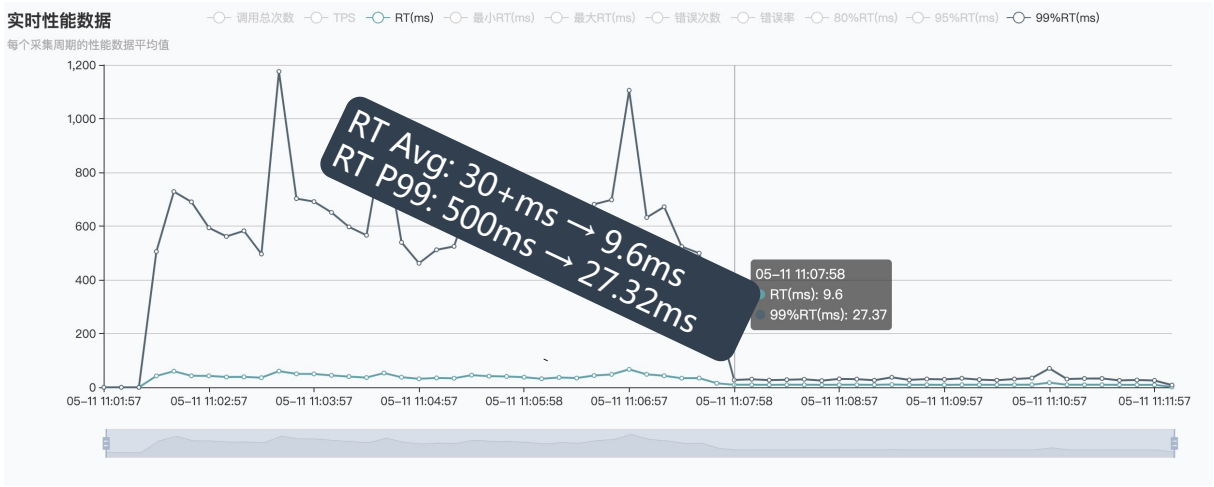
- 在 CPU，内存，IO，网络等领域，分析资源竞争原因，基于标准内核资源隔离能力扩展新特性
- 针对特定场景，提供更强自研隔离特性，并回馈社区

价值和效果

- 提高宿主机资源有效利用率，降低资源使用成本。
- 延迟敏感应用 QoS 保障

云原生：弹性CPU带宽控制 - CPU Burst





场景和挑战

- 容器调度中，设置CPU资源上限可以限制个别容器消耗过多资源，并且确保其他容器拿到足够多CPU资源。
- 设置CPU资源上限时的CPU限流是令人头疼的问题，即使容器的CPU利用率远低于上限，仍然可能出现CPU限流导致的延迟问题。在CPU Burst之前的答案是提高CPU上限甚至完全关闭上限设置，带来高TCO（Total Cost of Ownership, 总拥有成本）问题和潜在性能影响。

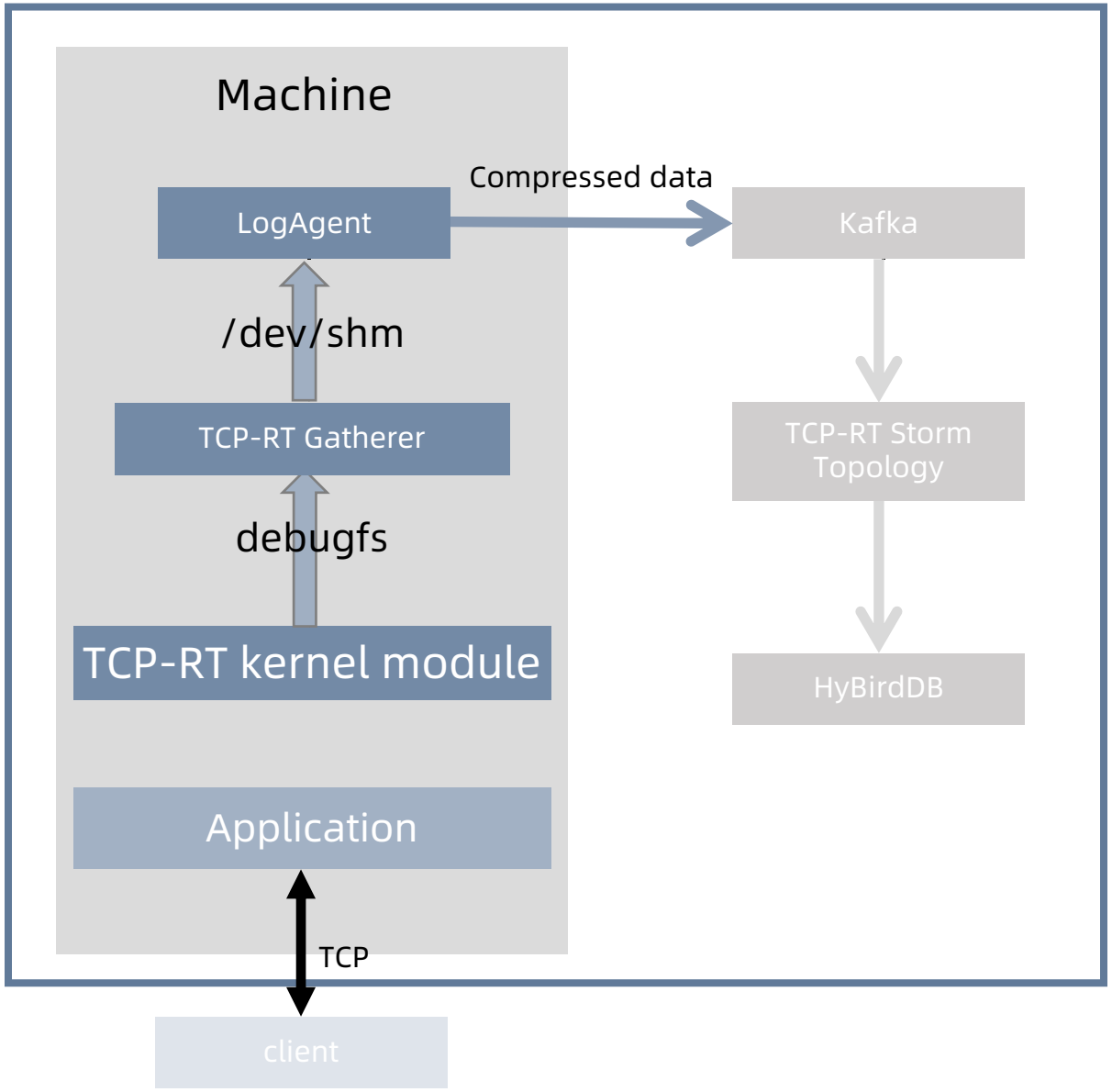
解决方案

- 提出 CPU Burst 技术，允许容器积累过去未使用的CPU资源，在需要时突发使用CPU资源，避免不必要的CPU限流。

价值和效果

- 消除不必要的CPU限流，使用户同时获得高CPU利用率和容器性能。

云原生：RASD 特性 - TCP-RT



Reliability Availability Supportability Debuggability

场景和挑战

- 当前大量互联网业务都是基于网络连接的，而通常如果应用出现网络异常后非常难于分析定位
- 传统手段一般只能跟踪系统整体的 TCP 指标, 难于定位每一个连接, 甚至每一个请求的详细信息。

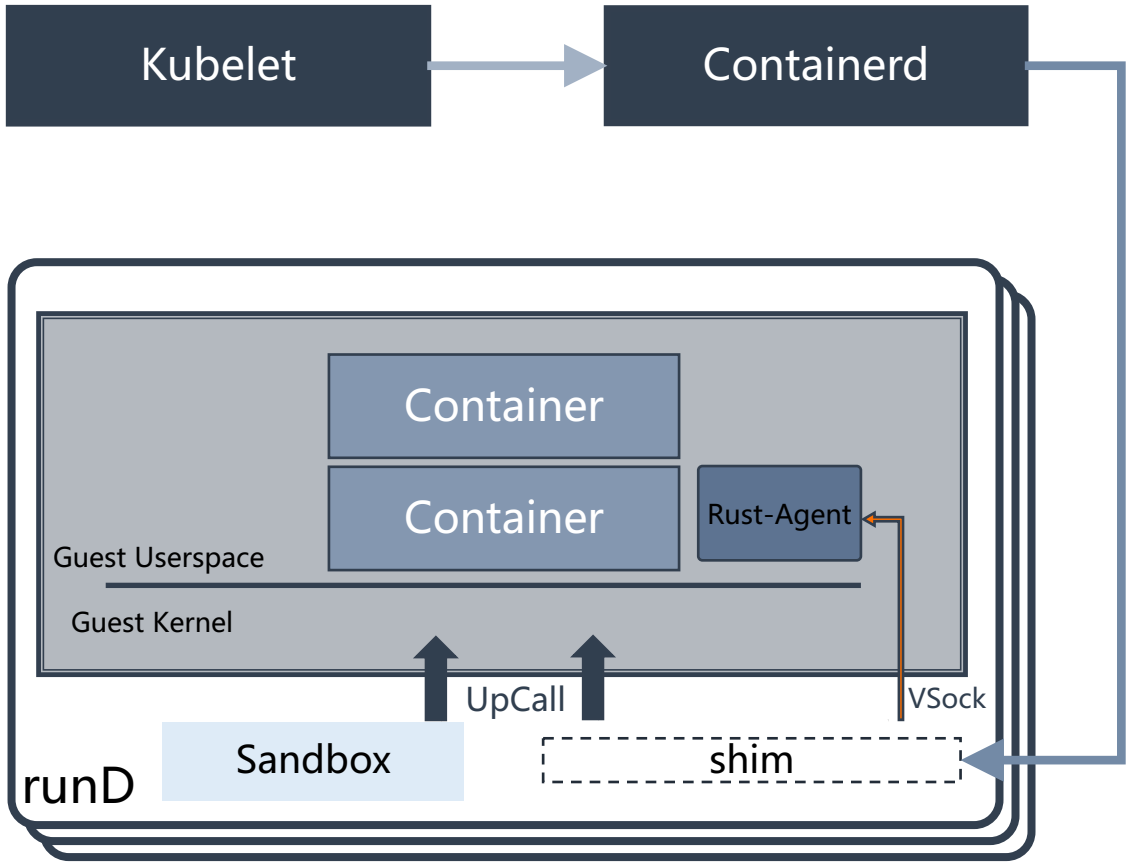
解决方案

- 提供 per-request 的 TCP RT 的数据采样能力

价值和效果

- 可以快速定位分析 TCP 连接中的每一个请求的状态变化
- 使用 TCP-RT 监控分析 TCP 流量, 快速定位异常。提升服务质量。

云原生： Serverless 运行时 - runD



场景和挑战

- 云原生 Serverless 服务需要支持多租户之间的强隔离，也需要细粒度的资源供给的能力
- 传统容器 runC无法保证租户级安全隔离/故障隔离/资源隔离
- 开源 kata 组件多，资源利用率低，运维复杂

解决方案

- 通过轻量级虚拟化技术实现强隔离
- Guest/hypervisor/image/shim 端到端全栈优化
- Shim 和 sandbox 融合为一，Rust 语言开发，安全高效

价值和效果

- 云原生一致用户体验
- 亚秒级快速启动，极致并发弹性
- 提高宿主资源有效利用率，降低资源使用成本。

安全可靠 - 全栈国密

龙蜥社区 OpenAnolis

商用密码生态及应用场景

商密支持进展情况

开源软件名称	SM2	SM3	SM4	SM4-avx2	PKCS#7	x509
gnulib	-	✅	-	-	-	-
libgcrypt	✅	✅	✅	Y	-	-
linux	✅	Y	Y	✅	开发中	✅
OpenSSL	✅	✅	✅	❌	✅	✅
coreutils	-	Y	-	-	-	-
RustCrypto	❌	✅	Y	❌	-	-
ima-evm-utils	✅	✅	-	-	-	-

场景和挑战

- 传统密码算法标准由国外制定，在特定场景存在合规问题
- 解决商用密码实现碎片化，没有统一技术方案的问题

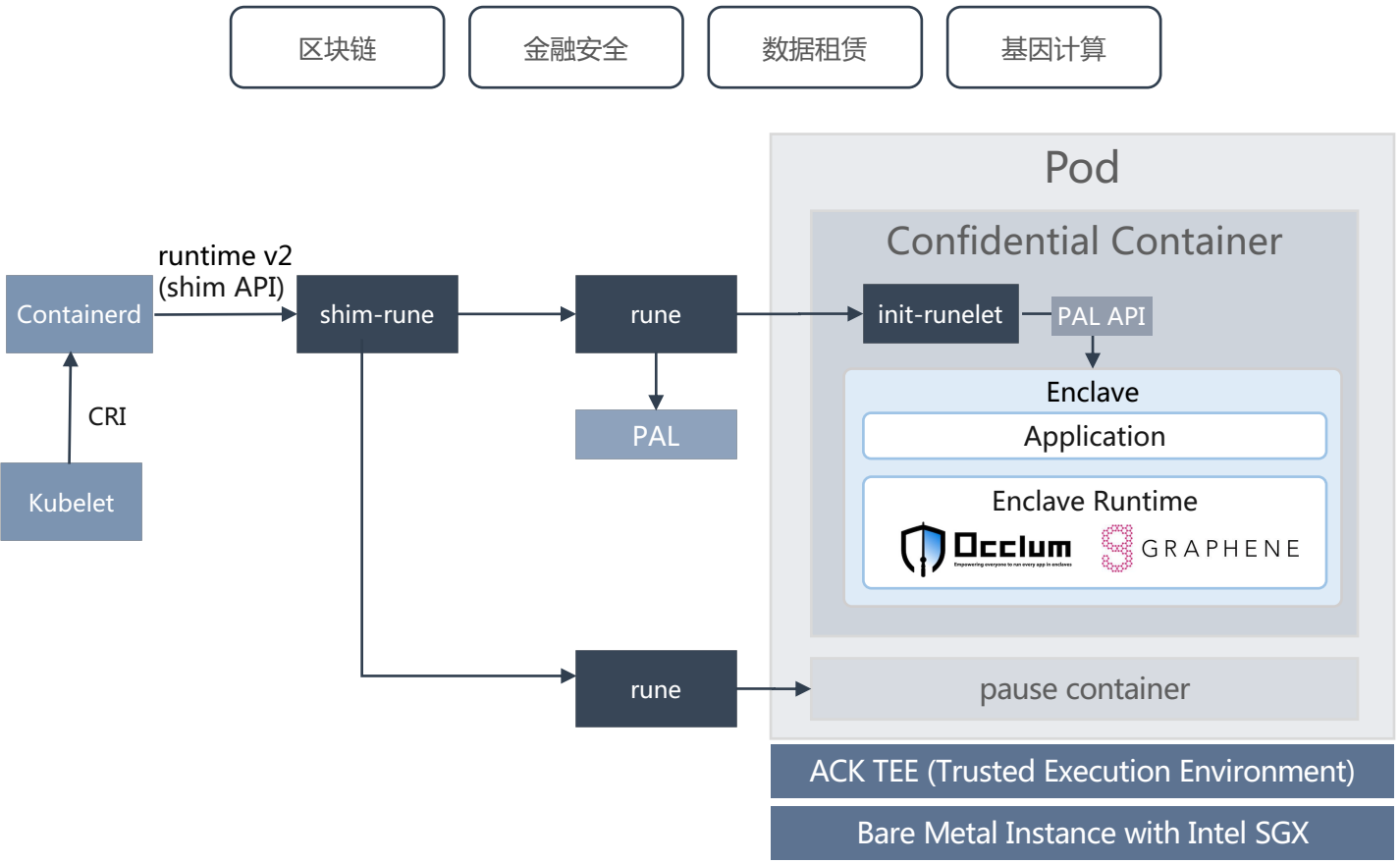
解决方案

- 从内核到基础密码算法库实现商用密码的主要标准，包括SM2/3/4基础算法，以及基于这些算法的TLS协议和安全认证机制
- 提供完整实现商用密码算法的BabaSSL算法库，完全兼容OpenSSL 1.1.1，支持QUIC 以及国内的安全协议TLCP，并提供一致的用户体验

价值和效果

- 只需要简单的配置开发，就可以基于这套商密基础设施，平滑切换到中国商用密码算法上来
- 对于有合规要求的客户可以获取软件密码模块合规资质
- 与国际统一的工程实现和API接口，回馈社区代码量23000多行

安全可靠 - 机密计算



场景和挑战

- 对拥有机密数据的租户来说，云服务提供商是完全不可信的。
- 机密计算技术提供了一种面向防御面的、新形态的云安全威胁模型。
- 传统的系统级安全威胁模型无法满足对机密数据有强安全防护诉求的用户群体，这已成为对隐私计算和敏感数据保护有需求的企业上云的最大障碍之一，也是云计算下一步必须亟待解决的问题。

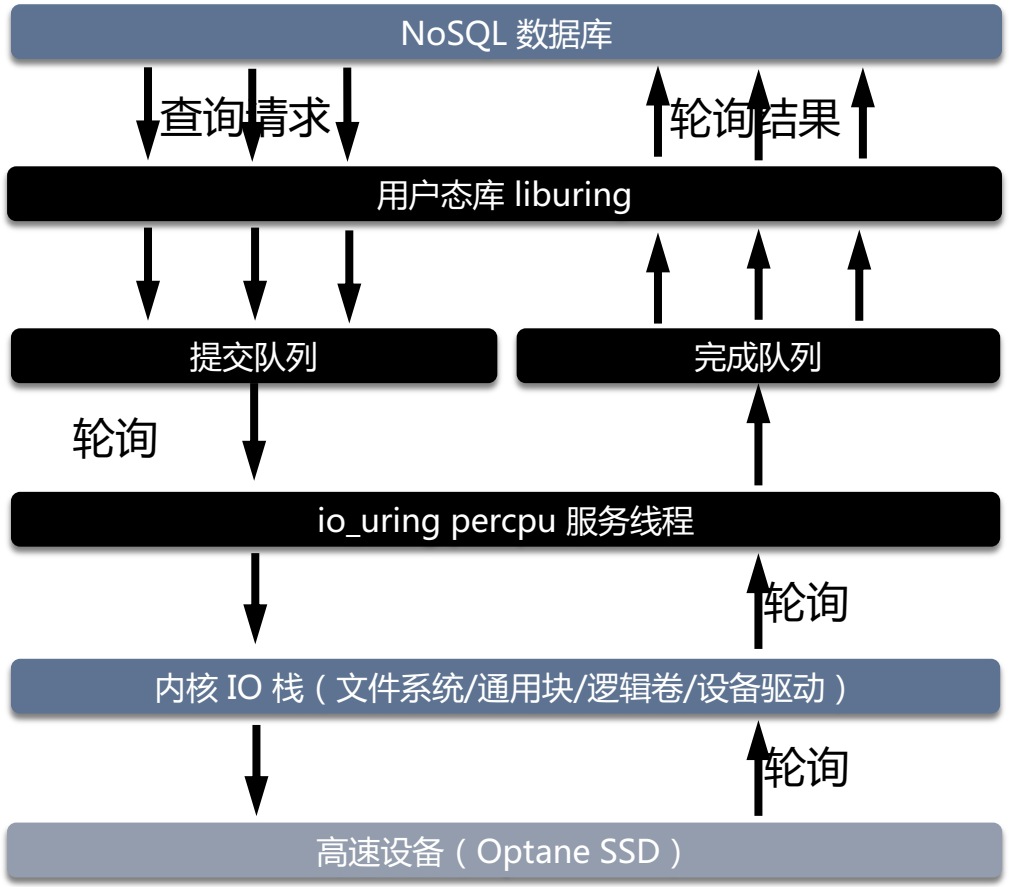
解决方案

- 基于 TEE SDK，ACK TEE，Inclavare Containers 等技术，能够全面支持裸金属服务器、机密虚拟机、机密计算容器和集群等多种机密计算形态来保护用户机密数据。

价值和效果

- 业务使用机密集群基础设施之后，可以为云用户的机密数据提供额外的数据安全防护能力。

高性能：充分释放存储性能 – io_uring



场景和挑战

- 某些 NoSQL 数据库对查询请求 RT 有很高的要求，一次查询请求会产生多次对存储设备的读操作，因此 IO 操作时延至关重要
- 传统的同步 IO 模型（pread）无法满足业务日益增长的诉求，使用 libaio 优化，无法发挥出 Optane SSD 的性能优势

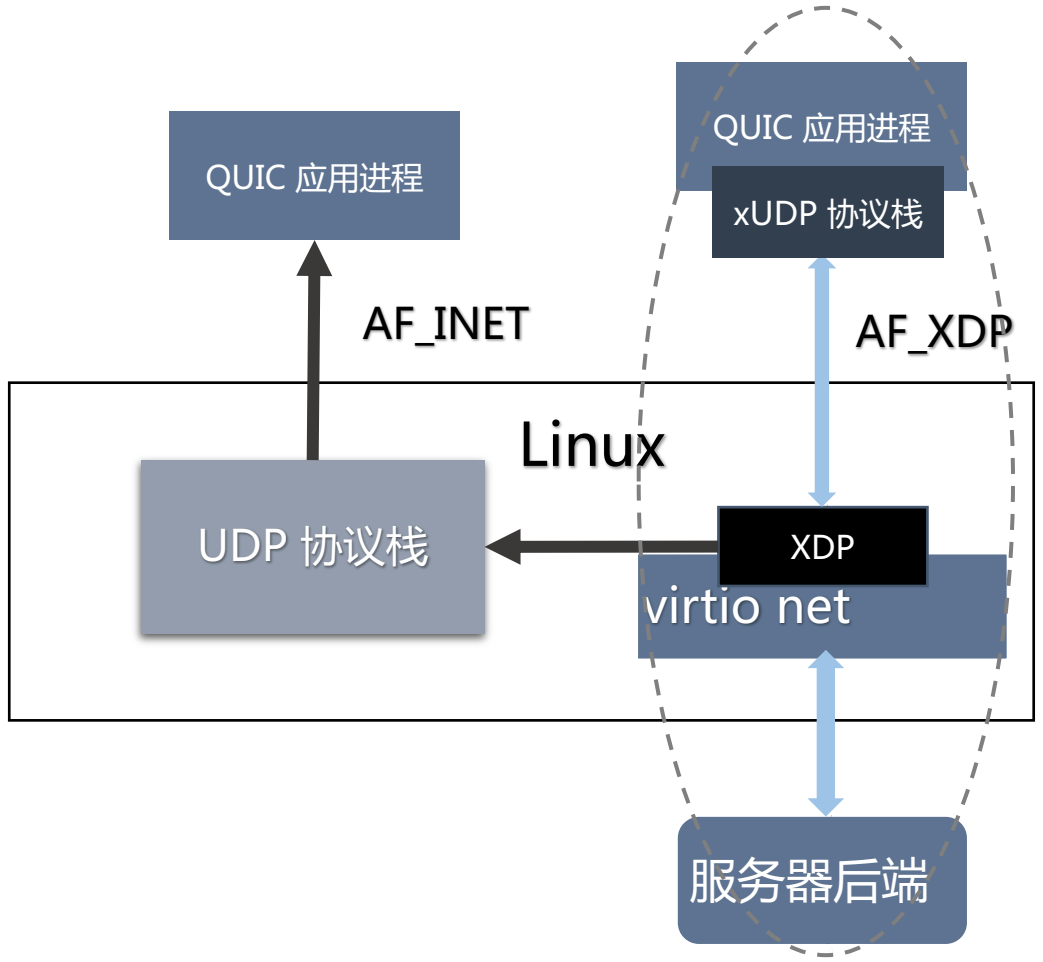
解决方案

- ANCK 内核内置新一代异步 IO 框架 io_uring，将 NoSQL 场景优化的特性回馈到 Linux 上游，并与社区协同确保稳定性
- NoSQL 数据库引擎同步修改，使用 io_uring 进一步优化

价值和效果

- 数据库端到端 RT 相比 libaio 版本优化 20%+。

高性能：精简 UDP 协议栈 - Express UDP



内核 UDP 协议栈　　XDP 绕开内核 UDP 协议栈

场景和挑战

- 传统内核 UDP 收发路径长，大量连接时扩展性差；大包卸载功能对 QUIC 不友好，服务端吞吐性能只有 TCP 的 30%
- virtio-net 对 xdp 支持不完备
- HTTP3/QUIC 已经正式标准化，新一代 Web 即将大规模到来
- HTTP3/QUIC 提升网络响应速度，优化用户体验

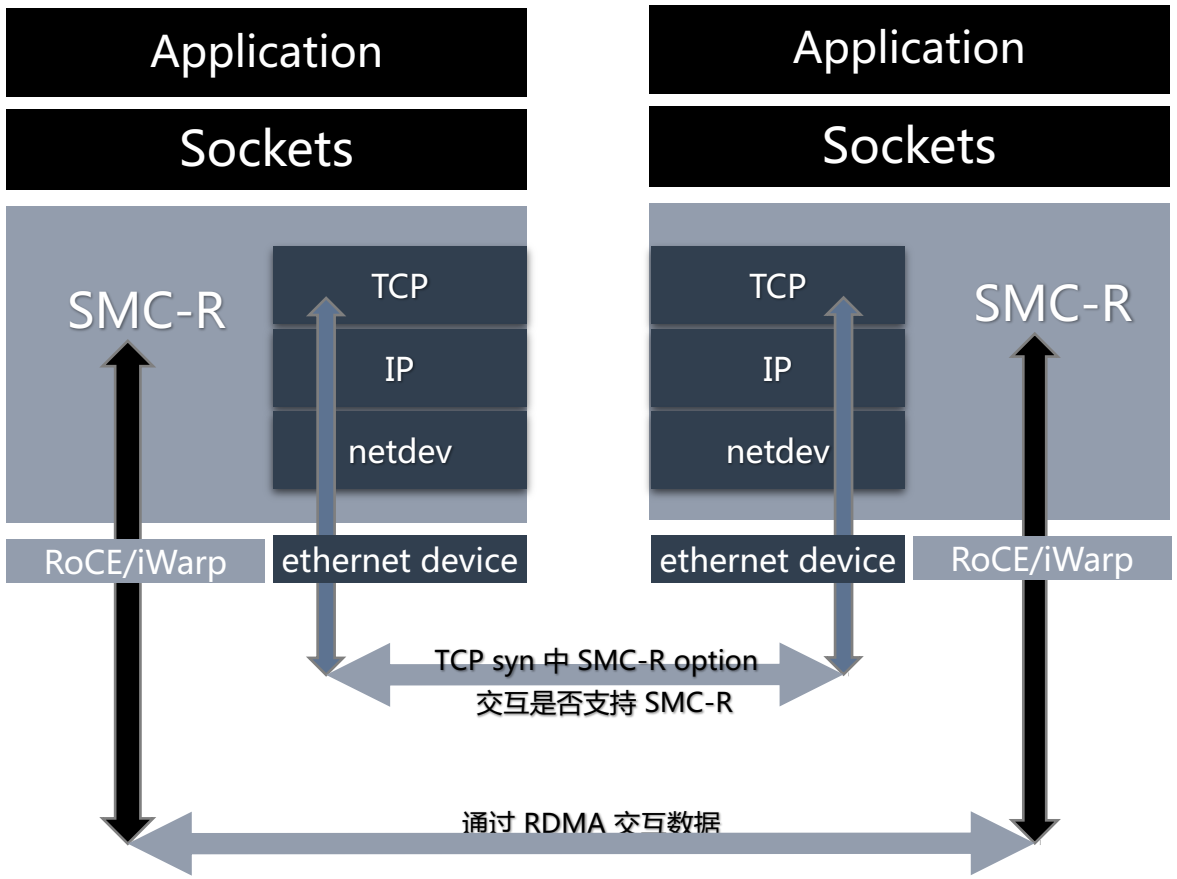
解决方案

- 在 virtio-net 队列数少于 CPU 数时，可以使用 XDP
- 在 virtio_net 上支持 XDP socket 零拷贝的功能
- 提供精简 UDP 协议栈，提升性能

价值和效果

- 收发性能 3~8 倍，QUIC 端到端 QPS 提升 50%

软硬协同：RDMA 高性能兼容协议栈 - SMC-R



场景和挑战

- 传统内核 TCP/IP 收发路径长，CPU 需要处理做大量协议处理，处理能力弱，大量宝贵的 CPU 资源耗费在网络收发处理上。
- CPU 性能提升不如 IO 提升速度快，数据中心硬件卸载成为趋势
- 业界 RDMA 能力在数据中心的普及，应用场景不够普适
- 阿里云神龙第四代服务器支持 RDMA 协议栈，也呼唤兼容普适方案

解决方案

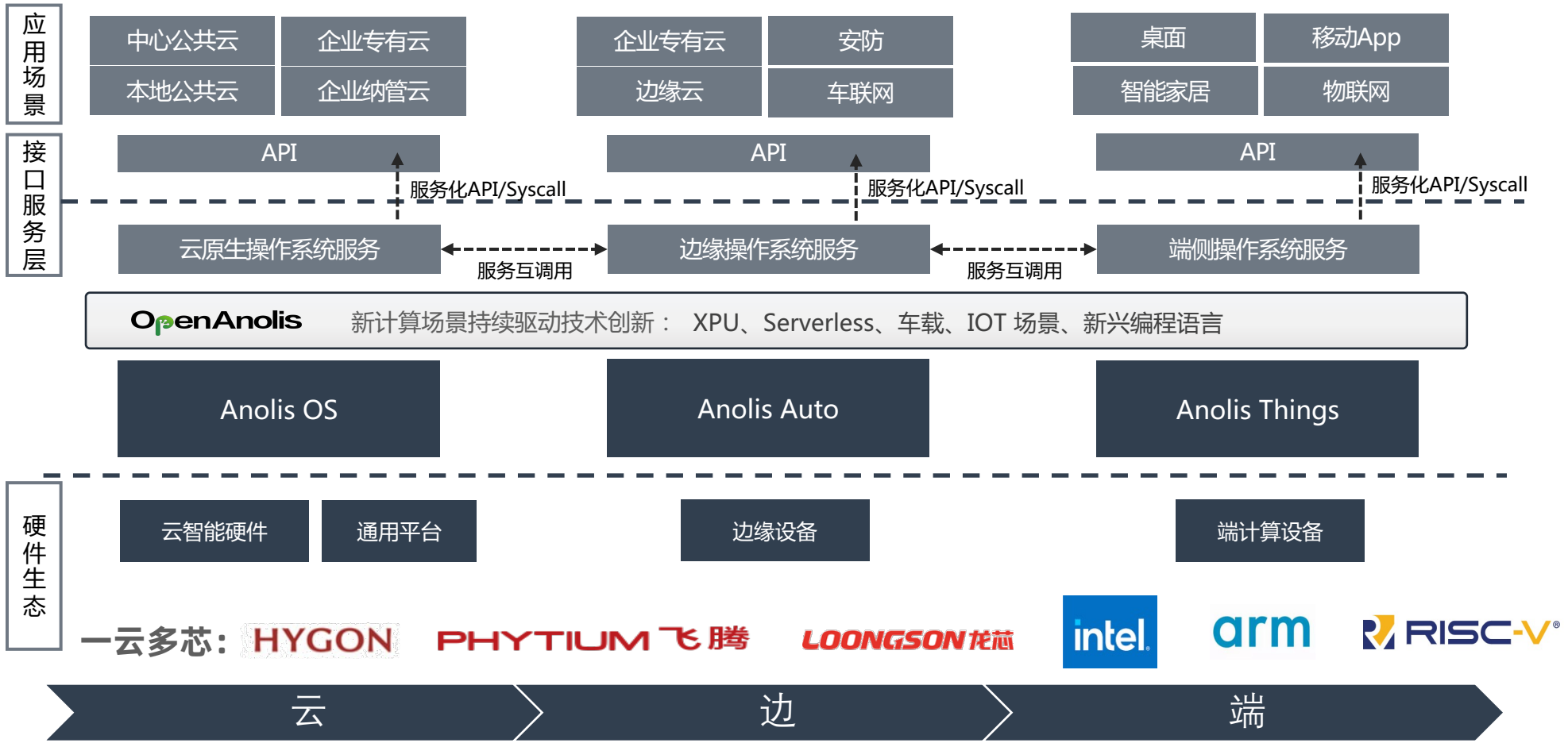
- 在第四代神龙 eRDMA 设备上支持 SMC-R 协议栈并实现规模化的管理
- 支持传统 RDMA 网卡
- 支持自适应识别能力，实现以太网和 RDMA 链路探测和切换

价值和效果

- 可以支持容器级别或者应用级别的无需任何修改的 TCP 到 SMC-R 的替换，部分应用性能提升 20%~50%

展望未来：龙蜥社区 - 不仅仅是 Linux 生态

Anolis - Anolis's Not Linux System，成为分布式云时代的数字基础设施助推器，促进云边端计算融合



持续演进

多架构的支持

- 一云多芯、一 OS 多芯片
- 标准化，解耦合
- 管理运维一致性

操作系统碎片化

- 整合产业链，有序演进
- 标准化，模块化
- 跨场景协同

全栈系统的割裂

- 场景容器化，应用流化
- 云原生化，应用解耦合
- 软硬协同，系统深度设计

OpenAnolis

关注社区微信公众号

加入社区钉钉交流群

谢谢