QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2018

《容器生态下的高性能负载均衡建设之路》

演讲者 / 韩建飞

主办方 Geekbang. 极客邦科技 InfoQ

目录

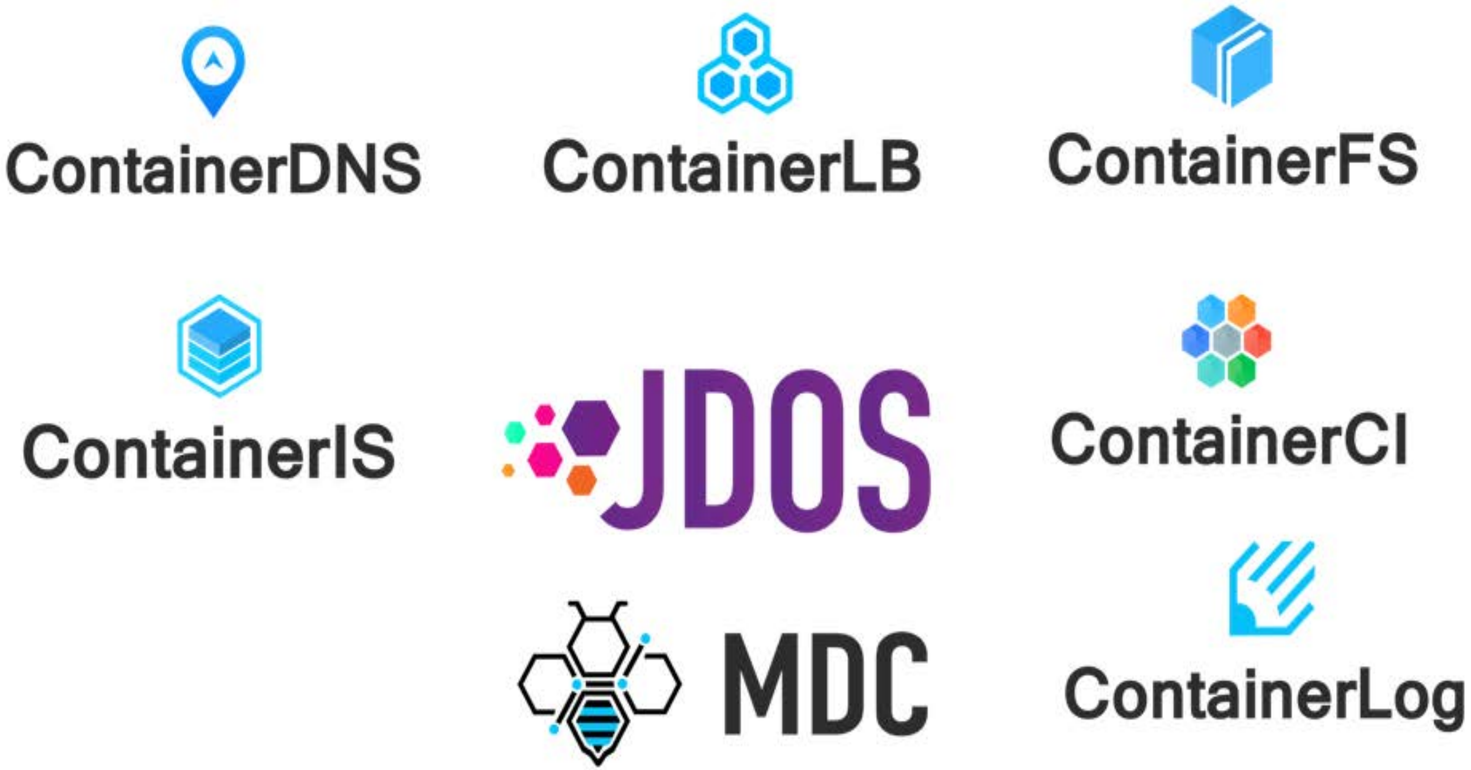
数据中心操作系统（JDOS）生态

当前开源：

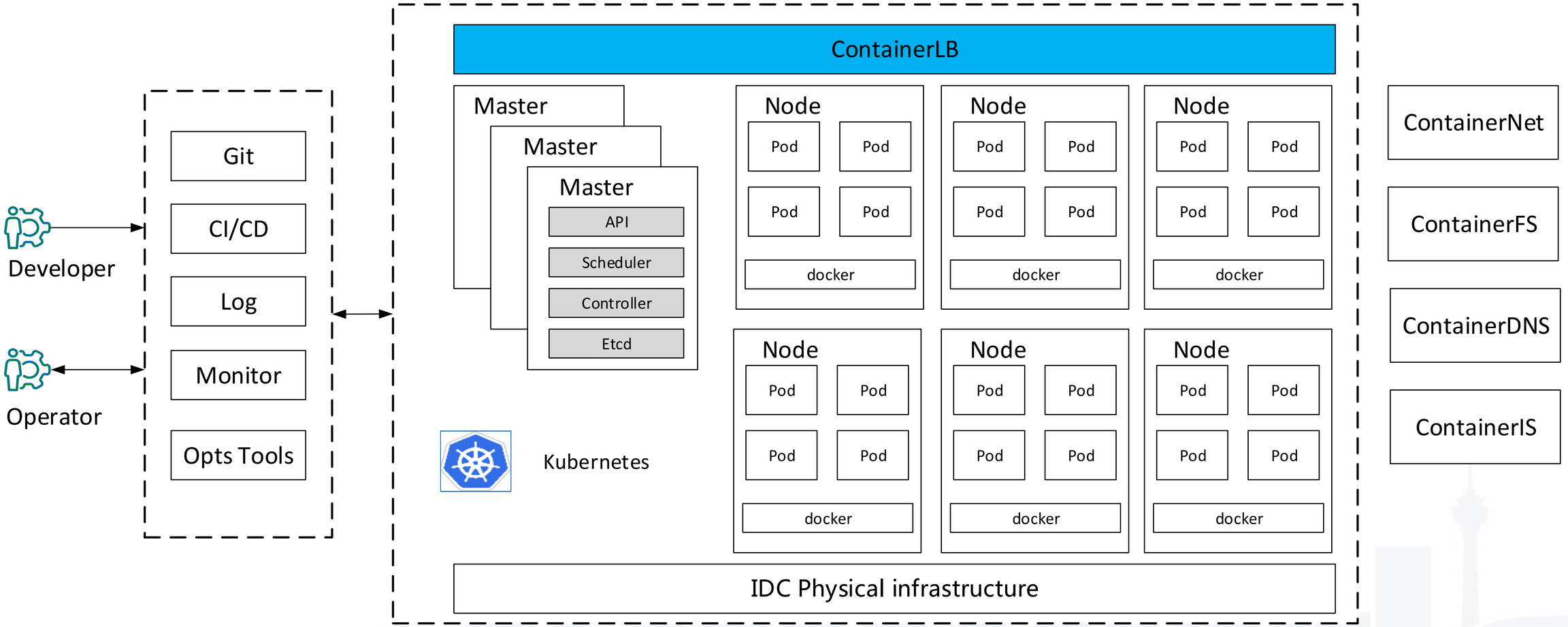
ContainerLB：
https://github.com/tiglabs/jupiter
https://github.com/tiglabs/ipoa

ContainerDNS：
https://github.com/tiglabs/containerdns

ContainerFS：
https://github.com/tiglabs/containerfs



数据中心操作系统（JDOS）架构



ContainerLB 研发背景：

JDOS(K8S) Service需求
✓需要满足严肃生产环境Service服务。

LVS/Nginx/Haproxy
✓DR,TUN, NAT模式需要修改容器配置。
✓FULLNAT内核版本要求。

性能
✓ 业务日益增长的高并发、低延时的需求。

成本
✓ 在同样流量下，需要更少的负载均衡机器，节约成本。

ContainerLB 目标：

- 接近网卡线速的报文转发性能。
- 支持常见的4层负载均衡功能，配置灵活。
- 充分发挥双路服务器性能，支持两个zone的网卡避免计算资源闲置。
- 融入JDOS生态，落地K8S Service。
- 部署简单，避免改动客户端和后端服务。业务应用无感知。
- 负载均衡集群高可用。做到条件热升级。

ContainerLB 功能特性：

1）协议支持

负载均衡支持包含TCP、UDP协议的四层负载均衡，配备健全的链路跟踪机制，以及多种调度策略，用户可以根据服务的需要创建合适自己的负载均衡。

2）链路跟踪

支持tcp链路跟踪，记录链路状态，支持快速会话释放。

3）高可用性

支持容器的健康检查，保证应用可用性： 负载均衡会定时探测后端服务是否正常运行，健康检查频率可自定义；探测到异常，则不会将流量再分配到这些异常实例，保证应用可用性。

负载均衡自己高可用通过配合bgp ecmp 来支持。

4）集群部署，多层次容错机制

负载均衡采用集群部署，支持热升级，机器故障和集群维护对用户完全透明，结合DNS使用还可支持全局负载均衡。

ContainerLB 功能特性：

5）灵活性

多种流量调度算法

6）易用性

提供多种管理途径，轻松操纵负载均衡， 用户可通过控制台轻松实现负载均衡器的配置、释放等功能。
开放标准的API或SDK提供给用户，从而自己开发对负载均衡的控制管理。
融入jdos 生态容器系统。

ContainerLB 调度算法：

- ✓ Source-Ip： 源ip hash 调度
- ✓ Source-Ip-Port： 源ip+port 一致性hash 调度
- ✓ Least-Connection： 最小连接数调度
- ✓ Round-Robin： 轮询调度
- ✓ Weighted Round-Robin： 加权轮询调度

支持会话保持，满足用户个性化需求： 负载均衡通过IP地址实现会话保持，可将一定时间内来自同一用户的访问请求，转发到同一个后端容器上进行处理，从而实现用户访问的连续性。

链路跟踪，支持vip 链路快速释放，防止在流量洪峰时，本地ip+port 资源不足。

ContainerLB 集群架构：

Container LB 集群架构

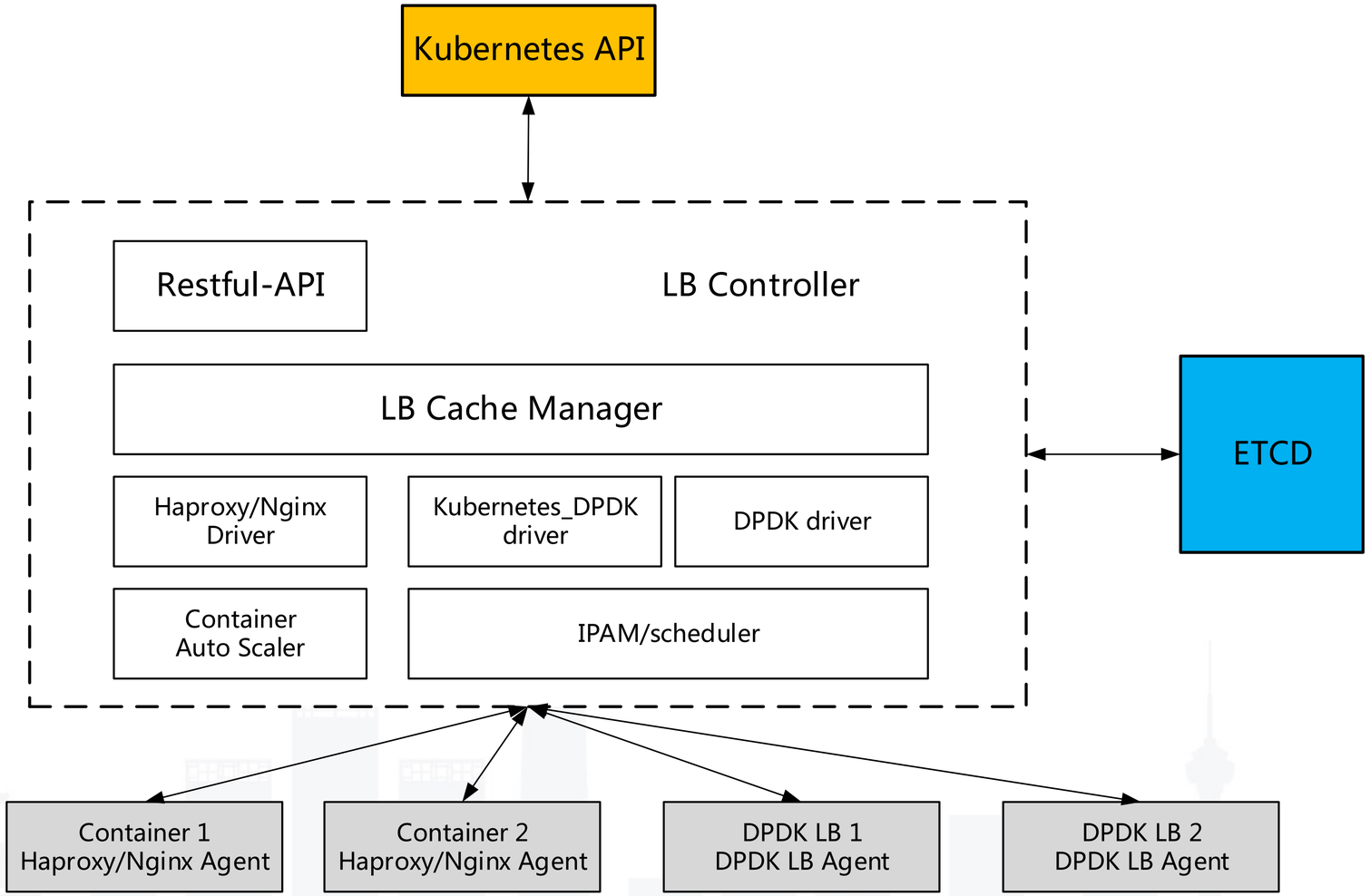
✓支持DPDK LB

✓支持Haproxy/Nginx。

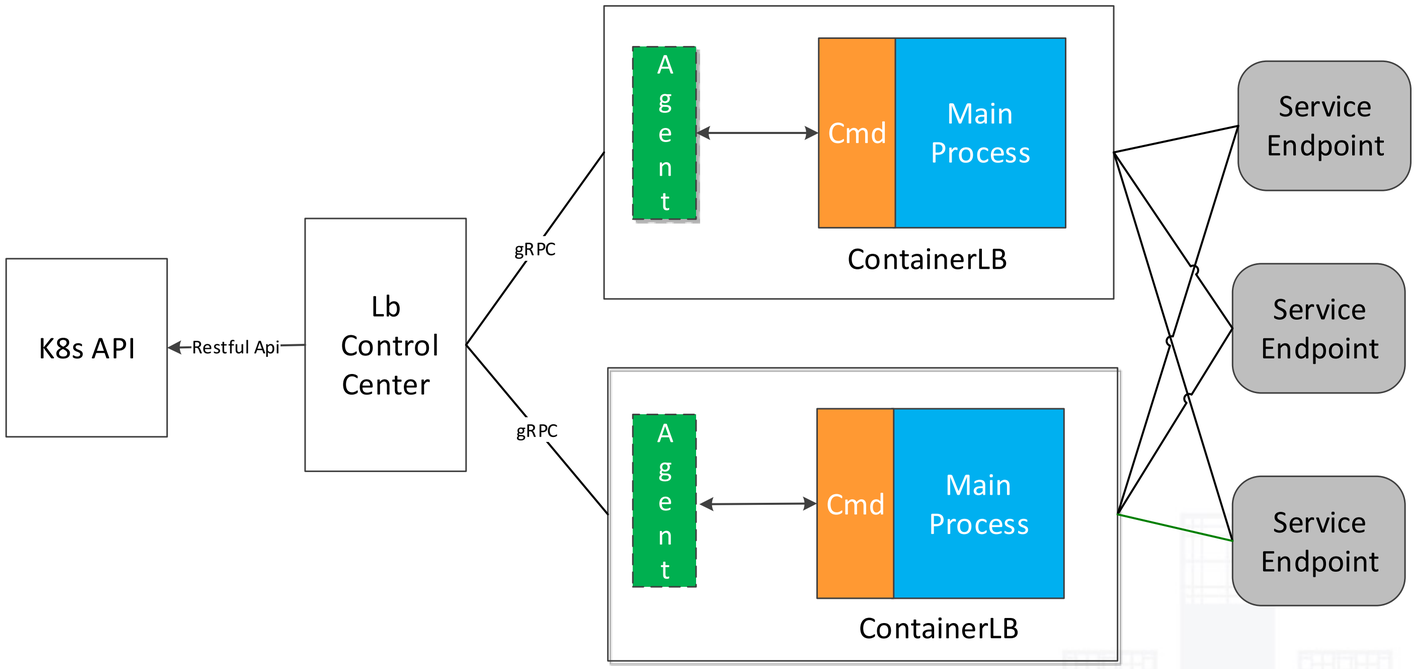
✓支持融入kubernetes的集群管理

✓支持单独的DPDK LB Driver
驱动单独集群

✓Haproxy/Nginx 负载均衡容器化
自动扩容和迁移



ContainerLB（DPDK）管理架构：



Container DPDK LB 管理架构

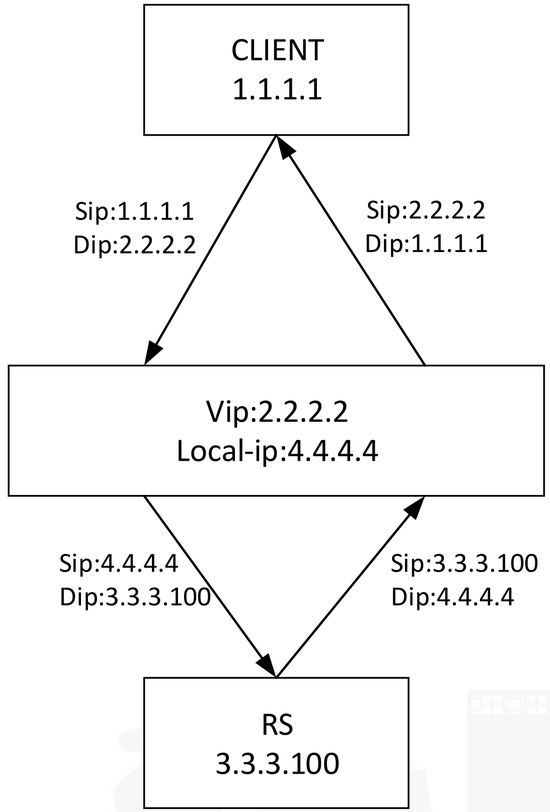
✓LB control center 与k8s 集群管理交接。

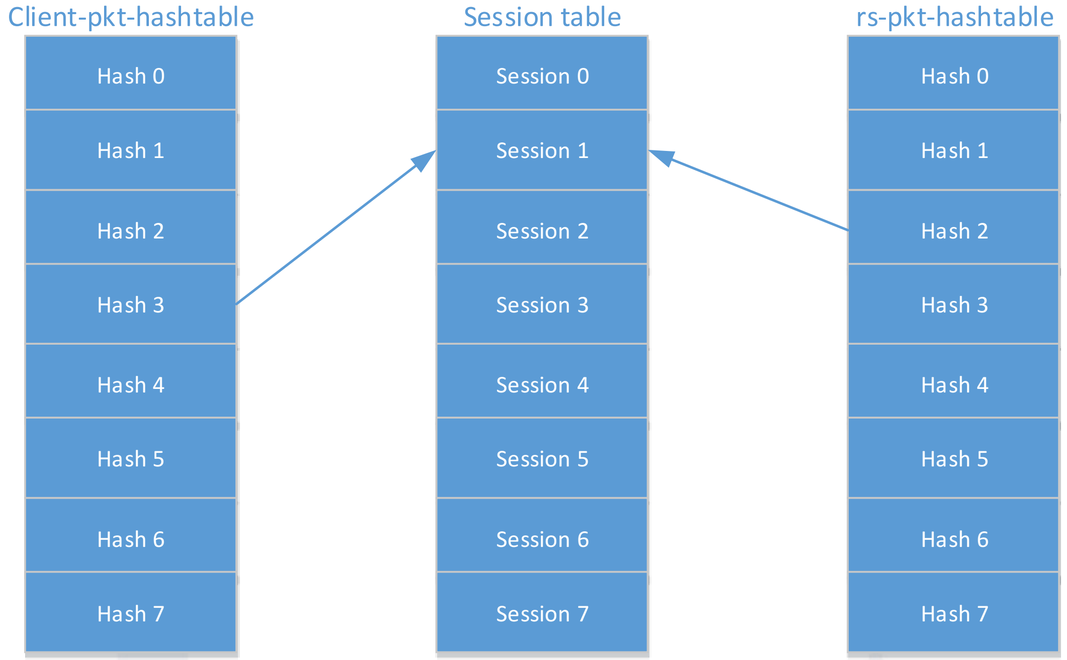
✓Agent 通过调用 cmd client 设置 Container LB参数。

✓Cmd client 通过与file socket 与 Dpdk Main Process 交互。

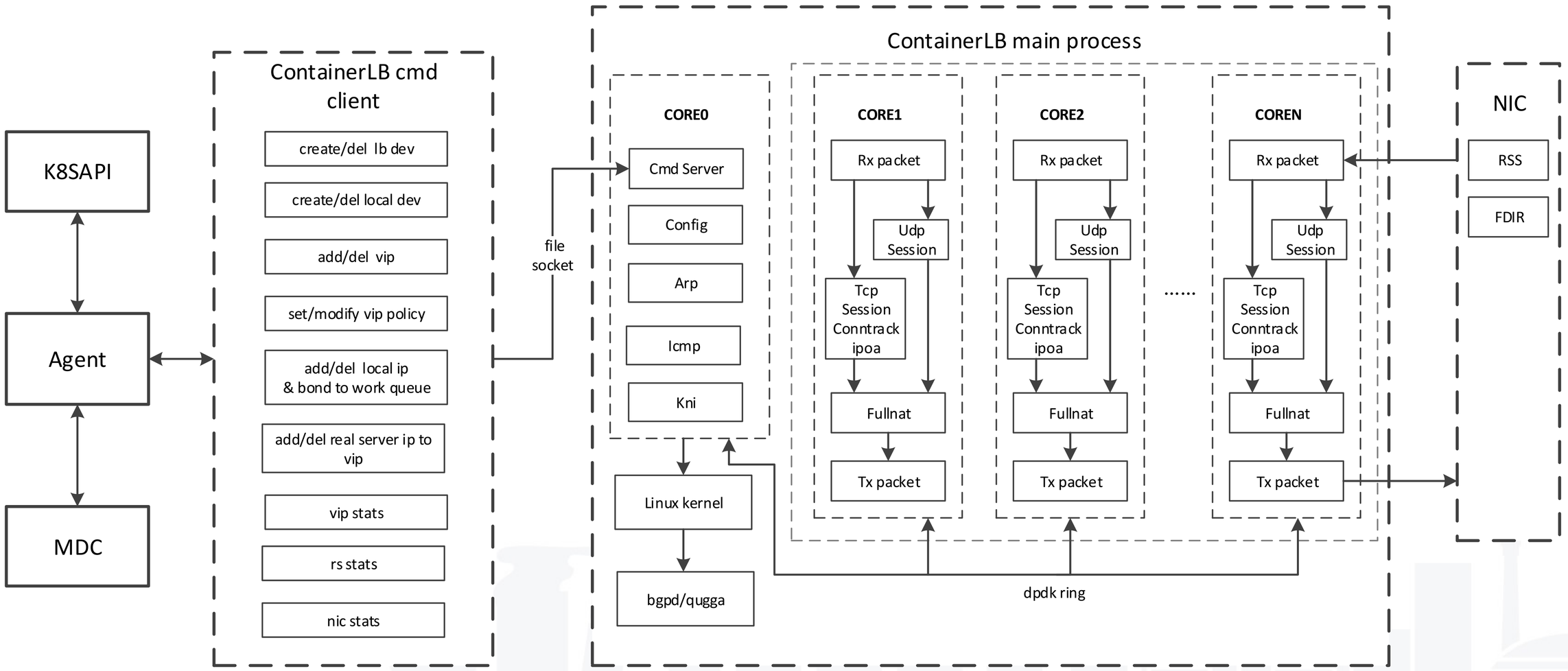
ContainerLB 模型：

- ✓FullNat
- ✓DR
- ✓NAT
- ✓TUNNEL





ContainerLB 核心流程：



ContainerLB 包处理：

✓ Run-to-completion(RTC)

核心都会独立执行报文接收，处理、发送任务，各个核心互不影响。

✓ Pipeline

报文接收，处理，发送任务将分别赋予不同的核心处理，各核心组成流水线工作。

✓ 包处理

ContainerLB选择一个核作为控制核，执行命令配置，与内核交换，ARP表维护等任务。

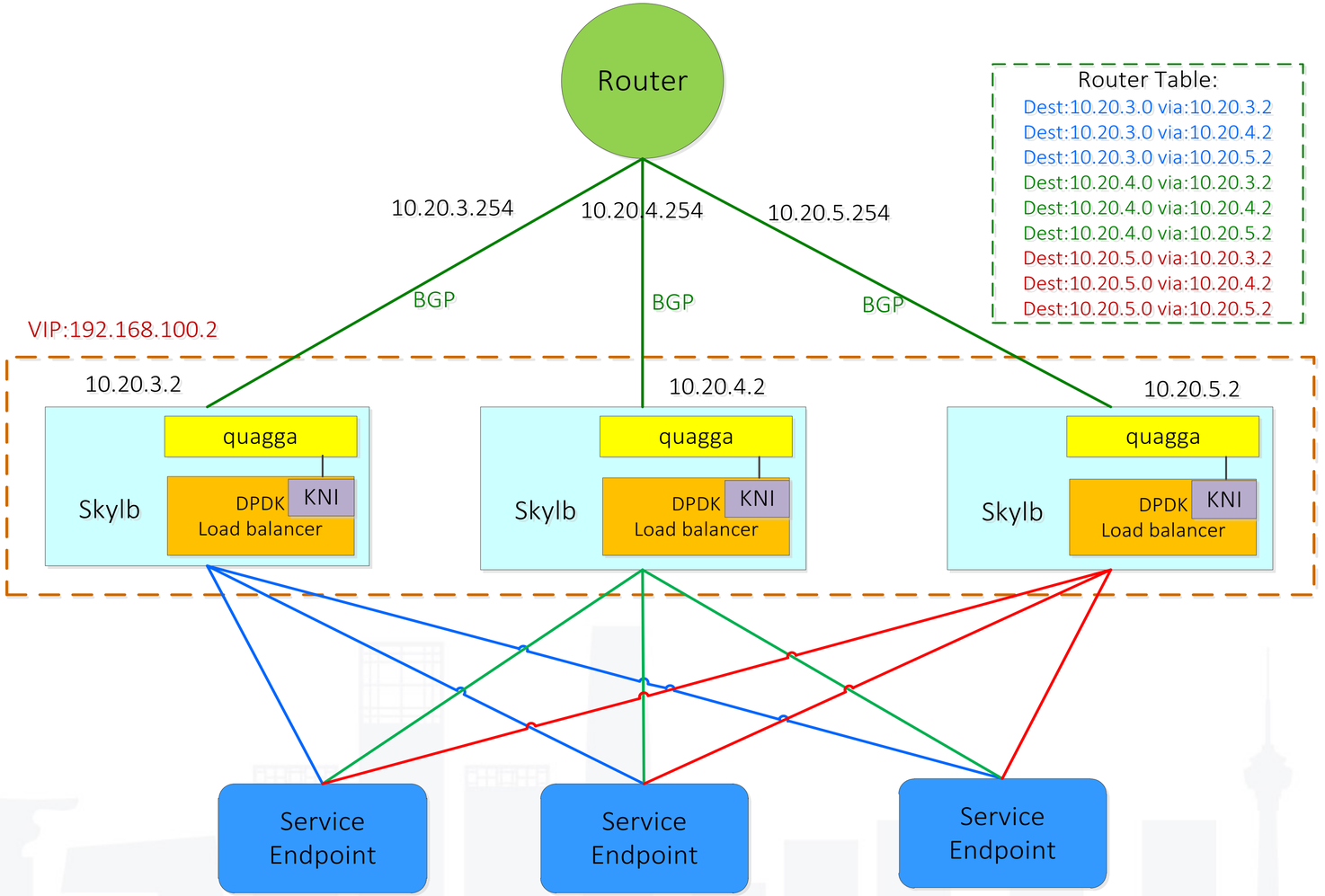
其他的核作为工作核，每个工作核轮询网卡的一个RX队列，执行数据包处理任务。

ContainerLB利用网卡的RSS功能实现客户端请求报文的分流，利用网卡FDIR功能实现后端服务响应报文的分流。 ContainerLB对报文分流目的是要保证客户端的请求报文和其对应的服务端响应报文到达同一个工作核上。

在这个目的达成后下， ContainerLB的业务实现会简单和高效很多。每个工作核维护一张session表，同于保存客户端和后端服务的连接信息。 ContainerLB不需要考虑对session表加锁，因为每个工作核的session表都是独立的。

ContainerLB组网及冗余：

- ✓ 支持传统二层vlan
- ✓ 支持三层bgp组网
- ✓ 配合交换机 ecmp
  实现负载均衡的高可用



ContainerLB性能提升：

- ✓ DPDK本身的性能优势，poll mode，无锁队列，内存零拷贝，用户态驱动。
- ✓ 尽量少的避免锁的使用。
- ✓ 恰当地使用rte_prefetch0()，可以减少cache-miss次数，避免当前函数成为性能热点。
- ✓ 恰当地使用likely()和unlikely()，可以减少分支预测失败的次数。

ContainerLB性能数据：

TCP性能测试数据：

调度算法	线程数	Min(ms)	Max(ms)	Avg(ms)	TP99(ms)	QPS(笔/秒)	交易失败数	交易失败率
tcp.IP_port	1000	0	124	0	1	2088316.1	0	0.00%
tcp.RR	1000	0	226	0	1	2070281.7	0	0.00%
tcp.LC	1000	0	410	1	6	696950.7	0	0.00%

UDP性能测试数据：

调度算法	线程数	Min(ms)	Max(ms)	Avg(ms)	TP99(ms)	QPS(笔/秒)	交易失败数	交易失败率
udp.IP_port	1000	0	58	0	1	4212952	0	0.00%
udp.RR	1000	0	226	0	1	4325008.4	0	0.00%
udp.LC	1000	0	58	1	3	812356.2	0	0.00%

谢　　谢