kata containers 2.0 性能调优探索与实践

李　宁

中移（苏州）软件技术有限公司

简介

李宁

中移（苏州）软件技术有限公司研发工程师，从事Linux系统、容器的定制开发、优化工作。近期的工作专注点是开源安全容器技术研究及安全容器技术在移动云产品中的落地开发。

中移苏研 基础软件产品团队

BC-Linux

主要负责为中国移动定制开发基础软件产品，为云平台提供稳定、可靠、安全的基础设施、业务基座。主要产品包括大云企业操作系统、虚拟化产品，软负载均衡产品。

目录

- 移动云与kata containers的联系
- 选择kata作为安全容器方案的考量
- kata的架构与网络模型概述
- 性能调优探索与优化思路分享
- 展望
- Q/A

移动云与Kata Containers的联系

考量 – 安全容器方案对比

安全容器方案	实现方式	隔离方式	性能、轻量化	通用、灵活性	适用场景	使用的厂商
Google gVisor	用户态内核	拦截系统调用	IO、网络性能不佳	较低，仅支持有限的系统调用，不通用	Serverless	Google
AWS Firecracker	轻量级虚拟化	虚拟化	高	一般，不完整的虚拟化，不够通用	Serverless	aws、ucloud
Kata Containers	轻量级虚拟化	虚拟化	高	高，支持多种轻量级虚拟化hypervisor	多种需求场景	Baidu、Huawei、Ant Financial、Alibaba

这里仅对比了3种较为主流的安全容器方案

总结：

- ✓ Kata contaienrs灵活性高，可以根据自己的需求场景选择合适的hypervisor，一套方案能覆盖多种场景：普通场景、serverless、边缘端
- ✓ 相比于gvisor，kata性能更好。相比于firecracker，更加云原生。
- ✓ 直接使用gvisor和firecracker的容器方案厂商较少，kata容器国内参与的厂商较多。

考量 – 我们的计划、打算

产品1：函数计算服务
方案：kata with fc

firecracker

优点：
- fc专为函数计算场景设计

缺点：
- 不完全的虚拟化功能，不通用

评价：
- 缺点也是优点，在函数计算场景下，最小的虚拟化功能，带来的是更好的安全性和低开销

产品2：通用安全容器服务
方案：kata with clh

cloud-hypervisor

优点：
- clh是为云原生设计的hypervisor，设计理念贴合容器场景。
- 轻量级，启动快，低开销，安全。
- 相比于fc更加通用
- 相比于qemu更加轻量安全。

缺点：
- 目前还不够成熟，尚不能用于生产

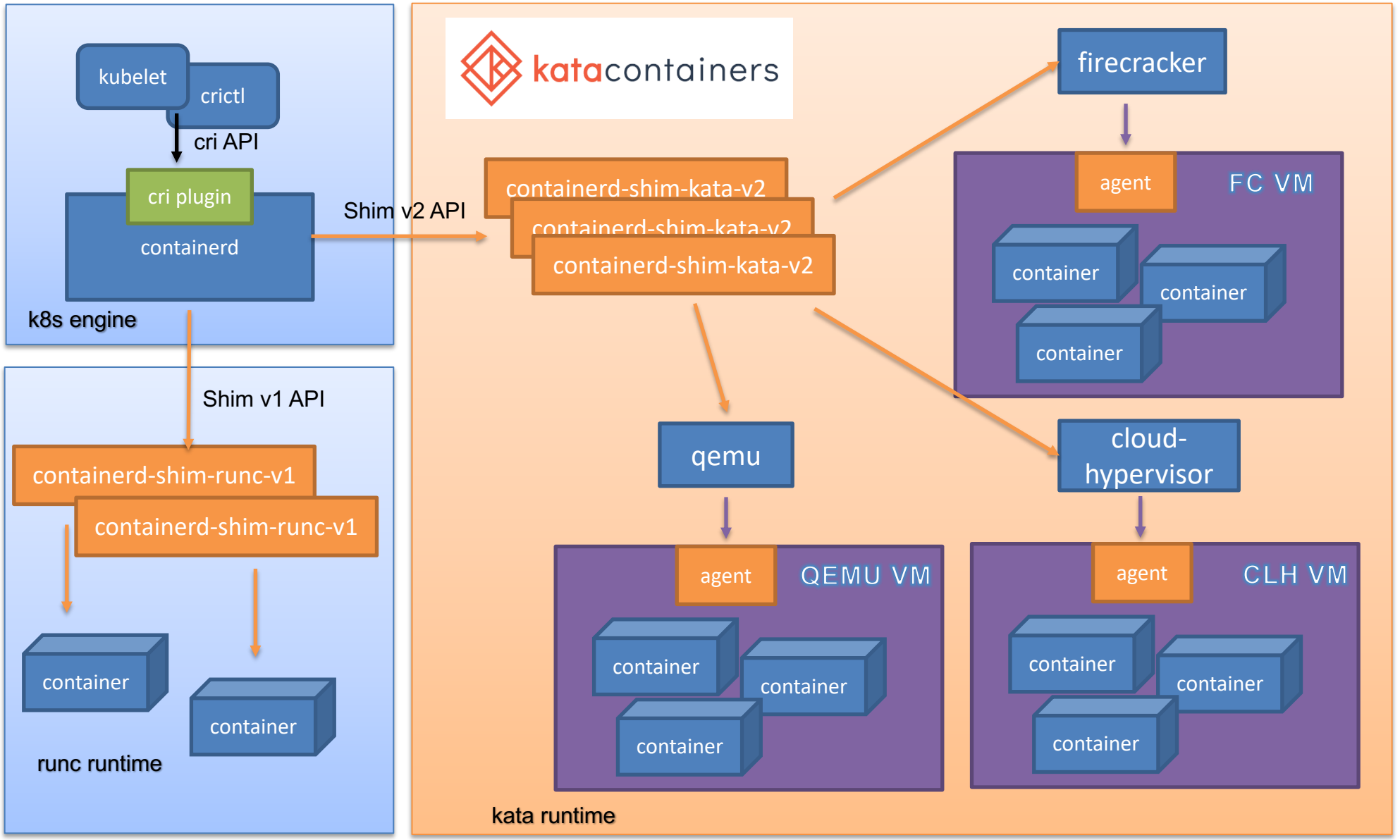
评价：对cloud-hypervisor抱有期待，未来kata的主力hypervisor，目前可以先使用kata with qemu过度，持续关注

Kata 整体架构

中国移动 China Mobile | 中移软件 CMSoft

技术要点总结：

1. VM作为安全沙箱
2. 支持多种hypervisor
3. 精简的guest kernel和guest os环境，快速和安全
4. VM中agent负责直接创建、更新、销毁容器
5. 使用vsock作为shim v2进程与agent通信渠道
6. 使用tc filter/macvtap连接veth和tap，打通CNI和VM网络
7. 通过virtio-9p、virtio-fs将host上容器rootfs目录挂载进VM
8. 通过块设备passthru，将host上block设备传进vm作为容器的rootfs



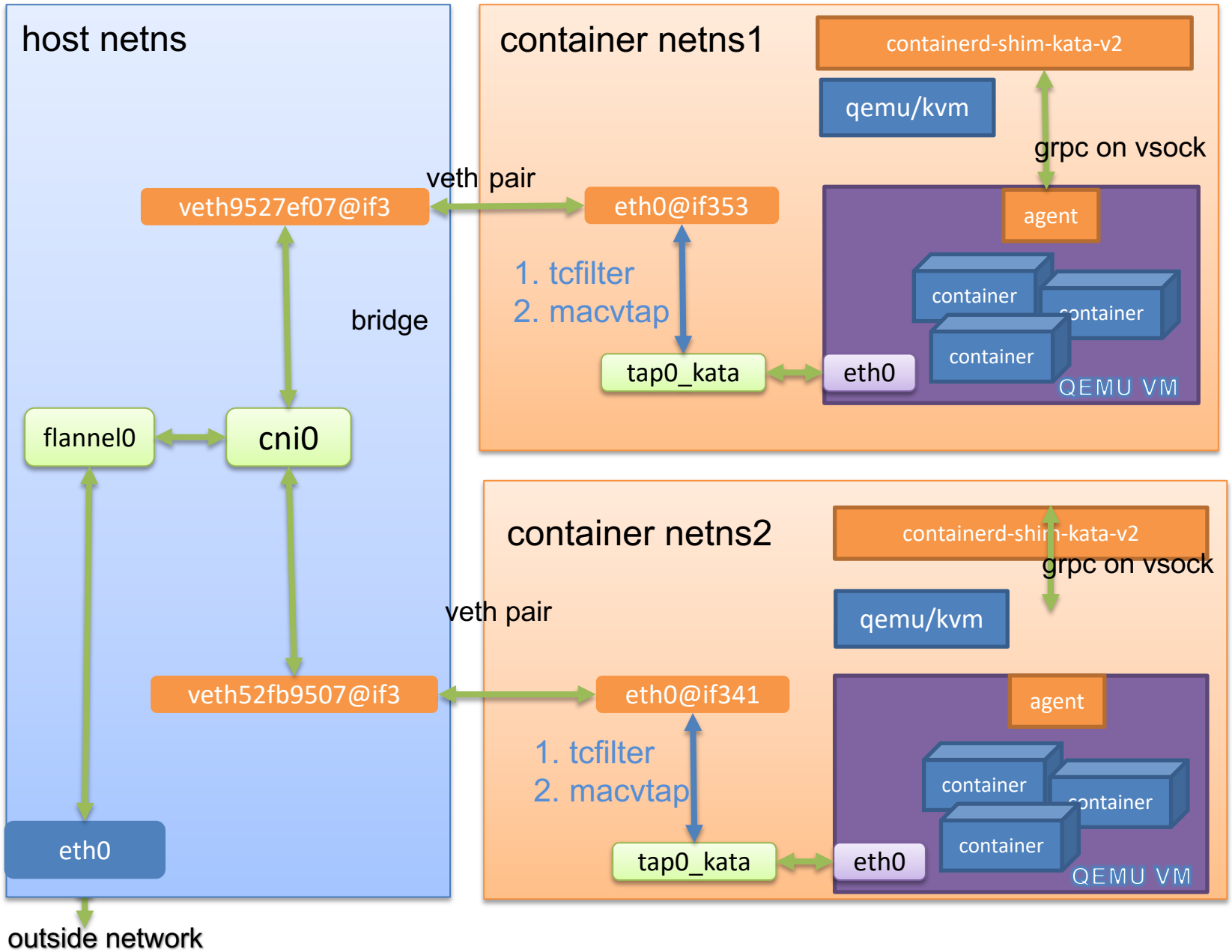
Kata 网络模型

技术要点总结：

- ✓ 主要解决的问题：现有容器网络模型与现有虚拟机网络模型不匹配的问题，将CNI网络和虚机网络对接。
- ✓ veth和tap连通方案：
  1. tcfilter：使用tc rules将veth的ingress和egress队列分别对接tap的egress和ingress队列实现veth和tap的直连
  2. macvtap：现有虚拟网卡连通技术

优化思路：

- ✓ 针对kata重新设计CNI
  1. 直接打通cni0网桥与tap设备，实现思路：直接attach tap到cni0
  2. 基于dpdk实现，cni0变为一个vswitch，与VM中eth0，基于dpdk + vhost-user
  3. 都是现有虚拟化中比较成熟的方案，重点在于如何针对kata设计CNI



关于kata容器的考量

我们从哪些维度去考量kata containers？

1. 容器启动速度
2. CPU、内存性能损耗
3. 资源开销
4. 网络性能
5. 文件IO性能
6. 整体表现

1 容器启动速度

- CPU: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
- Host kernel：4.19.0-193.1.3.bclinux.x86_64
- 磁盘：机械硬盘
- 测试命令：
  - **test1:** time ctr run --runtime io.containerd.kata.v2 -t --rm docker.io/library/busybox:latest hello uname –a
  - **test2:** ctr run --runtime io.containerd.kata.v2 -t --rm docker.io/library/busybox:latest dmesg

test1: 测试启动销毁总耗时

runtime + storage driver	耗时
runc + overlayfs	0.5s
runc + devmapper	1.3s
kata + virtio-9p	0.8s
kata + virtio-fs	1.1s
kata + devmapper	1.5s

test2: vm启动时间

hypervisor	耗时
kata + qemu	347ms (kernel) + 115ms (userspace) = 462ms
kata + cloud-hypervisor	272ms (kernel) + 79ms (userspace) = 352ms
kata + firecracker	176ms (kernel) + 137ms (userspace) = 314ms

说明：

1. test1中测试使用的qemu，测出来的时间实际上是容器启动 + 执行命令 + 容器销毁的总时间
2. test1中virtiofs 比virtio9p多耗时0.2s，virtiofs需要启动virtiofd，多出来的0.2s是virtiofsd启动、销毁时间开销

容器启动速度

- kata container启动时间构成分析：
  - ✓ Step1: qemu启动 + virtiofsd (vhost-user-fs server)启动 + kvm创建vm资源
  - ✓ Step2: guest os内核态：kernel bootup
  - ✓ Step3: guest os用户态：systemd + agent启动
  - ✓ Step4: vm中agent创建container + container启动

- 调优、优化思路：
  - ✓ 使用轻量化hypervisor，好处是进一步精简了VM的设备模型，减少了内核中设备初始化时间开销
  - ✓ 开启vm template，将Step1 ~ Step3变为了vm template clone and resume，减少了总的时间开销
  - ✓ host kernel打补丁加固：df12eb6d6cd9 ("net: virtio_vsock: Enhance connection semantics ")
  - ✓ 使用固态硬盘作为物理存储，加快文件读取

2 CPU、内存性能损耗

- CPU: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
- Host kernel：4.19.0-193.1.3.bclinux.x86_64
- Pod规格：4c4g
- 测试命令：
  - CPU损耗：kubectl exec -ti ${pod} -- sysbench cpu run
  - Mem损耗：kubectl exec -ti ${pod} -- sysbench memory --memory-block-size=4k --memory-total-size=4G run

OCI runtime	CPU损耗	Mem损耗
runc	< 0.5%	< 0.5 %
kata	< 2%	< 1 %

性能损耗低原因：

✓ 高版本的kernel kvm模块、qemu代码做了优化，虚拟化的开销 < 1.5%

3 资源开销

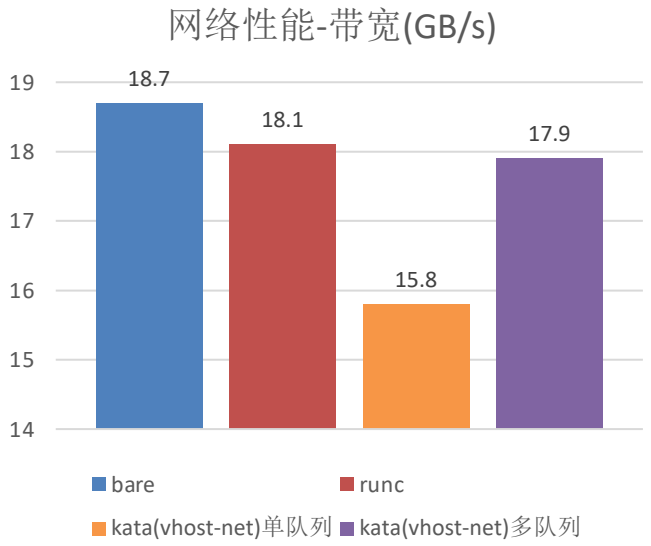
- CPU: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
- Host kernel：4.19.0-193.1.3.bclinux.x86_64
- Pod规格：4c4g
- 测试命令：
  - ctr run --runtime io.containerd.runc.v1 --rm -t docker.io/library/busybox:latest hello sh
  - cat /proc/${vm_pid}/state |grep –i vmrss

hypervisor	Mem Overhead
qemu	142820 kB
cloud-hypervisor	255428 kB
firecracker	88276 kB

4 网络性能

- 网卡：2个万兆卡组mode 4的bond，理论带宽是20Gb/s
- k8s网络插件: flannel，backend为vxlan
- Pod规格: 4c4g
- 测试命令：kubectl exec –ti ${pod} - - iperf3 –c ${server} –P 4

Runtime	带宽 (GB/s)
bare	18.7
runc	18.1
kata vhost-net单队列	15.8
kata vhost-net多队列	17.9



调优、优化思路：

- 开启虚拟网卡多队列（kata支持，但存在“缺陷”，默认单vcpu，单队列，需要改进）
- 虚机内部中断均衡（实测表现性能提升不明显）
- 虚拟网卡ring buffer长度默认是256，建议修改为1024

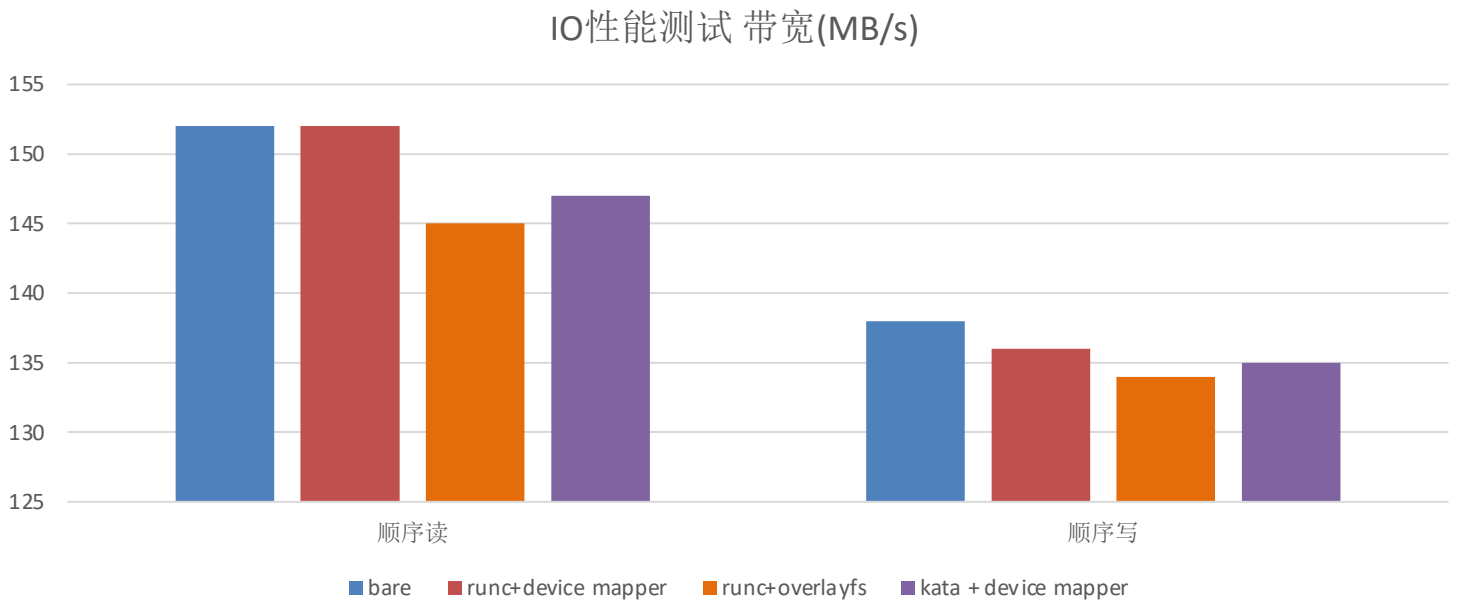
```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kata-bench
  labels:
    name: benchmark
spec:
  selector:
    matchLabels:
      name: benchmark
  template:
    metadata:
      labels:
        name: benchmark
    spec:
      runtimeClassName: kata
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
      - name: benchmark
        image: benchmark:ubuntu
        imagePullPolicy: IfNotPresent
        command: ["sleep", "60000"]
        resources:
          limits:
            cpu: 4
            memory: 4Gi
          requests:
            cpu: 4
            memory: 4Gi
```

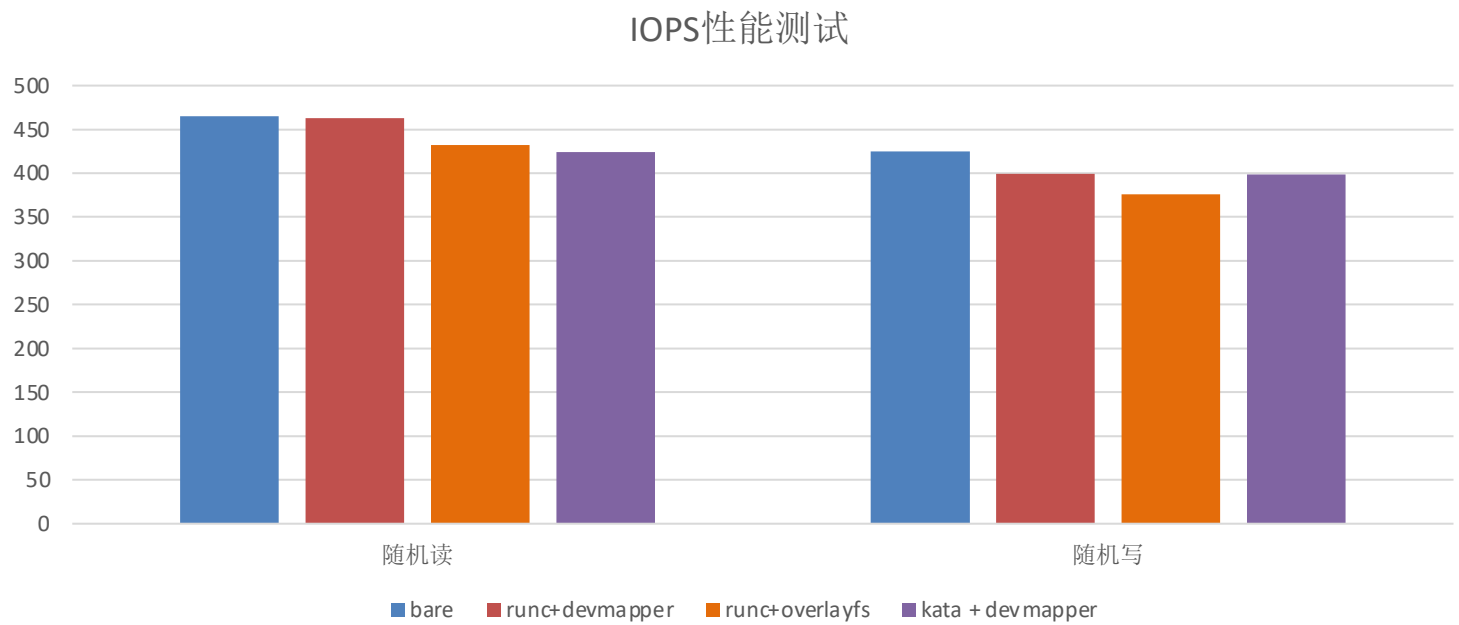
5 文件IO性能

- 磁盘：机械硬盘
- Pod规格：4c4g
- 测试命令：fio \
  -directory=/testdir \
  -rw=${rw} \
  -bs=${bs} \
  -size=4G \
  -numjobs=4 \
  -iodepth=128 \
  -runtime=30 \
  -direct=1 \
  -ioengine=libaio \
  -group_reporting \
  -name=rand${rw}_${bs}_test
- 参数：
  - 带宽测试：bs=128k，顺序读、顺序写
  - iops测试：bs=4k，随机读、随机写
- 调优、优化思路：
  - 使用device mapper作为存储驱动，性能较好且稳定，缺点是排查问题相对复杂
  - 打开iothread
  - 使用高性能固态盘、或是上spdk
- 遇到的问题/疑问：

1. virtio-fs cache设为none后，测试发现没有生效，还是会使用host page cache

2. virtio-9p无法关闭cache



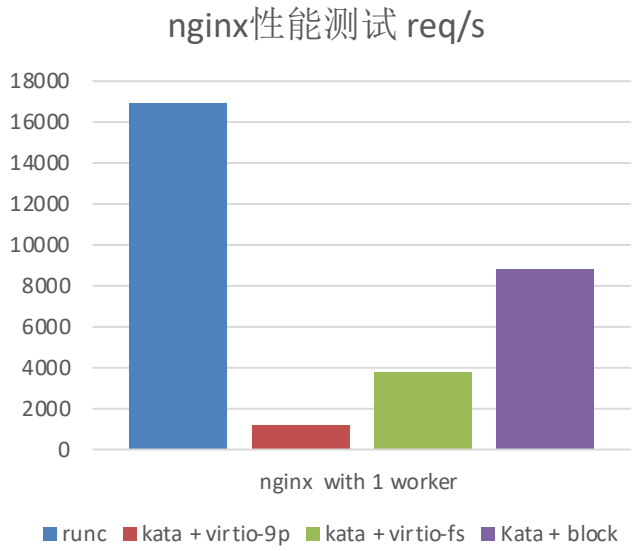


6 nginx性能

nginx压测

- Pod规格：4c4g
- 磁盘：机械硬盘
- 测试命令： ab -kn 100000 -c 100 http://${pod_ip}/

runtime&rootfs	nginx 1 worker
runc	16948.77 req/s
kata + virtio-9p	1181.99 req/s
kata + virtio-fs	3786.81 req/s
Kata + devmapper	8854.39 req/s




```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kata-nginx
  labels:
    name: benchmark
spec:
  selector:
    matchLabels:
      name: benchmark
  template:
    metadata:
      labels:
        name: benchmark
    spec:
      runtimeClassName: kata
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
      - name: nginx
        image: nginx:1.19
        resources:
          limits:
            cpu: 4
            memory: 4Gi
          requests:
            cpu: 4
            memory: 4Gi
```

调优、优化思路：

- 针对nginx、redis、mysql不同类型的具体应用，来设置内核调优参数
- 使用高性能固态硬盘

展望

- ✓ 轻量化，面向云原生，cloud-hypervisor可能是未来的主力hypervisor
- ✓ 软硬融合
- ✓ 定制化

中国移动 China Mobile | 5G 移动云

谢谢您的观看

中国移动 China Mobile | 5G 移动云

Q & A