



十年架构 成长之路

SACC 第十届中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2018

2018年10月17-10月21日 北京海淀永泰福朋喜来登酒店



腾讯GaiaStack容器产品私有云场景实践

腾讯 陈纯



第十届中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2018



自我介绍

陈纯，腾讯TEG数据平台部高级工程师

- Docker Libnetwork项目maintainer
- 《循序渐进学docker》作者之一
- 对容器生态的Kubernetes, Docker, runc, Libnetwork, Flannel等开源项目都有大量的源代码贡献



十年架构 成长之路



GaiaStack介绍

GaiaStack是腾讯基于Kubernetes打造的企业级容器云平台。

GaiaStack作为数据中心操作系统，可以调度CPU或GPU等海量计算资源，运行所有计算框架，监控任务执行结果，让数据中心的所有资源被合理使用，协同完成各类应用场景的计算。



关于GaiaStack Docker私有云

GaiaStack是腾讯基于Kubernetes打造的Docker私有云解决方案，腾讯内部所有BG都有产品在GaiaStack上运行，包括IEG、TEG、OMG、WXG、SNG、CDG的信鸽、MTA、游戏云、EasyCount、广点通、深度学习平台等众多产品和服务。

GaiaStack

提供了从构建至交付到运行的一整套的解决方案



容器服务



持续集成



镜像仓库



资源编排



私有集群



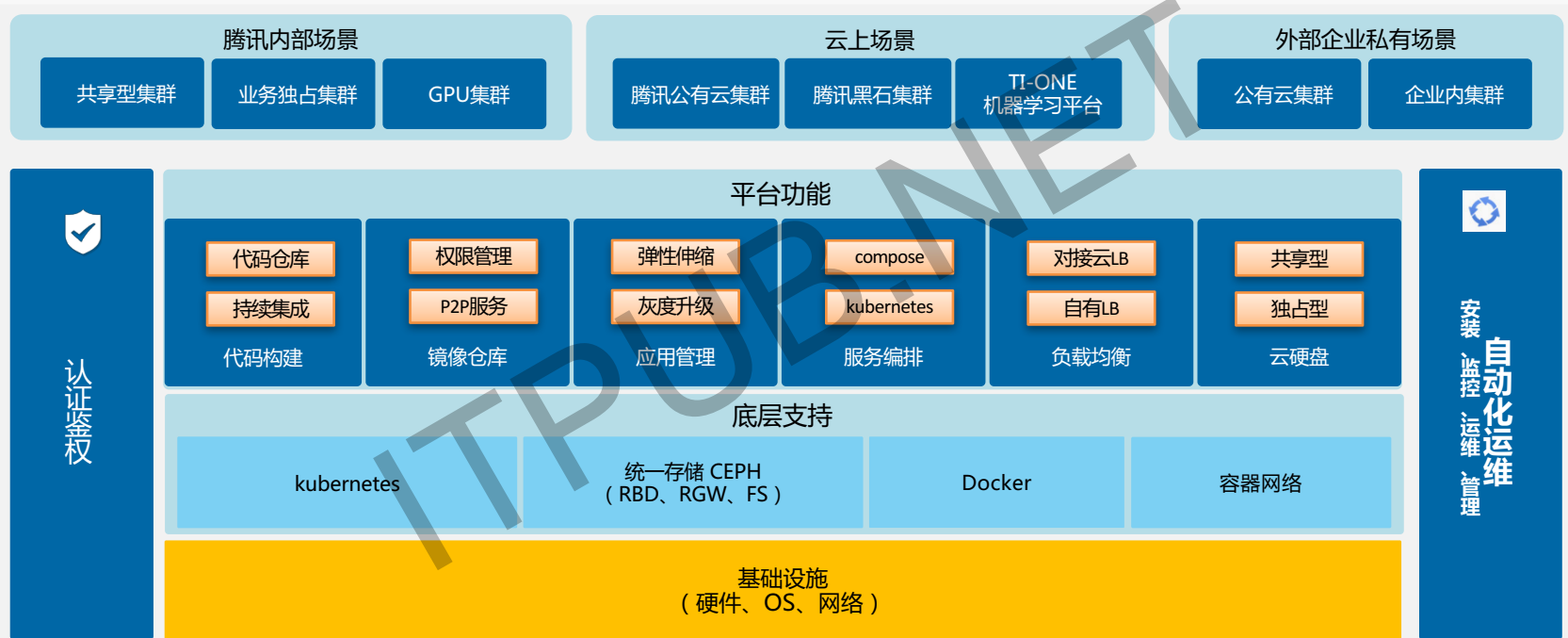
部署与管理



十年架构 成长之路



GaiaStack功能及应用场景



十年架构 成长之路



社区版Kubernetes私有云场景挑战

- 资源纬度仅支持CPU、Memory、ephemeral-storage (1.12 beta) , 网络出入带宽和磁盘IO如何管理？
 - <http://2017.qconbeijing.com/presentation/512>
- Deployment、Statefulsets、Job、CronJob应用类型真的好用吗？
- Flannel、calico、weave容器网络选用哪个？负载均衡用什么？
- 云硬盘用哪个？
- 升级版本不兼容？
- 日志，监控，告警问题

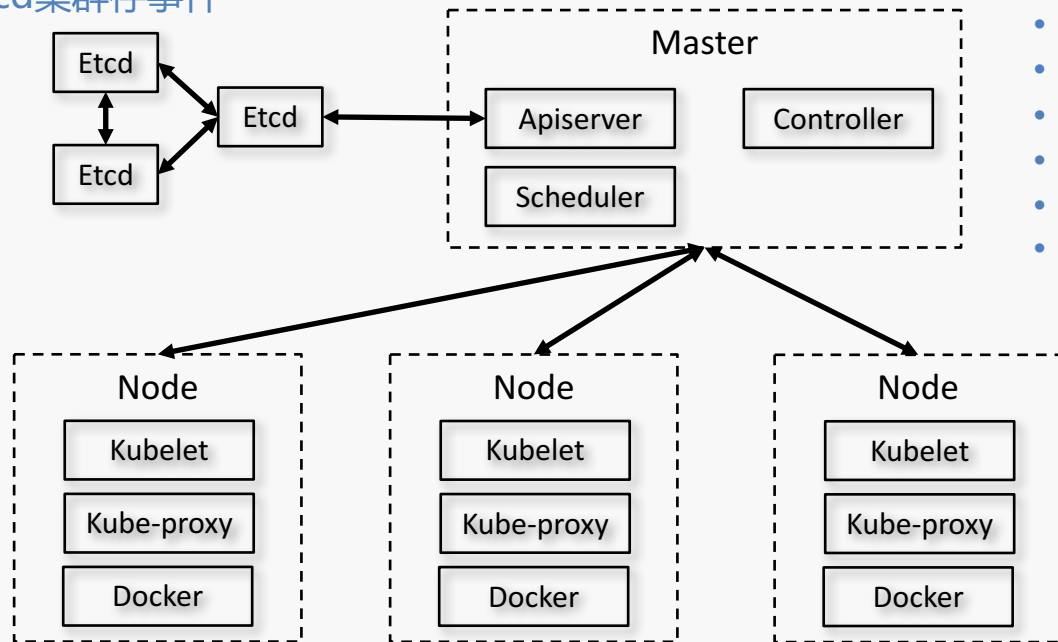


十年架构 成长之路



Kubernetes扩展接口

单独etcd集群存事件



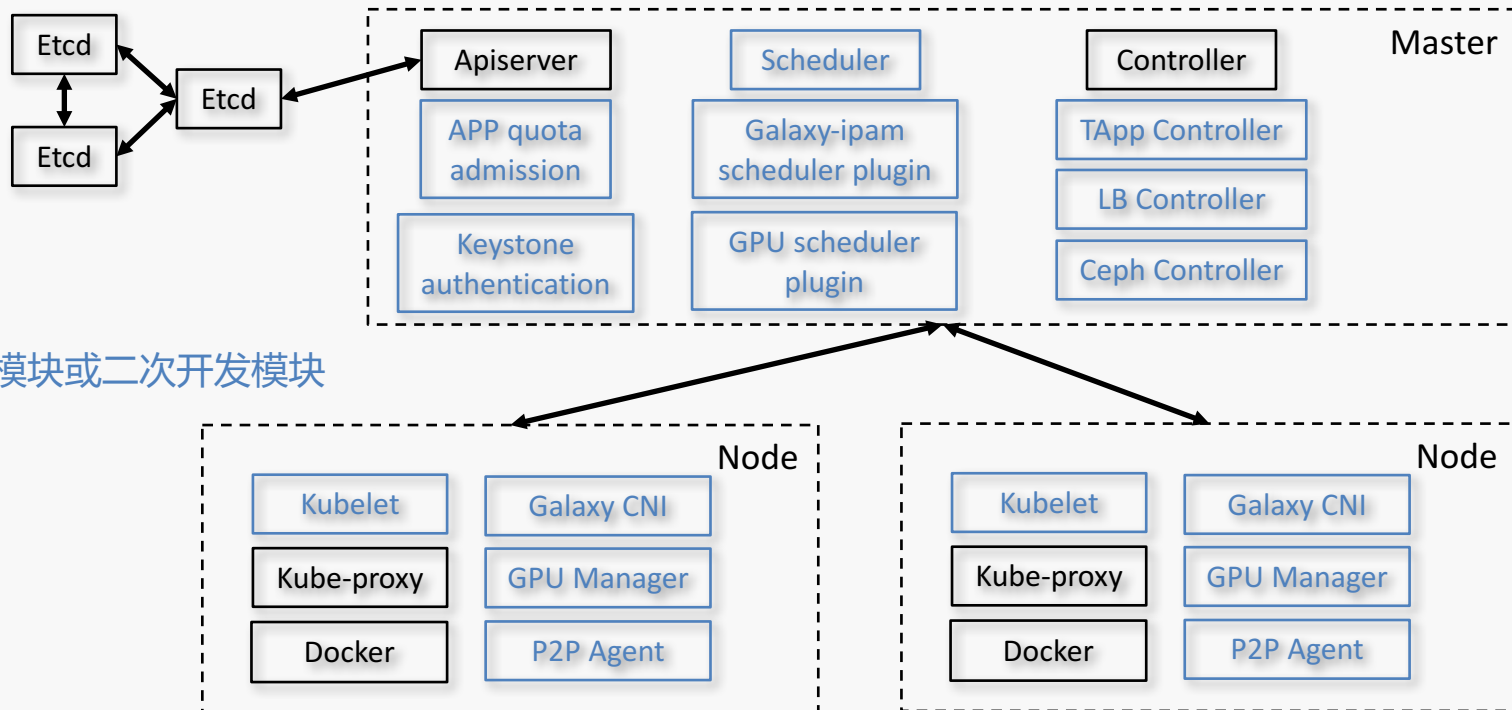
- 自定义API类型CRD
 - 认证、授权、准入扩展
 - 多个Apiserver存不同对象
 - 调度器插件
 - 多调度器
 - 自定义控制器
-
- 存储插件接口CSI
 - 设备插件DeviceManager
 - 网络插件CNI
 - 容器运行时接口CRI
 - Kube-proxy替代项目



十年架构 成长之路



GaiaStack Kubernetes



新增模块或二次开发模块



十年架构 成长之路



TAPP应用类型

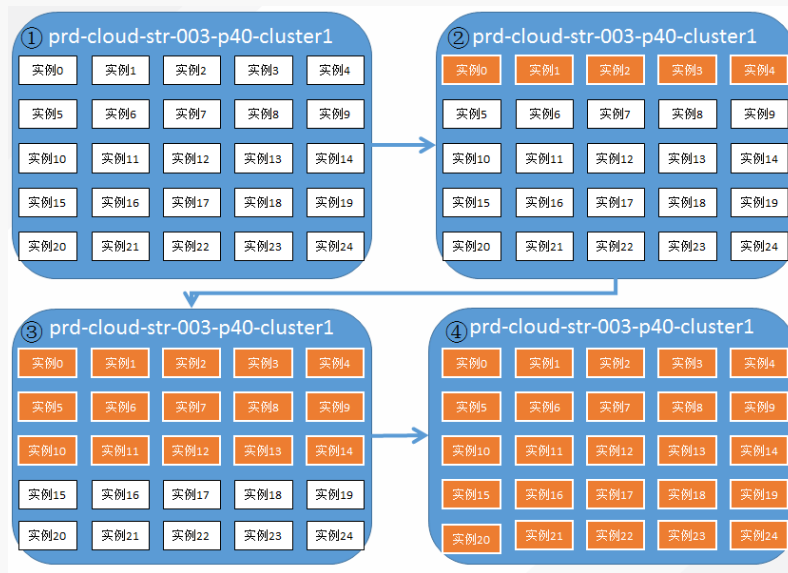
利用Kubernetes CRD功能自研TApp应用类型，与Kubernetes Statefulset应用类型相同点：

- Pod具有唯一自增ID
- 绑定单独云盘，迁移时数据盘跟随迁移

优势：

- **支持指定若干实例多次进行删除、停止、重启、原地灰度升级、回退Pod等操作**
- 单个TAPP应用的Pod支持N个版本
- 不只是修改镜像版本，甚至可以多加一个容器

```
type TAppSpec struct {  
    Template corev1.PodTemplateSpec //默认Pod模板  
    TemplatePool map[string]corev1.PodTemplateSpec  
    Statuses map[string]InstanceStatus  
}
```



十年架构 成长之路



Galaxy-CNI网络插件

因为各种不同场景的需要，自研网络项目Galaxy

- 同时提供Underlay + Overlay方案
- 普适性，多种网络适应不同场景
- 性能领先

选择应用适合的网络方案

- 不同的应用可以选择不同的网络模式
- **同一主机的不同容器可以选择不同的网络模式**

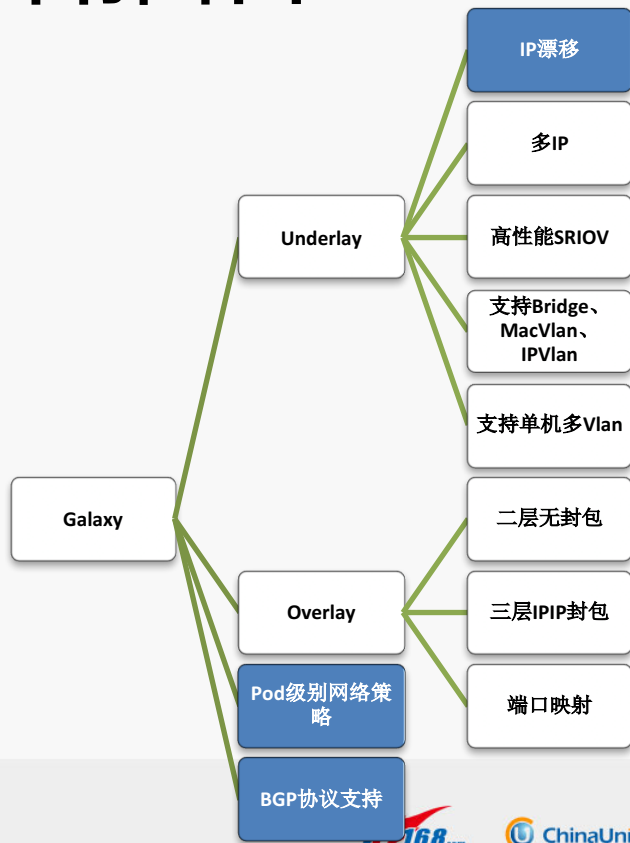
网络模式： Floating IP（浮动IP）

IP漂移： Floating IP（浮动IP）

Overlay（虚拟网络）

NAT（端口映射）

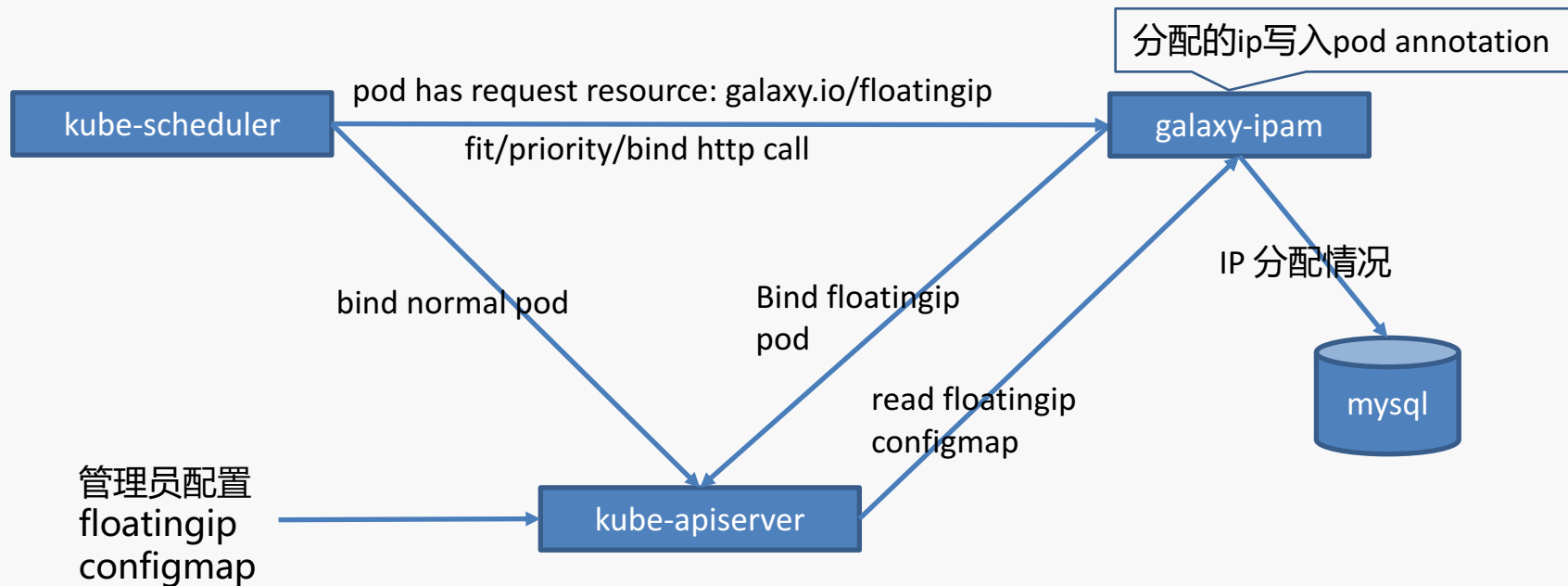
Host（宿主机网络）



十年架构成长之路



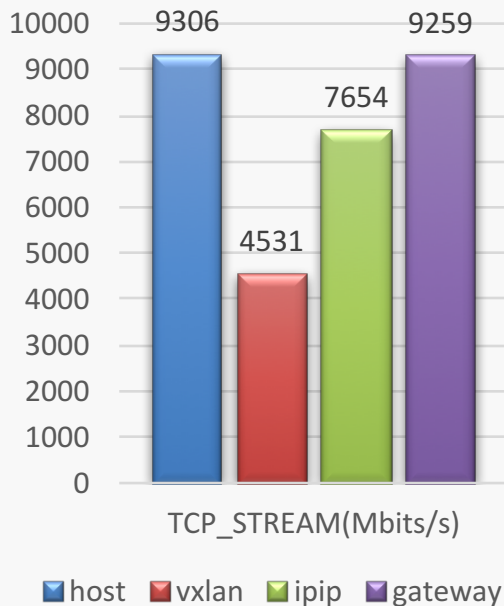
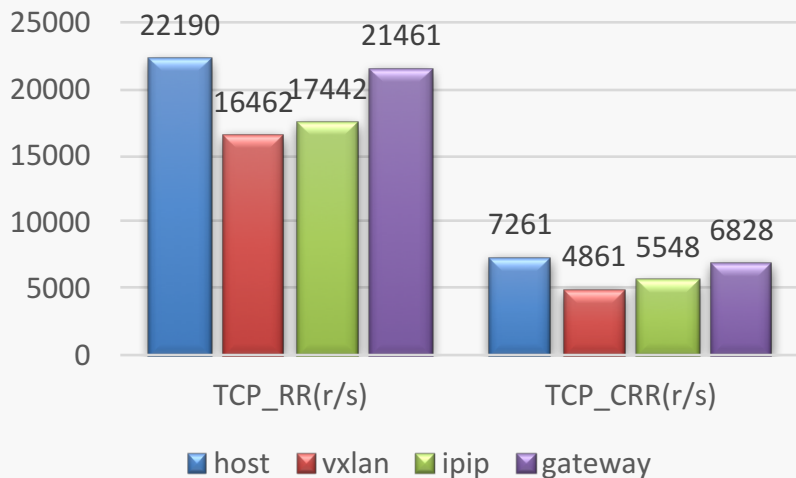
Pod Underlay IP漂移



十年架构 成长之路

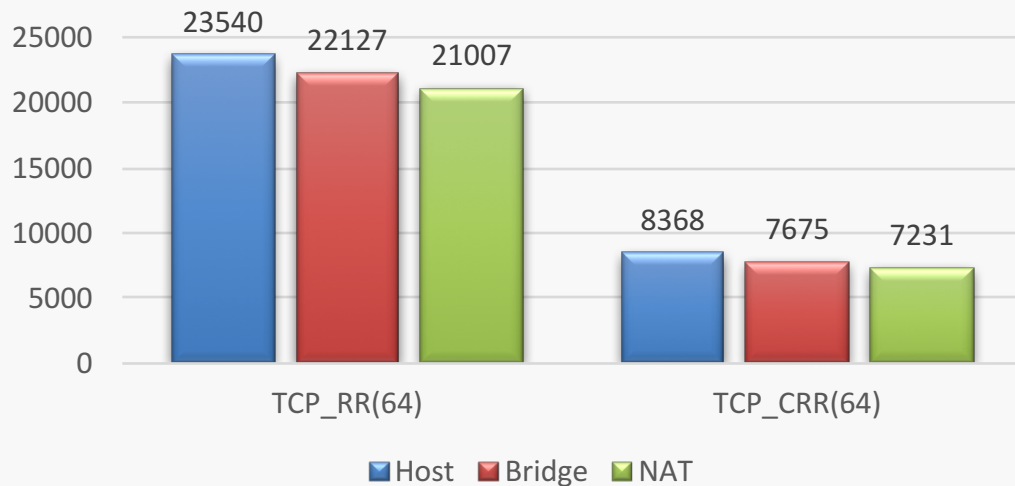


Overlay方案性能



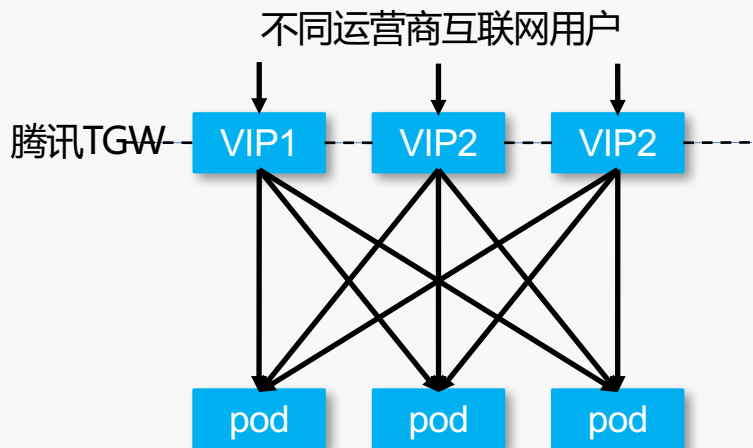
GaiaStack提供Overlay方案是IPIP + Host Gateway 混合方案，性能仅次于calico3层方案
短链接 Vxlan比HOST差33%，IPIP比HOST差23%，Gateway比HOST只差6%
方案被社区合并 <https://github.com/coreos/flannel/pull/842>

Underlay方案性能



Bridge比Host仅差6%

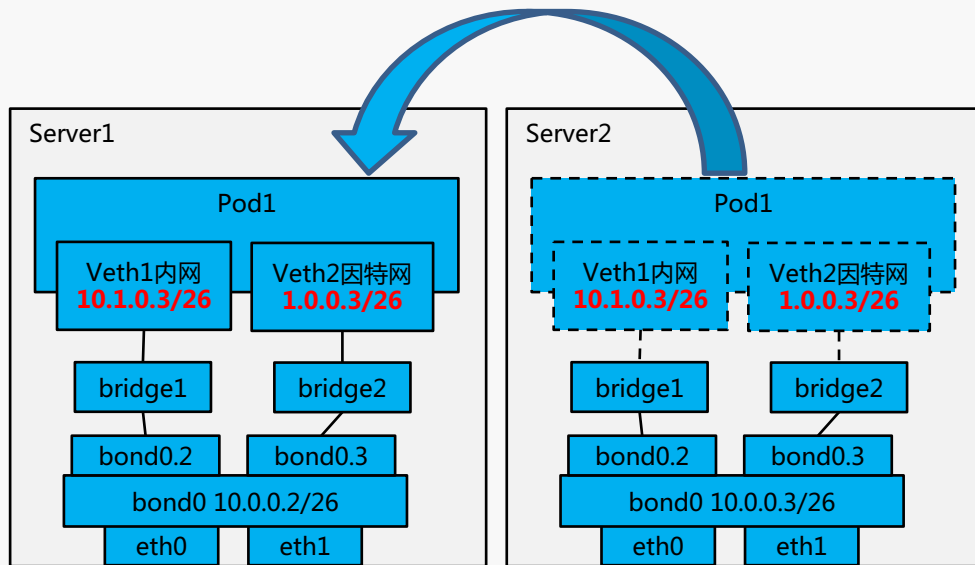
Underlay网络案例



Bridge/SRIOV Underlay网络

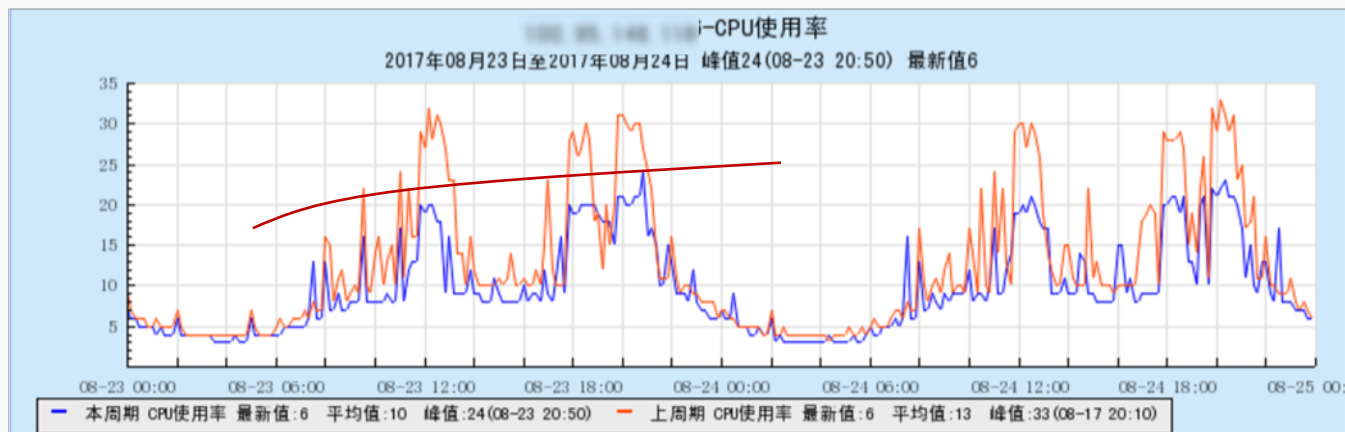
将容器的IP和端口直接绑定到TGW
将TGW的Tunnel网卡创建到容器中

Server2宕机，Pod跨机迁移



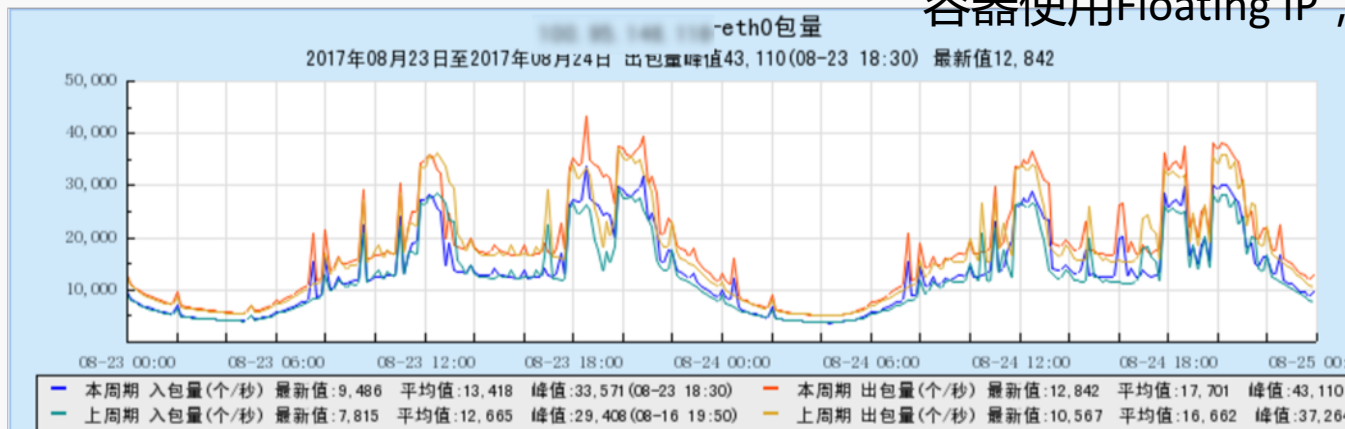
十年架构 成长之路





CPU使用降低1/3

容器使用Floating IP, SRIOV网桥方案



包量+6%

实现kubernetes网络策略

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
```

```
spec:
  podSelector:
    matchLabels:
      role: db
```

```
policyTypes:
  - Ingress
  - Egress
```

```
ingress:
```

```
  - from:
      - ipBlock:
          cidr: 172.17.0.0/16
        except:
          - 172.17.1.0/24
```

```
  - namespaceSelector:
      matchLabels:
        project: myproject
    podSelector:
      matchLabels:
        role: frontend
```

```
  ports:
    - protocol: TCP
      port: 6379
```

```
egress:
  - to:
      - ipBlock:
          cidr: 10.0.0.0/24
```

```
  ports:
    - protocol: TCP
      port: 5978
```

Ingress ipsets

src-ip-xxxx

dst-ip-xxxx

dst-port-xxxx

Egress ipsets

src-ip-xxxx

dst-ip-xxxx

dst-port-xxxx

Hash:IP (零散的IP集合)
Hash:Net (cidr : 10.0.0.0/24)

Iptables ipset extension
-p tcp -m set --match-set ipset-src-r3v4h src \
-m set --match-set ipset-dst-53uf3 dst \
-j ACCEPT

Iptables multiport extension
-p tcp -m multiport --dports 80,81



十年架构 成长之路



BGP协议的使用

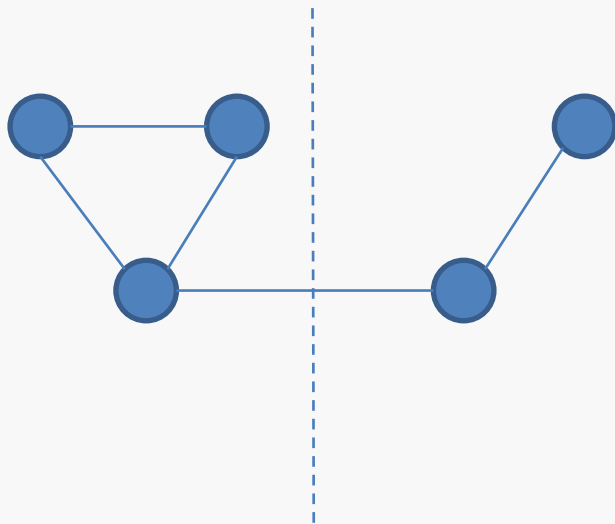
Calico/contiv提供了BGP L3 network的实现，给容器提供一个完全无封包的网络

- 要求交换机开启BGP功能，实质是一种基于BGP协议的数据中心网络架构。

BGP对于容器网络的价值：

- 无封包的网络方案
- 解决不完全连通网络的容器跨主机访问的问题。相比Weave用gossip协议实现P2P网络，标准路由协议的优势：AS Path防环路；Route Reflector/Route Server解决P2P的性能问题

相互隔离网络



Galaxy借助GoBGP实现BGP协议的支持

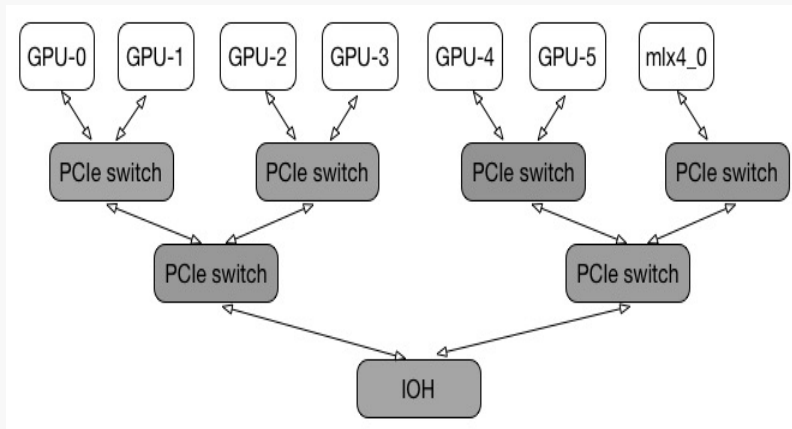


十年架构 成长之路



GPUManager设备插件

- 自研Kube-scheduler调度器插件 + Device Plugin GPU Manager
- 异构集群的精细化管理
 - 管理不同型号的GPU（如：M40、P40等）组成的集群限定不同业务可以使用的卡种类和数量
 - 保证业务能够使到预期的GPU资源（GPU卡数，GPU型号）
- 调度分配优化
 - Drain node调度解决资源碎片问题
 - 多卡需求按GPU架构优先分配距离较近的卡



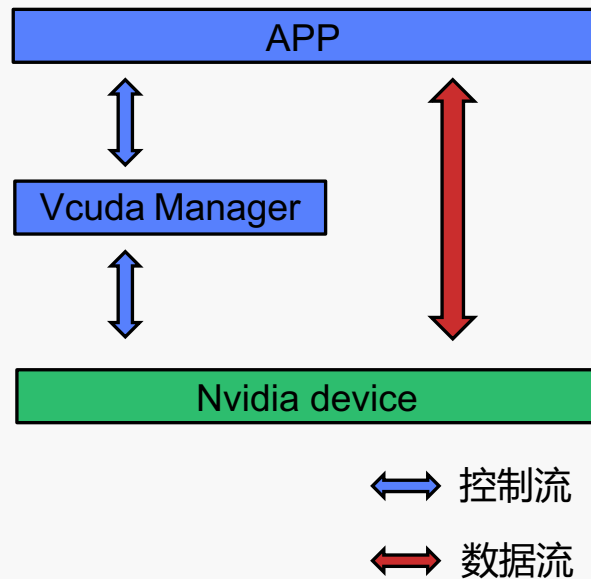
十年架构 成长之路



GPU虚拟化

业内首创容器场景的GPU虚拟化技术

- 多个程序共享同一张GPU卡，提高资源利用率
- 支持GPU卡，GPU内存的虚拟
tencent.com/vcuda-core=0-100、
tencent.com/vcuda-memory=0-100
- GPU卡和内存资源申请通过Manager进行限速，
用户程序的数据流不经过Manager
- 对用户程序零入侵
- 基于kubernetes device plugin实现，对
kubernetes、kernel无入侵



十年架构 成长之路



```

2018-05-24 07:04:10.060819: I tensorflow/core/common_runtime/nccl_allocator.cc:219) Allocator (GPU_0_bfc) ran out of memory 5
5053. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory is
2018-05-24 07:04:10.020418: W tensorflow/core/common_runtime/nccl_allocator.cc:219) Allocator (GPU_0_bfc) ran out of memory 5
5053. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory is
2018-05-24 07:04:10.060819: W tensorflow/core/common_runtime/nccl_allocator.cc:219) Allocator (GPU_0_bfc) ran out of memory 5
5053. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory is
Done warm up
Step images/sec loss
1 images/sec: 43.0 +/- 0.0 (Stdev = 0.4) 7.430
10 images/sec: 43.3 +/- 0.6 (Stdev = 1.3) 7.362
20 images/sec: 43.7 +/- 0.5 (Stdev = 1.3) 7.454
30 images/sec: 43.7 +/- 0.4 (Stdev = 1.4) 7.372
40 images/sec: 43.9 +/- 0.4 (Stdev = 1.3) 7.308
50 images/sec: 43.9 +/- 0.3 (Stdev = 2.0) 7.414
60 images/sec: 43.9 +/- 0.3 (Stdev = 2.1) 7.409
70 images/sec: 43.8 +/- 0.3 (Stdev = 2.1) 7.383
80 images/sec: 43.8 +/- 0.3 (Stdev = 2.1) 7.383

ssh
2018-05-24 07:04:15.549773: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1322) Found device 0 with properties:
name: Tesla M40 24GB major: 5 minor: 2 memoryClockRate(GHz): 1.112
pciBusID: 0000:05:00.0
totalMemory: 23.96018 FreeMemory: 23.78618
2018-05-24 07:04:15.549813: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1322) Adding visible gpu devices: 0
2018-05-24 07:04:15.912347: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1902) Creating TensorFlow device (CudaLocalHe
/device:GPU0 with 23804 MB memory) -> physical GPU device(s): 0, name: Tesla M40 24GB, pci bus id: 0000:05:00.0, compute capab
Running warm up
Done warm up
Step images/sec loss
1 images/sec: 86.7 +/- 0.0 (Stdev = 0.4) 7.452
10 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.366
20 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.463
30 images/sec: 86.7 +/- 0.0 (Stdev = 0.1) 7.377
40 images/sec: 86.7 +/- 0.0 (Stdev = 0.1) 7.382
50 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.411
60 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.442
70 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.435
80 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.388
90 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.442
100 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.398
110 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.369
120 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.343
130 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.369
140 images/sec: 86.6 +/- 0.0 (Stdev = 0.1) 7.483

-# curl -s localhost:1234/graph/jq -r .graph
ROOT@6
|--SOC (eval: 6, pids: [31921 32267], usedMemory: 3228042160, totalMemory: 26529306432, allocatableCores: 0, allocat
ableMemory: 0)
|--P0 (eval: 2, pids: [31921 32267], usedMemory: 3228042160, totalMemory: 102646153216, allocatableCores: 0, all
locatableMemory: 0)
|--P1 (eval: 0, pids: [31921 32267], usedMemory: 3228042160, totalMemory: 51323076608, allocatableCores: 0, all
locatableMemory: 0)
|--GPU0 (pids: [31921], usedMemory: 7781613568, totalMemory: 25661538304, allocatableCores: 50, allocatabl
eMemory: 17680474624)
|--GPU1 (pids: [32267], usedMemory: 24498798592, totalMemory: 25661538304, allocatableCores: 0, allocatabl
eMemory: 0)
|--P2 (eval: 2, pids: [], usedMemory: 0, totalMemory: 51323076608, allocatableCores: 0, allocatableMemory: 0)
|--GPU2 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--GPU3 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--P0 (eval: 4, pids: [], usedMemory: 0, totalMemory: 102646153216, allocatableCores: 0, allocatableMemory: 0)
|--P1 (eval: 2, pids: [], usedMemory: 0, totalMemory: 51323076608, allocatableCores: 0, allocatableMemory: 0)
|--GPU4 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--GPU5 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--P2 (eval: 2, pids: [], usedMemory: 0, totalMemory: 51323076608, allocatableCores: 0, allocatableMemory: 0)
|--GPU6 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--GPU7 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)

```

vcuda-1容器器 申请了了50%的利用率，
7680MB，运行行行在0号GPU，
vcuda-2容器器独占卡，
运行在1号GPU；vcuda-1的训练速率
是平均43.8/s，vcuda-2的训练速度是
平均86.6/s

vcuda-1容器器 申请了了30%的利利用
率，7680MB，运行在0号GPU，
vcuda-2容器器申请了
60%利用率，12800MB，运行在0号
GPU；vcuda-1的训练速率是平均
25.22/s，vcuda-2的训练速度是平均
54.7/s

```

Step images/sec loss
1 images/sec: 50.7 +/- 0.0 (Stdev = 0.8) 7.334
10 images/sec: 57.1 +/- 1.6 (Stdev = 7.4) 7.312
20 images/sec: 54.9 +/- 1.3 (Stdev = 8.3) 7.346
30 images/sec: 50.0 +/- 1.5 (Stdev = 8.4) 7.363
40 images/sec: 50.1 +/- 1.3 (Stdev = 8.2) 7.424
50 images/sec: 49.9 +/- 1.2 (Stdev = 10.4) 7.460
60 images/sec: 48.4 +/- 1.1 (Stdev = 10.0) 7.349
70 images/sec: 48.3 +/- 1.0 (Stdev = 11.0) 7.394
80 images/sec: 48.9 +/- 0.9 (Stdev = 9.3) 7.400
90 images/sec: 49.2 +/- 0.9 (Stdev = 8.3) 7.355
100 images/sec: 49.7 +/- 0.9 (Stdev = 11.0) 7.457
110 images/sec: 50.4 +/- 0.8 (Stdev = 10.7) 7.385
120 images/sec: 50.1 +/- 0.8 (Stdev = 10.4) 7.481
130 images/sec: 50.1 +/- 0.8 (Stdev = 10.4) 7.395
140 images/sec: 50.2 +/- 0.7 (Stdev = 10.3) 7.440
150 images/sec: 50.3 +/- 0.7 (Stdev = 10.3) 7.377
160 images/sec: 50.3 +/- 0.7 (Stdev = 10.2) 7.392
170 images/sec: 50.5 +/- 0.9 (Stdev = 10.1) 7.531
180 images/sec: 50.6 +/- 0.6 (Stdev = 10.3) 7.469
190 images/sec: 50.6 +/- 0.6 (Stdev = 10.3) 7.388
200 images/sec: 50.8 +/- 0.6 (Stdev = 10.0) 7.380
210 images/sec: 50.5 +/- 0.6 (Stdev = 10.3) 7.383
220 images/sec: 50.4 +/- 0.6 (Stdev = 8.3) 7.384
230 images/sec: 50.1 +/- 0.6 (Stdev = 10.1) 7.384
240 images/sec: 50.3 +/- 0.6 (Stdev = 8.2) 7.430
250 images/sec: 50.1 +/- 0.6 (Stdev = 9.3) 7.387
260 images/sec: 50.1 +/- 0.5 (Stdev = 10.0) 7.406

ssh
2018-05-24 08:15:10.211551: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1322) Found device 0 with properties:
name: Tesla P40 24GB major: 5 minor: 2 memoryClockRate(GHz): 1.112
pciBusID: 0000:0A:00.0
totalMemory: 7.58618 FreeMemory: 7.48618
2018-05-24 08:15:10.211551: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1322) Adding visible gpu devices: 0
2018-05-24 08:15:10.681077: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1902) Creating TensorFlow device (CudaLocalHe
/device:GPU0 with 7342 MB memory) -> physical GPU device(s): 0, name: Tesla P40 24GB, pci bus id: 0000:0A:00.0, compute capab
Running warm up
Done warm up
Step images/sec loss
1 images/sec: 21.8 +/- 0.0 (Stdev = 0.4) 7.430
10 images/sec: 21.4 +/- 0.1 (Stdev = 0.3) 7.373
20 images/sec: 21.4 +/- 1.5 (Stdev = 7.3) 7.350
30 images/sec: 22.1 +/- 1.1 (Stdev = 7.3) 7.387
40 images/sec: 22.9 +/- 0.1 (Stdev = 7.3) 7.426
50 images/sec: 22.2 +/- 0.9 (Stdev = 7.3) 7.347
60 images/sec: 22.1 +/- 0.8 (Stdev = 7.3) 7.374
70 images/sec: 21.9 +/- 0.9 (Stdev = 7.3) 7.350
80 images/sec: 22.2 +/- 0.9 (Stdev = 6.5) 7.396
90 images/sec: 21.8 +/- 0.6 (Stdev = 6.5) 7.368
100 images/sec: 22.0 +/- 0.6 (Stdev = 6.4) 7.433

-# curl -s localhost:1234/graph/jq -r .graph
ROOT@7
|--SOC (eval: 6, pids: [25380 25477], usedMemory: 20663500800, totalMemory: 26529306432, allocatableCores: 0, allocat
ableMemory: 0)
|--P0 (eval: 3, pids: [25390 25477], usedMemory: 30663500800, totalMemory: 102646153216, allocatableCores: 0, all
ocatableMemory: 0)
|--P1 (eval: 1, pids: [25380 25477], usedMemory: 20663500800, totalMemory: 51323076608, allocatableCores: 0, all
ocatableMemory: 0)
|--GPU0 (pids: [25390 25477], usedMemory: 20663500800, totalMemory: 25661538304, allocatableCores: 10, all
ocatableMemory: 648010544)
|--GPU1 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--P2 (eval: 2, pids: [], usedMemory: 0, totalMemory: 51323076608, allocatableCores: 0, allocatableMemory: 0)
|--GPU2 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--GPU3 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--P0 (eval: 4, pids: [], usedMemory: 0, totalMemory: 102646153216, allocatableCores: 0, allocatableMemory: 0)
|--P1 (eval: 2, pids: [], usedMemory: 0, totalMemory: 51323076608, allocatableCores: 0, allocatableMemory: 0)
|--GPU4 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--GPU5 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--P2 (eval: 2, pids: [], usedMemory: 0, totalMemory: 51323076608, allocatableCores: 0, allocatableMemory: 0)
|--GPU6 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)
|--GPU7 (pids: [], usedMemory: 0, totalMemory: 25661538304, allocatableCores: 100, allocatableMemory: 2566
1538304)

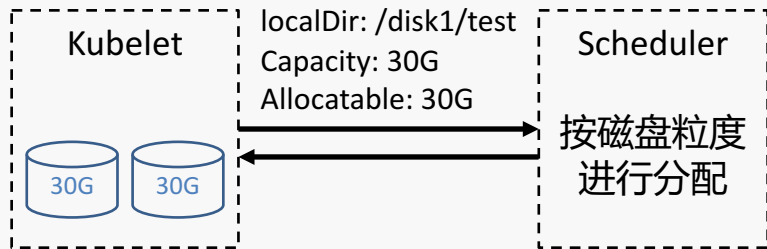
```

磁盘空间管理

GaiaStack修改了调度器和Kubelet以支持本地磁盘管理，相比Kubernetes emptyDir优势：

- 支持多块盘
- 支持弹性管理，允许临时超出
 - 单个Pod超出limit大小后若机器磁盘仍有余量可以暂时使用

目前耦合在Kubernetes代码中，后续计划用scheduler plugin和CSI接口进行优化



十年架构 成长之路



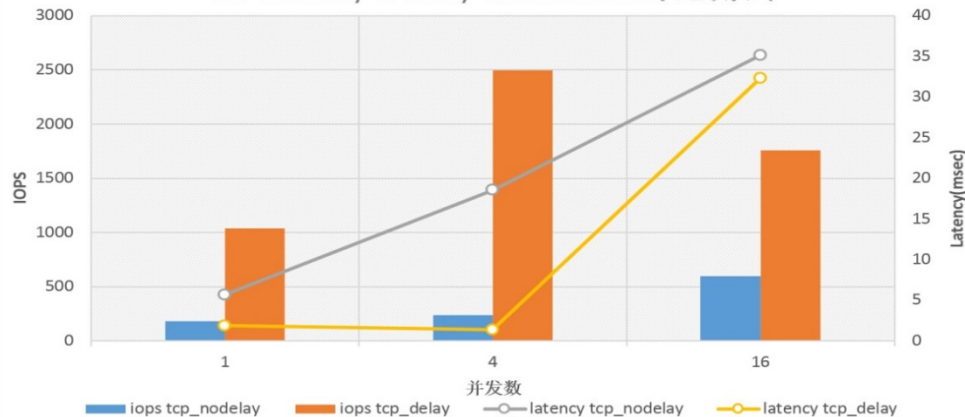
云盘管理

- 支持CephRBD、CephFS，腾讯公有云CBS、CFS等
- 外部controller支持CephFS dynamic provisioning
- Kubelet支持Volume在线扩缩容，
<https://github.com/kubernetes/kubernetes/pull/62460>

一名Ceph官方组织成员

- 提升mds对大量文件的目录处理速度**6~10倍**
- 提高了mds主备切换的速度
- Cephfs内核模块稳定性改进，bug fix
- 支持quota
- 支持Jewel, 支持keyring挂载权限

TCP nodelay & delay 4k randread对比测试

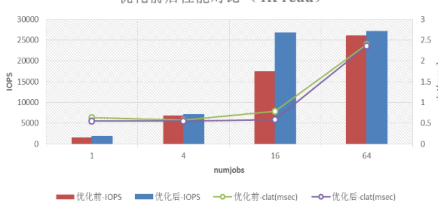


Tethys块设备(RBD) 优化前后对比

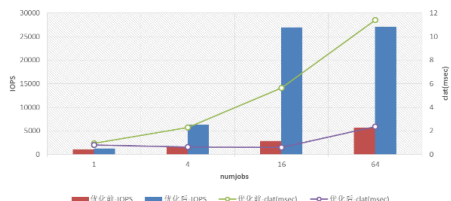
1. TS5 * 4台, 1 Monitor, 44 OSD (MON与OSD混布, 每台机器11个OSD)
2. 千兆网卡, 磁盘大小: 2T, 系统盘为2T SATA,
3. 操作系统: Tencent linux release 2.0 (Final)
4. 开启writcache, 关闭swap
5. 内存64G, CPU: Intel(R) Xeon(R) CPU E5640 @ 2.67GHz, 16核
6. I/O主要参数: direct=0 sync=1 ioengine=sync
7. 副本数 2

注: numjobs 为并发数 4K 为blocksize

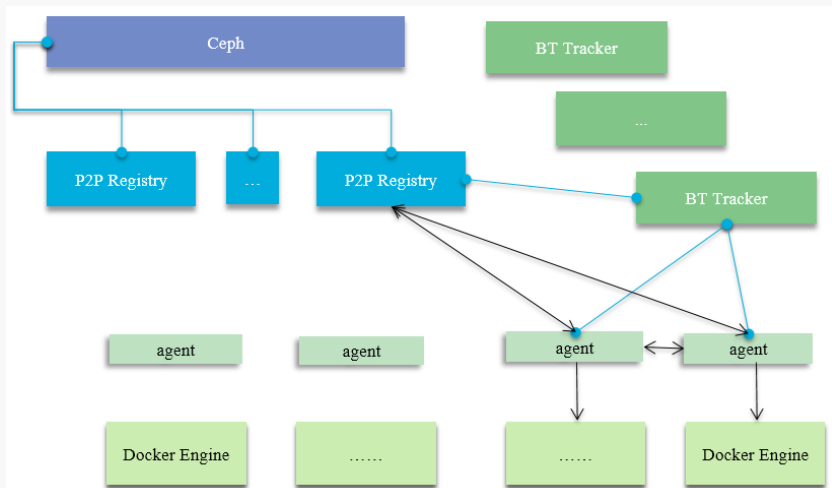
优化前后性能对比 (4K-read)



优化前后性能对比 (4K-randread)



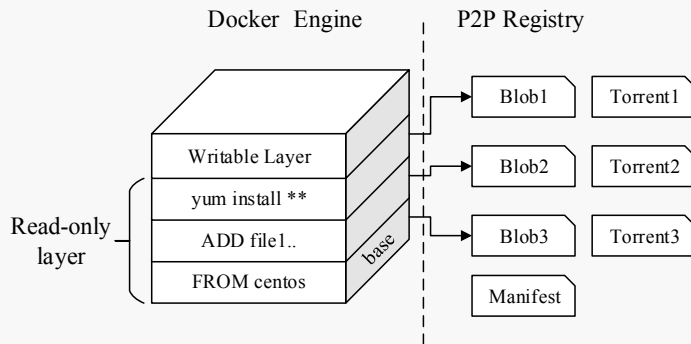
P2P Docker registry



- **P2P Registry** : 镜像仓库, 种子生成, 种子下载, 文件初始提供者
- **Agent** : Docker透明代理, 下载任务的主要功能组件
- **BT Tracker** : P2P下载资源查询定位组件

主要设计思想：

- 在镜像下载过程中, 引入BT协议
- 在Blob上传时, 对Blob生成种子
- 每层分别做种
- 在下载镜像的Blob时, 先下载种子, 再通过种子文件下载数据



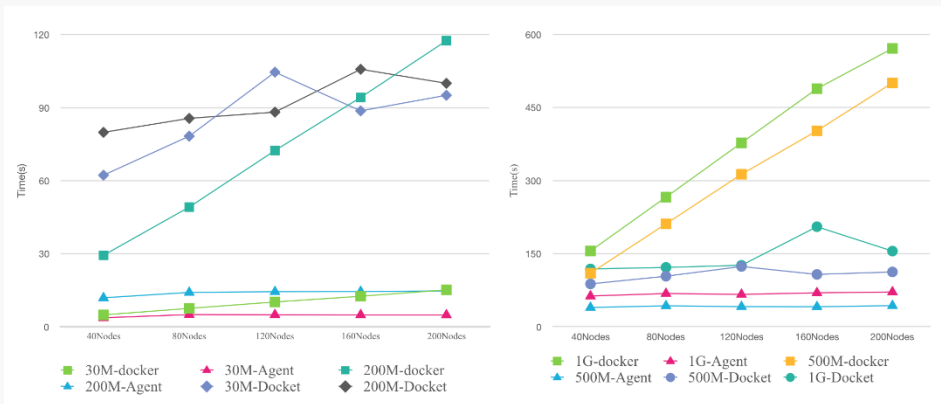
十年架构 成长之路



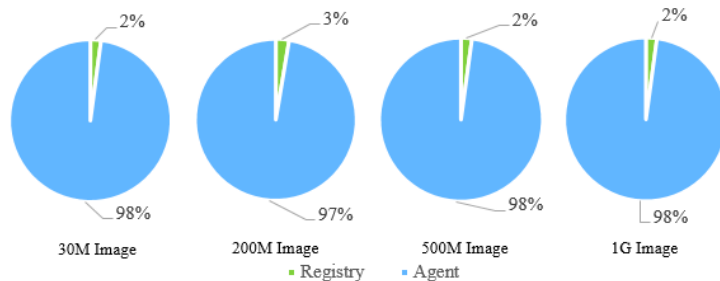
P2P Docker registry

- 每层分别做种，下载速度更快
- 优化流量调度算法，节省registry流量
- 代理方式，对Docker Daemon零入侵

Docker、Docket、Gaiastack P2P Agent 下载镜像对比



Registry与P2P Agent 流量占比对比



十年架构 成长之路



热升级问题

1.4升级1.9版本

- Pod Hash发生变化
- Container名称发生变化，点分隔改为了下划线分隔
- 容器标签发生变化
 - pause容器的标签io.kubernetes.container.name=POD改为io.kubernetes.docker.type=podsandbox
 - io.kubernetes.container.restartCount改为annotation.io.kubernetes.container.restartCoun
- Cgroup目录结构发生变化，新增Pod层级



十年架构 成长之路





THANKS