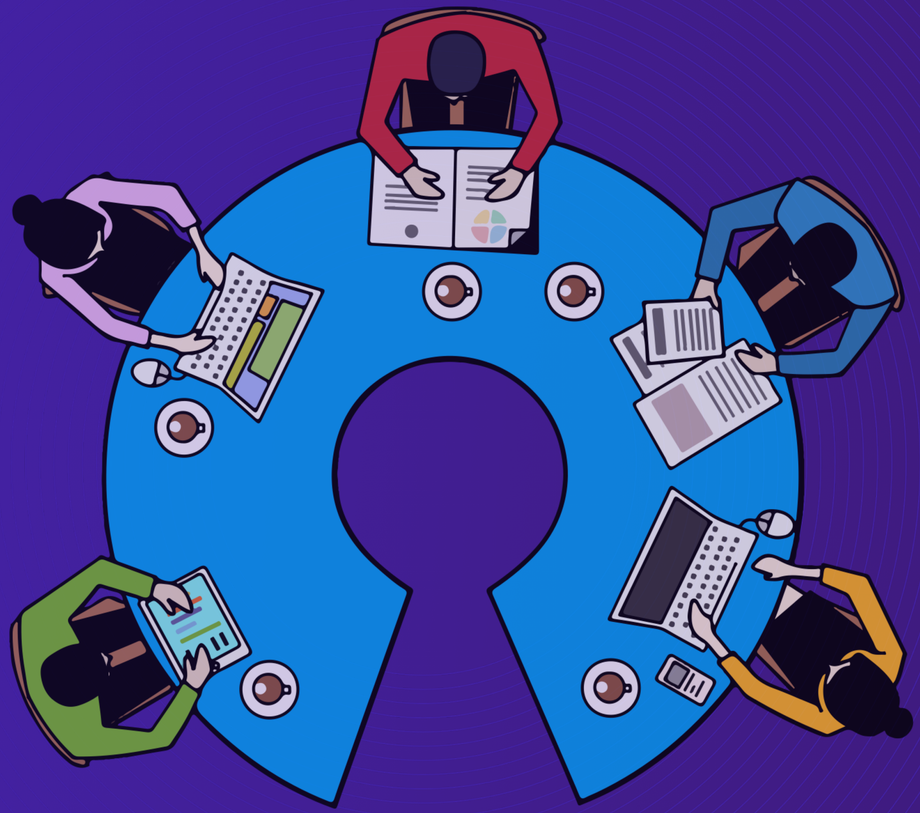
木兰开源社区
MULAN OPEN SOURCE

移动云容器调度云原生架构演进



董江 / 中国移动云能力中心 高级系统架构专家

2022木兰峰会

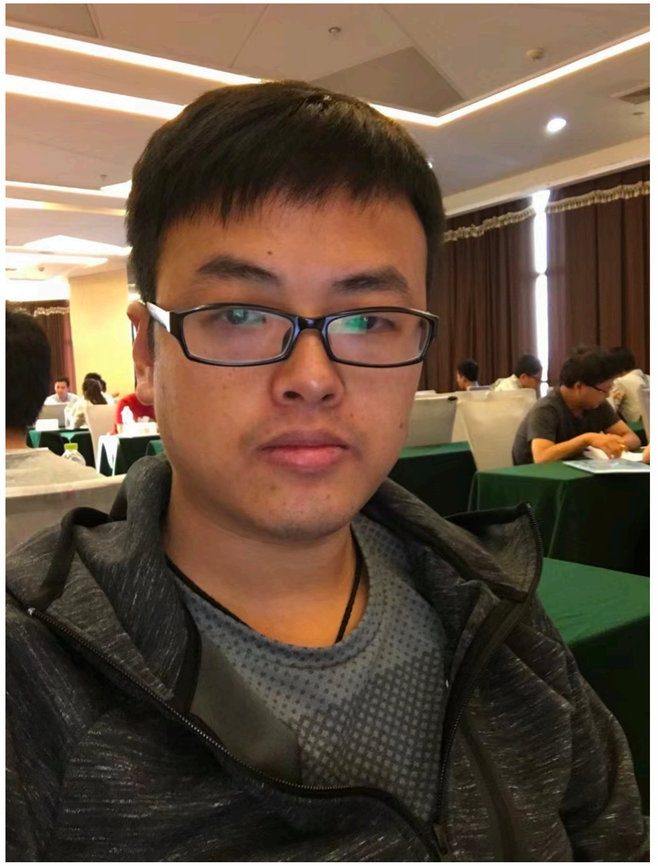
自我介绍

董江 中国移动云能力中心 高级系统架构专家

容器技术布道者及实践者，Apache ServiceComb微服务框架核心开发者， 云原生社区成员、Knative Sig核心成员、华为云MVP

KubeService Stack社区发起者 https://stack.kubeservice.cn/

曾就职于 百度、阿里、滴滴、华为、贝壳



木兰开源社区
MULAN OPEN SOURCE

CONTENTS

木兰开源社区
MULAN OPEN SOURCE

移动云容器调度现状

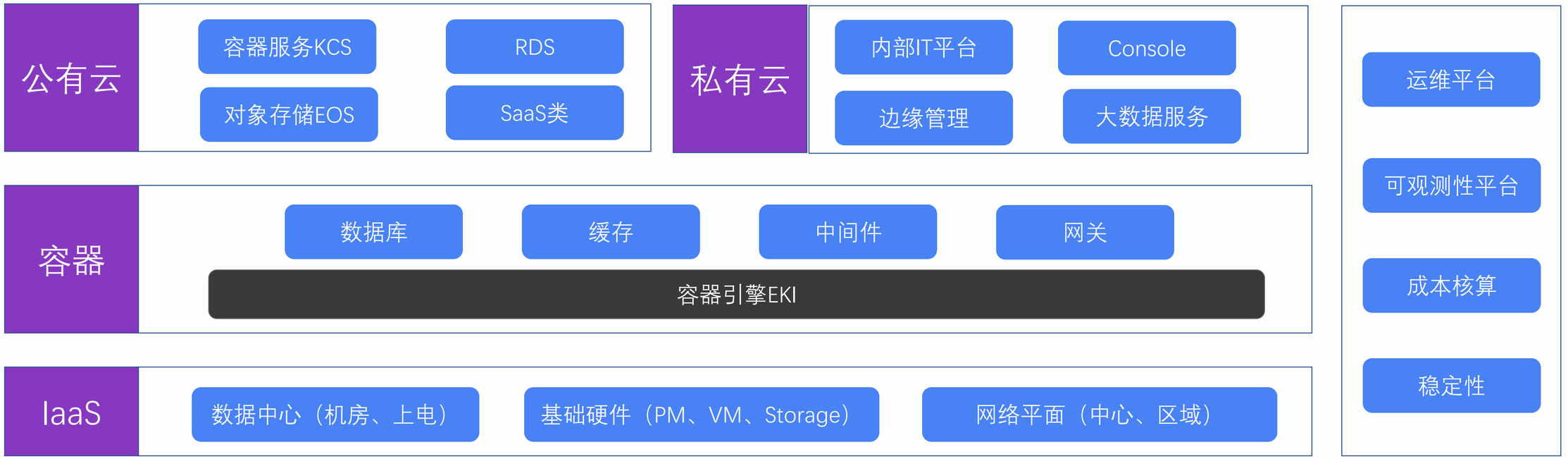
2022木兰峰会

移动云容器现状

集群数
280+

在线服务数
8000+

实例数
10w+



移动云『过去』面临的问题

1.『容器化』程度较高，但是**『云原生化』程度低**

2. 业务工程能力参差不齐，集群**『难以升级』或『不能升级』**

3. 前期集群规划不合理，没有**『上云管控限制』**

4. 业务无特地『上云、云原生改造』，**没有解决上云过程问题能力**

5. 按照功能划分集群，无**『多集群管理能力』**

思考

1. 如何支持『存量』业务升级云原生架构，充分释放云原生的能力

2. 如何提升资源效能，降低业务上云难度

3. 如何让容器引擎Kubernetes作为基础设施，提供通用能力，支撑各种内部业务

- ***合理多租管理：*** CaaS（Clusters as a Service） + NaaS （Nmaespace as a Service）
- ***高效服务管理：*** 开源和自研的各种通用插件能力，一键完成常见运维操作，提 升业务服务管理的效率，降低业务在服务上云成本
- ***资源管理策略：*** 资源混部、资源隔离、资源精细化调度，释放云原生的能力

木兰开源社区
MULAN OPEN SOURCE

云原生上的实践

2022木兰峰会

准入Controller/Webhook集合

多租户模型

Webhook forbidden：
- **ClusterRole**
- **ClusterRoleBinding**
- **Privileged**
- **readOnlyRootFS**
- **daemonset**

Controller：
- **Auto-Add-NetworkPolicy**

Webhook forbidden：
- **HostPIC**
- **HostPID**
- **HostNetwork**
- **Default DnsPolicy**

Controller：
- **CustomSecret**

Webhook forbidden：
- **Auto-add-serviceAccount**
- **NodePort**
- **Auto-add-imagepullsecrets**

Controller：
- **AccountQuotaSet**
- **Rootfs 限制**

访问隔离

●使用RBAC实现指定资源在命名空间粒度的细化授权；

●禁止cluster级别RBAC

●禁止Pod特权模式

●必须readOnlyRootFS模式

网络隔离

●默认添加NetworkPolicy拒绝所有Namespace直接网络互通

●禁止HostPIC、HostPID、hostNetwork、Default dnsPolicy等

资源隔离

●通过Namespace 亲和Node，将部署在其中的所有Pod默认亲和到相关Node节点上；

●Namespace层级限制每个rootfs使用大小，默认注入到Pod中；

安全隔离

●Pod网络隔离：Pod Security Policy限制容器的行为；

●镜像使用隔离：每个Pod 和Deployment添加ServiceAccount; 并且在其中添加各自imagePullSecrets

●禁止使用 Default ServiceAccount、禁止NodePort

敏感RSA

●升级secret加密默认RSA加密

●默认：imagePullSecrets、TLS、dockerconfigjson等以RSA加密

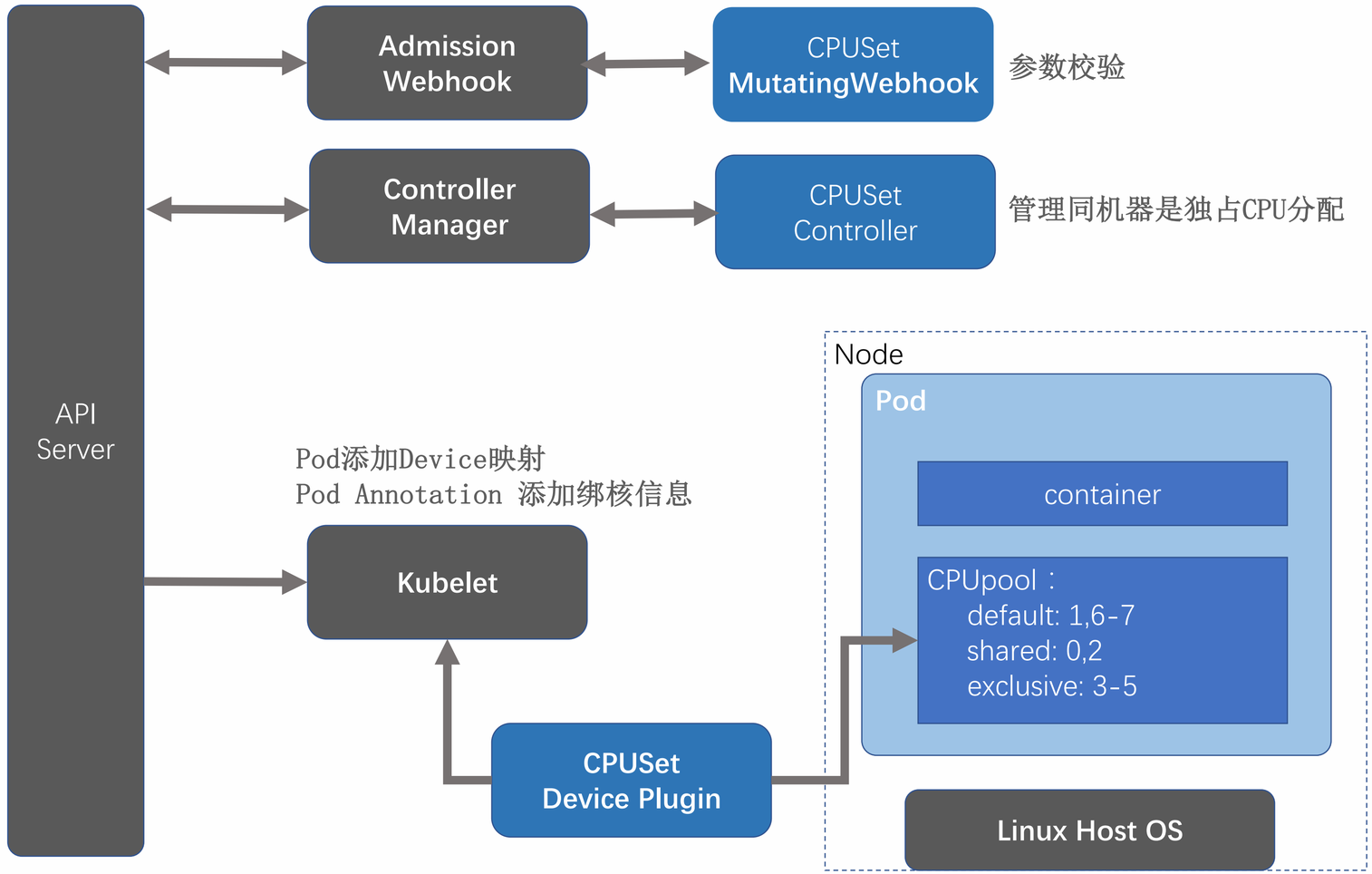
资源限制

●自动化判断租户使用资源是否超过集群限制，如果超限，禁止使用集群；

●集群特别关键字禁止使用

●集群特别类型资源禁止扩namespace处理

容器CPU - 精细化管理CPUSet




```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cpuset-configmap
  namespace: kube-system
data:
  cpuset-config.yaml: |
    pools:
      exclusive:
        cpus : "3-5"
      shared:
        cpus : "0,2"
      default:
        cpus: "1,6-7"
    nodeSelector:
      name: xxxx
```

```
---
apiVersion: v1
kind: Pod
metadata:
  name: cpuset-exclusive
spec:
  containers:
...
    resources:
      requests:
        memory: 256Mi
        ecloud.cn/exclusive-cpu: "1"
      limits:
        memory: 256Mi
        ecloud.cn/exclusive-cpu: "1"
```

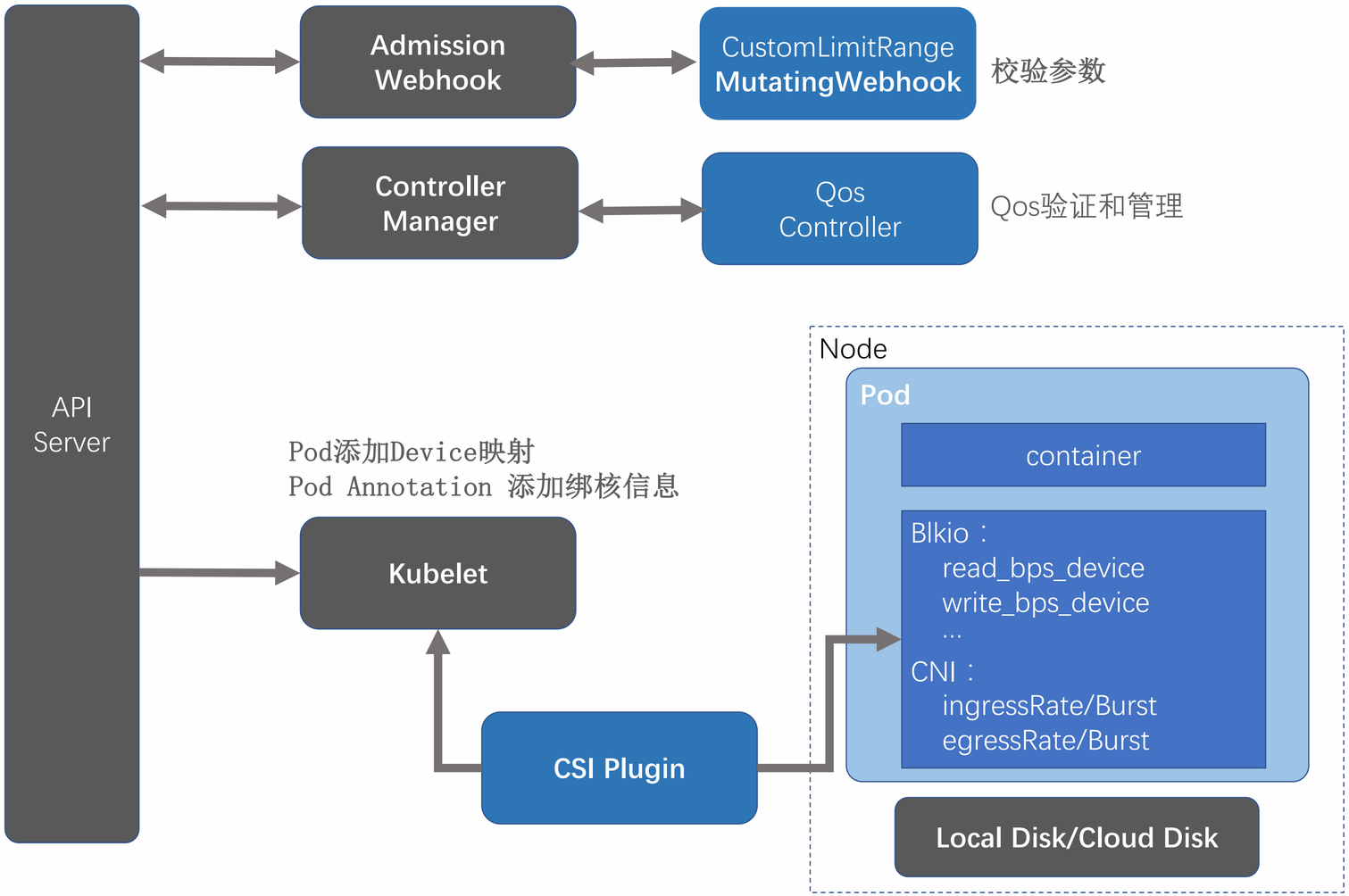
优点：

- **对CPU进一步池化，实现部分绑核能力；**
- **对现有的部署不影响**
- **特定绑核使用，使大颗粒高性能服务可上云；**

缺点：

- **default 与 shared共享问题**

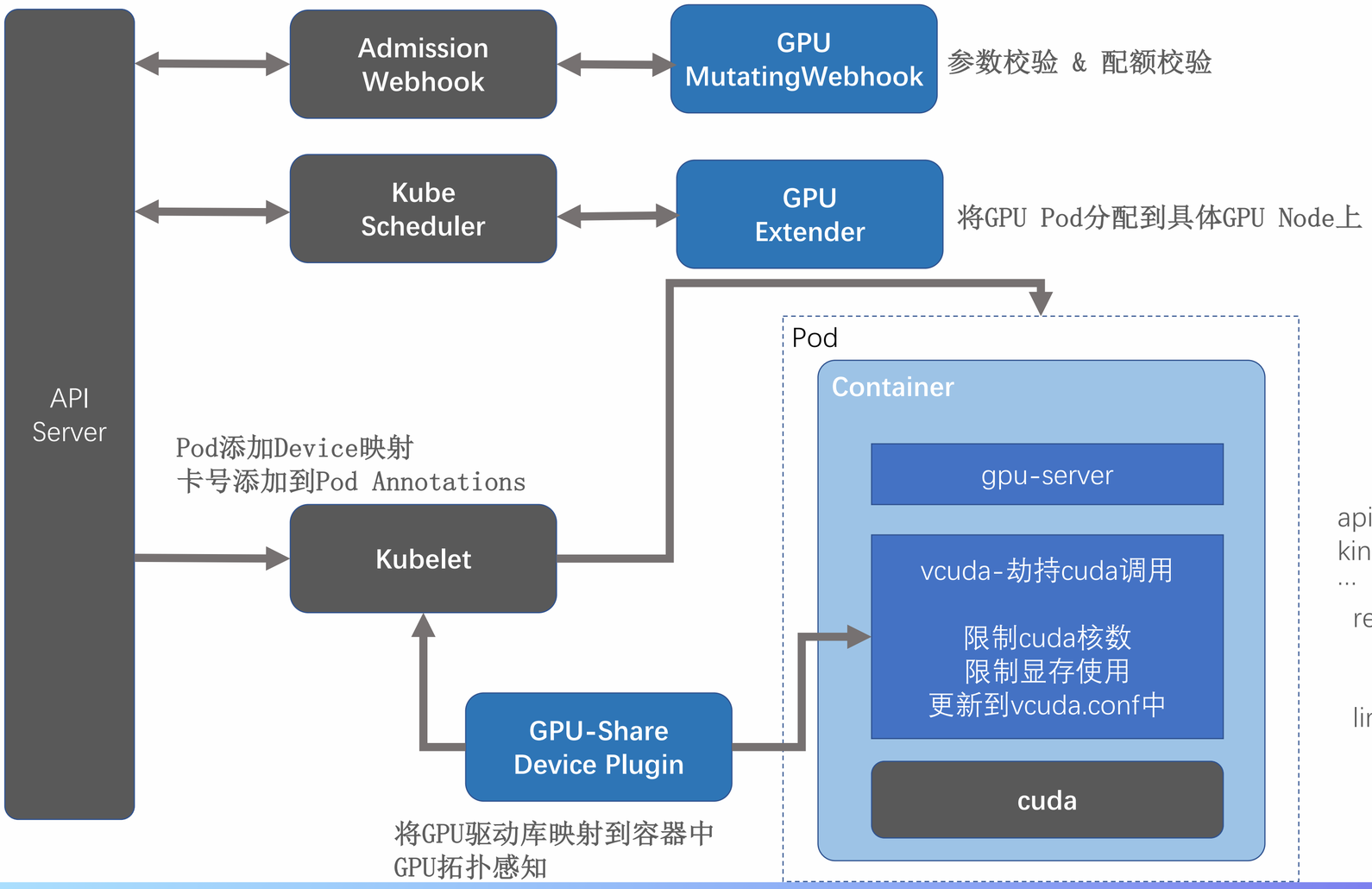
容器QoS-网络/磁盘流量控制实现




```
apiVersion: custom.xxx.com/v1
kind: CustomLimitRange
metadata:
  name: test-rangelimit
spec:
  limitrange:
    type: pod      # 对pod类型限制
    max:           # max和min是限制的上下线，如果pod自定义的值不在其中，
ValidatingAdmissionWebhook校验报错
      ingress-bandwidth: "1G"
      egress-bandwidth: "1G"
    min:
      ingress-bandwidth: "10M"
      egress-bandwidth: "10M"
    default:       # 定义了default，如果pod annotation为空，
MutatingAdmissionWebhook自动注入此数据；未定义default，不作强注入操作
      ingress-bandwidth: "128M"
      egress-bandwidth: "128M"

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
    name: csi-local
provisioner: local.csi.cmsss.com
parameters:
    volumeType: LVM
    vgName: volumegp
    fsType: ext4
    lvmType: "striping"
    readBPS: 1M  # read 此类卷的带宽是1MB/s
    writeBPS: 100K # 写 此类卷的带宽是100KB/s
    readIOPS: 2000  # read 此类卷的tps是2000
    writeIOPS: 1000  # write 此类卷的tps是1000
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
```

容器GPU - GPU Share Scheduler实现



优点：

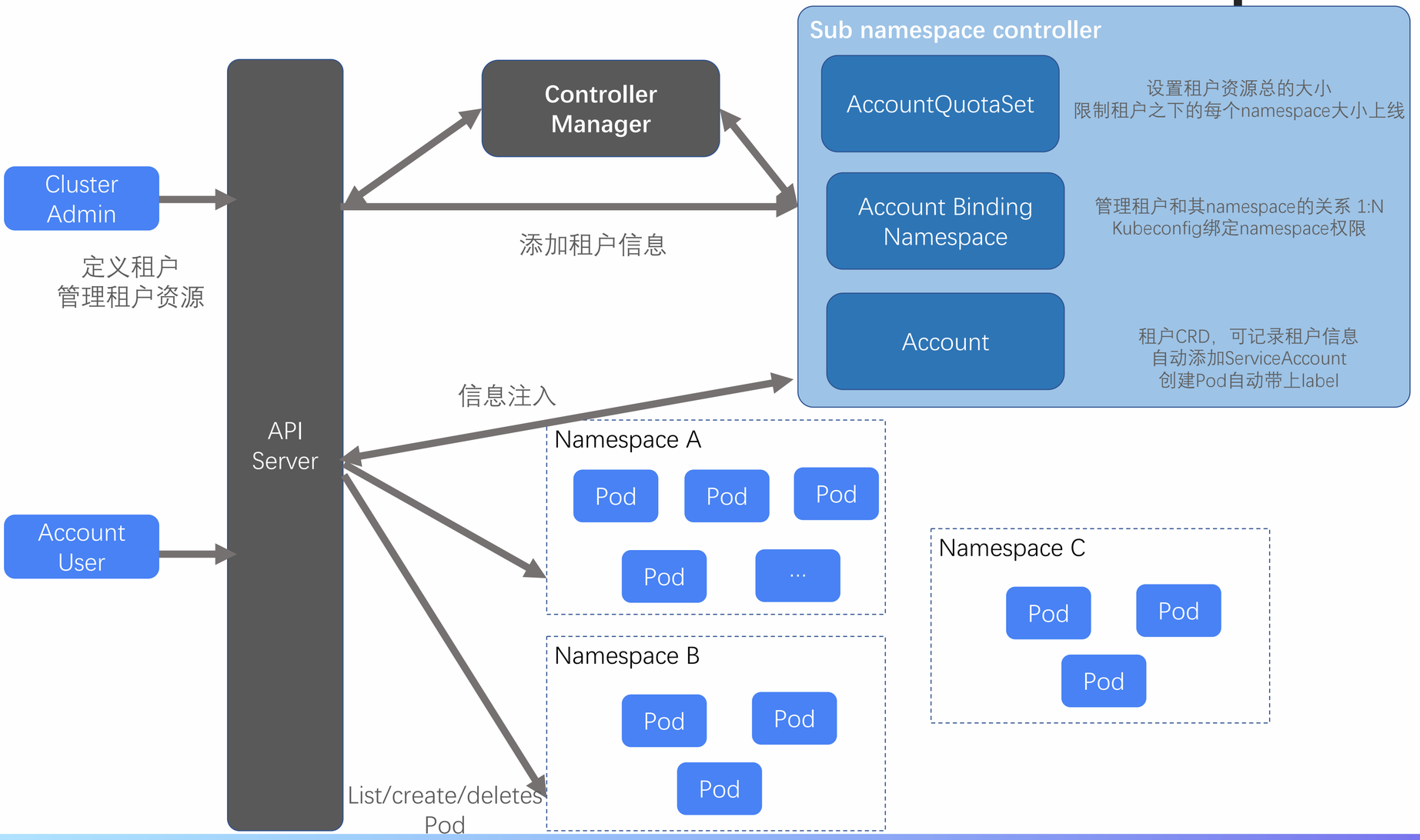
- 支持GPU share，提高GPU资源利用率
- 支持同一张卡上容器间GPU和显存的使用隔离
- 基于拓扑感知，提供最优的调度策略
- 对用户程序无侵入，用户无感

缺点：

- 驱动和加速库的兼容性依赖于厂商（<=cuda 11.5.1）
- 存在约5%的性能损耗

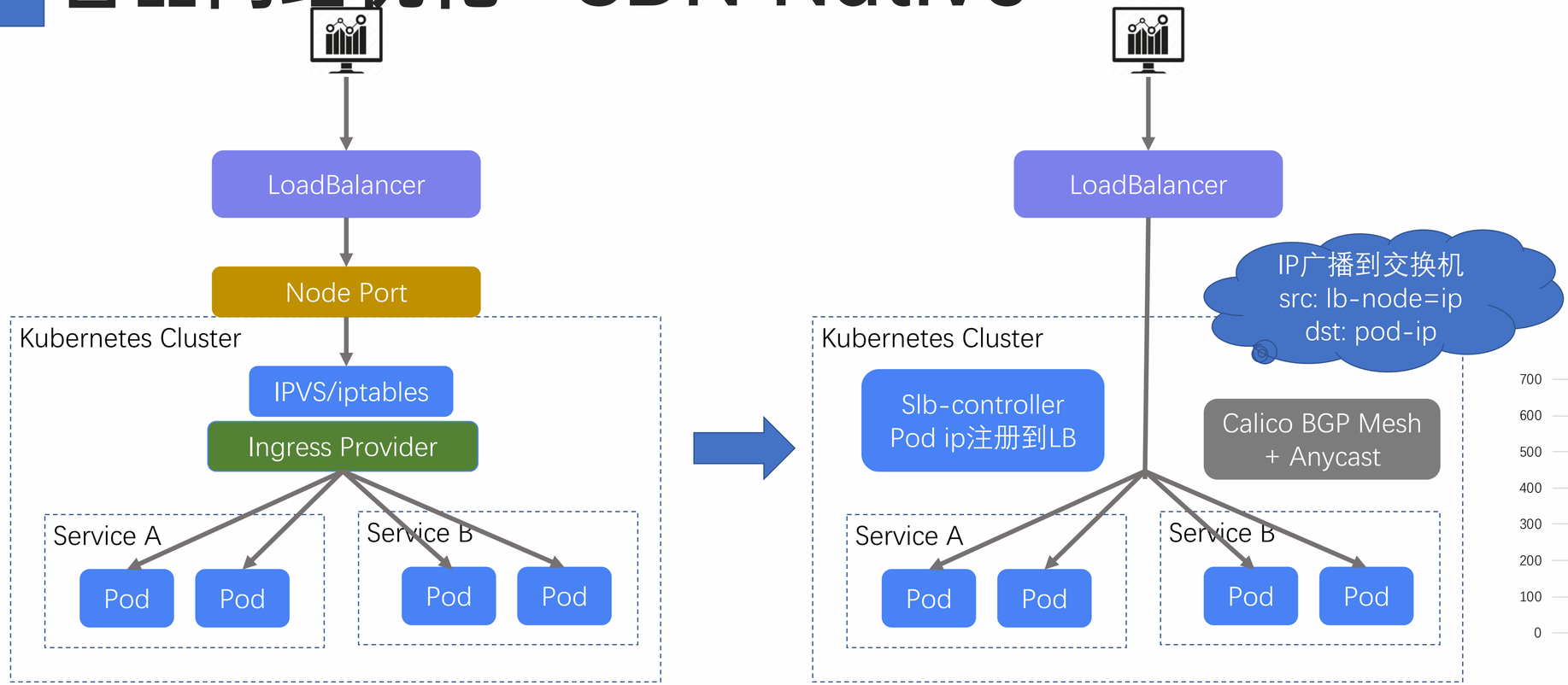
```
apiVersion: v1
kind: Pod
...
  requests:
    ecloud.cn/gpu-core: 2000m
    ecloud.cn/gpu-memory:100Mi
  limits:
    ecloud.cn/gpu-core: 2000m
    ecloud.cn/gpu-memory:100Mi
```

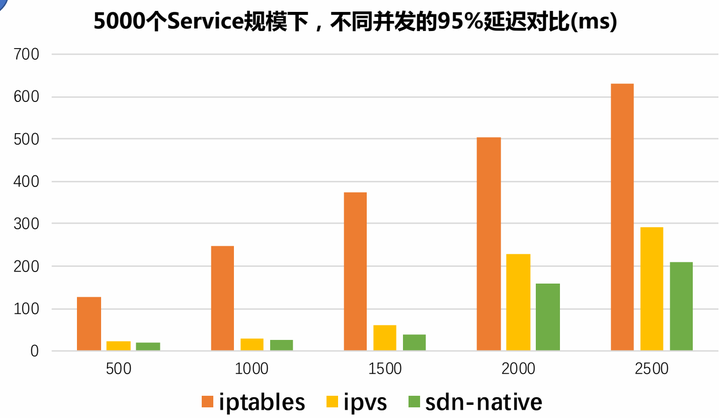
容器多租户 – 多租户管理sub-namespace controller




```
apiVersion: subnamespace.xxx.com/v1
kind: Account
metadata:
  name: accountA
  namespace: kube-system
---
apiVersion: subnamespace.xxx.com/v1
kind: AccountQuotaSet
metadata:
  name: AccountAQuotaSet
  namespace: kube-system
spec:
 selector:
    matchLabels:
       name: accountA
  hard :
    cpu: 20
    memory: 10Gi
---
apiVersion: subnamespace.xxx.com/v1
kind: AccountBindingNamespace
metadata:
  name: AccountABindingNamespace
  amespace: kube-system
spec:
 selector:
    matchLabels:
       account: accountA
  namespaces :
    - Namespace A
    - Namespace B
    - Namespace C
```

容器网络优化 - SDN-Native



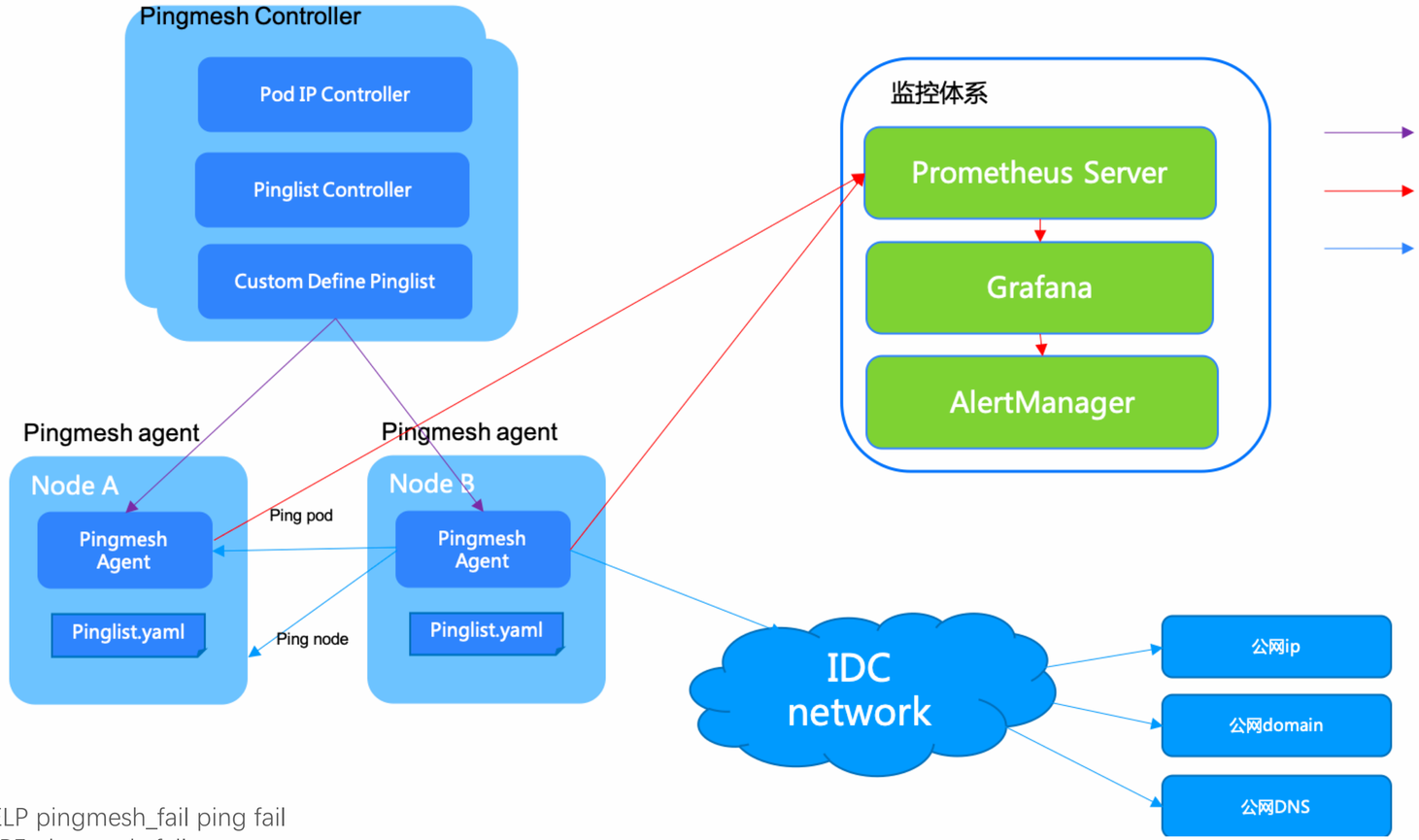


优势

- 省去kube-proxy、Overlay，更短的链路带来低延迟。
- SLB集群基于DPDK版本的IPVS（DPVS）实现，性能是内核IPVS的4-10倍。并且不依赖Conntrack，无SNAT带来的冲突问题。
- 链路简单，容器网络无Overlay，扁平化的网络，易于网络的维护与排查。

缺点： 与网络强耦合

容器网络监控 – PingMesh实现

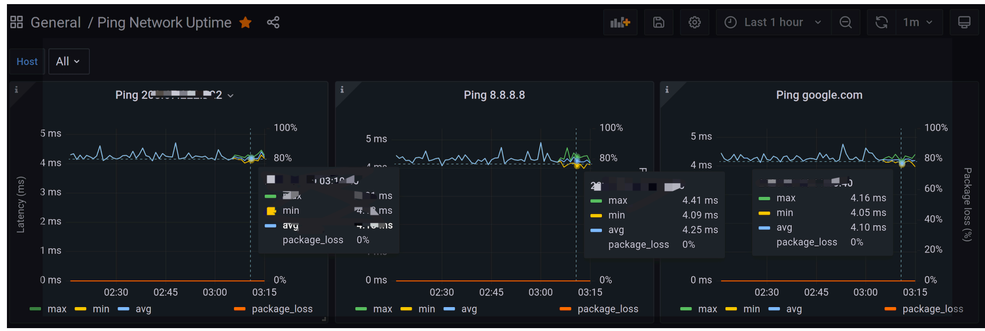



```
setting:
  # the maximum amount of concurrent to ping, uint
  concurrent_limit: 20
  # interval to exec ping in seconds, float
  interval: 60.0
  # The maximum delay time to ping in milliseconds, float
  delay: 200
  # ping timeout in seconds, float
  timeout: 2.0
  # send ip addr
  source_ip_addr: 0.0.0.0
  # send ip protocal
  ip_protocol: ip6

mesh:
  add-ping-public:
    name: ping-public-demo
    type: OtherIP
    ips :
      - 127.0.0.1
      - 8.8.8.8
      - www.baidu.com
      - docker.io
      - kubernetes.default.svc.cluster.local
```

```
# HELP pingmesh_fail ping fail
# TYPE pingmesh_fail gauge
pingmesh_fail{target="8.8.8.8",tor="ping-public-demo"} 1

# HELP pingmesh_duration_milliseconds duration of ping rtt
# TYPE pingmesh_duration_milliseconds gauge
pingmesh_duration_milliseconds{target="docker.io",tor="ping-public-demo"} 245
```



木兰开源社区
MULAN OPEN SOURCE

开源规划

开源规划

社区官网： https://stack.kubeservice.cn/
Github地址： https://github.com/kubeservice-stack

全功能集生态

KubeserviceStack 提供了一整套完整DevOps解决方案，覆盖服务注册中心、配置中心、业务监控、流量监控、平台监控、事件中心和操作记录等全方位的可观测性能力，帮助用户快速构建可观测性能力。

支持企业级应用服务使用

为企业级服务提供7x24保障

KubeServiceStack

KubeService Stack = Kubernetes + Custom Service

构建 Kubernetes/KubeEdge 企业级周边生态体系

可观测性平台、CNI插件、CSI插件、Kubernetes Controller/Webhook、自定义调度器等

Kubernetes无侵入plugin/addon

构建 CNI插件、CSI插件、CRI插件 以及 Kubernetes Webhook/Controller 实现对K8s无侵入扩展

兼容 Kubernetes 多版本

整个生态支持 Kubernetes 社区版本，云服务版本. 包括阿里云ACK、华为云CCE、腾讯云TKE、移动云KCS、百度云CCE

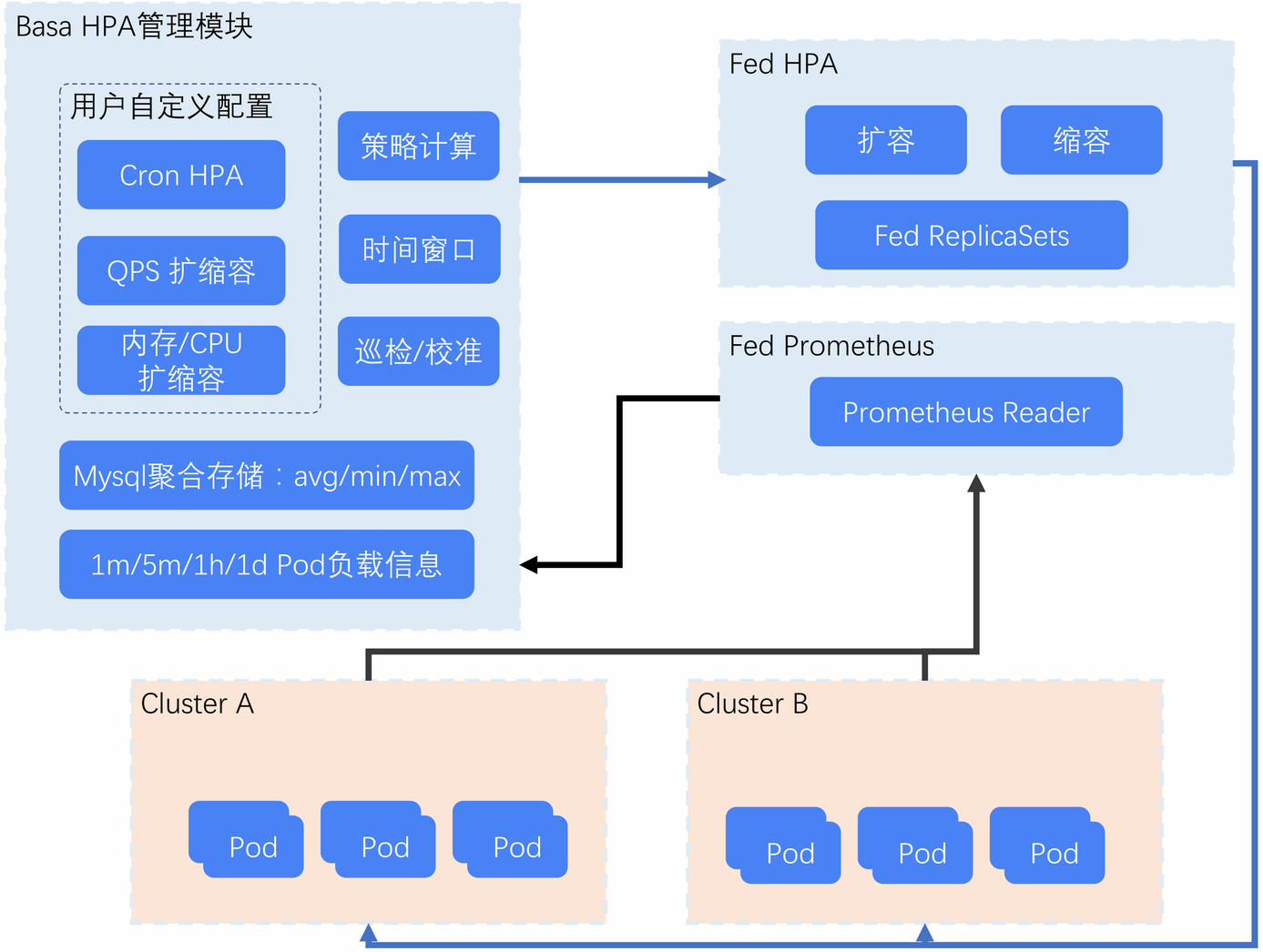
多集群管理平台 - Basa

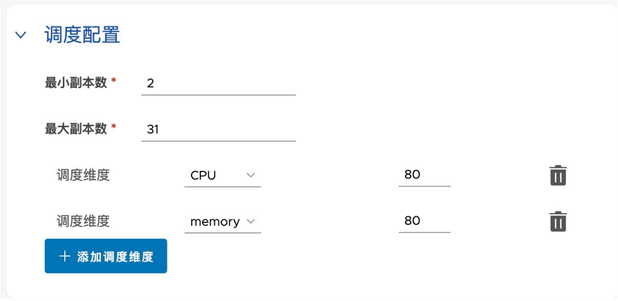
社区官网： https://stack.kubeservice.cn/
Github地址： https://github.com/kubeservice-stack/basa

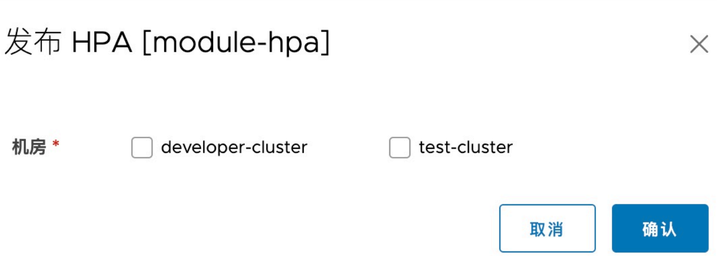
- 融入 **部门、项目、模块**概念
- 纳管多套集群：
  元数据自管理，可快速纳管全部集群
- 权限细力度控制：
  按人员角色设计不同权限；**运维视角 与 研发视角隔离**
- FaaS能力：
  支持OpenFaaS 无差别部署
- 成本核算：
  对 **部门、团队、项目级别** 资源可做到 成本分摊
- 上线效率：
  即支持webssh 通过命令行部署；也支持页面引导式部署
- 报表：
  数字化报表，支持部门、项目、模块的呈现
- 多集群、多租户能力：
  动态协同总发布实例数-Fed HPA

跨集群弹性Fed HPA

社区官网： https://stack.kubeservice.cn/
Github地址： https://github.com/kubeservice-stack/basa







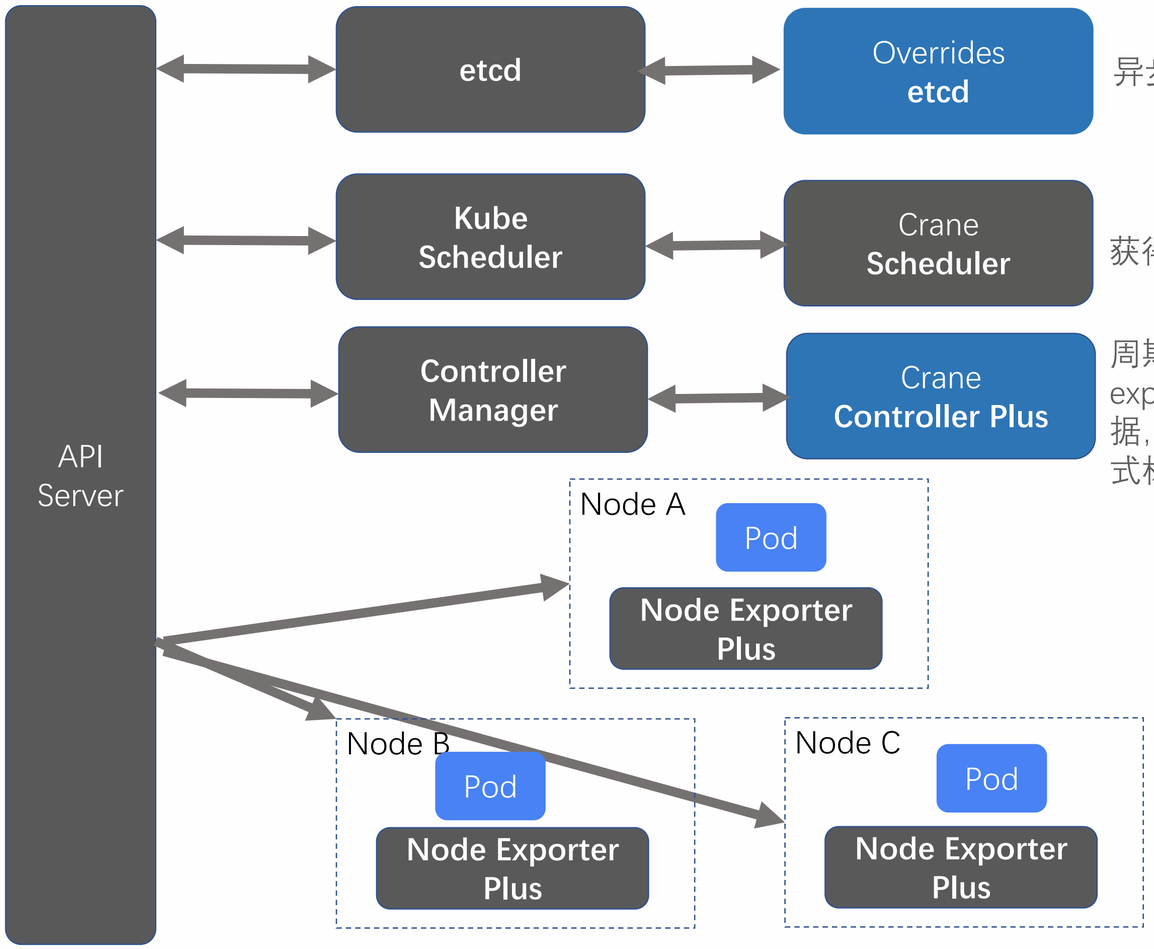
目前支持多种探测策略，包括

- cpu和mem
- cron定时
- 基于业务自定义指标，比如qps
- 预测 HPA

基本上满足在线服务弹性需求。

节点真实负载情况调度

社区官网： https://stack.kubeservice.cn/
Github地址： https://github.com/kubeservice-stack/crane-scheduler



动态上报Pod 和 Node 真实负载

•Crane-scheduler-plus:

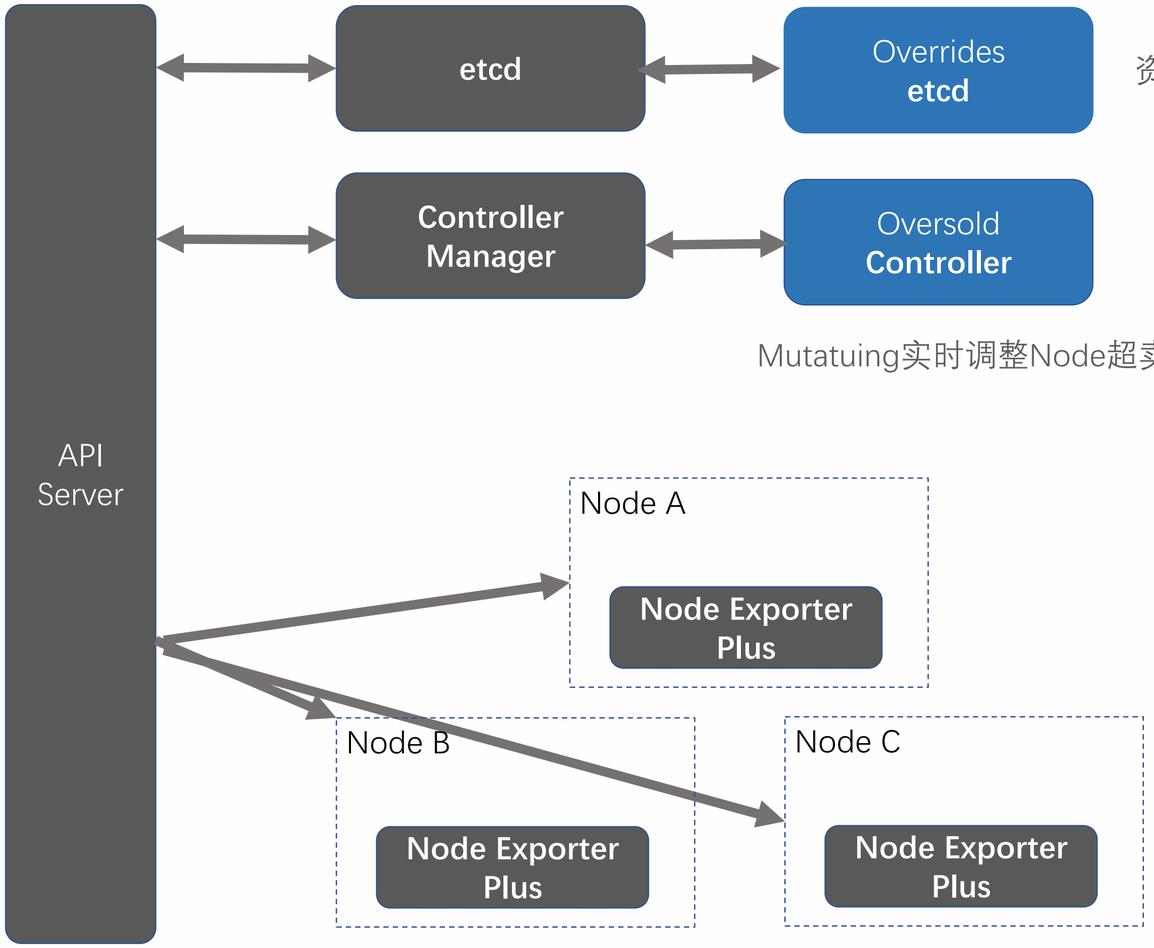
① 如果通过endpoints 来访问的node节点过多， 在一个周期（比如15s）处理不完， 会保证本次处理完成后，在执行下一次， 优先一个轮回完成
② metrics中是通过Promethrus Gauge, 在每个周期中单独计算, 保证即使网络异常丢失 pull 请求，也可以通过下一次请求进行补足

•Node-Exporter-plus:

① 设计 BaseCollector 实现 prometheus.GaugeValue 数据内存计算和收集

动态资源超卖 & 混部

社区官网： https://stack.kubeservice.cn/
Github地址： https://github.com/kubeservice-stack



•资源画像：

基于历史 Node 节点监控数据，预测未来一个周期 的资源使用量，避免热点问题

•超卖控制器：

通过资源画像上报的数据，来动态调整节点可超 卖的比例;Mutating 根据超卖比例来计算节点可分配资源

•巡检：

对服务稳定性以及资源使用情况告警自动化处理，解决 一些热点问题

多层级调度

社区官网： https://stack.kubeservice.cn/
Github地址： https://github.com/kubeservice-stack

一级调度	**业务模型策略** 基于服务QoS特性调度
二级调度	**多集群调度策略**
三级调度	**单集群内调度策略** crane-scheduler-plus　descheduler

•基于业务模型调度：

实现服务 QoS 资源保障模型。 按照服务的 QoS 等级， 给与优先级不同的算力保障

•多集群协同调度:

通过用户输入的调度需求、统一的全局资源视图，根据不同的调度策略产生对应的集群调度结果，满足不同应用对于跨集群调度 的需求

•单集群调度:

主要用于集群内节点调度。主要包含了 crane-schdeduler-plus（基于 crane-scheduler 调度器升级版）， 可基于真实负载感知调度、 抢占等策略

混部QoS统一管理

社区官网： https://stack.kubeservice.cn/
Github地址： https://github.com/kubeservice-stack

- 容器CPU QoS
  - CPU Burst 内核补偿,避免限流
  - CPU Group Identity干扰抑制
- 容器内存QoS
  - 优先保证高Qos Pod
- 容器L3 Cache & 内存带宽隔离
- 容器负载感知调度

木兰开源社区
MULAN OPEN SOURCE

挑战与未来

未来发展

1.推动内部服务 『云原生架构』改造升级

2.大云：超大规模集群（5000+Node、5W Service、20w Pod）商用，并且精细化运营

木兰开源社区
MULAN OPEN SOURCE

Thanks

@KubeService DongJiang

2022木兰峰会