Tencent 腾讯

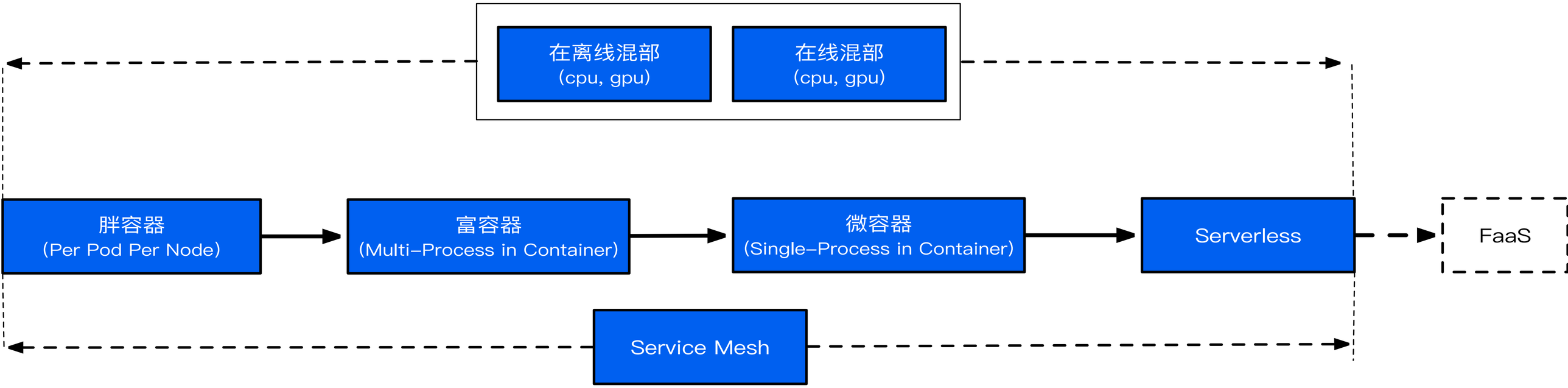
如何管理超千万核资源的容器规模

王涛 / 腾讯云容器平台负责人

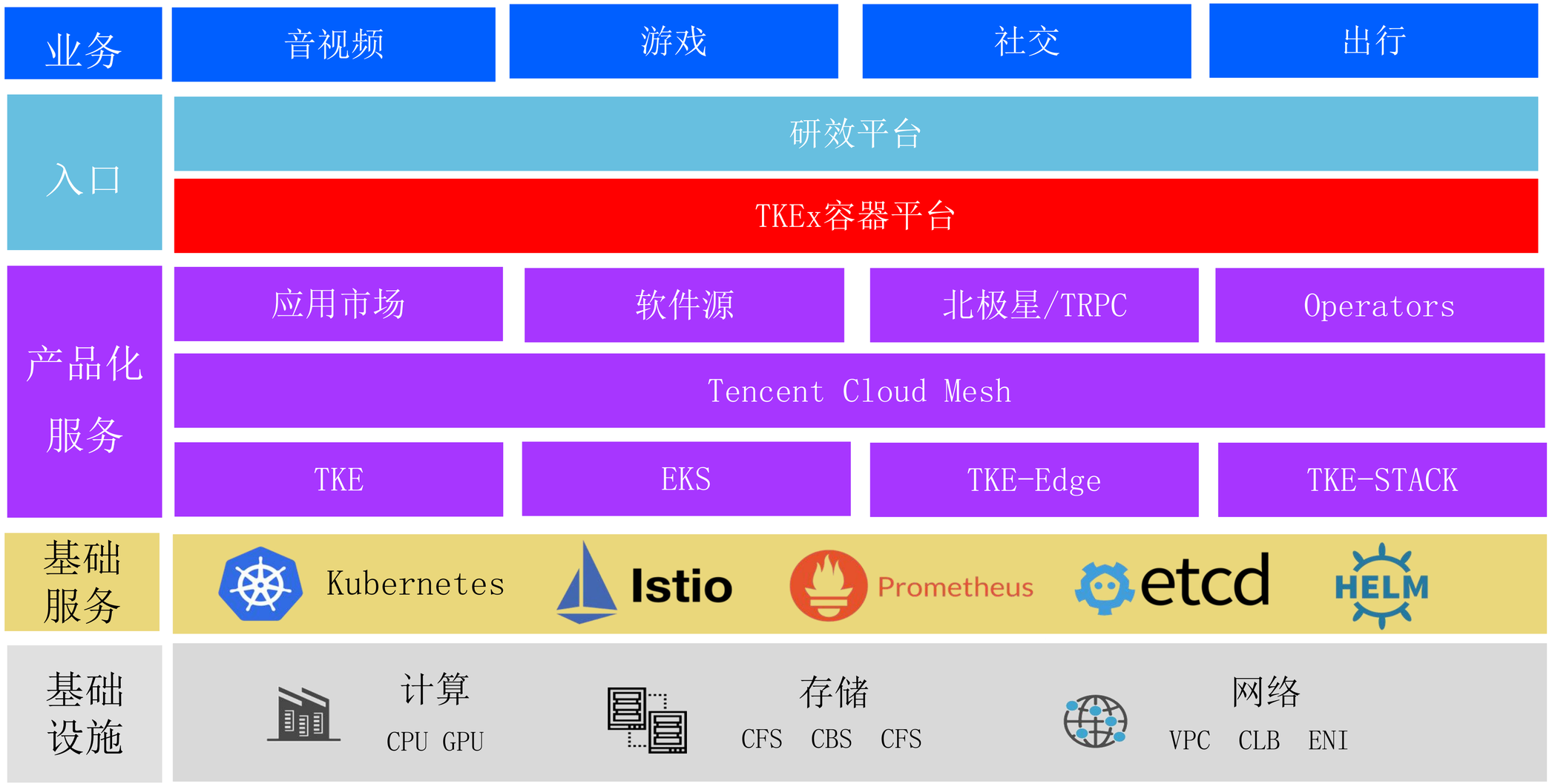
CONTENTS

1 腾讯自研业务容器化上云历程

容器化上云的技术路线


在离线混部
(cpu, gpu)
在线混部
(cpu, gpu)
胖容器
(Per Pod Per Node)
富容器
(Multi-Process in Container)
微容器
(Single-Process in Container)
Serverless
FaaS
Service Mesh

自研业务容器化上云产品简略架构



持续沉淀容器技术

经过3-4年的大规模自研业务云原生实践，在各个技术方向和业务场景都沉淀了大量的解决方案。

容器调度

- 节点负载均衡动态调度
- Pod升级资源预留
- 重调度

弹性伸缩

- Cluster AutoScaler
- HPA，VPA，CronHPA
- 基于预测的智能弹性伸缩

有状态服务容器化

- 分批灰度发布
- 原地升级/快速升级/热升级
- 长连接服务无损升级

资源管理

- 资源超卖技术
- 混部技术
- 云原生配额管理系统
- 降本增效产品

稳定性提升

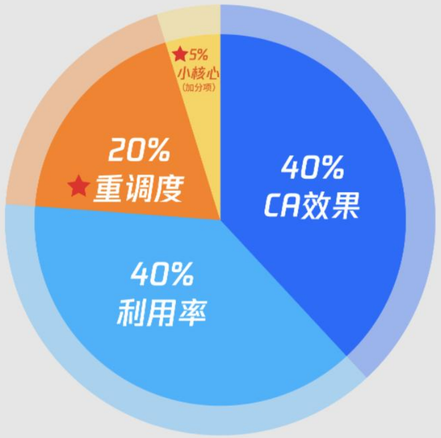
- 节点稳定性检测&自愈
- Pod异常检测&自愈
- 混沌工程&故障演练
- 集群稳定性系统化建设

业务运维效率提升

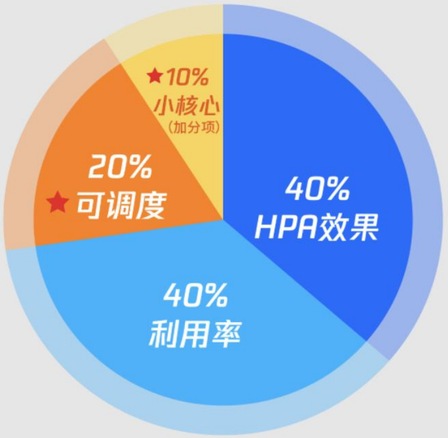
- 业务快速容灾部署&可视化
- 对接研效平台
- 业务大盘视图
（异常分布、高负载分布）
- 应用智能化诊断及自愈

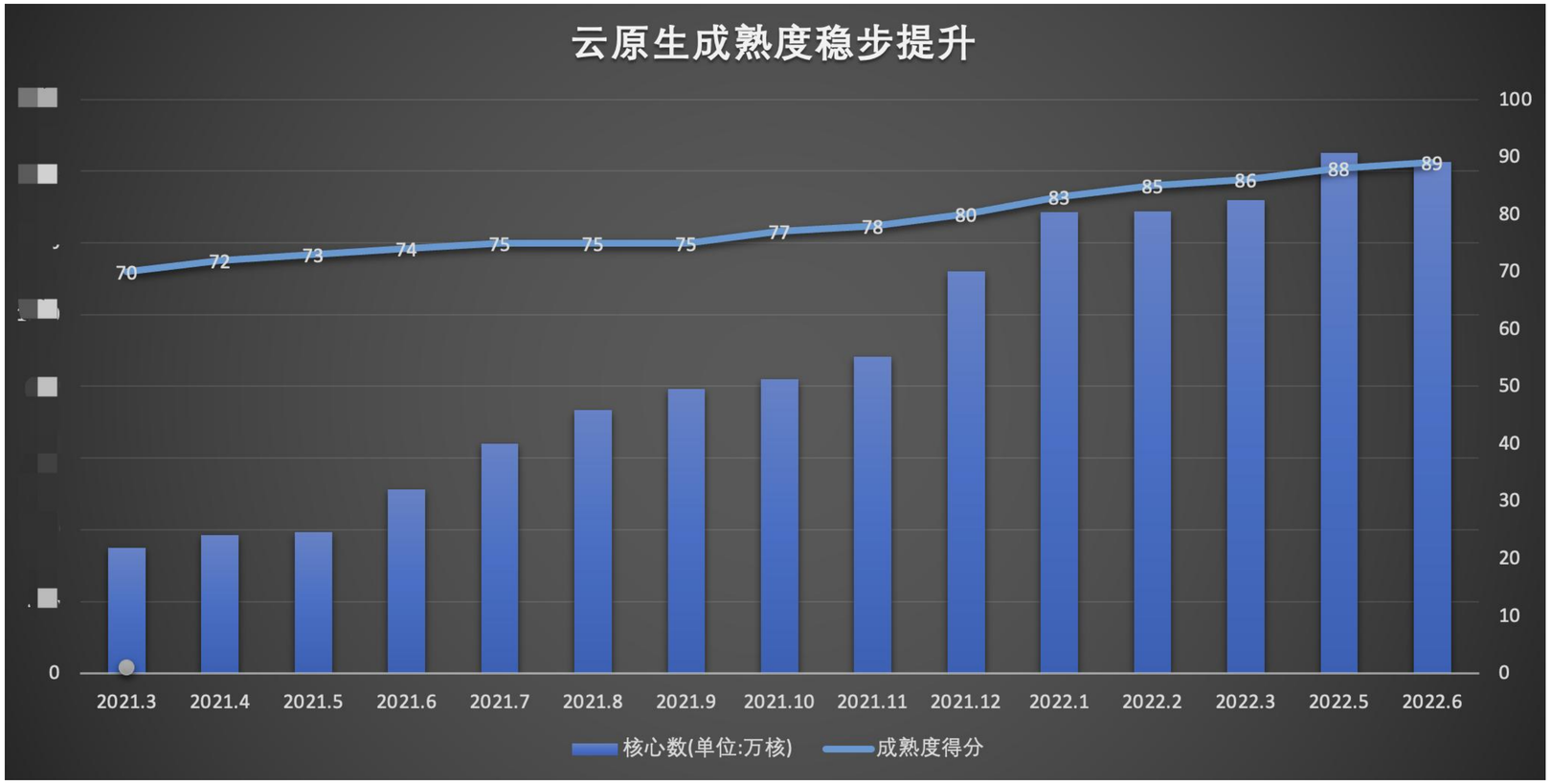
自研上云规模

平台侧打分规则：



业务侧打分规则：





2 各种混部场景下利用率提升方案

2.1 在线离线混部集群

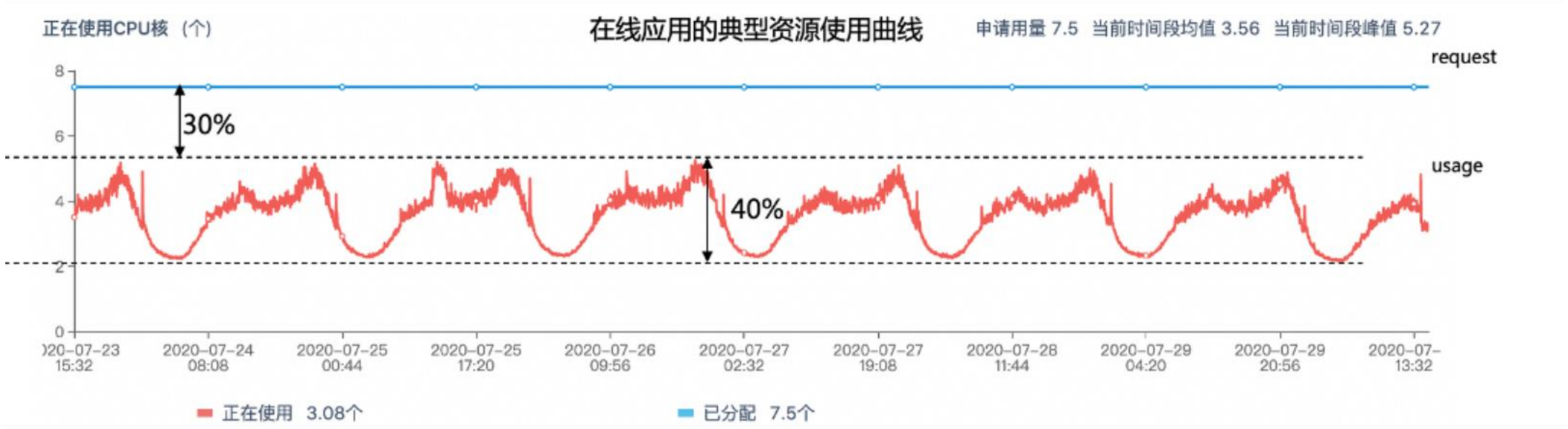
在离线混部面临的问题

在线作业资源利用率低

- 非容器化部署，未能充分利用整机资源
- 业务容灾占用资源
- 申请资源高于实际使用资源，“占而不用”
- 资源使用波峰波谷的潮汐现象
- 业务之间相互隔离

离线作业需要大量碎片资源

- CPU密集型任务
- 延迟不敏感，实时性不高
- 批量任务，可以容忍一定的失败率
- 执行周期时间短
- 资源申请量少，可以填充碎片资源

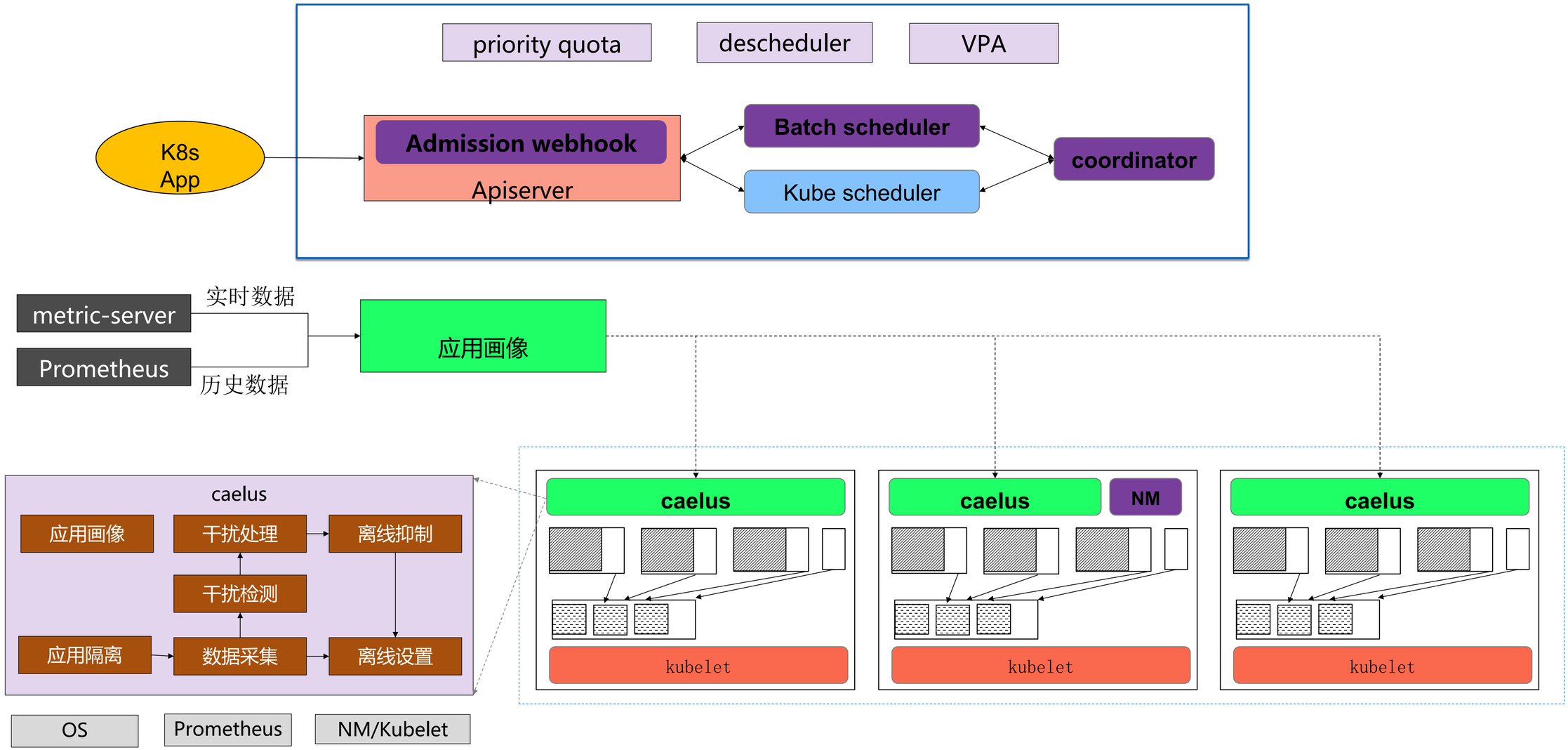


在离线混部原则&目标

设计原则	设计目标
通用技术，公司内外都可以使用，方便开放到社区以及输出到腾讯云客户	在线作业SLO受保证，离线作业不能无限填充
符合云原生方式	离线作业能快速上线下线，在线作业需要更多资源时，能及时避让
降低对应用的依赖，不能引入太多假设	离线作业的成功率受保证，不能因为频繁受限，导致失败率很高
兼容生态，K8s和Hadoop	在保证在线和离线服务质量的前提下，尽可能提升资源利用率

对 Kubernetes 零入侵！

Caelus全场景在离线混部



Caelus在离线混部流程

本地预测：

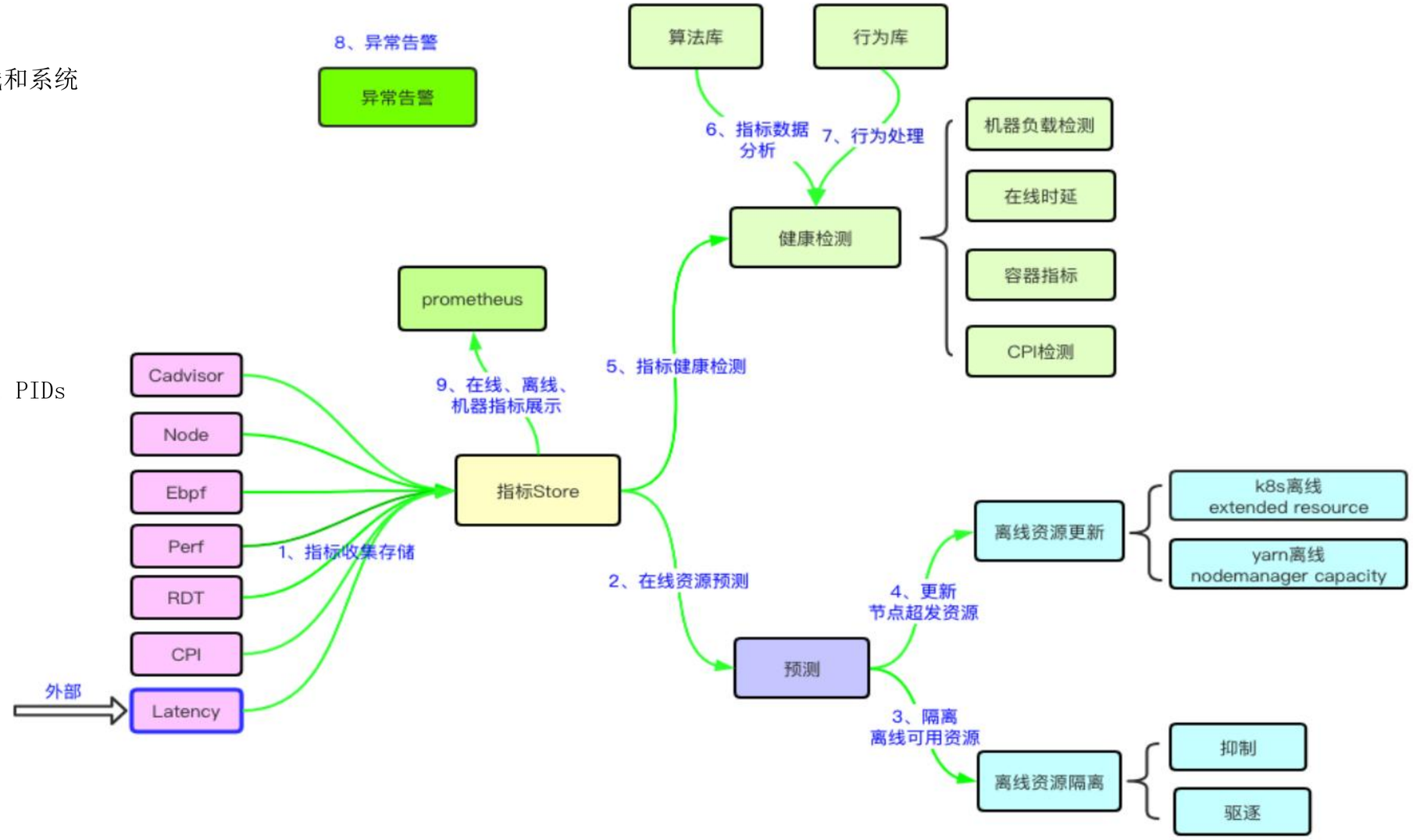
- 根据节点资源实际使用量预测，包括在线和系统进程
- 在线作业资源突变，可快速作出反应
- VPA组件优化，适配节点场景

全维度资源隔离：

- CPU、内存、磁盘IO、网络IO、本地磁盘、PIDs等
- 资源弹性，提升资源利用率
- 优先级抢占，满足在线资源需求

兼容主流离线生态：

- K8s生态，如云原生大数据
- Hadoop生态



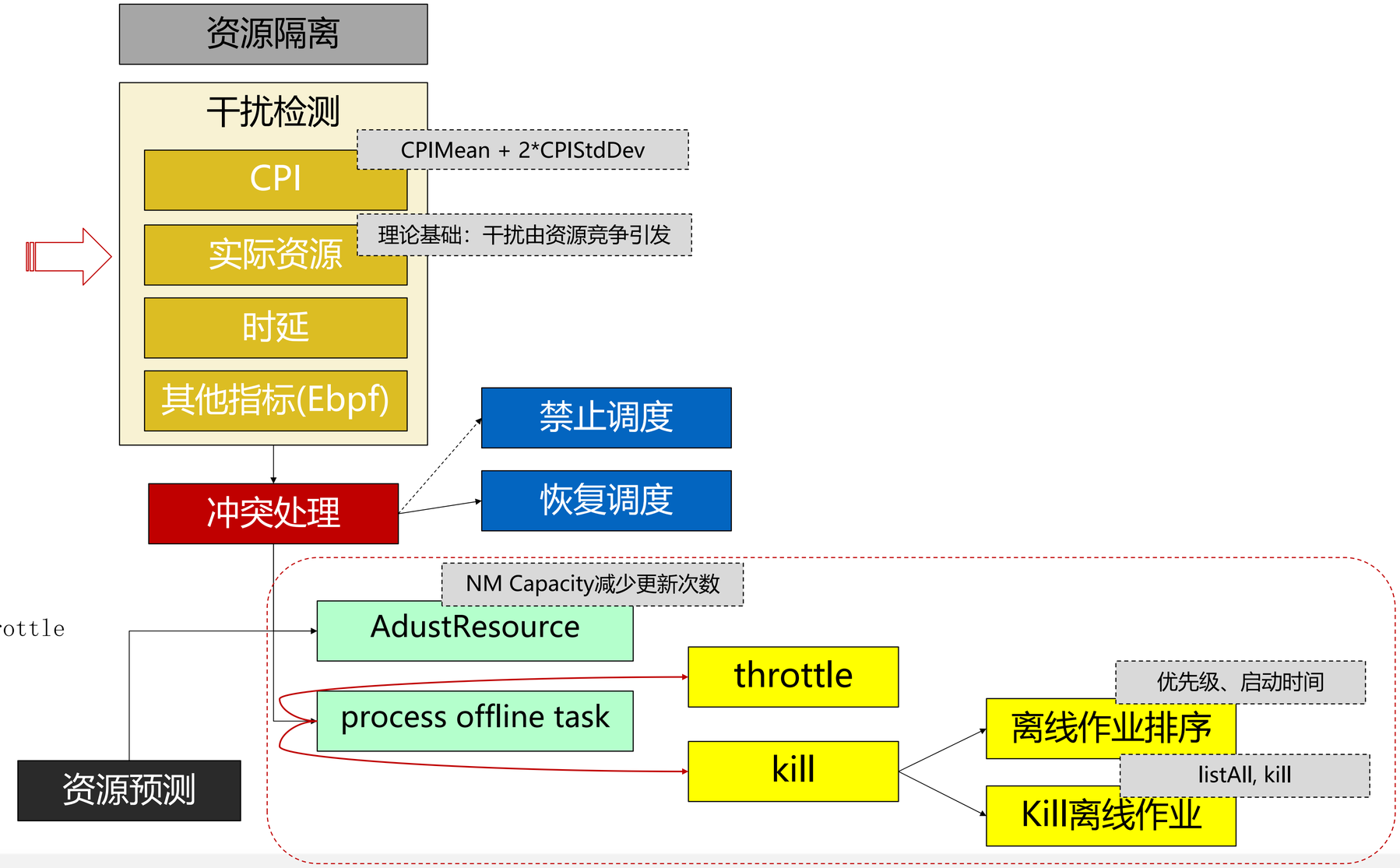
Caelus混部干扰检测与处理，全面提升混部应用的服务质量SLA

干扰检测：

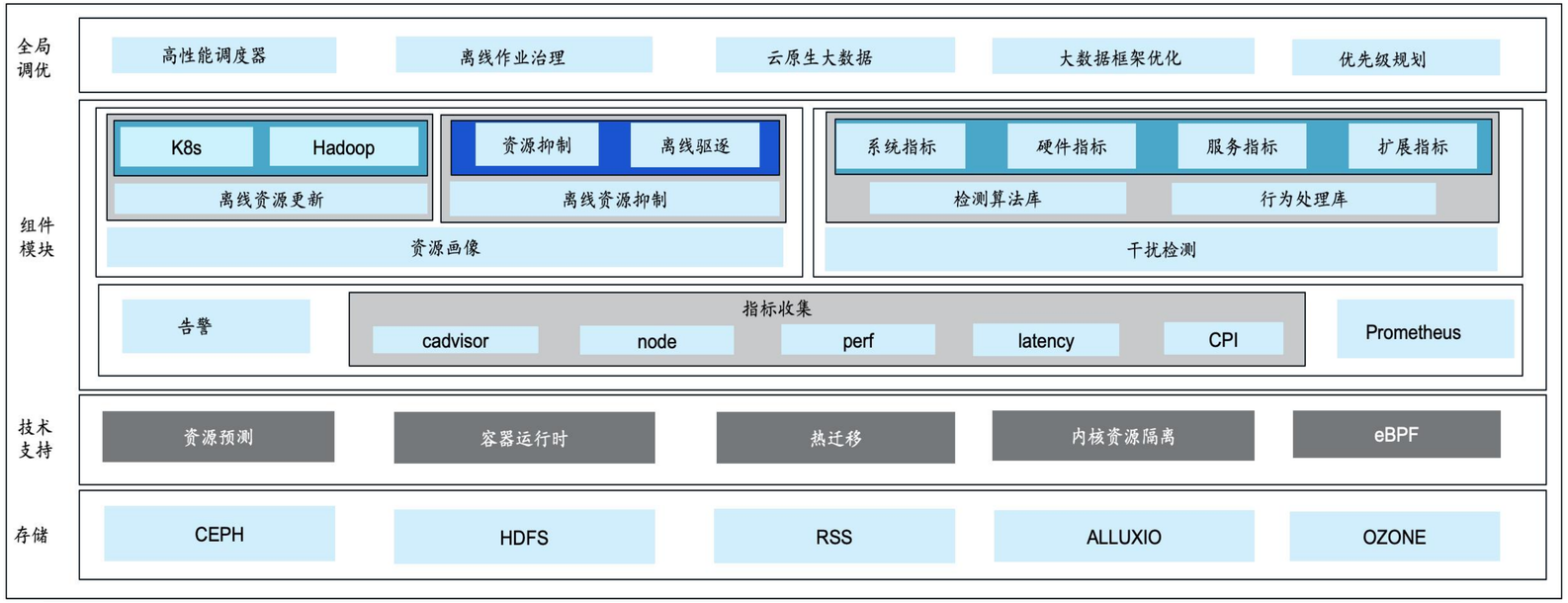
1. 检测干扰的方式：
   - 指标
   - 资源
2. 指标获取方式：
   - 需要应用配合
   - 无需应用配合，系统自动采集

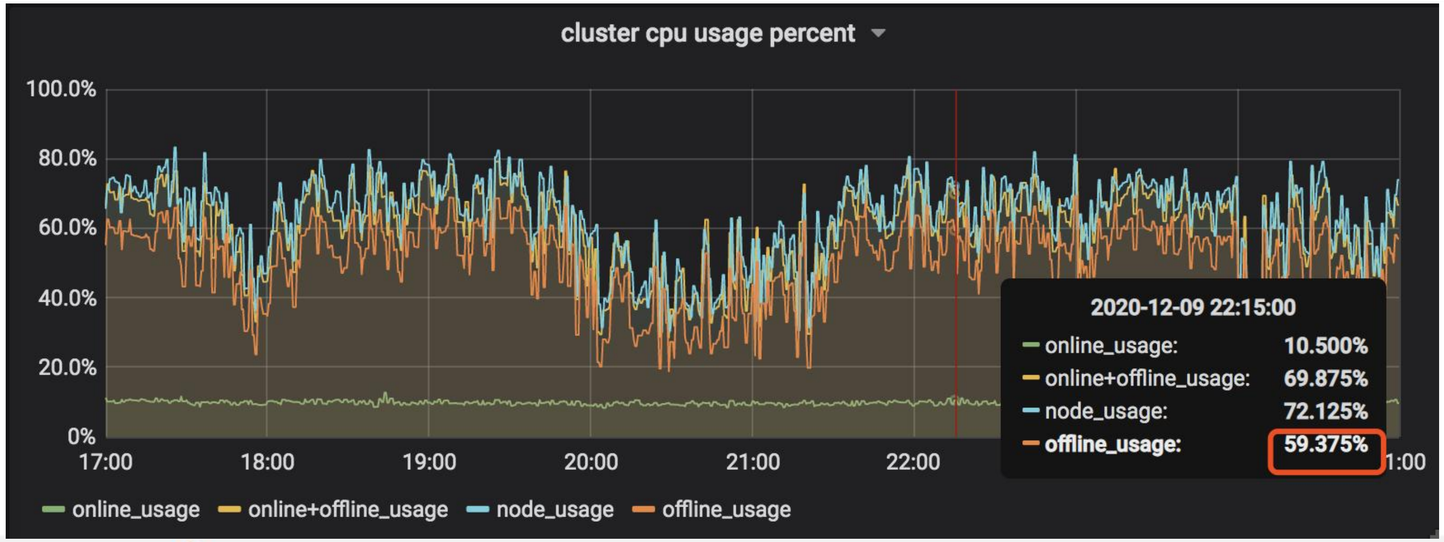
冲突处理：

1. 调整资源，快降低慢恢复
2. 处理离线进程
   - 不同类型资源处理不同：Kill/throttle
   - 按离线作业重要性排序驱逐



在离线混部通过Caelus对外输出





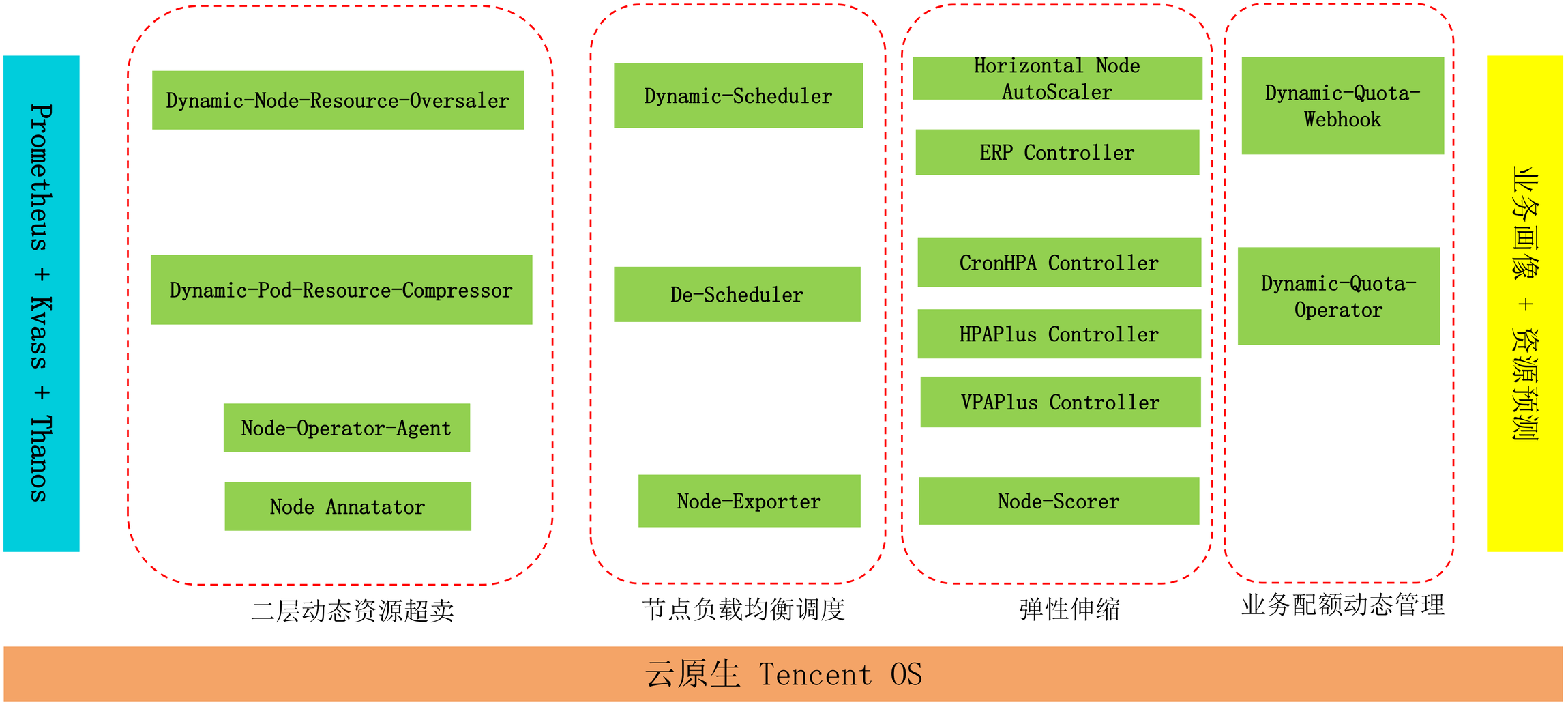
Caelus协同存储、内核、运行时、调度及离线框架等多层面，协同合作。在保障在线服务质量的同时，保障离线任务服务质量

通过Caelus在离线混部能力，集群CPU利用率可提升到60%

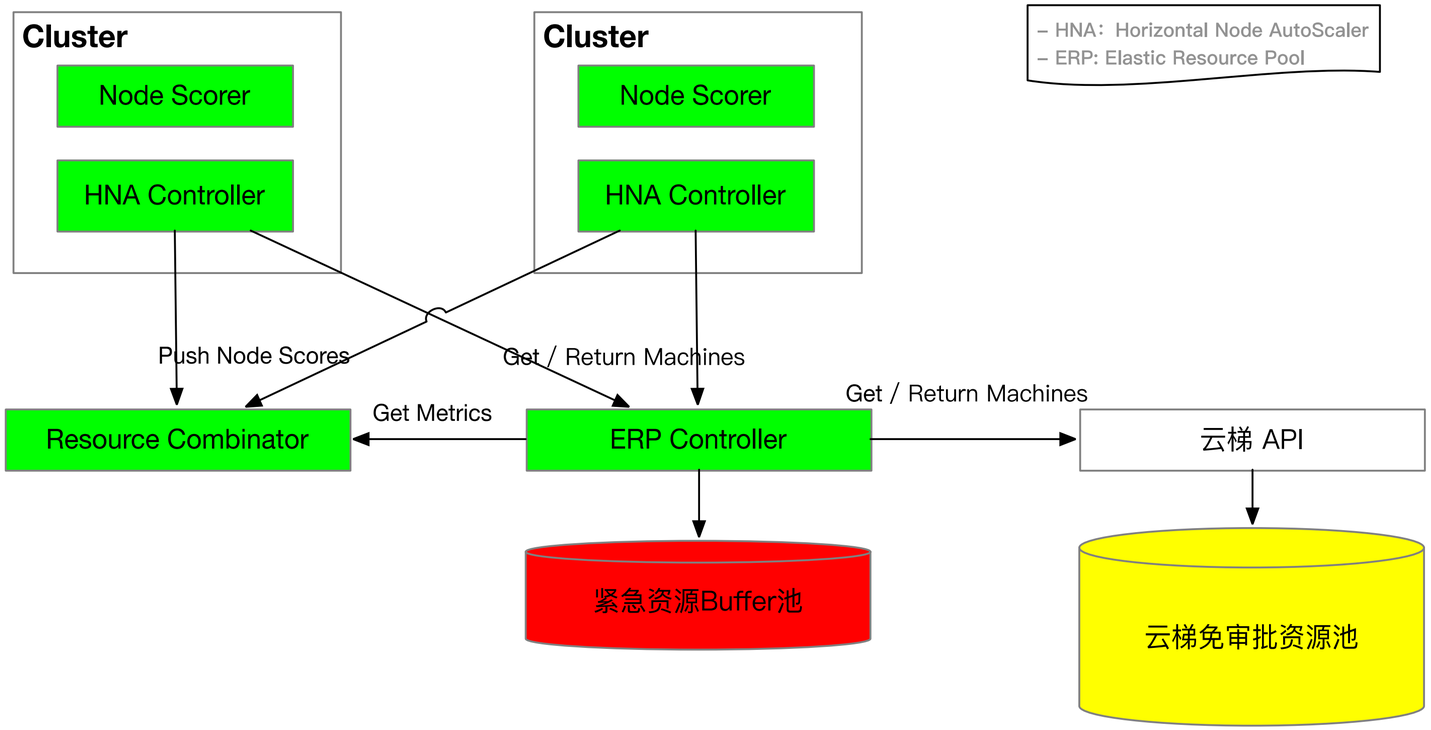
Caelus相关的能力已开源：https://github.com/Tencent/caelus

2.2 在线混部集群

在线混部利用率提升手段



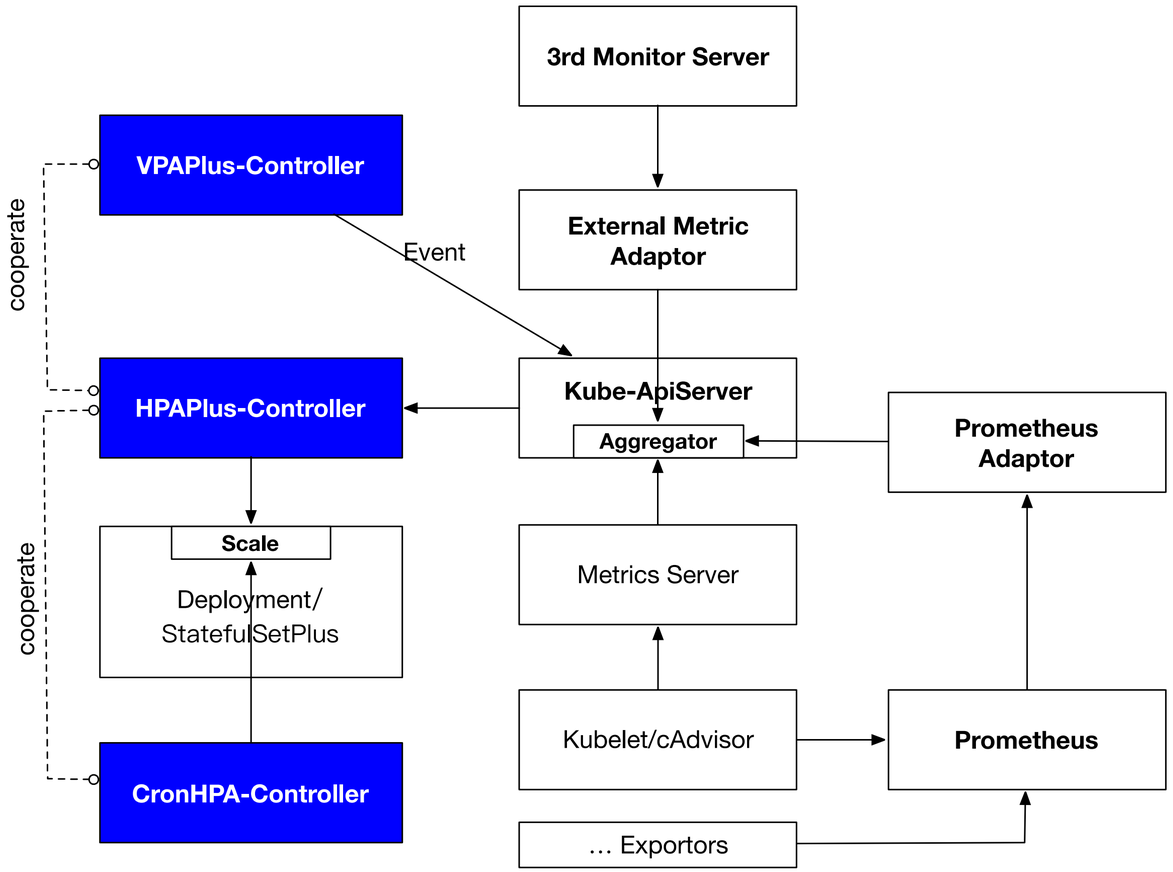
弹性伸缩-集群



- 二级弹性资源池方案，支持常规扩缩容和紧急扩缩容2种场景。
- 自研多集群资源协调器，将多集群的闲置资源构建统一的平台级Buffer池，让资源在多集群高效流转。
- 节点上下线实现自动化和标准化。
- 集群负载高或者资源不足时，最快可实现小于1分钟的扩容速度，这样对提升集群负载有极大的益处。

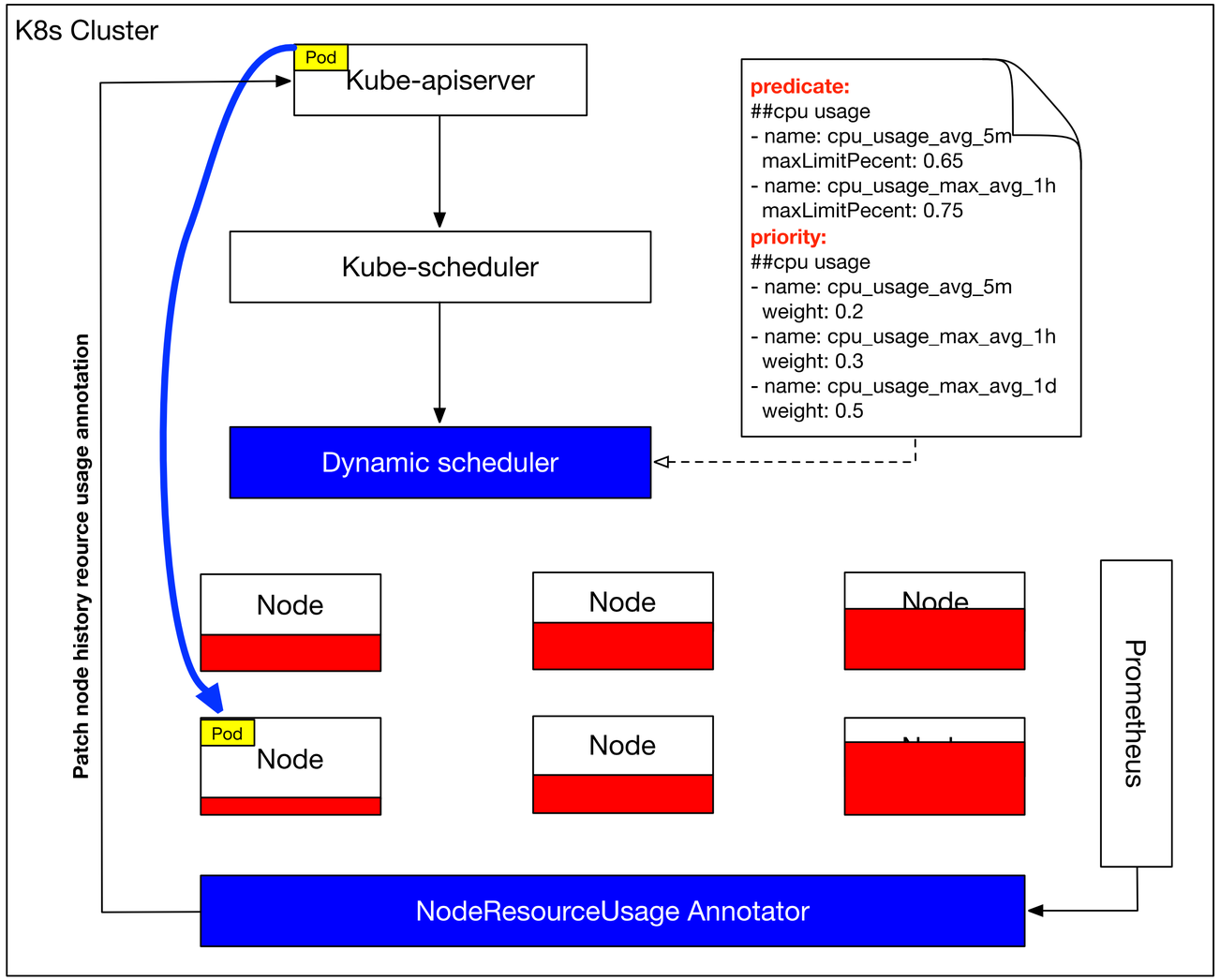
弹性伸缩-业务

给有状态/无状态服务提供全面、高性能的AutoScale服务，实现无边界弹性。



- HPAPlus-Controller：支持业务常规弹性伸缩场景
  - 支持HPA对象自定义关键配置：扩缩容速率/计算周期/指标容忍度等。
  - 支持弹性的maxReplicas策略，避免超出预期的流量受限于maxReplicas配置太低，导致业务雪崩。
  - 性能优化：支持几千个业务HPA对象并行弹性伸缩计算逻辑。
- CronHPA-Controller：支持业务周期性弹性伸缩场景
  - HPA与CronHPA联动决策：支持业务计划内的定时扩容策略，如果业务实际流量超过预估流量，仍能自动扩容。
- VPAPlus-Controller：支持有状态服务无感知垂直扩缩容场景
  - 根据业务历史负载监控，对有状态服务进行无感知扩缩容。
  - 联动VPA和HPA：当Pod VPA达到Node资源上限时自动触发HPA进行横向扩容。

动态调度器

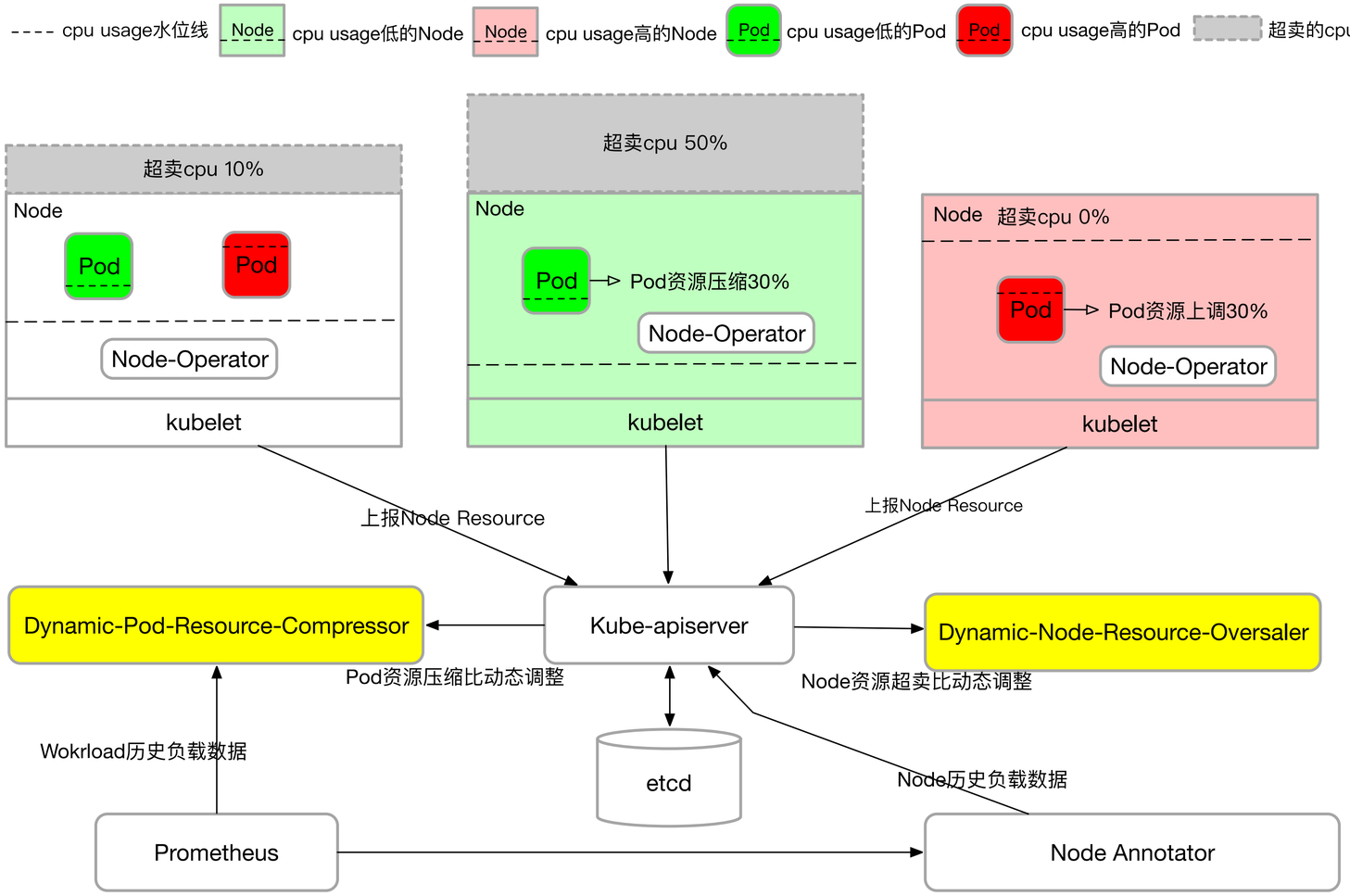


Kubernetes原生调度器属于静态调度，当大量业务混部在一个集群时，必然出现节点负载不均衡，Pod调度时仍可能往高负载节点上调度，造成业务服务质量下降。

- 自研动态调度器：让节点的关键资源在集群节点中均衡分布
  - Cpu/ Memory/Disk usage
  - Network io / System load / Iowait / softirq
- 如何避免调度热点问题，是一个有趣的问题。
  - 自研热点动态补偿算法

二层动态资源超卖

L1: Node L2: Pod



资源超卖是所有资源调度平台必须深耕的技术，是降本增效的最有效手段，动态超卖更加安全可控。

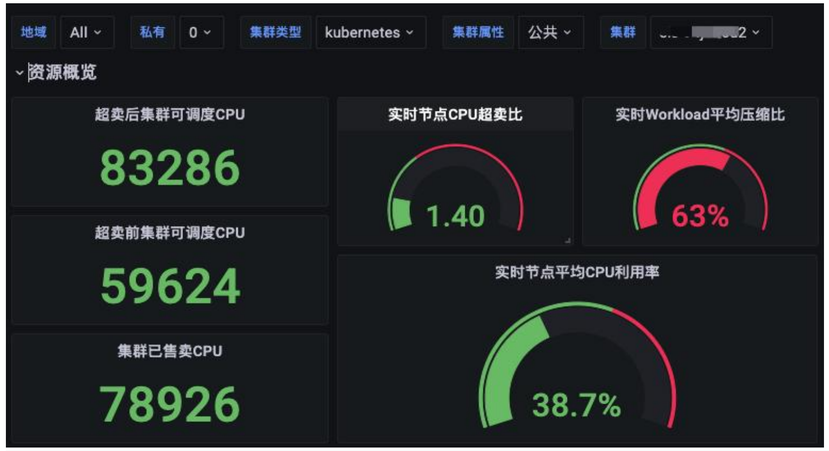
技术挑战：

- 节点超卖比的安全控制，防止影响业务稳定。
- 节点资源超卖对Kubernetes驱逐机制和资源预留机制的影响。

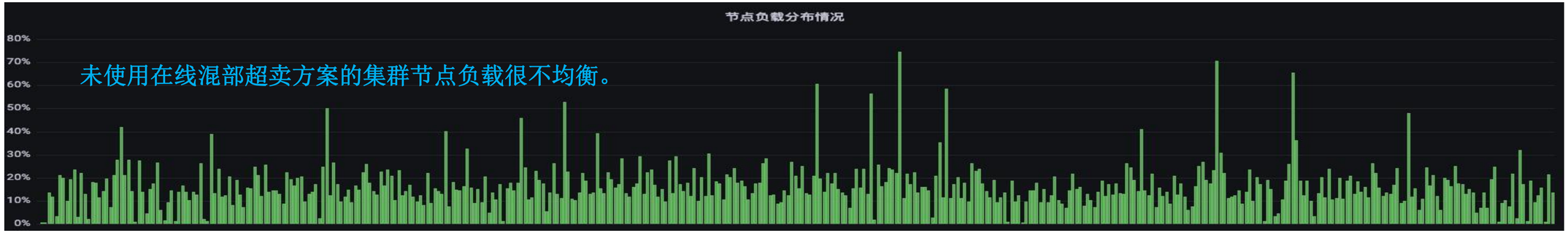
关键手段：

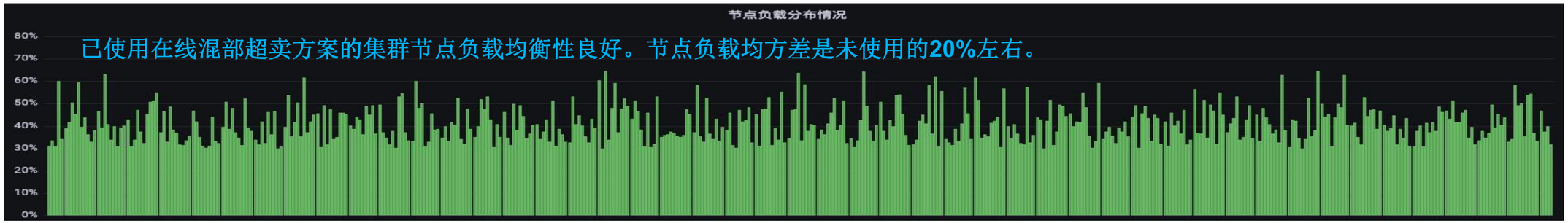
- 超卖比需要根据节点实时的负载数据进行动态调整，防止造成节点负载过高。
- 节点负载和容器负载可预测，提前规避节点出现高负载。
- 如果出现预料外的节点高负载，通过de-scheduler及时降低节点负载。
- 极端情况如果出现大面积节点高负载，通过HNA进行秒级扩容。
- 超卖比和Kubernetes节点稳定性机制（驱逐和资源预留）有关联，超卖比变化需要动态调整kubelet对应的配置。

在线混部利用率提升方案效果



通过在线业务混部超卖方案，集群CPU平均利用率提升到 **30%~40%**

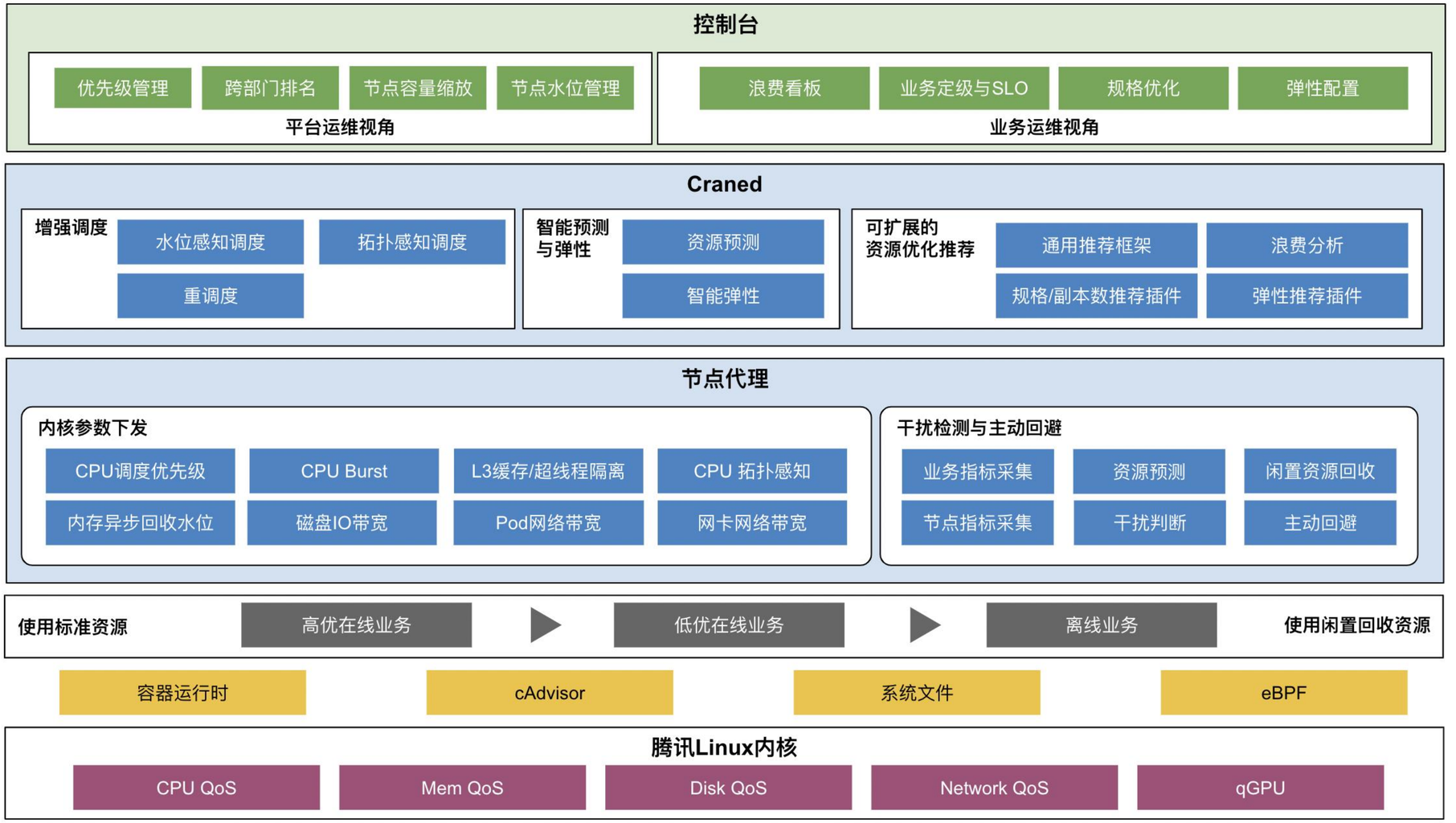




通过Crane对外开源全套技术

在离线混部、在线混部等全场景的混部技术通过Crane项目对外开源

crane: https://github.com/gocrane/crane

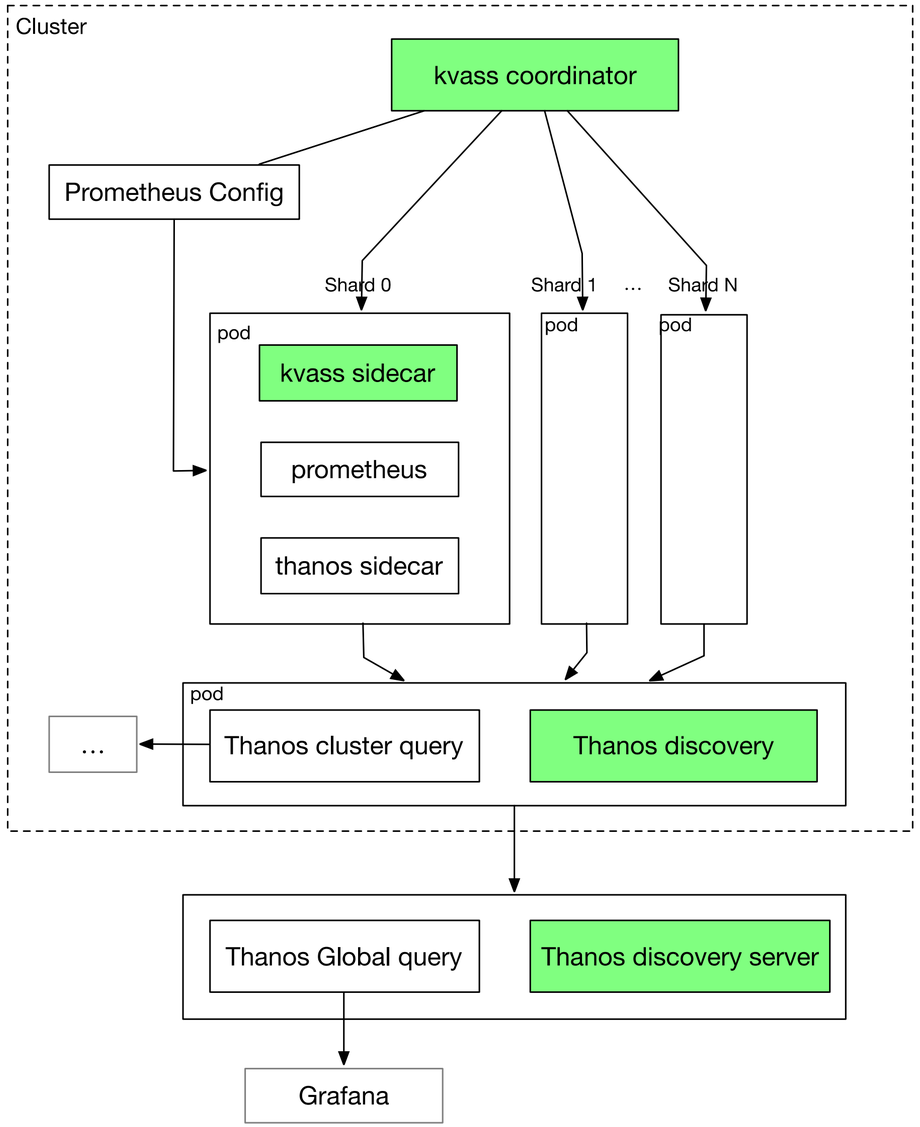


3 稳定性面临的挑战及其破解之法

稳定性全面剖析

平台层稳定	集群层稳定	节点层稳定	业务层稳定
➢ 完善的监控告警体系，让稳定性指标可视化。 ➢ (平台→集群→组件 →节点→Workload/Pod→Container→Process）	➢ 自动探测集群组件的状态及异常告警。	➢ Dockerd/Containerd/Kubelet核心节点组件状态监测与告警、自愈。	➢ 关键业务稳定性指标可观测
➢ 使用云上PaaS(Redis, TMDQ, TDSQL…)	➢ 集群组件高可用部署，配置健康检查，异常自愈。	➢ OS/Kernel稳定性指标监测与告警。	➢ De-Scheduler对异常Pod进行重调度。
	➢ 集群建设要自动化、标准化。	➢ 节点内核版本基线化管理，自动化监测内核版本和热补丁集不一致，自动进行节点内核升级和Patch热补丁。	➢ 一键多地域多可用区容灾部署。
		➢ 容器场景内核参数优化。	➢ 多集群协同编排调度，拉通多集群的资源。
			➢ 业务基础镜像优化。（精简版、标准版）

平台&集群稳定性



Case 1: 提升Prometheus集群的稳定性，具备动态弹性能力。

- 稳定：根治Prometheus OOM造成监控数据断点
- 运营：数百个集群，需要有主动服务发现能力，降低人力运营成本
- 性能：大规模的监控数据查询，如何保证关键查询的性能

技术方案：

- Kvass：Prometheus Auto Shard，让Prometheus集群基于targets和内存HPA。
- 自研Thanos discovery组件，具备多集群服务发现的能力。
- 分级查询：集群内查询 - Thanos cluster query，平台全局的查询 - Thanos global query。

Case 2：集群规模巨大，配置项巨多，集群标准化建设非常关键。

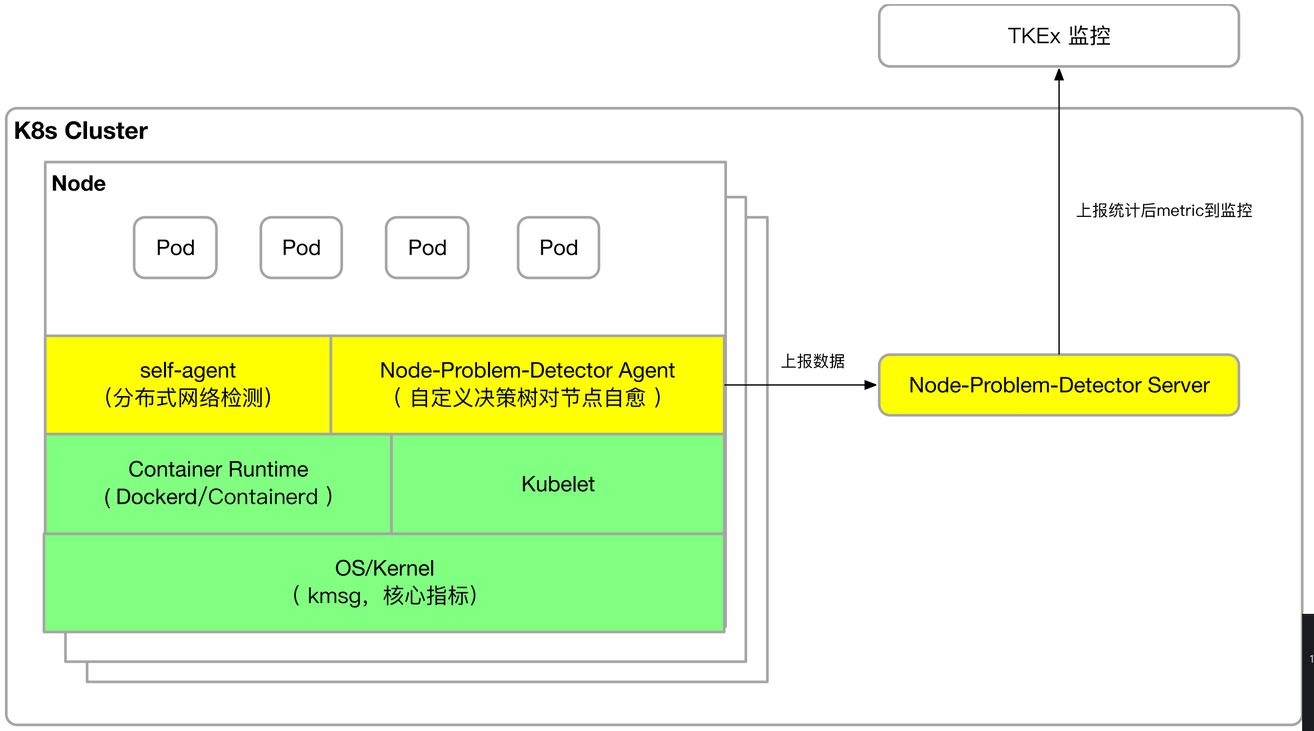
组件更新进度

ID	组件名	插件名	探测集群数	更新进度
270	node-inspection	VolumeMount	78	98.72%
319	loadbalancer-webh...	ImageVersion	86	98.84%
1	dynamicquota-oper...	ImageVersion	86	100.00%
4	dynamicquota-web...	ImageVersion	86	100.00%
6	dynamicquota-hpa...	ImageVersion	90	100.00%
10	image-gc	ImageVersion	72	100.00%
12	coredns	ImageVersion	90	100.00%
13	tke-eni-ipamd Confi...	ConfigMapKeyExist	91	100.00%
14	gpu-manager-pro	ImageVersion	9	100.00%

组件运行状态

组件名	正常集群数	正常集群比
cls-provisioner	90	98.9%
add-initcontainer	91	100.0%
admission-webhoo...	7	100.0%
cbs-provisioner	77	100.0%
cmdb	91	100.0%
coredns	90	100.0%
dcgm-exporter	9	100.0%
descheduler	45	100.0%
dynamic-scheduler-...	44	100.0%

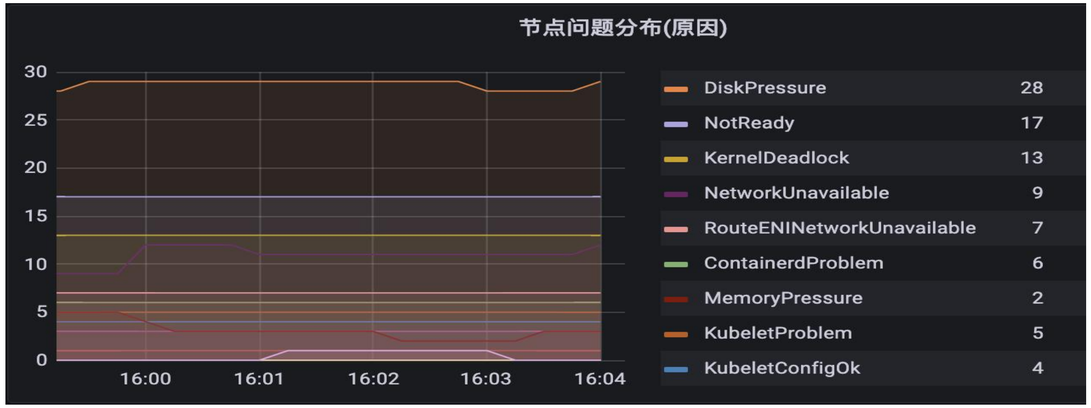
节点稳定性

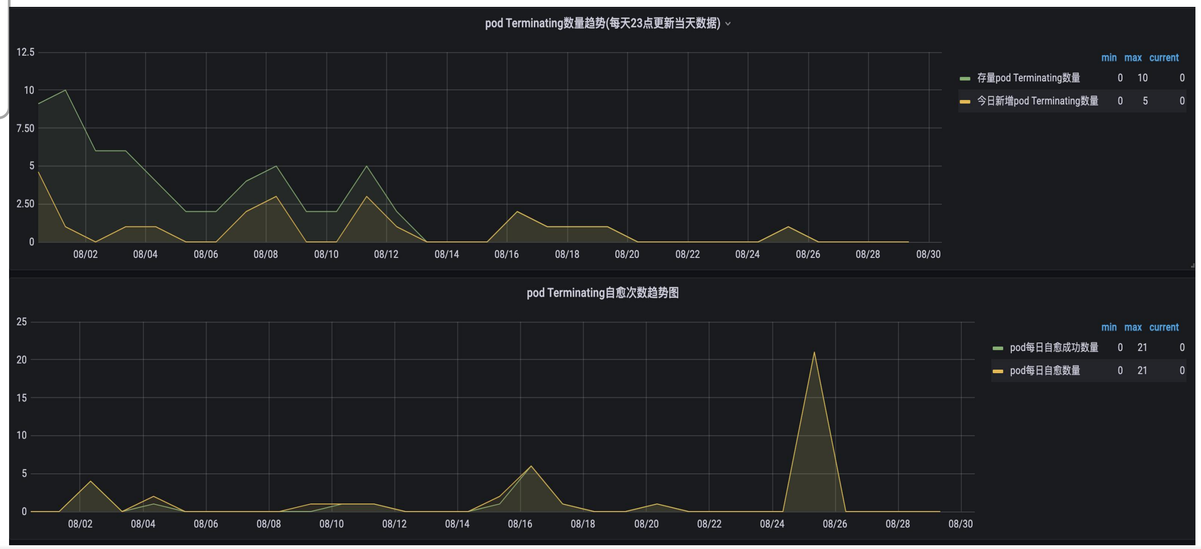


Case 1: 对核心组件/OS/Kernel的状态和日志、指标监控进行自动化分析，并基于预定义的决策树进行自愈决策。

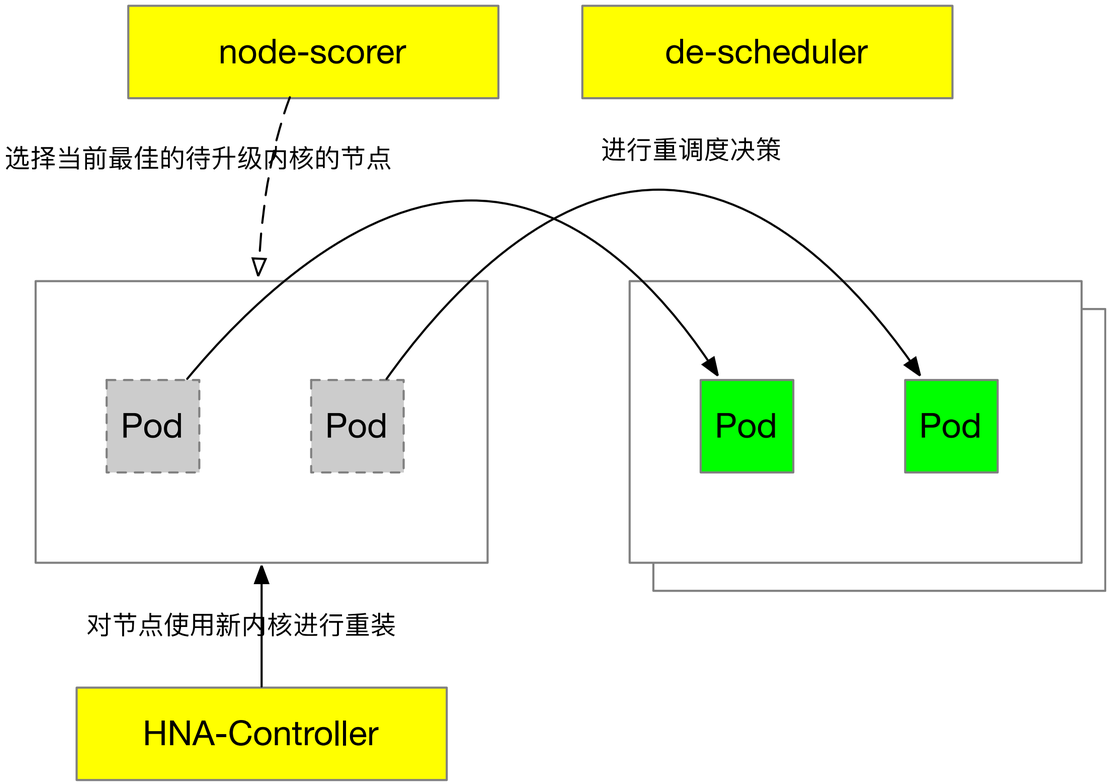
NPD(Node-Problem-Detector)支持的节点稳定性检测指标如下，支持自定义自愈动作。

- Dockerd/Containerd/Kubelet状态和异常日志分析
  - Umount Failed | Shim残留 | Syncloop hung住 | 进程D状态 | Cgroup 泄露/残留 | Container残留
- OS稳定性指标检测
  - Memory usage | 数据盘 usage | PID Pressure | D状态进程数 | Iowait | System load | FD Pressure |节点网络异常检测
- 内核稳定性事件检测 （云原生TencentOS）
  - Kernel死锁 | Softlockup | Hungtask | RCU Stall | Kernel Panic





节点稳定性



Case 2: 当前的基线内核暴露问题增多，有些问题无法提供热补丁，为了提升节点稳定性，升级内核势在必行。

如何让节点内核升级对业务尽量无感知，做到风险低、自动化、常态化，是一件极具挑战的事情。

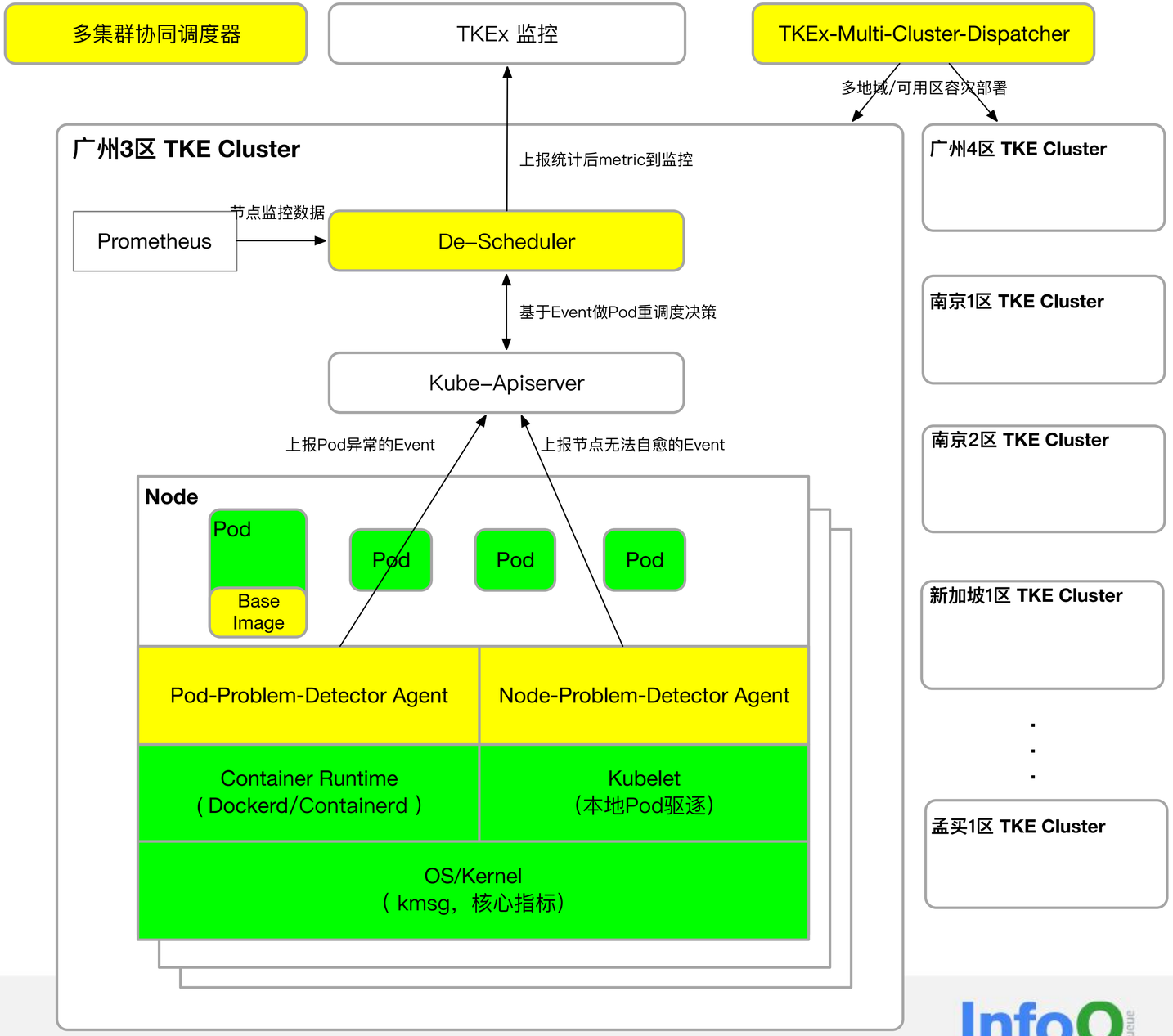
解决方案：自研node-scorer组件，对节点上的业务Wokload进行分析（高可用、负载、有状态性等），选择最佳的待升级的节点集，复用集群扩缩容组件HNA Controller对节点进行新内核的重装。

业务稳定性

业务经常因节点关键资源抢占导致业务服务质量下降。

深入内核，从内核层面提供更丰富的容器级的稳定性指标，在容器层面进行协同调度编排。

- Prometheus
  - Node cpu/mem/fd/disk/load/ iowait/softirq
- Node-Problem-Detector
- Pod-Problem-Detector
  - Pod restart frequency/fd
  - Pod Load.r/load.d
  - Pod Long sys
  - Pod Cpu调度延时
  - Pod Iowait
  - Pod 内存分配延时
- 多可用区容灾部署
- 多集群协同调度器
- 基础镜像和内核参数优化



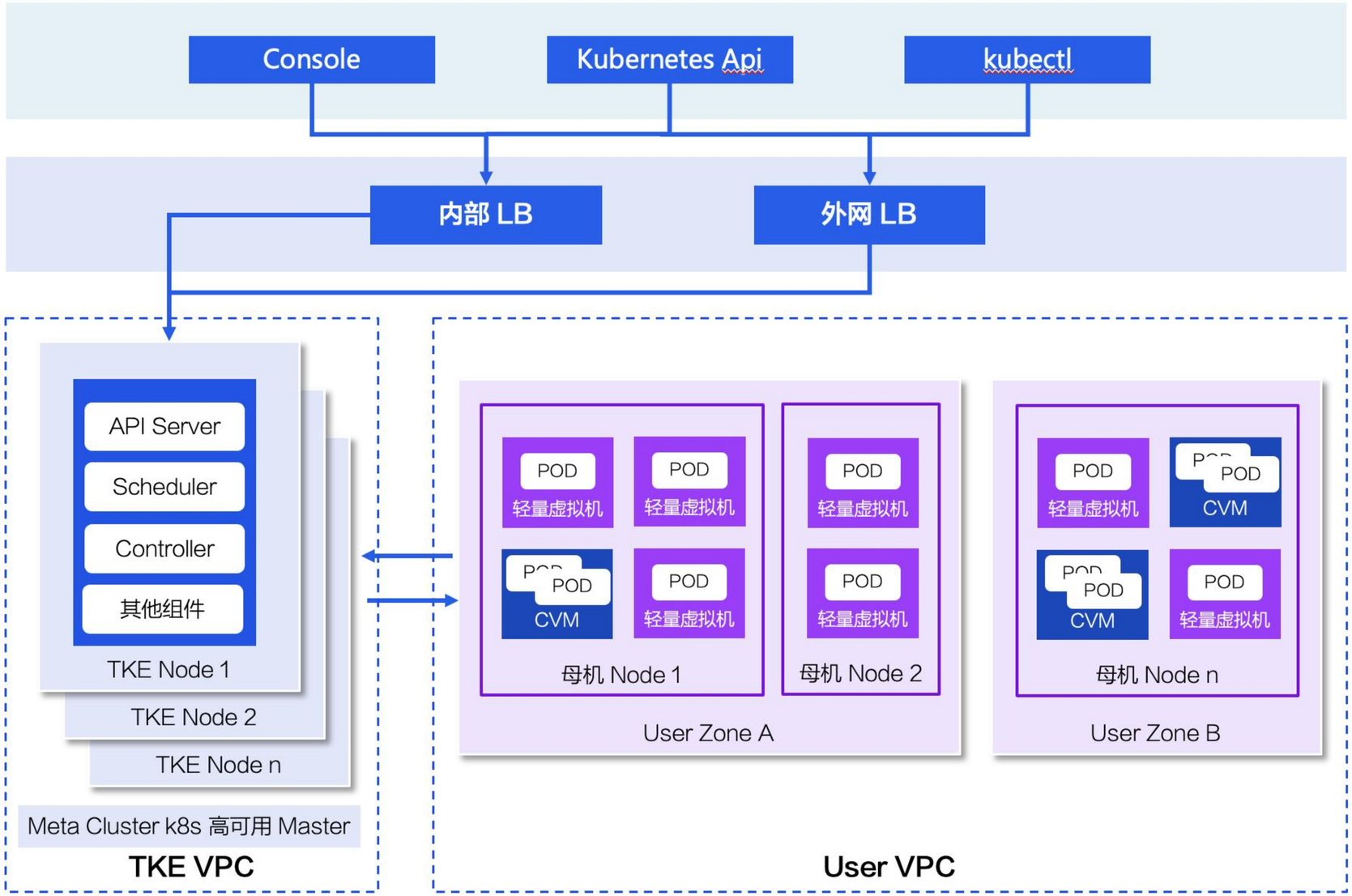
逐步大规模使用Serverless K8s （EKS）

架构

- Master 组件托管在 Meta Cluster。
- Pod 运行在轻量级虚拟机里，具备良好隔离性。
- 轻量级虚拟机和 CVM 共享大盘母机资源，资源量充足。

优势

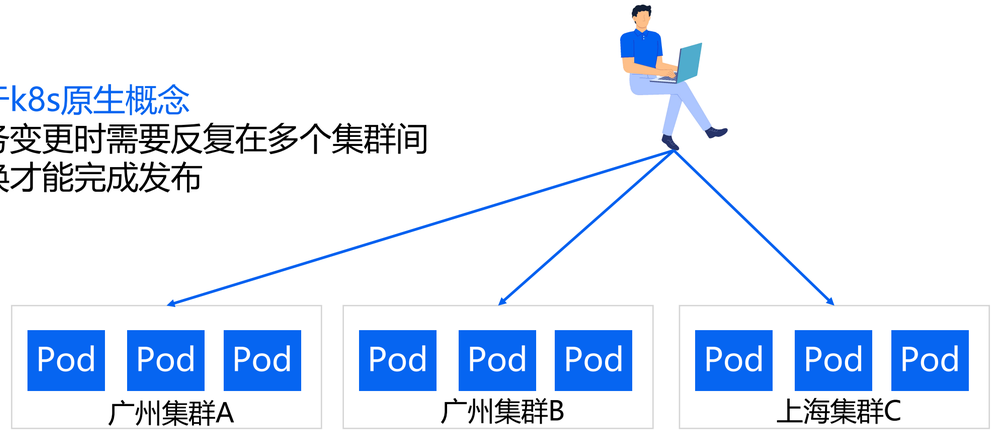
- 云原生标准协议
- 高性能，高可用，安全可靠
- 轻运维，支持异构算力
- 计费灵活，弹性效率



4 从面向集群到面向应用的调度编排

面向集群业务管理效率低

基于k8s原生概念
业务变更时需要反复在多个集群间切换才能完成发布



案例：某个业务全网有1w个工作负载、5w个Pod，分布在17个region 的80个集群中，一次变更需要灰度多天，灰度期间运维实时值守

从资源视角到
应用视角

抽象出应用概念-业务可在统一视图下一次性完成应用管理，无需切换集群
基于TAD实现应用视角的跨级群应用管理，业务无需关心集群与资源概念

- 跨集群应用统一变更
- 跨集群应用配置管理
- 跨集群应用业务看板
- 跨集群应用弹性伸缩
- 跨集群应用流量调度
- 跨集群应用容灾检查

基于Clusternet-腾讯云开源的多集群管理项目

目标效果：一键点击发起应用的全球灰度变更，灰度期间偶尔关注应用视图即可了解大盘情况，无需实时关注，解放人力

Clusternet Core Features

➢Kubernetes Multi-Cluster Management and Governance

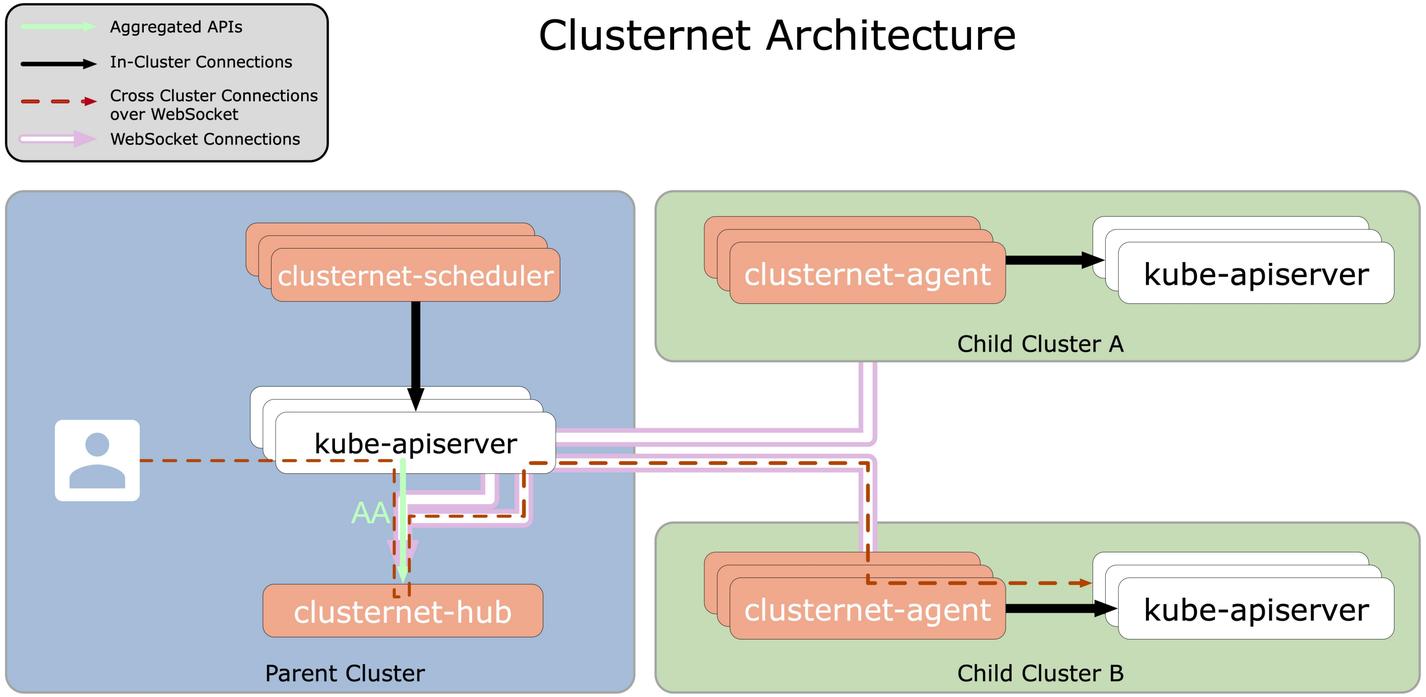
- managing Kubernetes clusters running in cloud providers
- managing on-premise Kubernetes clusters
- managing Kubernetes clusters running at the edge

➢Application Coordinations

- Cross-Cluster Scheduling
  - ✓ cluster label selectors
  - ✓ cluster taints & tolerations
- Various Resource Types
  - ✓ Kubernetes native objects, such as Deployment, StatefulSet, etc
  - ✓ CRD
  - ✓ helm charts
- Setting Overrides
  - ✓ two-stage priority based override strategies
  - ✓ easy to rollback
  - ✓ cross-cluster canary rollout

Clusternet: https://github.com/clusternet/clusternet

Clusternet Architecture

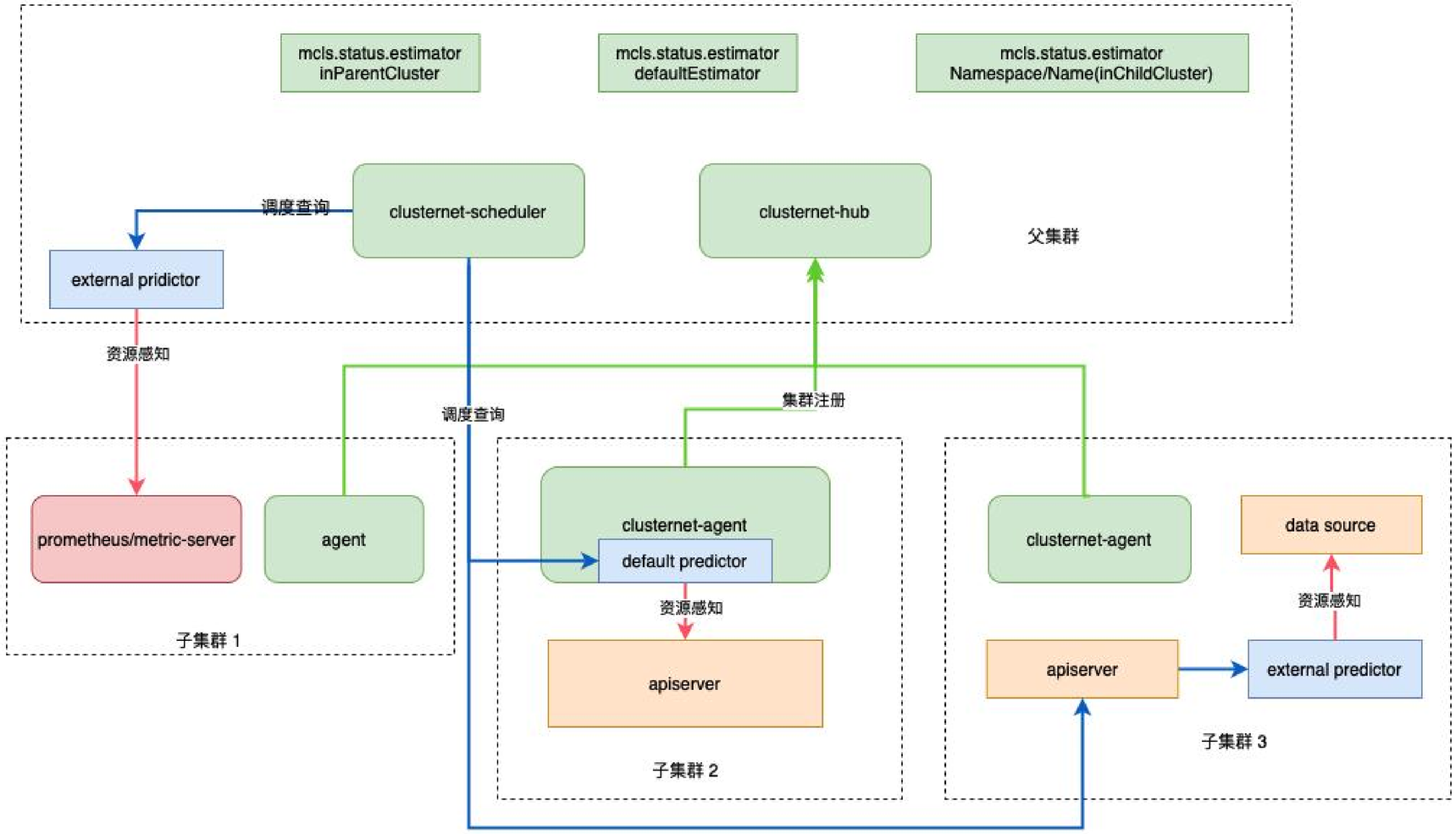


动态拆分调度

感知子集群在集群、节点等维度的各种资源使用情况，提供标准接口供scheduler调用，为精准调度提供支持。

调度流程：

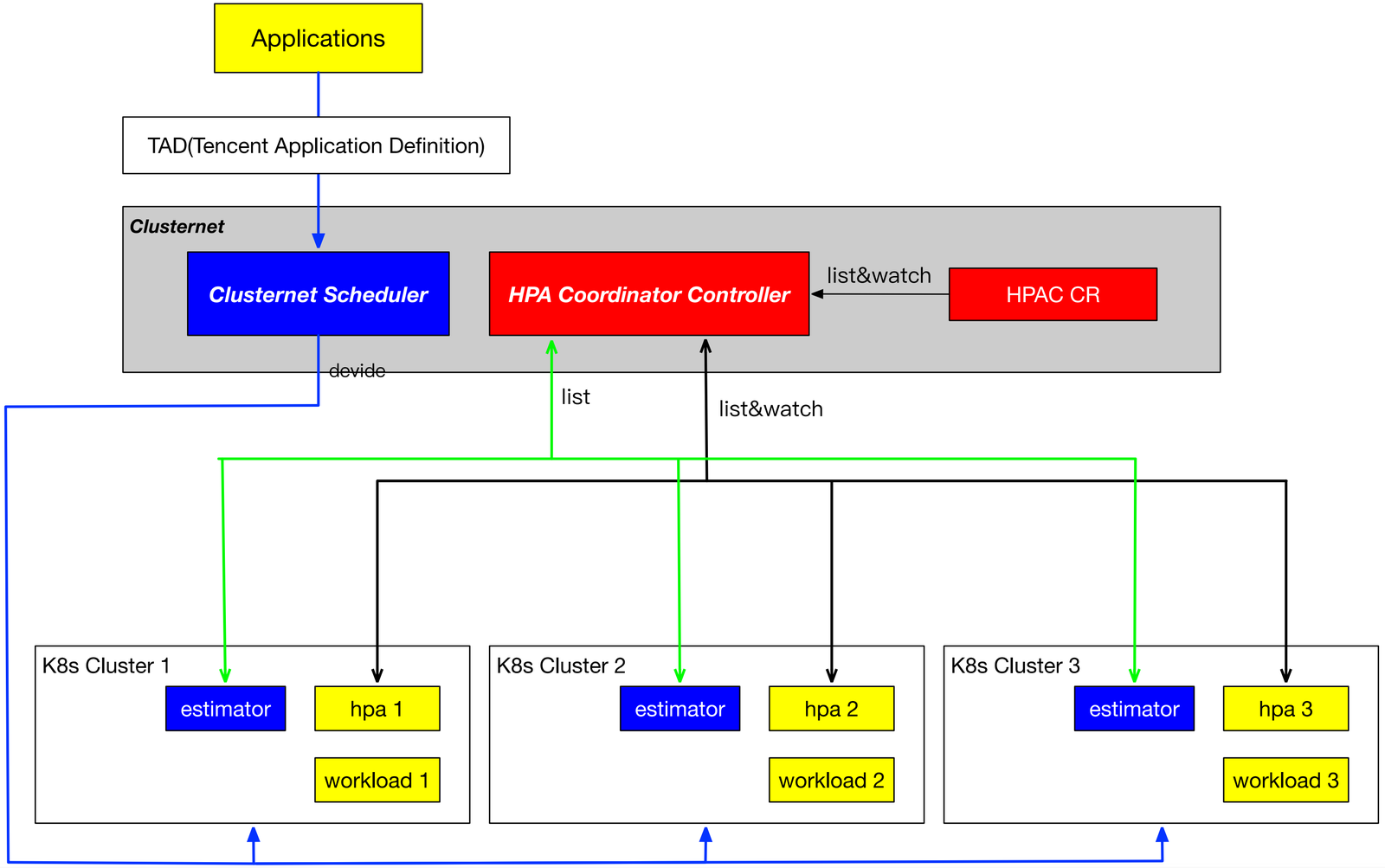
- 1. clusternet-scheduler根据workload中podTemplateSpec向predictor发起查询请求：
  - request resources
  - nodeSelector
  - Affinity
  - tolerations
  - priority class
- predictor根据请求，匹配得出本集群可容纳副本数，返回给clusternet-scheduler
- 根据predictor返回的结果，scheduler优选部署集群并设置副本数



面向应用的多集群协同弹性伸缩

通过Hub-Cluster中的 HPA Coordinator Controller，根据应用的多集群拓扑分布，子集群资源状态，动态调整应用在多个集群中的弹性能力，从而实现应用副本在子集群的动态协同弹性。

```yaml
apiVersion: tkex.io/v1alpha1
kind: HorizontalPodAutoscalerCoordinator
metadata:
  name: app-demo
  namespace: default
spec:
  tarfre deploy
  strategy: auto/percent  # 自动策略，还是有一定的半分比策略
  tolerance: 10 # 如果使用百分比策略，那么每个集群的范围±10%
  feeds:
    - cluster: cls1
      apiVersion: autoscaling/v1
      name: hpa-demo
      namespace: default
      percent: 50
    - cluster: cls2
      apiVersion: autoscaling/v1
      name: hpa-demo
      namespace: default
      percent: 20
    - cluster: cls3
      apiVersion: autoscaling/v1
      name: hpa-demo
      namespace: default
      percent: 30
```



云原生应用治理平台

- 基于TAD(Tencent Application Definition)声明应用
- 支持面向应用的多集群分批灰度发布
- 多集群的应用故障自愈，应用异常诊断，应用弹性伸缩
- 多集群的流量编排调度
- 面向应用的配额管理、核算计费、成本优化视图
- 应用可观测性：容灾健康度、应用拓扑（位置拓扑、异常拓扑、利用率拓扑）

感谢倾听

了解行业动向，交流容器技术

请扫二维码关注腾讯云原生