
Architect
SACC
2022 中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2022
激发架构性能 点亮业务活力
云上会议 网络直播 | 2022年10月27-29日
ITPUB

容器云调度优化及实践

YY直播 高级SRE运维工程师 王琼

1 YY直播容器云介绍

2 服务资源智能推荐

3 基于实际负载调度

4 运行中二次调度

5 弹性调度

YY直播容器云介绍

A 自研容器云管理平台:10+ 自建集群分布在不同机房（1.20.8）

B 2000+ 节点

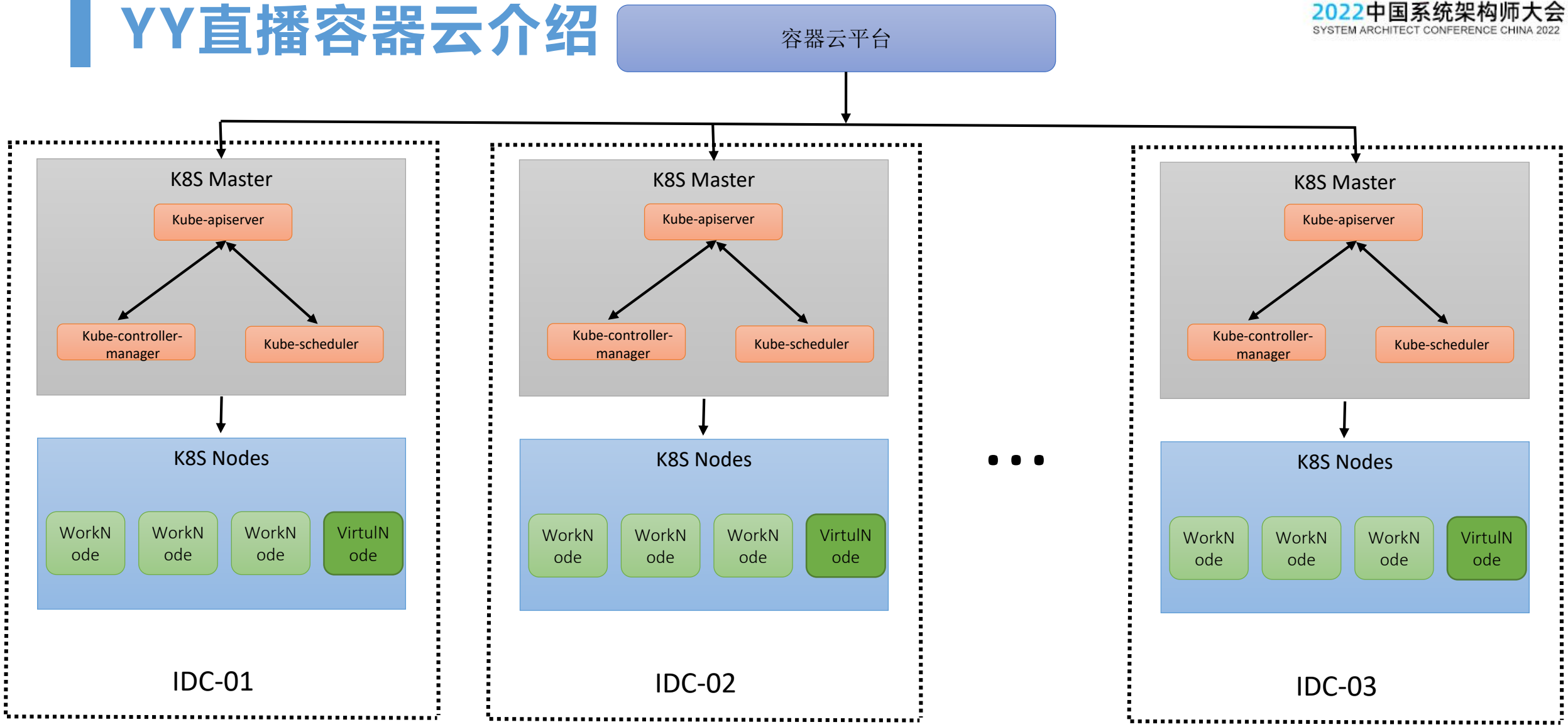
C 6w+ Pod

D 自研CNI插件，Pod IP 三层互通，支持固定内、外网 IP；IDC机房与阿里、腾讯云、百度云内网专线互通

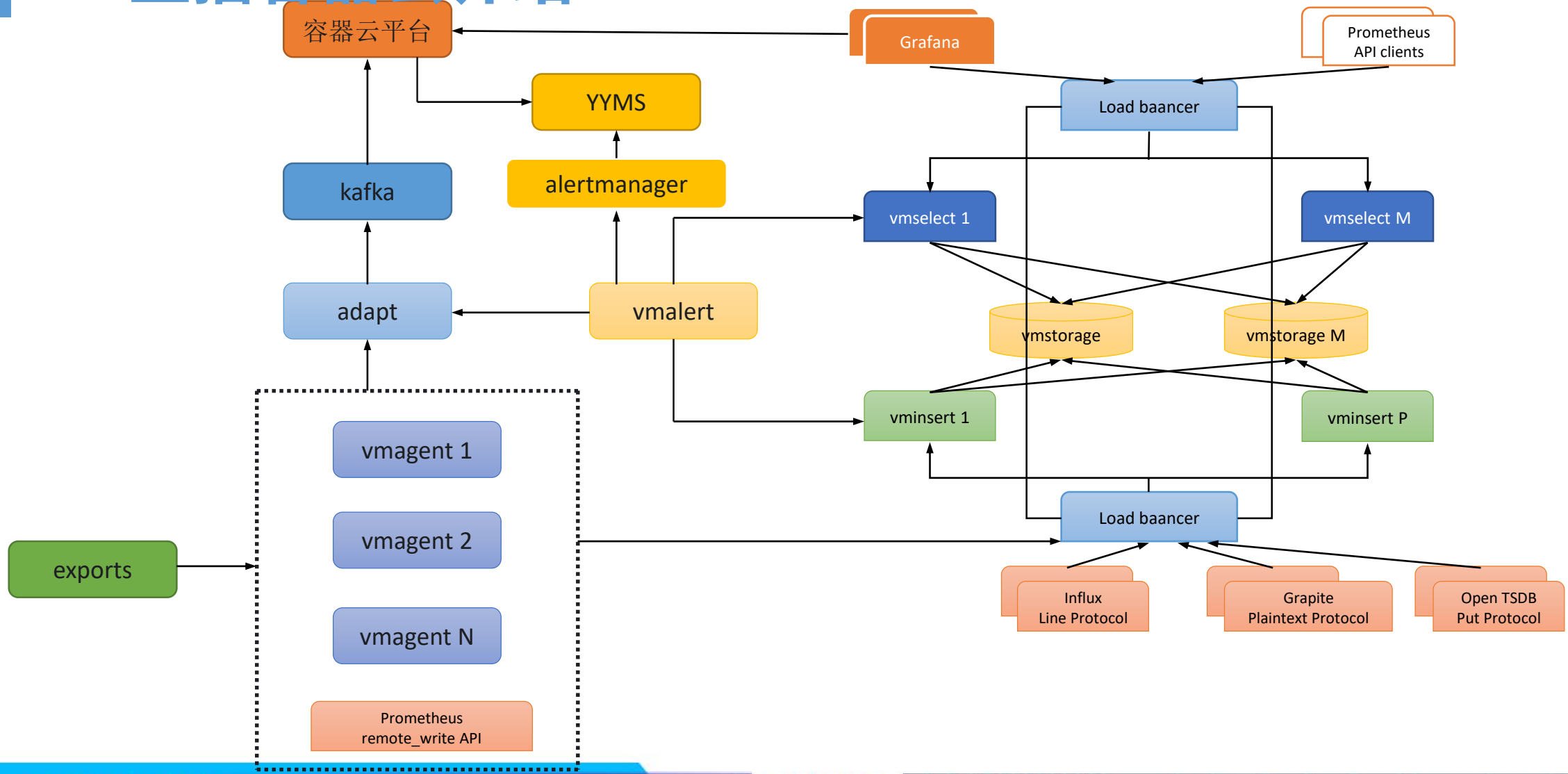
E Victoria Metrics 监控 Metrics

F 阿里 Logtail + Loki 存储业务日志

YY直播容器云介绍


容器云平台
K8S Master
Kube-apiserver
Kube-controller-manager
Kube-scheduler
K8S Nodes
WorkNode
WorkNode
WorkNode
VirtulNode
IDC-01
K8S Master
Kube-apiserver
Kube-controller-manager
Kube-scheduler
K8S Nodes
WorkNode
WorkNode
WorkNode
VirtulNode
IDC-02
K8S Master
Kube-apiserver
Kube-controller-manager
Kube-scheduler
K8S Nodes
WorkNode
WorkNode
WorkNode
VirtulNode
IDC-03

YY直播容器云介绍


容器云平台
Grafana
Prometheus
API clients
YYMS
Load baancer
kafka
alertmanager
vmselect 1
vmselect M
adapt
vmalert
vmstorage
vmstorage M
vminsert 1
vminsert P
vmagent 1
vmagent 2
vmagent N
exports
Load baancer
Prometheus
remote_write API
Influx
Line Protocol
Grapite
Plaintext Protocol
Open TSDB
Put Protocol

容器调度存在的问题

01

背景：

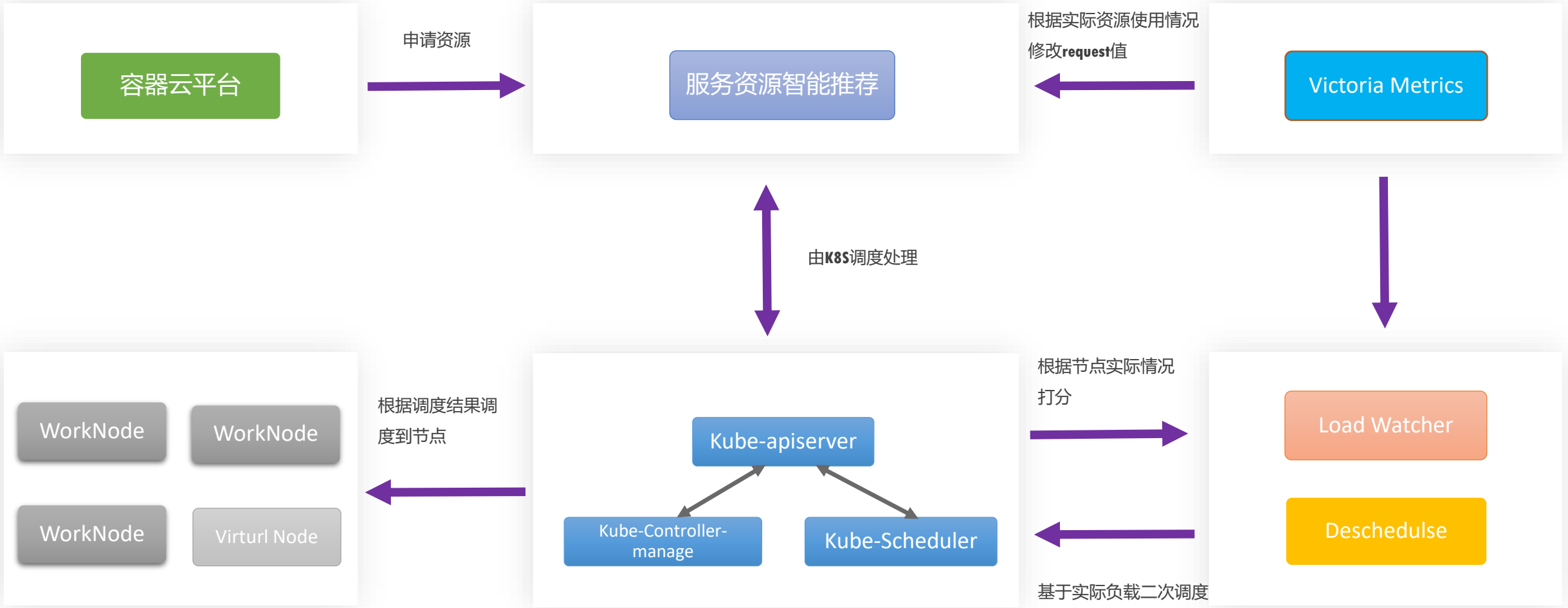
Kubernetes默认调度器策略在小规模集群下有着优异表现，但是随着业务量级的增加以及业务种类的多样性变化，默认调度策略则逐渐显露出局限性，企业在服务迁移至Kubernetes过程依然存在很多挑战

02

问题：

- 业务方不清楚服务应该申请多少资源
- 资源整体实际使用率低，低空载率高
- 集群调度不均衡，部分资源机器负载过高
- 业务突发将单个节点或者整个集群打挂

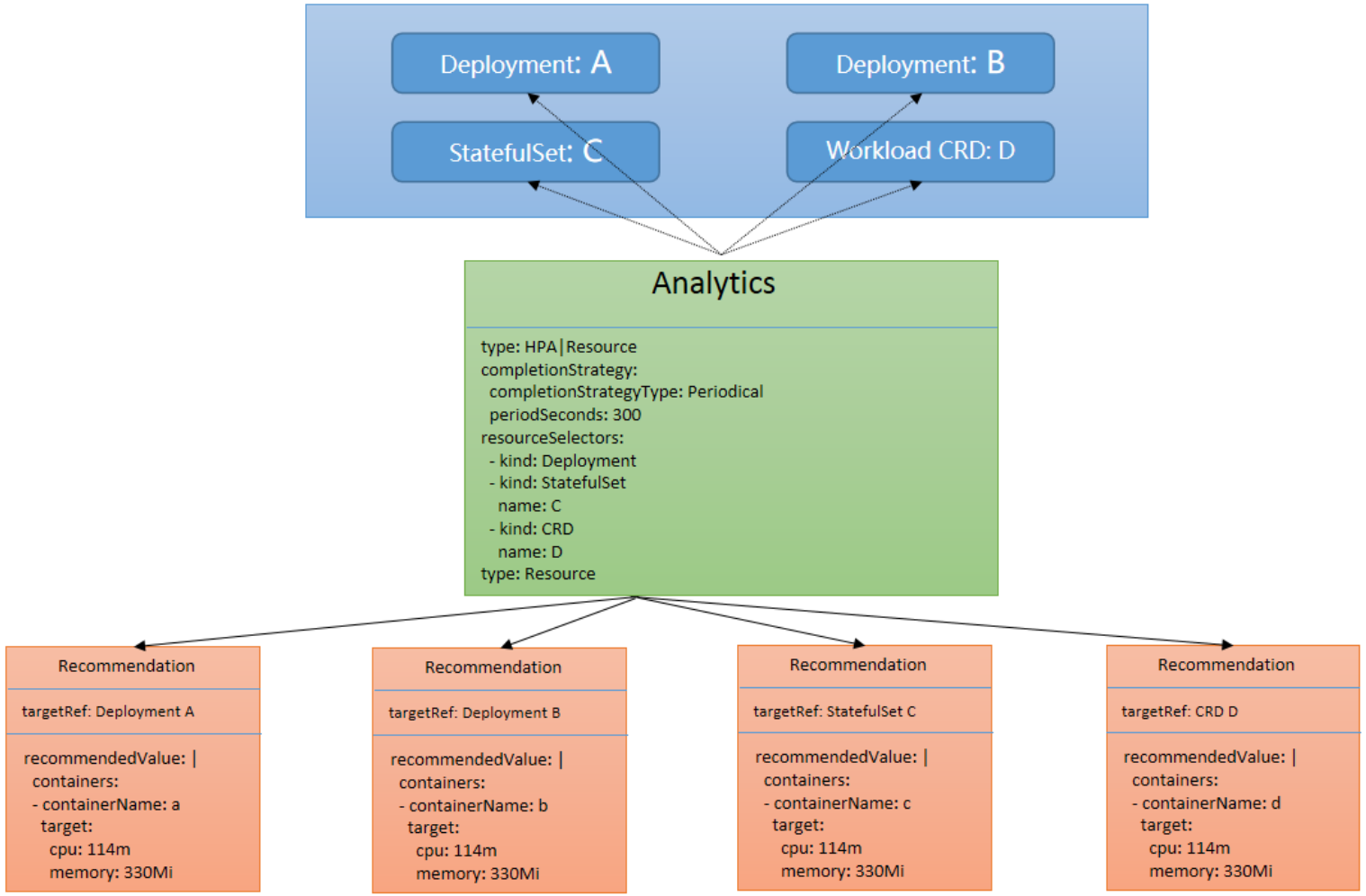
调度整体架构


容器云平台
申请资源
服务资源智能推荐
根据实际资源使用情况
修改request值
Victoria Metrics
由K8S调度处理
WorkNode
WorkNode
WorkNode
Virturl Node
根据调度结果调
度到节点
Kube-apiserver
Kube-Controller-
manage
Kube-Scheduler
根据节点实际情况
打分
Load Watcher
Deschedulse
基于实际负载二次调度

B

服务资源智能推荐

服务资源智能推荐


Deployment: A
Deployment: B
StatefulSet: C
Workload CRD: D
Analytics
type: HPA|Resource
completionStrategy:
completionStrategyType: Periodical
periodSeconds: 300
resourceSelectors:
- kind: Deployment
- kind: StatefulSet
name: C
- kind: CRD
name: D
type: Resource
Recommendation
targetRef: Deployment A
recommendedValue: |
containers:
- containerName: a
target:
cpu: 114m
memory: 330Mi
Recommendation
targetRef: Deployment B
recommendedValue: |
containers:
- containerName: b
target:
cpu: 114m
memory: 330Mi
Recommendation
targetRef: StatefulSet C
recommendedValue: |
containers:
- containerName: c
target:
cpu: 114m
memory: 330Mi
Recommendation
targetRef: CRD D
recommendedValue: |
containers:
- containerName: d
target:
cpu: 114m
memory: 330Mi

服务资源智能推荐

算法模型采用了 VPA 的滑动窗口（Moving Window）算法进行推荐

01

通过监控数据，获取 Workload 过去一周（可配置）的 CPU 和 Memory 历史用量

02

算法考虑数据的时效性，较新的数据采样点会拥有更高的权重

03

CPU 推荐值基于用户设置的目标百分位值计算，Memory 推荐值基于历史数据的最大值

服务资源智能推荐

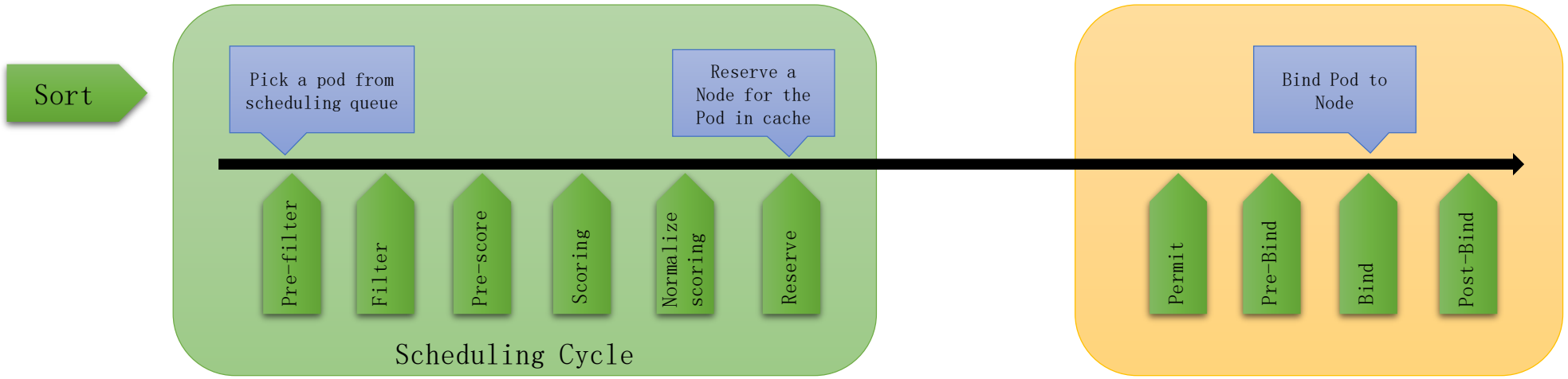

镜像配置：
资源 CPU(参考值0.919) 内存(参考值4.824G)
镜像版本*
配置名
12核
6G
t09161538.3d0e2cf3.r(m...
jvm参数
参数变量
secret挂载：
0
启用镜像托管：
本地日志保存(待下线)：
健康检查：* 存活：
None
TCP端口检查
HTTP检查
Exec
详情
就绪：
None
TCP端口检查
HTTP检查
Exec
详情

C

基于实际负载调度

基于实际负载调度

Pod Scheduling Context



Scheduling-framework

- 增强 Kubernetes 原有调度器的可扩展性
- 调度框架中可设置多个扩展点

https://github.com/kubernetes-sigs/scheduler-plugins/blob/master/pkg/trimaran/README.md

基于实际负载调度

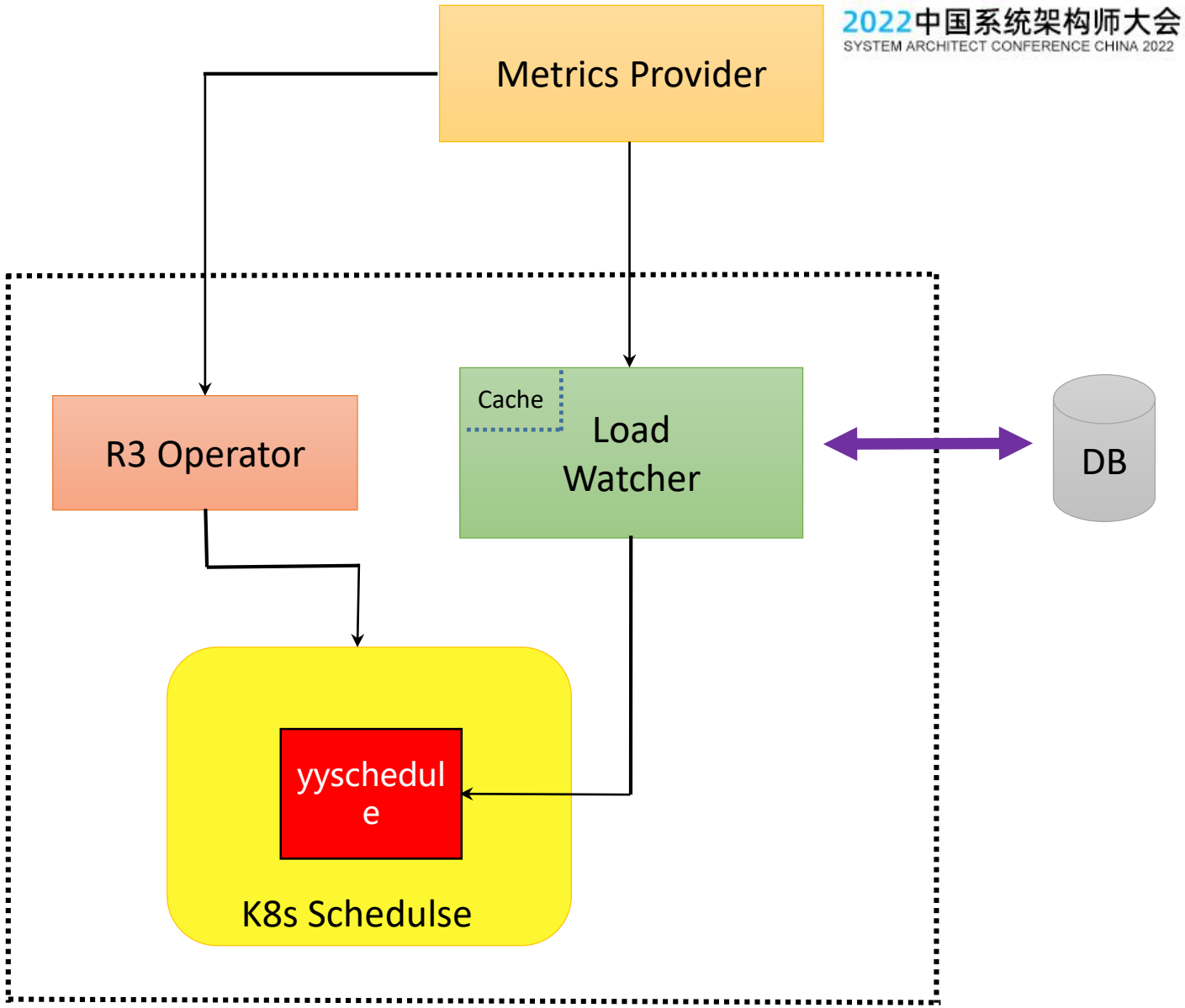
Load Watcher

解耦监控数据源、缓存监控数据、支持多种数据源

- Metrics Server
- Prometheus

YYschedule

- 基于节点实际负载去筛选节点
- 基于节点实际负载去进行打分



基于实际负载调度

预选策略-基于节点实际负载筛选

- CPU、内存超过指定值则排除该节点
- node.UsedAvg + pod.Req < Max

基于实际负载调度

优选策略-基于节点实际负载打分

```
risk = [ average + margin * stDev^{1/2} ] / 2
score = ( 1 - risk ) * maxScore
```

- average： node.UsedAvg + pod.Req
- stDev：方差
- 节点平均值及方差通过Prometheus
- CPU及内存打分低着为最后得分

基于实际负载调度

禁用默认打分插件

01 NodeResourcesBalancedAllocation

02 NodeResourcesLeastAllocated

03 ImageLocality

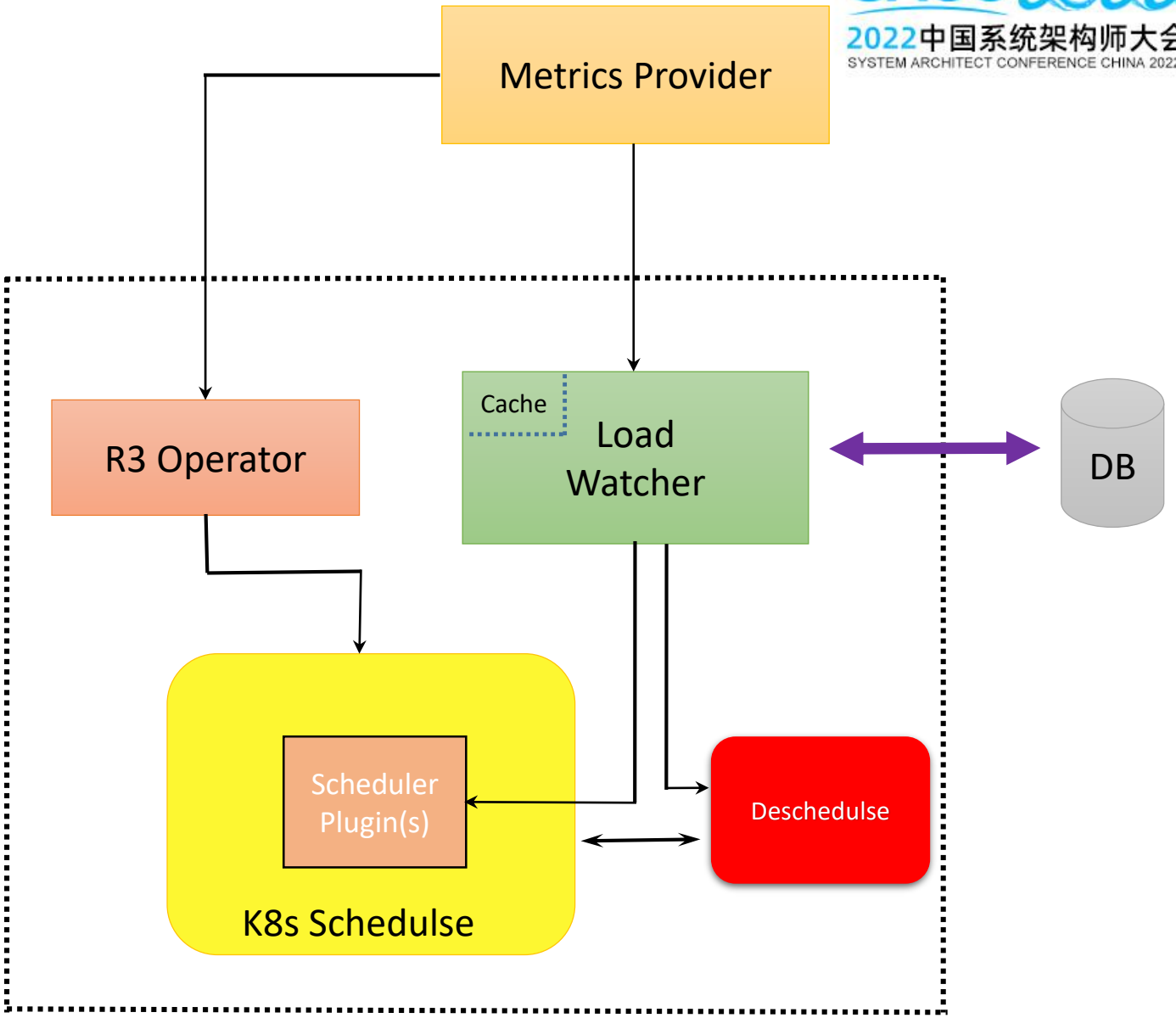
D

运行中二次调度

运行中二次调度

Deschedulse

- 避免突发业务导致单节点负载过高
- 修改 deschedulse 基于实际负载进行二次调度
- Pod运行过程中二次调度，驱逐指定的实例



运行中二次调度

1 定时获取节点实际负载

2 通过 Annotations 标识哪些实例可被驱逐

3 判断 Ready 实例数，低于指定的值不驱逐

4 判断实例启动时间，低于指定的时间不驱逐

https://github.com/kubernetes-sigs/descheduler

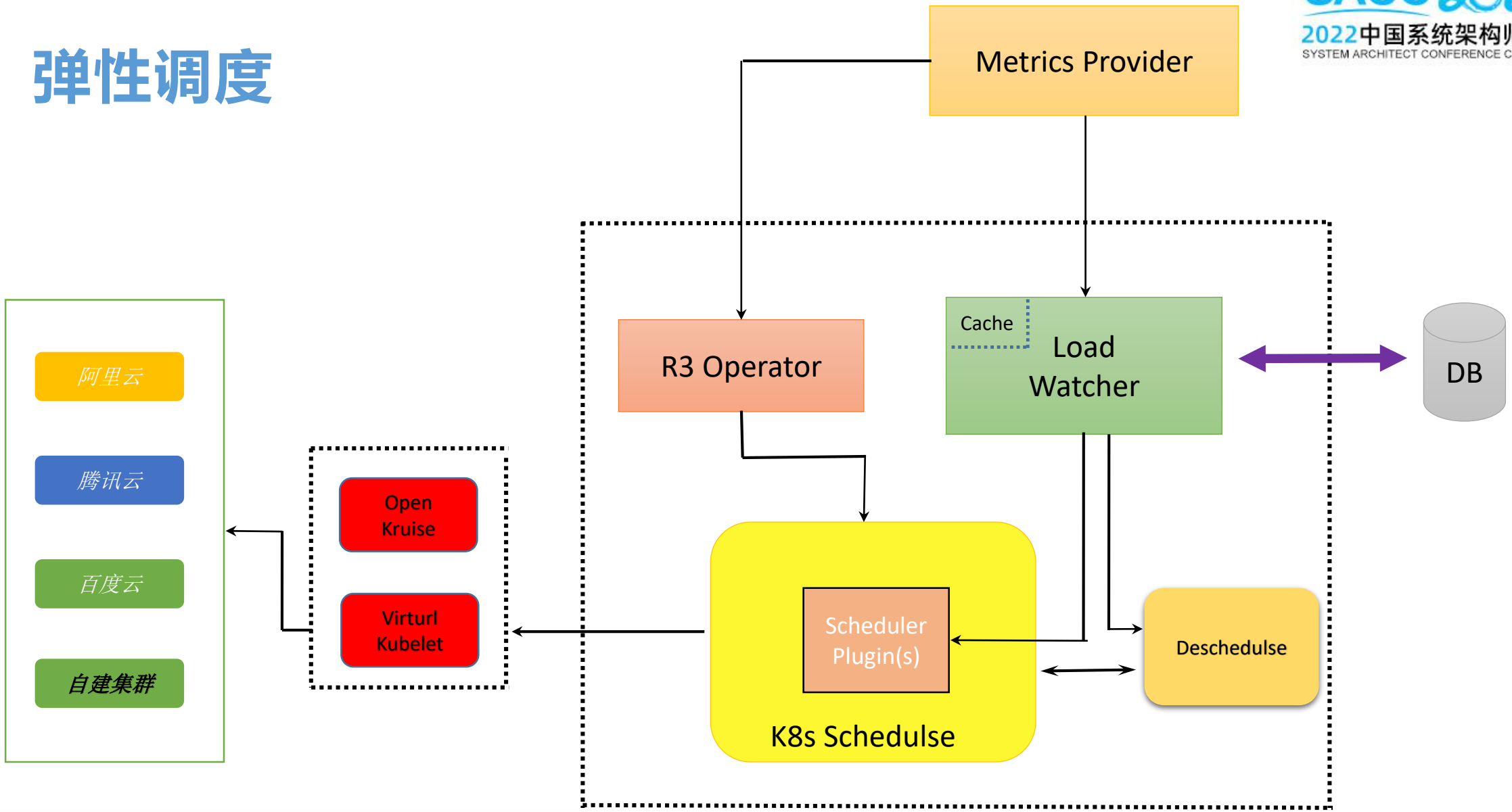
运行中二次调度


调度器名:
调度器名
反亲和性设置:
本地目录挂载:
/data/k8s/ssl/
/tmp
快捷选项
容忍污点:
可被驱逐:
就绪副本比例:
70%
副本启动时间:
10min

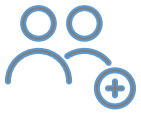
E

弹性调度

弹性调度


Metrics Provider
R3 Operator
Cache
Load Watcher
DB
阿里云
腾讯云
百度云
自建集群
Open Kruise
Virturl Kubelet
Scheduler Plugin(s)
K8s Schedulse
Deschedulse

弹性调度



Virturl Kubelet

- 依靠vk实现秒级弹性扩容
- 云厂商即用即计费，优化计算资源成本
- 同机房多集群调度

Openkruise

- WorkloadSpread 能够将 Workload 的 Pod 按一定规则分布到不同类型的 Node 节点上，赋予单一 Workload 多区域部署和弹性部署的能力
- 优先部署到自建机房，资源不足时部署到 VK
- 优先部署固定数量个 Pod 到自建机房，其余到 VK

弹性调度

启用debug： 注:debug模式适用于调式,不会执行业务的启动脚本,启用debug模式容器会定时清理掉,不适用于线上环境

启用测试： tag： 请选择

多容器间共享目录：

GPU：

弹性调度： 开启公网ip： 区域选择：

最大物理节点： -1 注:-1表示全部调度到物理节点，0表示全部调度到弹性节点

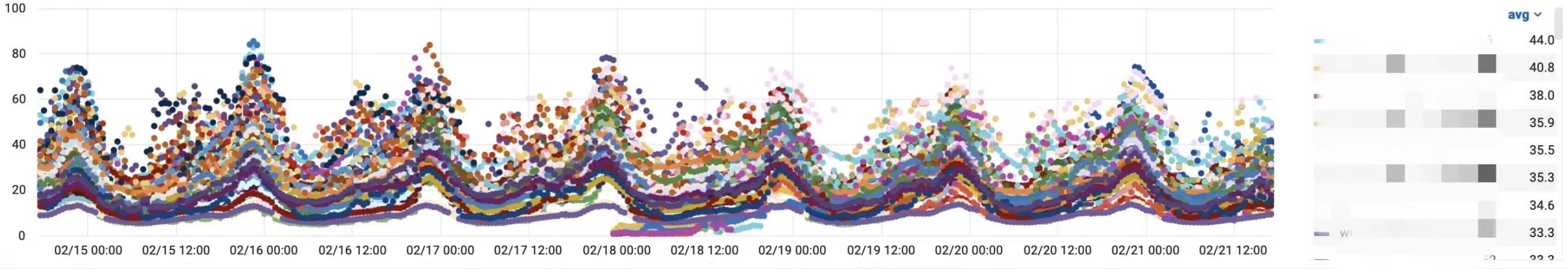
本地日志保存(NEW)：

E

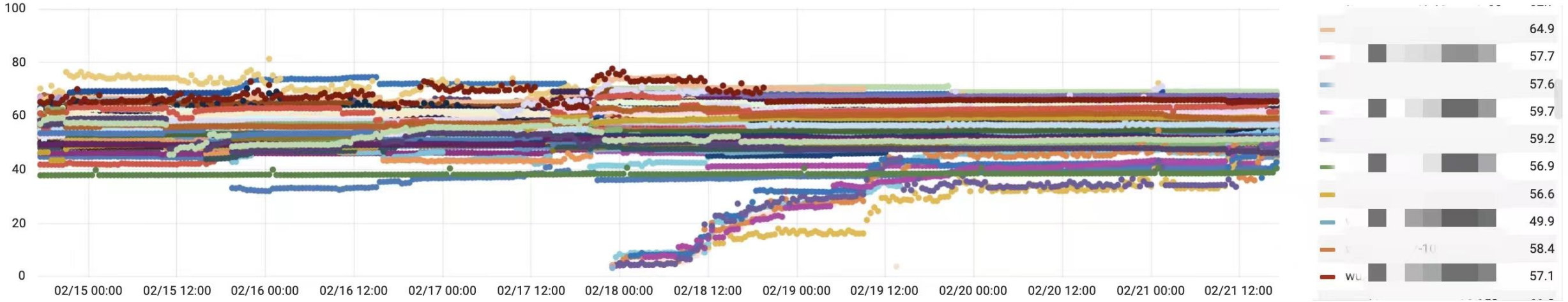
效果展示

优化前

CPU 实际使用率分布

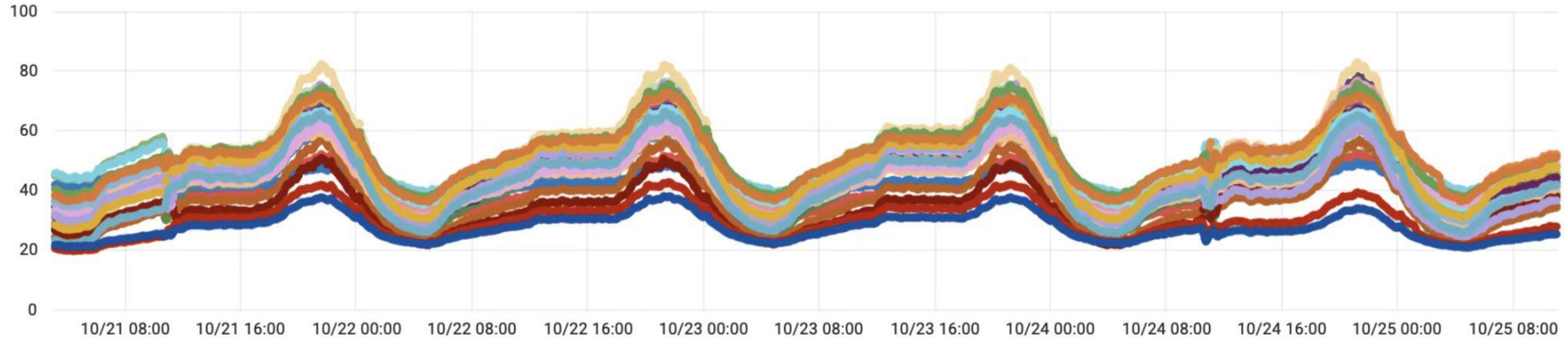


内存实际使用分布



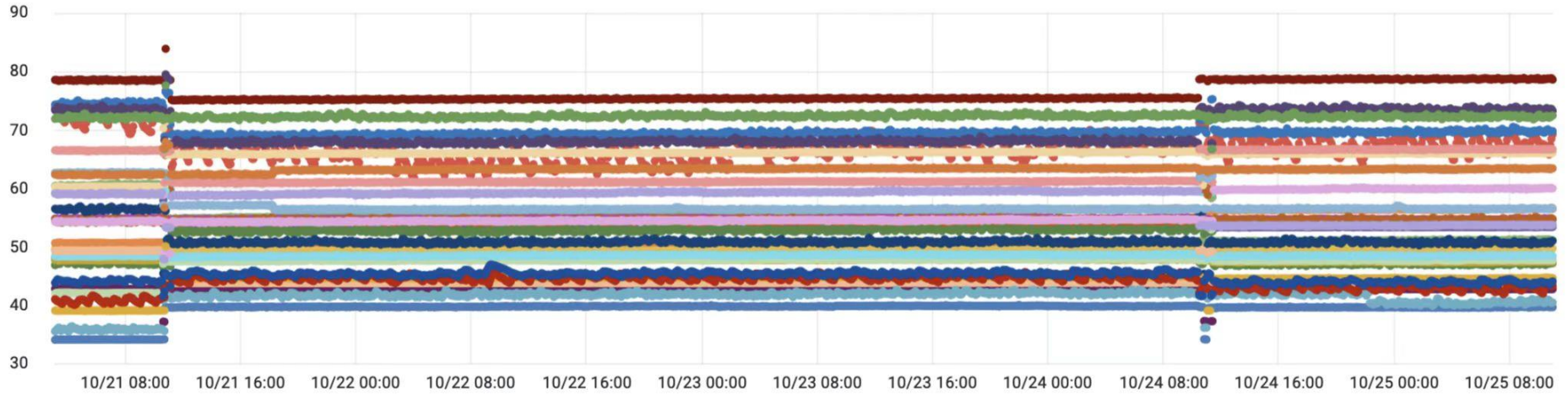
优化后

CPU 实际使用率分布



	min	max	current
	38.8	69.1	48.6
	37.8	75.9	51.3
	36.4	72.9	51.4
	35.6	49.5	41.6
	34.0	67.7	47.8
	32.9	67.8	39.2
	32.1	77.0	51.2
	31.7	83.0	51.3
W	31.1	76.1	49.3

内存实际使用分布



	min	max	current
	73.0	83.9	78.6
	67.5	79.5	73.3
	66.7	77.7	72.5
	61.7	76.7	69.3
	68.6	76.5	69.8
	60.3	71.5	66.1
	56.8	68.0	63.5
	60.8	66.9	66.7
W	54.1	65.8	54.8
WUX	56.4	62.9	56.7

THANKS