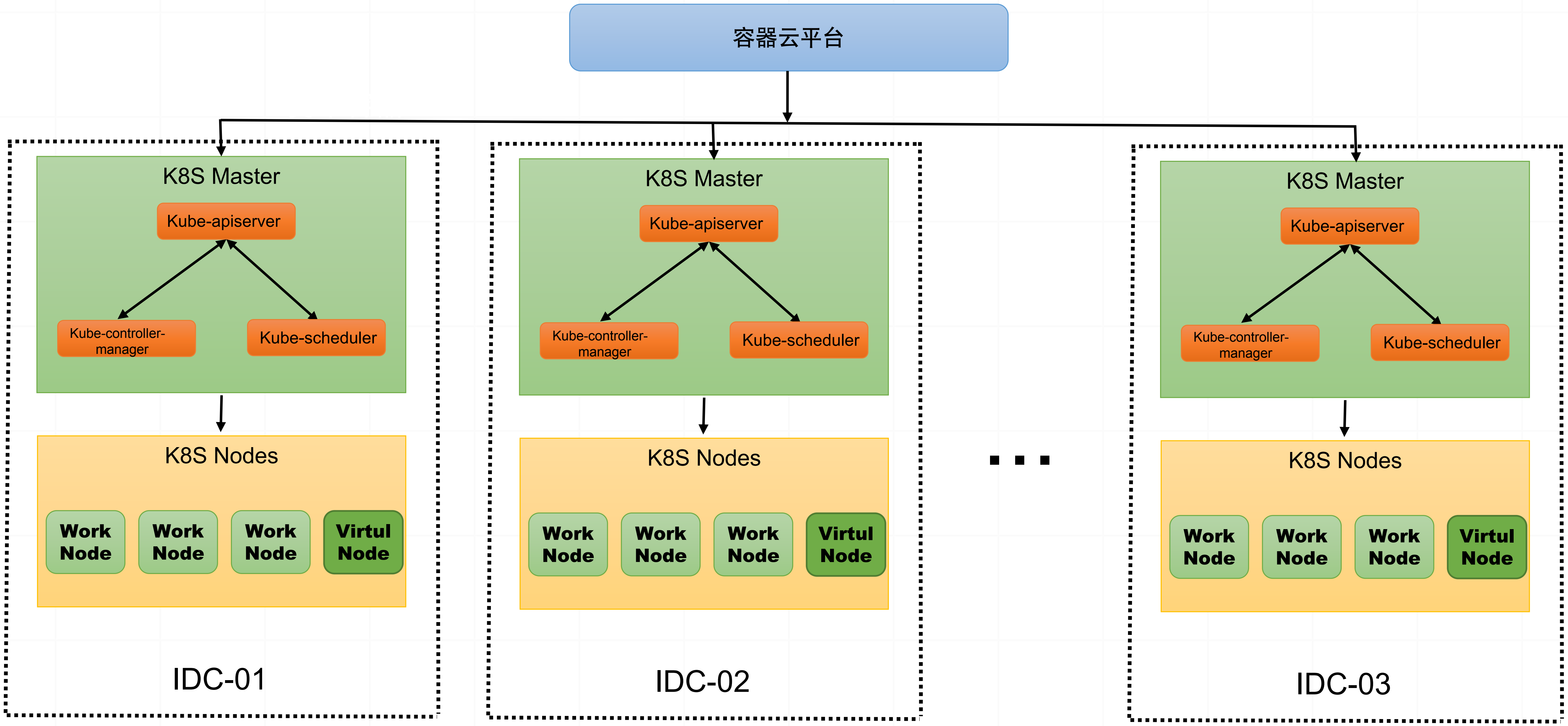
调度优化及实践

王琼 (YY 直播 高级SRE工程师)

- YY直播容器云介绍
- 服务资源智能推荐
- 基于实际负载调度
- 二次调度
- 弹性调度

YY直播容器云介绍

- 自研容器云管理平台
- 10+ 自建集群分布在不同机房（1.20.8）
- 2000+ 节点
- 6w+ Pod
- 50% 业务迁移至容器（java、go、python、c++等等）
- 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
- Victoria Metrics 监控 Metrics
- 阿里 Logtail + Loki 存储业务日志


容器云平台
K8S Master
Kube-apiserver
Kube-controller-manager
Kube-scheduler
K8S Nodes
Work Node
Work Node
Work Node
Virtul Node
IDC-01
K8S Master
Kube-apiserver
Kube-controller-manager
Kube-scheduler
K8S Nodes
Work Node
Work Node
Work Node
Virtul Node
IDC-02
...
K8S Master
Kube-apiserver
Kube-controller-manager
Kube-scheduler
K8S Nodes
Work Node
Work Node
Work Node
Virtul Node
IDC-03

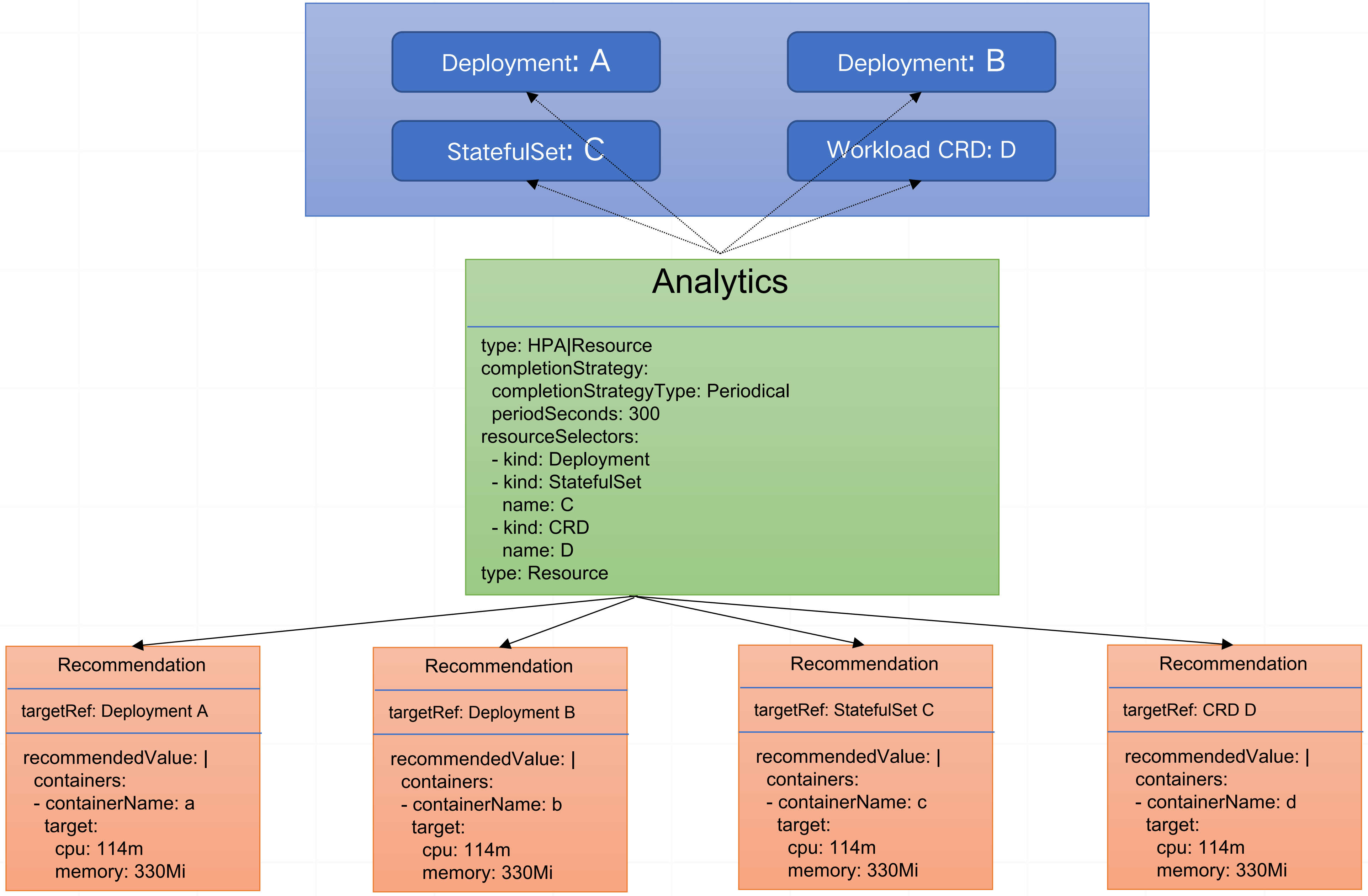
背景：

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

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

服务资源智能推荐



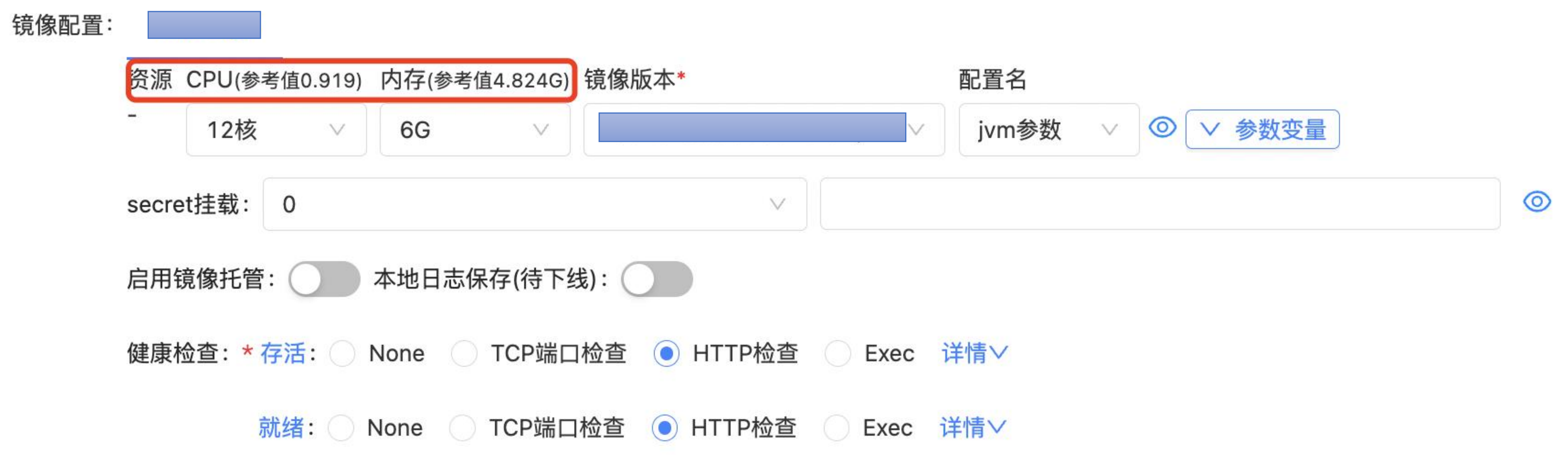
```
type: HPA|Resource
completionStrategy:
  completionStrategyType: Periodical
  periodSeconds: 300
resourceSelectors:
  - kind: Deployment
  - kind: StatefulSet
    name: C
  - kind: CRD
    name: D
type: Resource
```

```
recommendedValue: |
  containers:
  - containerName: a
    target:
      cpu: 114m
      memory: 330Mi
```

```
recommendedValue: |
  containers:
  - containerName: b
    target:
      cpu: 114m
      memory: 330Mi
```

```
recommendedValue: |
  containers:
  - containerName: c
    target:
      cpu: 114m
      memory: 330Mi
```

```
recommendedValue: |
  containers:
  - containerName: d
    target:
      cpu: 114m
      memory: 330Mi
```



镜像配置： [redacted]

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 [redacted] 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

https://github.com/gocrane/crane
https://docs.gocrane.io/dev/zh/tutorials/analytics-and-recommendation/

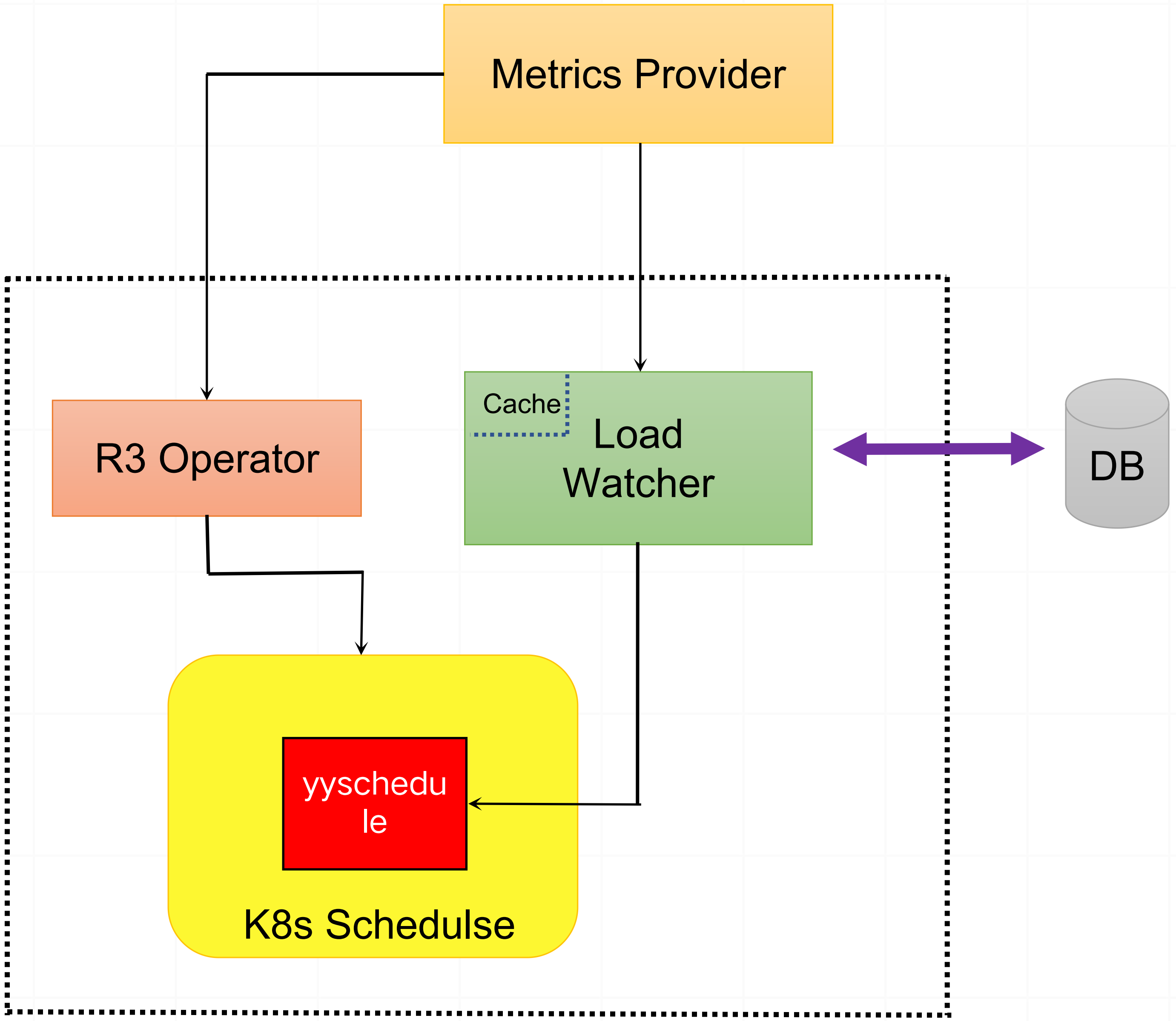
基于实际负载调度

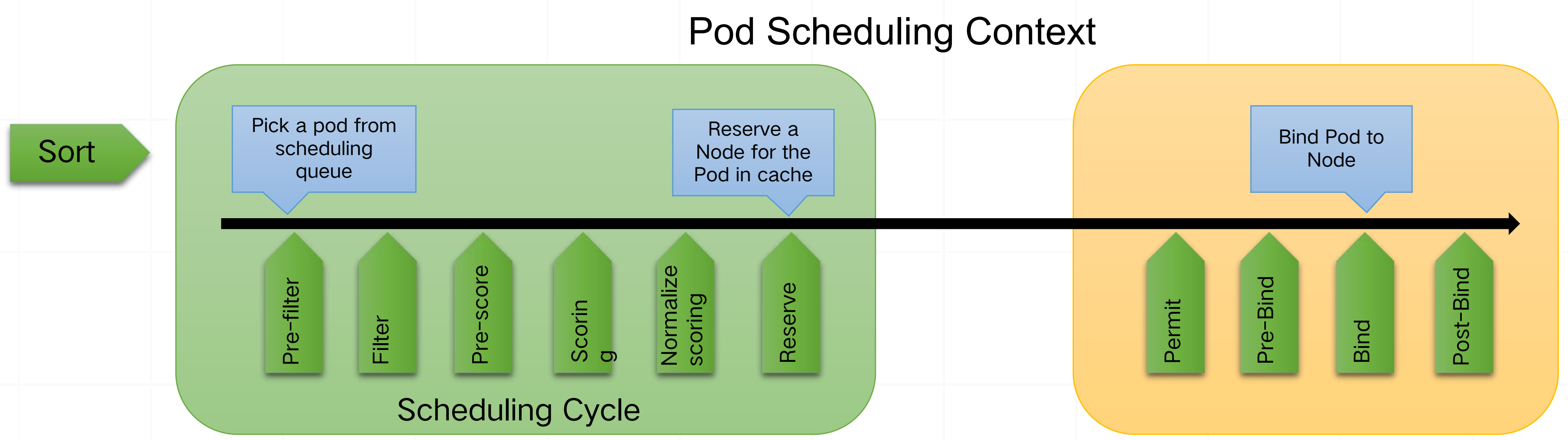
Load Watcher

解耦监控数据源、缓存监控数据、支持多种数据源
- Metrics Server
- Prometheus

YYschedule

- Kubernetes 默认基于Request进行调度
- Kubernetes原生调度策略并不考虑节点的实时利用率





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

```
type ScorePlugin interface {
        Plugin
        Score(ctx context.Context, state *CycleState, p *v1.Pod, nodeName string) (int64, *Status)
        ScoreExtensions() ScoreExtensions
}

type YYLoadBalancingstruct {
        handle       framework.Handle
        eventHandler *yyschedule.PodAssignEventHandler
        collector    *yyschedule.Collector
        args         *pluginConfig.YYLoadBalancingArgs
}

func New(obj runtime.Object, handle framework.Handle) (framework.Plugin, error) {
        ......
        return pl, nil
}

func (pl *YYLoadBalancing) Score(ctx context.Context, cycleState *framework.CycleState, pod *v1.Pod, nodeName string) (int64,
*framework.Status) {
        ......
        return score, framework.NewStatus(framework.Success, "")
}
```

```
func main() {
        command := app.NewSchedulerCommand(
                app.WithPlugin(capacityscheduling.Name, capacityscheduling.New),
                app.WithPlugin(coscheduling.Name, coscheduling.New),
                app.WithPlugin(loadvariationriskbalancing.Name, loadvariationriskbalancing.New),
                app.WithPlugin(noderesources.AllocatableName, noderesources.NewAllocatable),
                app.WithPlugin(noderesourcetopology.Name, noderesourcetopology.New),
                app.WithPlugin(preemptiontoleration.Name, preemptiontoleration.New),
                app.WithPlugin(targetloadpacking.Name, targetloadpacking.New),
                app.WithPlugin(podstate.Name, podstate.New),
                app.WithPlugin(qos.Name, qos.New),
                  app.WithPlugin(yyloadbalancing.Name, yyloadbalancing.New),
        )

        code := cli.Run(command)
        os.Exit(code)
}
```

➢ YYLoadBalancing

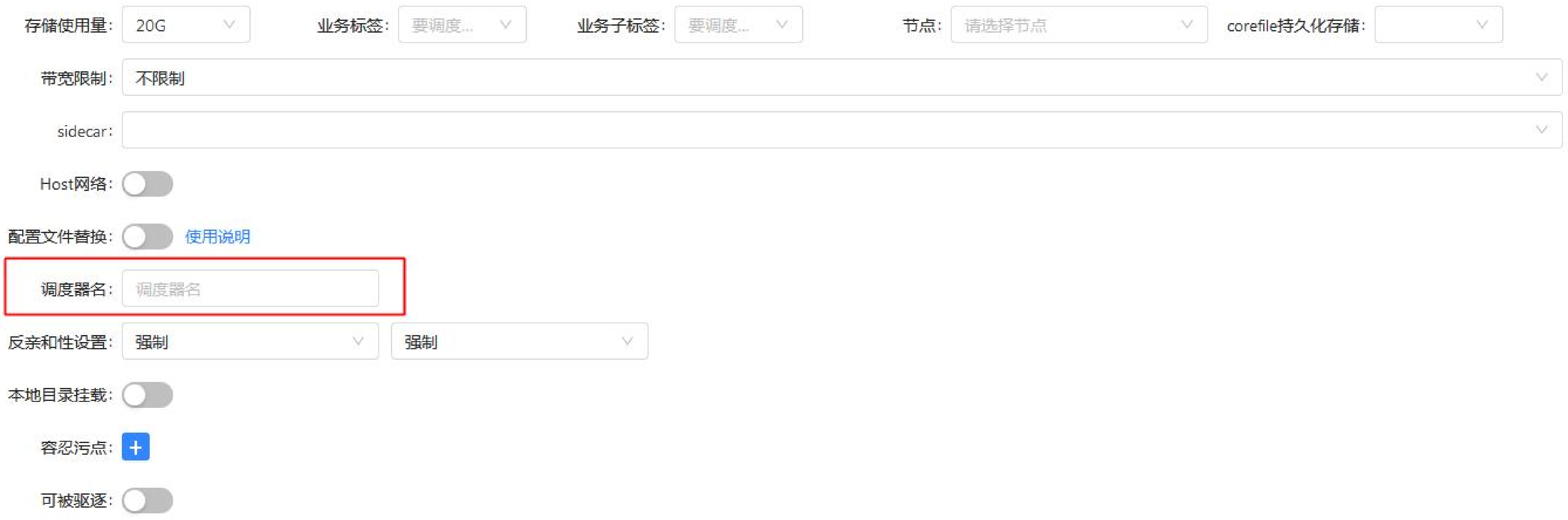
基于节点实际负载进行打分，包括 CPU、内存等

➢ 禁用默认打分插件

NodeResourcesBalancedAllocation
NodeResourcesLeastAllocated
ImageLocality

```
apiVersion: kubescheduler.config.k8s.io/v1beta1
kind: KubeSchedulerConfiguration
leaderElection:
  leaderElect: false
profiles:
- schedulerName: yyschedule
  plugins:
    score:
      disabled:
      - name: NodeResourcesBalancedAllocation
      - name: NodeResourcesLeastAllocated
      - name: ImageLocality
      enabled:
      - name: YYLoadBalancing
  pluginConfig:
  - name: YYLoadBalancing
    args:
      watcherAddress: http://192.168.0.1:2020
      safeVarianceMargin: 1
      safeVarianceSensitivity: 2
```

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



存储使用量：20G
业务标签：要调度...
业务子标签：要调度...
节点：请选择节点
corefile持久化存储：
带宽限制：不限制
sidecar：
Host网络：
配置文件替换： 使用说明
调度器名：调度器名
反亲和性设置：强制 强制
本地目录挂载：
容忍污点：+
可被驱逐：

基于业务画像自动计算业务类型

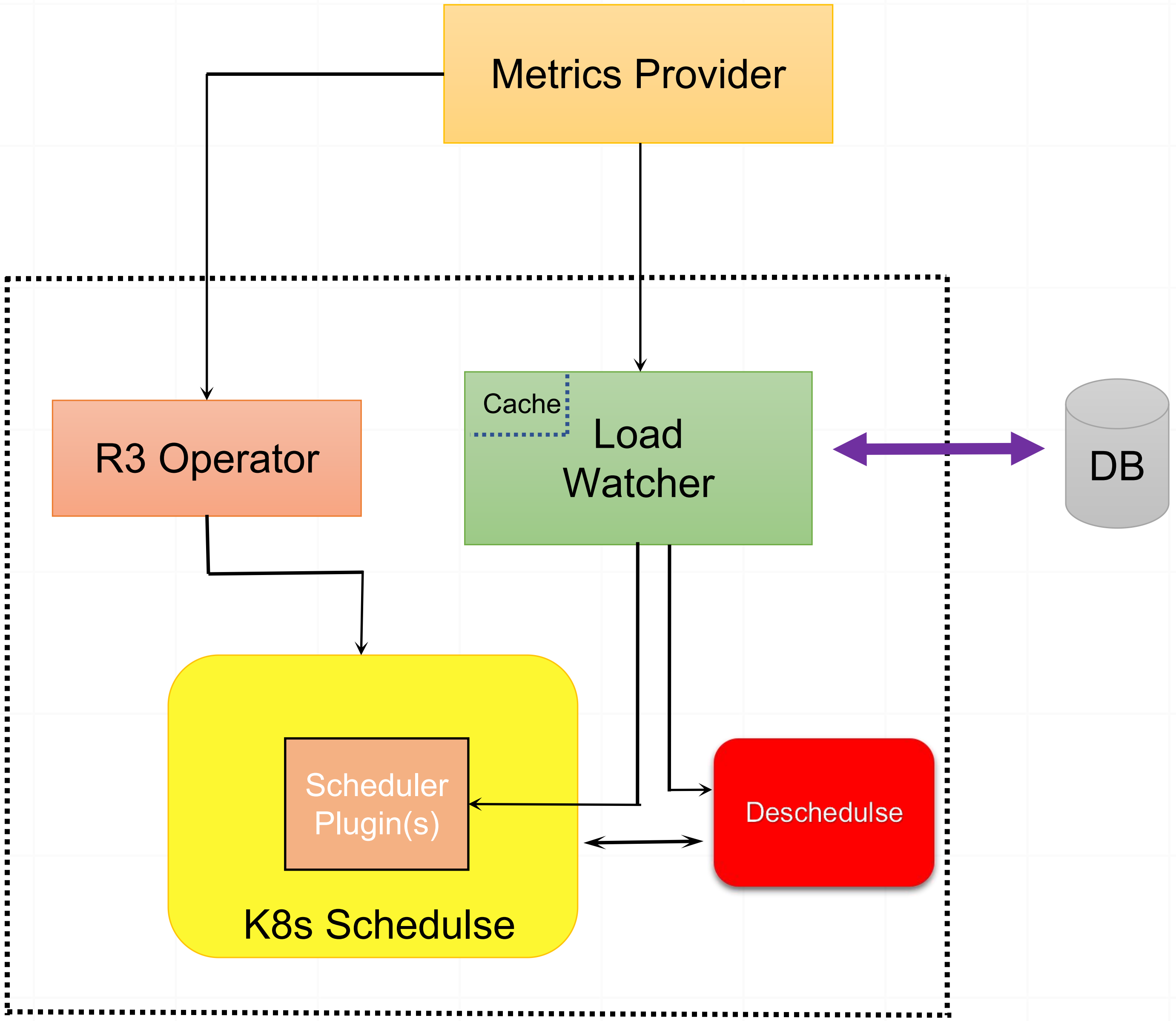
- 计算型
- 存储型
- 网络型
- IO型

基于不通的业务类型使用不同功能的调度插件

二次调度

Deschedulse

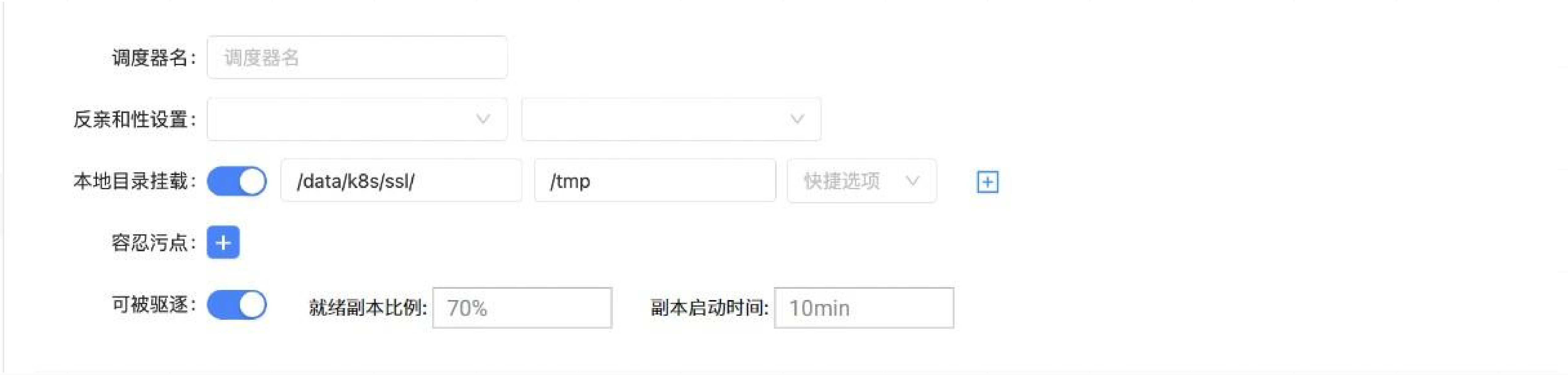
- ➢ 避免突发业务导致单节点负载过高
- ➢ 修改 deschedulse 基于实际负载进行二次调度
- ➢ Pod运行过程中二次调度，驱逐指定的实例



- 定时获取节点实际负载
- 通过 Annotations 标识哪些实例可被驱逐
- 判断 Ready 实例数，低于指定的值不驱逐
- 判断实例启动时间，低于指定的时间不驱逐

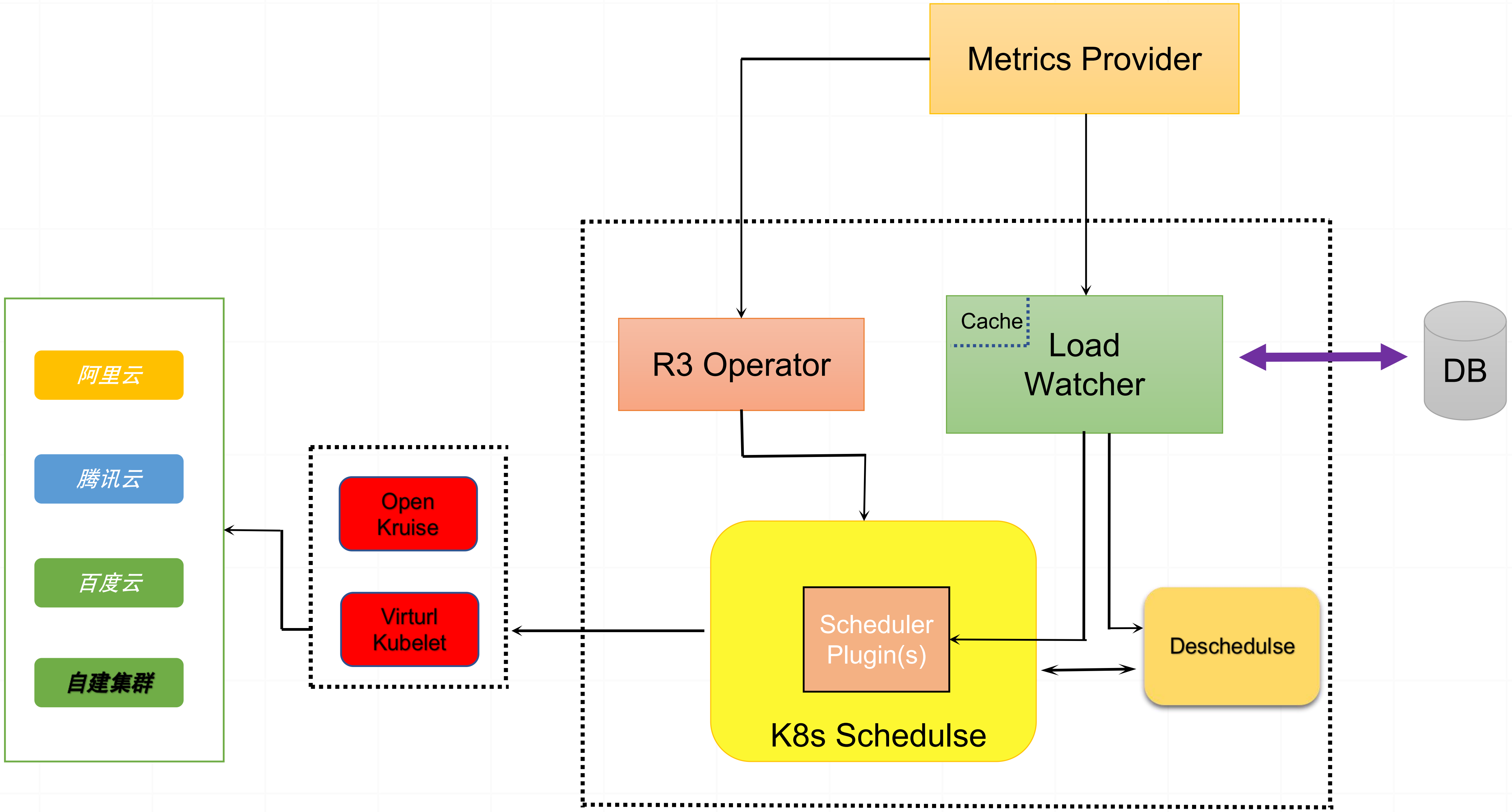
```
policy.yaml: |
  apiVersion: "descheduler/v1alpha1"
  nodeSelector: "biz.type=common"
  maxNoOfPodsToEvictPerNode: 3
  kind: "DeschedulerPolicy"
  strategies:
    "RemoveDuplicates":
       enabled: false
    "RemovePodsViolatingInterPodAntiAffinity":
       enabled: false
    "LowNodeUtilization":
       enabled: true
       params:
         nodeResourceUtilizationThresholds:
           thresholds:
             "cpu" : 40
             "memory": 70
             "pods": 100
           targetThresholds:
             "cpu" : 80
             "memory": 85
             "pods": 100
```

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



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		

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

- Virturl Kubelet

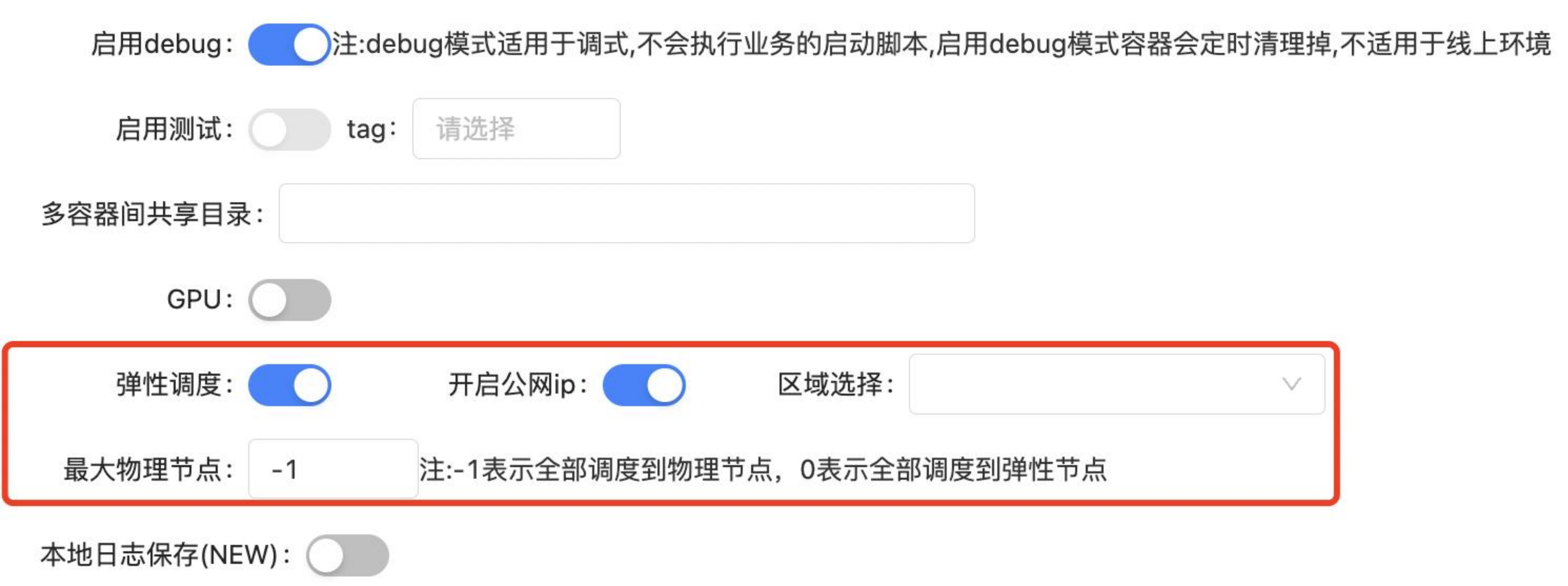
- 依靠vk实现秒级弹性扩容
- 云厂商即用即计费，优化计算资源成本
- 同机房多集群调度

- 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

```
apiVersion: apps.kruise.io/v1alpha1
kind: WorkloadSpread
metadata:
  name: workloadspread-demo
spec:
  targetRef:
    apiVersion: apps/v1 | apps.kruise.io/v1alpha1
    kind: Deployment | CloneSet
    name: workload-xxx
  subsets:
    - name: subset-a
      requiredNodeSelectorTerm:
        matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
              - common
      maxReplicas: 3
    - name: subset-b
      requiredNodeSelectorTerm:
        matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
              - vk
```



启用debug： [✓ 开] 注:debug模式适用于调试,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： [关] tag： 请选择

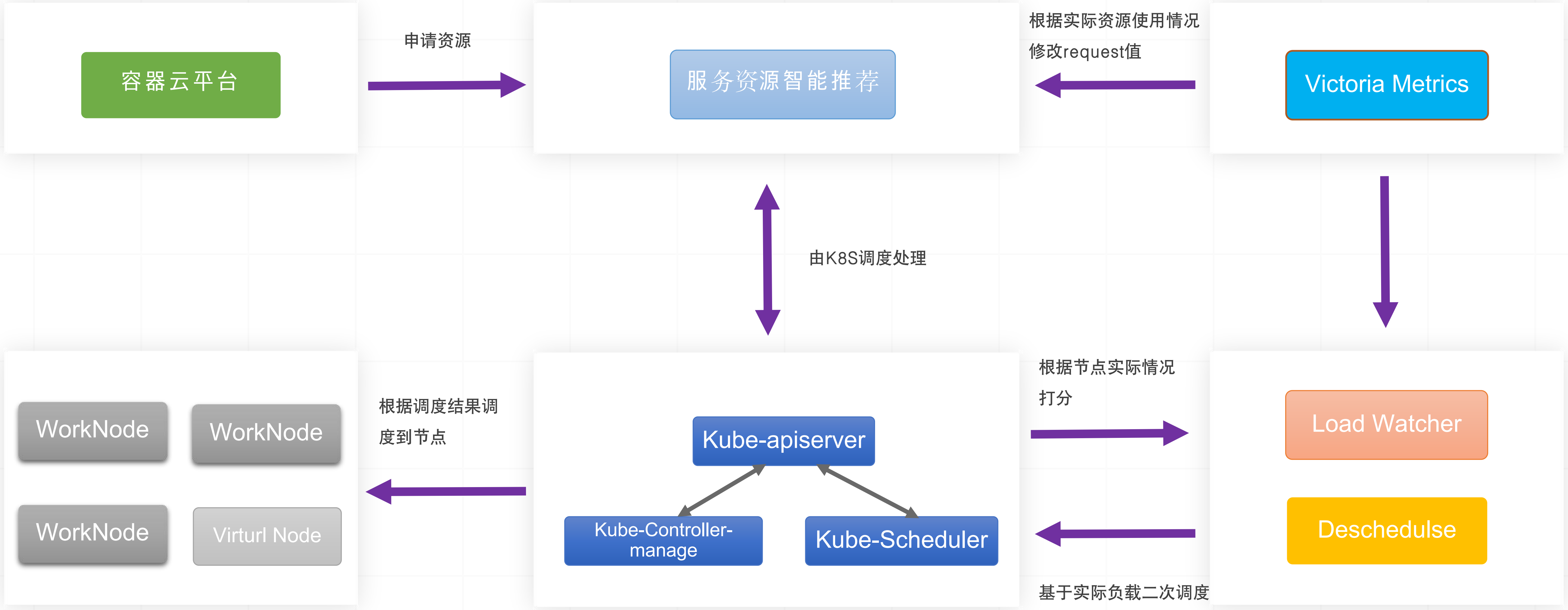
多容器间共享目录：

GPU： [关]

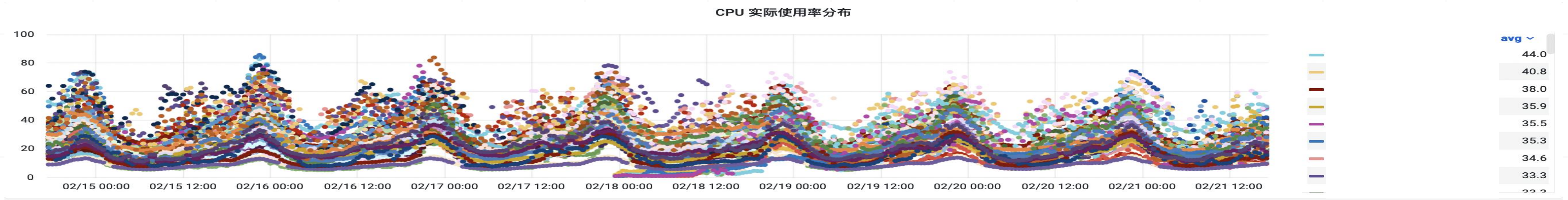
弹性调度： [✓ 开] 开启公网ip： [✓ 开] 区域选择：

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

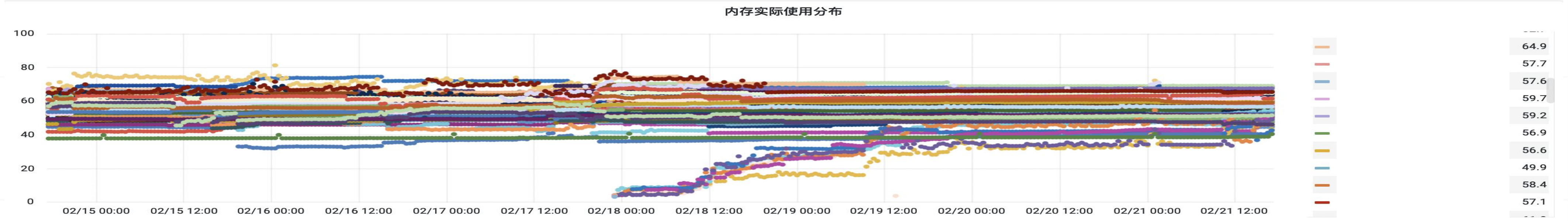
本地日志保存(NEW)： [关]


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
基于实际负载二次调度
Load Watcher
Deschedulse

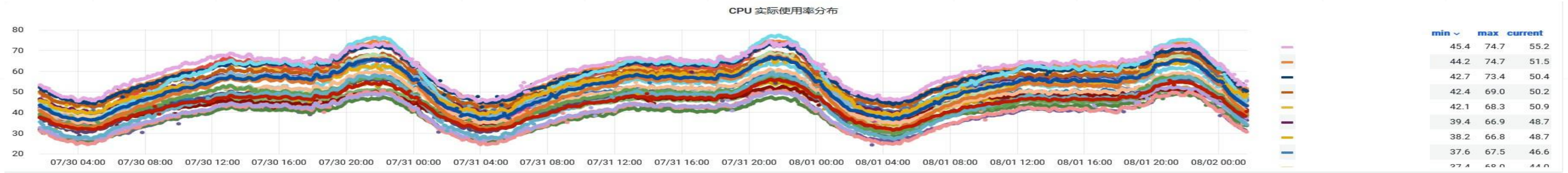
效果展示



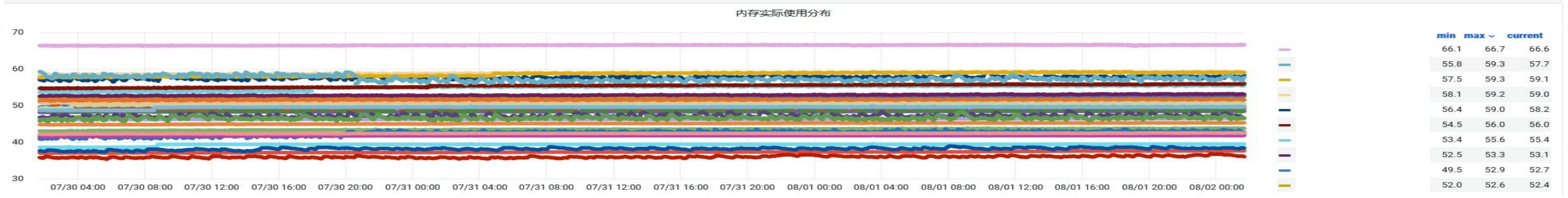
	avg
🟦	44.0
🟨	40.8
🟥	38.0
🟨	35.9
🟪	35.5
🟦	35.3
🟥	34.6
🟪	33.3



🟧	64.9
🟥	57.7
🟦	57.6
🟪	59.7
🟪	59.2
🟩	56.9
🟨	56.6
🟦	49.9
🟧	58.4
🟥	57.1



	min	max	current
🟪	45.4	74.7	55.2
🟧	44.2	74.7	51.5
🟦	42.7	73.4	50.4
🟧	42.4	69.0	50.2
🟨	42.1	68.3	50.9
🟪	39.4	66.9	48.7
🟨	38.2	66.8	48.7
🟦	37.6	67.5	46.6



	min	max	current
🟪	66.1	66.7	66.6
🟦	55.8	59.3	57.7
🟨	57.5	59.3	59.1
🟧	58.1	59.2	59.0
🟦	56.4	59.0	58.2
🟥	54.5	56.0	56.0
🟦	53.4	55.6	55.4
🟪	52.5	53.3	53.1
🟦	49.5	52.9	52.7
🟨	52.0	52.6	52.4

感谢观看

云原生社区

Cloud Native Community