



Kubernetes服务质量



前言

- 本章节介绍了Kubernetes中QoS的原理与实现方式。



目标

- 学完本课程后，您将能够：
 - 描述QoS定义
 - 区分不同QoS级别
 - 创建不同QoS级别的Pod



目录

1. QoS原理与使用



Kubernetes QoS概述

- Kubernetes集群中运行大量的容器，如果容器数量过多，负载过大时，会陷入无资源可用的情况。因此需要一定的机制来确保重要的容器拥有足够的资源，甚至不惜杀死一部分容器。服务质量（QoS）就是这样一种机制，它用于确定哪些容器可以使用多少资源。



Kubernetes资源模型 (1)

- 使用 `kubectl get node k8s-node1 -o yaml` 命令可以查看节点 `k8s-node1` 的详细情况。其中如右侧所示，`allocatable` 字段指这台机器可以被容器所使用的资源量。`Capacity` 字段表示这台 `node` 的资源真实量。
- 我们主要关注其中的 `cpu` 和内存资源，可以看到，使用的节点上有 4 个 `cpu`（其实是线程）和大约 8G 的内存。

```
allocatable:
  cpu: "4"
  ephemeral-storage: "42495643169"
  hugepages-1Gi: "0"
  hugepages-2Mi: "0"
  memory: 7906792Ki
  pods: "110"
capacity:
  cpu: "4"
  ephemeral-storage: 46110724Ki
  hugepages-1Gi: "0"
  hugepages-2Mi: "0"
  memory: 8009192Ki
  pods: "110"
```



Kubernetes资源模型 (2)

- kubernetes默认带有两类基本资源
 - CPU: K8S中CPU的设置单位是“CPU个数”。一个单位的CPU资源都会被标准化为一个标准的“Kubernetes Compute Unit”，大致和x86处理器的一个单个超线程核心是相同的。

CPU资源的基本单位是millicores，因为CPU资源其实准确来讲，指的是CPU时间。所以它的基本单位为millicores，1个核等于1000 millicores。
 - Memory: 内存的限制和请求以Bytes为单位。可以用T, G, M, K等形式表示，也可用Ti, Gi, Mi, Ki等形式表示。
 - 100M内存，表示100*1000*1000 Bytes内存；
 - 100Mi内存，表示100*1024*1024 Bytes内存。
 - 100Mi这样的表示方式更接近我们通常对100兆的定义。



Kubernetes资源模型 (3)

- Kubernetes中资源可以分为两类：可压缩资源和不可压缩资源。
 - CPU是可压缩资源，如果CPU不足了，可以通过减少分配给现有业务的时间分片来腾出一部分资源，也许现有业务会变慢，但仍然能运行。能做到这个也取决于CPU无状态的特点。
 - 内存和硬盘之类的资源是不可压缩资源，一旦一块内存或硬盘空间分配给了现有业务，除非Pod结束了使命，否则这些资源不能被释放给其他Pod。因为这些空间可能是有状态的，而且内存和磁盘不具备CPU那样敏捷的切片特性。不可压缩资源的使用一旦超限，就意味着有Pod将被停止。



Kubernetes资源模型 (4)

- Kubernetes中Pod对资源的申请是以容器为最小单位进行的，针对每个容器，它都可以通过如下两个信息指定它所希望的资源量：
 - request
request指针对这种资源，这个容器希望能够保证获取到的最少的量。只有节点上的富余资源大于request值时，容器才会被调度到该节点。
 - limit
limit对于CPU，还有内存，指的是容器对这个资源使用的上限。
对CPU来说，容器使用CPU过多，内核调度器就会切换，使其使用的量不会超过limit。
对内存来说，容器使用内存超过limit，这个容器就会被OOM kill掉，从而发生容器的重启。



Kubernetes QoS模型

- Kubernetes支持用户容器通过request、limit两个字段指定自己的申请资源信息。而根据容器指定资源的不同情况，Pod也被划分为3个不同的QoS级别。分别为：
 - Guaranteed
 - Burstable
 - BestEffort
- 优先级Guaranteed > Burstable > Best-Effort



Guaranteed级别

- Pod设置满足以下条件时，K8S会将其划分到Guaranteed级别：
 - Pod里的每个容器都必须有内存 requests和limits，且数值一样。
 - Pod里的每个容器都必须有CPU requests和limits，且数值一样。
- Guaranteed level的Pod是优先级最高的，系统管理员一般对这类Pod的资源占用量比较明确。

```
apiVersion: v1
kind: Pod
metadata:
  name: gua-pod
spec:
  containers:
  - name: gua-container
    image: nginx
    resources:
      limits:
        memory: "200Mi"
        cpu: "700m"
      requests:
        memory: "200Mi"
        cpu: "700m"
```



Burstable级别

- Pod设置满足如下情况，则K8S会将其划分到Burstable级别：
 - 该Pod不满足Guaranteed级别的条件。
 - Pod里至少有一个容器设置了CPU或内存的requests。
- Burstable level的Pod优先级其次，管理员一般知道这个Pod的资源需求的最小量，但是当机器资源充足的时候，还是希望他们能够使用更多的资源，所以一般limit > request。

```
apiVersion: v1
kind: Pod
metadata:
  name: bur-pod
spec:
  containers:
  - name: bur-container
    image: nginx
    resources:
      limits:
        memory: "200Mi"
      requests:
        memory: "100Mi"
```



BestEffort级别

- 若一个Pod既没有设置requests，也没有设置limits，则K8S将其划分到BestEffort级别。
- BestEffort level的Pod优先级最低。当机器资源充足的时候，它可以充分使用，但是当机器资源被Guaranteed、Burstable的Pod所抢占的时候，它的资源也会被剥夺，被无限压缩。

```
apiVersion: v1
kind: Pod
metadata:
  name: best-pod
spec:
  containers:
  - name: best-container
    image: nginx
```



超出容器的内存限制

- 使用polinux/stress容器测试
Kubernetes在容器运行时资源使用超过限制值时的表现。如右图配置，容器的内存限额是100Mi，但运行时实际使用150M。
- 查看Pod的状态：已被OOMKilled

```
Last State:      Terminated
Reason:          OOMKilled
Exit Code:       1
```

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo
  namespace: qos-namesapce
spec:
  containers:
  - name: memory-demo-ctr
    image: polinux/stress
    resources:
      limits:
        memory: "100Mi"
      requests:
        memory: "50Mi"
    command: ["stress"]
    args: ["--vm", "1", "--vm-bytes",
"150M", "--vm-hang", "1"]
```



超出节点可用资源

- 申请 Guaranteed 类型的 Pod，内存 1000G。由于节点并没有如此大量的资源，因此 Pod 会处于 Pending 状态。
- 从该 Pod 的详细信息可以看到：

```
Warning FailedScheduling 44s (x3
over 105s) default-scheduler 0/3
nodes are available: 3 Insufficient
memory.
```

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo2
  namespace: qos-namespaces
spec:
  containers:
  - name: memory-demo2-ctr
    image: polinux/stress
    resources:
      limits:
        memory: "1000Gi"
      requests:
        memory: "1000Gi"
    command: ["stress"]
    args: ["--vm", "1", "--vm-
bytes", "150M", "--vm-hang", "1"]
```



实验任务

- 实验任务
 - 请按照实验手册2.13章节完成QoS实验内容。



本章总结

- 本章节介绍了Kubernetes中QoS的原理与实现方式。三种不同的QoS级别的定义，Kubernetes中是如何分配资源的。

The background of the slide features a blue-tinted image of several business professionals in a modern office environment. They are standing on a highly reflective floor, and their silhouettes are clearly visible. The individuals are engaged in various interactions, some holding documents or tablets. The overall aesthetic is professional and corporate.

谢谢

www.huawei.com