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

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。

- QoS（Quality of Service，服务质量）

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。

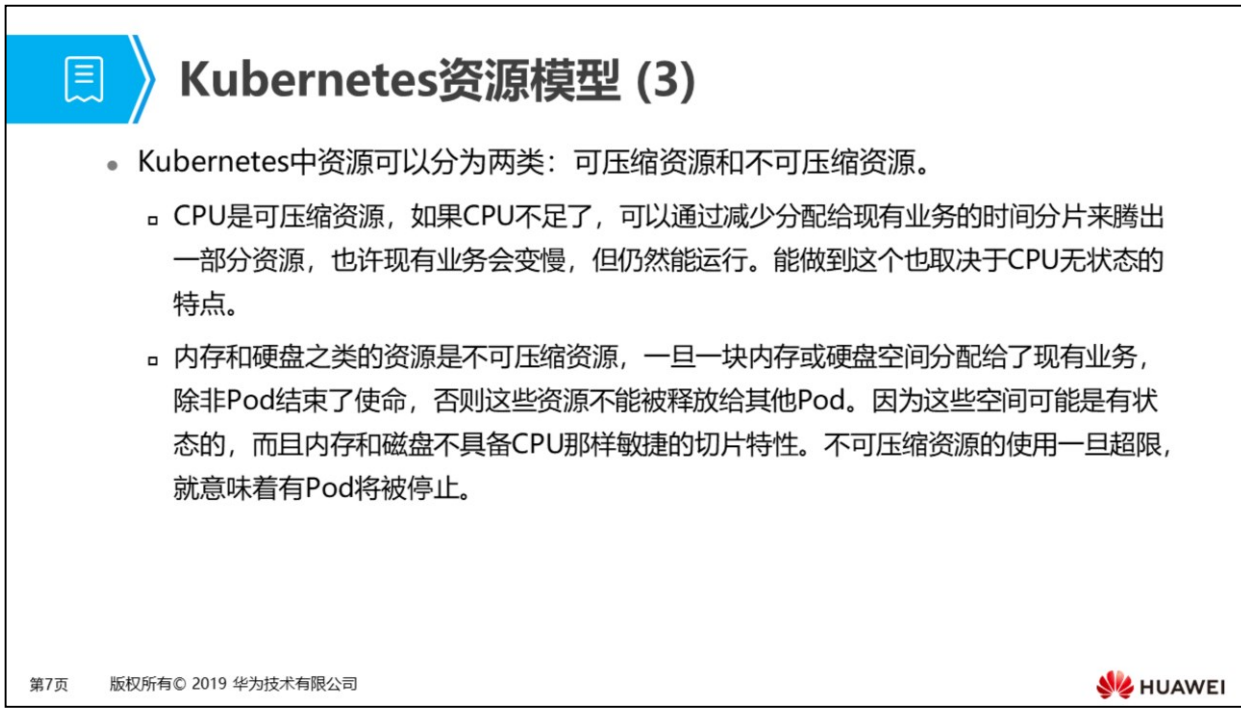
```
allocatable:
   cpu: "4"
   ephemeral-storage: "42495643169"
   hugepages-1Gi: "0"
   hugepages-2Mi: "0"
   memory: 7906792Ki
   pods: "110"
 capacity:
   cpu: "4"
   ephemeral-storage: 46110724Ki
   hugepages-1Gi: "0"
   hugepages-2Mi: "0"
   memory: 8009192Ki
   pods: "110"
```


- -o yaml命令的意思是显示node信息，输出模式（output）为yaml格式。

Kubernetes资源模型 (2)

- kubernetes默认带有两类基本资源
  - CPU：K8S中CPU的设置的单位是“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主要管理Pod，而Pod中最重要的是计算资源。所以大多资源管理都是基于CPU和内存的。
- MiB=mebibyte；1Mi=1024*1024；
- MB=megabyte；1M=1000*1000；



- 内存和硬盘在虚拟化等场景下是可压缩的，但在目前版本的Kubernetes中不可压缩。

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。

- OOM：out of memory
- Kubernetes的requests+limits的做法，其思路是：用户在提交Pod时，可以声明一个相对较小的requests值供调度器使用，而Kubernetes真正设置给容器Cgroups的，则是相对较大的limits值。

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

- 优先级低的Pod会在资源不足时优先被杀死。
- K8S QoS划分的主要应用场景，是当宿主机资源紧张的时候，kubelet对Pod进行Eviction（即资源回收）时需要用到的。

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"
```

  HUAWEI

- Kubernetes 中，不同的requests和limits的设置方式，会将这个Pod划分到不同的QoS级别当中。

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"
```

  HUAWEI

- Burstable：爆发
  - 当有资源的时候，pod运行于最佳性能，当资源不足时，pod运行在最低限度。

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
```


- 也被称为“尽力型”Pod，能用多少则用多少。

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-namespace
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"]
```


- 使用不可伸缩资源（内存）超过限度触发OOMKilled，如果使用可伸缩资源超限，不会杀死Pod。

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-namespace
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"]
```


- 出现该问题是由于schedule阶段就已经失败了，无法进入创建Pod步骤。

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。


谢 谢
www.huawei.com