Kubernetes资源调度

版权所有© 2019 华为技术有限公司

前言

- 本章主要讲述Kubernetes的资源管理， Kubernetes调度机制，常用调度策略，调度优先级和抢占机制。

版权所有© 2019 华为技术有限公司

目标

- 学完本课程后，您将能够：
  - Kubernetes资源管理和Eviction
  - 描述Kubernetes资源调度过程
  - 列举常用的Kubernetes资源调度策略
  - 描述Kubernetes调度优先级和抢占机制

版权所有© 2019 华为技术有限公司

目录

版权所有© 2019 华为技术有限公司

Kubernetes资源管理 - 节点资源

- Kubernetes集群中的每一台计算节点都包含一定的资源，通过`kubectl get node xxx -o yaml`得到节点有关资源的描述：

```
status:
  addresses:
  - address: 192.168.137.13
    type: InternalIP
  - address: worker01
    type: Hostname
  allocatable:
    cpu: "4"
    ephemeral-storage: "33919543240"
    hugepages-2Mi: "0"
    memory: 7906436Ki
    pods: "110"
  capacity:
    cpu: "4"
    ephemeral-storage: 36805060Ki
    hugepages-2Mi: "0"
    memory: 8008836Ki
    pods: "110"
```

思考：Capacity资源数据是从何处获取的?

版权所有© 2019 华为技术有限公司

HUAWEI

- capacity：该Node的资源真实量。比如这台机器是4核，8G内存，那么在capacity一栏就会显示CPU资源有4核，内存资源有8G（内存资源单位为Ki）。
- allocatable：指的是当前这台机器可以被容器使用的资源量，即排除了该节点上已经被占用的资源后剩余的资源。由图中可见allocatable是小于capacity的。

Kubernetes资源管理 - capacity

- capacity资源的详细数据从哪里获取?
  - 由于capacity反映的是这台机器的资源真实数量，所以确认这个信息的任务理所应当交给运行在每台机器上面的kubelet上。
  - kubelet目前把Cadvisor的大量代码直接作为vendor引入。在kubelet启动后，这个内部的Cadvisor子模块也会相应启动，并且获取这台机器上面的各种信息。其中就包括了有关这台机器的资源信息，而这个信息也自然作为这台机器的真实资源信息，通过kubelet再上报给APIserver。

版权所有© 2019 华为技术有限公司

- Cadvisor是google开源的一个容器监控项目，可以对节点机器上的资源及容器进行实时监控和性能数据采集，包括CPU使用情况、内存使用情况、网络吞吐量及文件系统的使用情况。Kubernetes项目已经将Cadvisor作为子模块内嵌入了kubelet中，当kubelet启动时，Cadvisor也随之启动。

Kubernetes资源管理 - Eviction (1)

- 在Kubernetes集群中的每一个节点上，除了用户容器还会运行很多其他的重要组件，他们并不是以容器的方式来运行的，这些组件的类型主要分为两个大类：kubernetes daemon和system daemon

kubernetes daemon	• kubernetes相关的daemon程序，比如kubelet，dockerd等等。
system daemon	• 与kubernetes不相关的其他系统级daemon程序，比如sshd等等。

- 为了保证这两类组件有足够的资源运行，同时为了保证物理节点的稳定性，kubernetes引入了eviction policy特性。（当节点上的内存以及磁盘资源这两种不可压缩资源严重不足时，很有可能导致物理机自身进入一种不稳定的状态，很显然这是不能接受的）

版权所有© 2019 华为技术有限公司

- 为保证物理节点的稳定性，即保证daemon类程序的稳定运行，必须要引入一种资源保护机制，即eviction。

Kubernetes资源管理 - Eviction (2)

- 实际上，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（资源回收）时需要用到的。
- 当kubernetes所管理的不可压缩资源短缺时，就有可能触发Eviction。此类不可压缩资源包括：可用内存（memory.available），可用的宿主机磁盘（nodefs.available），容器运行镜像存储空间（imagefs.available）等。
- 在kubernetes中，设置的eviction默认阈值为：

```
memory.available<100Mi
nodefs.available<10%
nodefs.inodesFree<5%
imagefs.available<15%
```


版权所有© 2019 华为技术有限公司
HUAWEI

- 这些默认阈值在安装的时候写入到了kubelet的配置文件中，如无必要，不需要改变它们。
- 可以通过kubelet命令修改这些参数。

Kubernetes资源管理 - Eviction (3)

- 可以通过如下命令配置Eviction触发的阈值

```
kubelet --eviction-
hard=imagefs.available<10%,memory.available<500Mi,nodefs.available<5%,no
defs.inodesFree<5% --eviction-
soft=imagefs.available<30%,nodefs.available<10% --eviction-soft-grace-
period=imagefs.available=2m,nodefs.available=2m --eviction-max-pod-
grace-period=600
```

- Eviction分为以下两种模式：
  - Soft模式：允许为某一项参数设置一个“优雅时间”，这段时间是为了给Pod预留出退出时间，当达到阈值之后一定时间之后，kubelet才会启动Eviction过程。
  - Hard模式：当某一项参数达到阈值之后立刻触发，直接杀掉Pod。

版权所有© 2019 华为技术有限公司

- Kubelet计算Eviction阈值的数据来源，主要依赖于Cgroups读取到的值，以及cAdvisor监控到的数据。

Kubernetes资源管理 - Eviction (4)

- 当宿主机的Eviction阈值达到之后，就会进入MemoryPressure或DiskPressure的状态，从而避免新的Pod被调度到该节点上（通过给节点打上“污点”的方式实现）。
- 当Eviction发生的时候，kubelet具体会挑选哪些Pod进行删除操作，就要参考这些Pod的类型了：
  - 首先被删除的是Besteffort类别的Pod；
  - 其次是Burstable类别、并且发生“饥饿”的资源使用量已经超出了requests的Pod；
  - 最后才是Guaranteed类别的Pod，并且Kubernetes会保证只有当该类型的Pod使用资源量超过了其limits的限制，才会被Eviction选中；
  - 对于同类别Qos的Pod，kubernetes会根据**Pod优先级**来选择移除的顺序

版权所有© 2019 华为技术有限公司

- Eviction驱逐动作发生的时候首先会参照Qos规则，当Qos规则相同的时候，再根据pod priority来进行移除。

目录

 版权所有© 2019 华为技术有限公司

HUAWEI

Kubernetes调度器

- API Server在接受客户端提交Pod对象创建请求后，然后是通过调度器（kube-scheduler）从集群中选择一个可用的最佳节点来创建并运行Pod。而这一个创建Pod对象，在调度的过程当中有3个阶段：**节点预选、节点优选、节点选定**，从而筛选出最佳的节点。
- Kubernetes-scheduler调度流程：首先剔除掉不符合条件的节点，筛选出一些符合条件的节点；然后根据算法比较所有符合条件的节点；最后选出一个最佳节点。

版权所有© 2019 华为技术有限公司

Kubernetes调度器

- **节点预选**：在具体的调度流程中，默认调度器会首先调用一组Predicate的算法，来检查每个Node。将那些不符合条件的节点过滤，从而完成节点的预选。
- **节点优选**：然后再调用一组Priority的调度算法，来给上一步筛选出的每个Node打分，以便选出最合适运行Pod对象的节点。
- **节点选定**：最终得分最高的那个Node就是调度结果。当这类节点多于1个时，则进行随机选择。

- 当调度器对一个Pod调度成功，就会在它的spec.nodeName字段填写上调度结果的名字。

版权所有© 2019 华为技术有限公司
HUAWEI

- 第一步的预选过程中，默认调度器会首先调用一组Predicate的调度算法，来检查每个Node，第二步的优选过程是通过一组叫做Priority的调度算法进行权值的打分的。

目录

 版权所有© 2019 华为技术有限公司

HUAWEI

Predicates预选策略

- 预选策略（Predicates）实际上就是节点过滤器。例如节点标签必须能够匹配到Pod资源的标签选择器，以及Pod容器的资源请求量不能大于节点上剩余的可分配资源等等。
- 执行预选操作，调度器会逐一根据规则进行筛选，如果预选没能选定一个合适的节点，此时Pod会一直处于Pending状态，直到有一个可用节点完成调度。

版权所有© 2019 华为技术有限公司

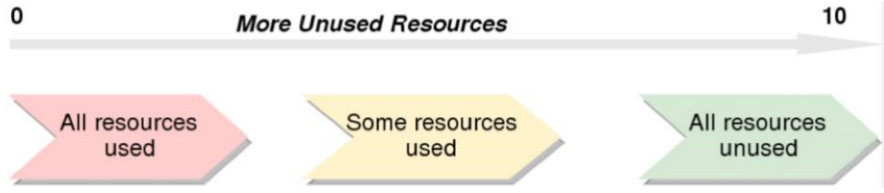
Predicates常用策略

- GeneralPredicates基础调度策略：
  - PodFitsResources：检查节点上的CPU和内存资源是否满足Pod的requests字段的资源要求。
  - PodFitsHostPorts：检查Pod申请的宿主机端口是否已经被占用。
  - PodMatchNodeSelector：检查Pod的nodeSelector或nodeAffinity指定的节点，是否与待选节点匹配。
- 宿主机相关的过滤规则
  - PodToleratesNodeTaints：如果Pod对象中定义了spec.tolerations属性，则需要检查Pod是否容忍Node Taints 。
  - CheckNodeMemoryPressure：检查Pod是否可以调度到MemoryPressure的节点上。
- Pod相关的过滤规则
  - PodAffinityPredicate：检查待调度Pod与Node上已有Pod之间的亲和(affinity)和反亲和(anti-affinity)关系。

版权所有© 2019 华为技术有限公司

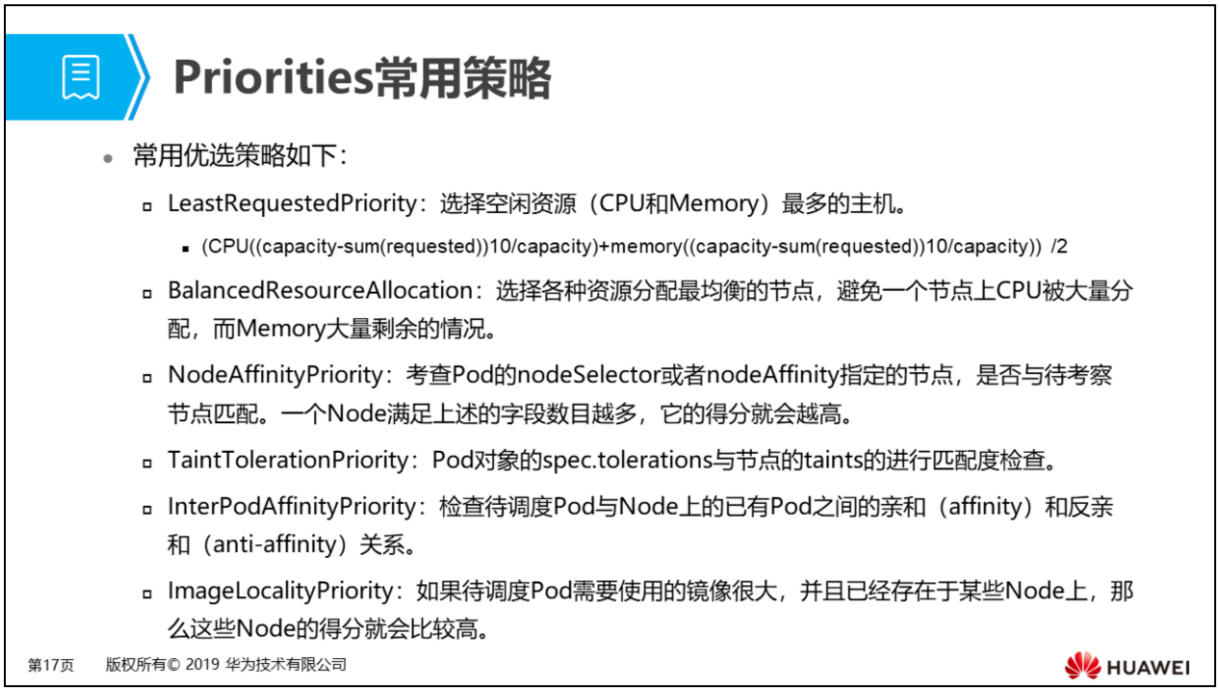
Priorities优选策略

- 预选策略筛选出一个节点列表就会进入优选阶段（Priorities）。在这个过程调度器会向每个通过预选的节点传递一系列的优选函数来计算其优先级分值，优先级分值介于0-10之间，其中0表示不适用，10表示最适合托管该Pod对象。
- 另外，调度器还支持给每个优选函数指定一个简单的值，表示权重。进行节点优先级分值计算时，它首先将每个优选函数的计算得分乘以权重，然后再将所有优选函数的得分相加，从而得出节点的最终优先级分值。权重可以让管理员定义优选函数倾向性的能力。



版权所有© 2019 华为技术有限公司
HUAWEI

- Kube-scheduler policy支持为各Priorities函数设置权重值，来控制调度器的调度行为。



- LeastRequestedPriority举例说明：例如CPU的可用资源为100，运行容器申请的资源为15，则cpu分值为8.5分，内存可用资源为100，运行容器申请资源为20，则内存分值为8分。则此评价规则在此节点的分数为[（100-15）*10/100+（100-20）*10/100]/2=8.25分。

目录

 版权所有© 2019 华为技术有限公司

HUAWEI

Kubernetes调度优先级和抢占机制

- 正常情况下，当一个Pod被调度失败时，它就会进入Pending状态，直到Pod被更新，或者集群状态发生变化，调度器才会对这个Pod进行重新调度。但某些高优先级的Pod在调度失败后，并不会Pending，而是会驱逐某个Node上一些低优先级的Pod，这样就可以保证这个高优先级的Pod调度成功。
- 优先级（Priority）和抢占（Preemption）这两种机制就是用于解决Pod调度失败之后该如何处理的问题（而非在Pod最开始的调度阶段被触发）
  - 调度器里维护着一个调度队列。当Pod拥有了优先级之后，高优先级的Pod就可能会比低优先级的Pod提前出队，从而尽早完成调度过程。这个过程，就是“优先级” 在Kubernetes里的主要体现。
  - 当一个高优先级的Pod调度失败的时候，调度器的抢占能力就会被触发。这时，调度器就会试图从当前集群里寻找一个节点，使得当这个节点上的一个或者多个低优先级Pod被删除后，待调度的高优先级Pod就可以被调度到这个节点上。这个过程，就是“抢占”在Kubernetes里的主要体现。

版权所有© 2019 华为技术有限公司

- 优先级和抢占机制并非在Pod最开始调度的阶段就被触发，而是在Pod调度失败时才会触发。

PriorityClass (1)

- Kubernetes规定，优先级是一个32bit的整数，最大值不超过1000000000（10亿），并且值越大优先级越高。
- 超过10亿的，是被Kubernetes保留下来分配给系统Pod使用的，目的是保证系统Pod不会被用户Pod抢占掉。

```
[root@master EFK]# kubectl get priorityclass
NAME                      VALUE        GLOBAL-DEFAULT   AGE
system-cluster-critical   2000000000   false            45d
system-node-critical      2000001000   false            45d
[root@master EFK]# kubectl describe priorityclass system-cluster-critical
Name:           system-cluster-critical
Value:          2000000000
GlobalDefault:  false
Description:    Used for system critical pods that must run in the cluster,
Annotations:    <none>
Events:         <none>
[root@master EFK]# kubectl describe priorityclass system-node-critical
Name:           system-node-critical
Value:          2000001000
GlobalDefault:  false
Description:    Used for system critical pods that must not be moved from th
Annotations:    <none>
Events:         <none>
```


版权所有© 2019 华为技术有限公司

PriorityClass (2)

- 而在Kubernetes中，优先级和抢占机制是在1.10版本后才逐步可用的。要使用这个机制，首先需要在Kubernetes里提交一个Priorityclass的定义：

```
apiVersion:
scheduling.k8s.io/v1beta1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000
globalDefault: false
```

- 上述yaml文件中
  - globalDefault被设置成true的话，那就意味着这个PriorityClass的值会成为系统的默认值。
  - globalDefault被设置成false表示只希望声明使用该PriorityClass的Pod拥有值为1000000的优先级，而对于没有声明Priority的Pod来说，优先级就是0。

版权所有© 2019 华为技术有限公司

PriorityClass (3)

- PriorityClass创建完成后就可以在Pod创建过程中引用该对象。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  priorityClassName: high-priority
```


版权所有© 2019 华为技术有限公司

实验&实训任务

- 实验任务
  - 请按照手册2.14章节完成Scheduler相关实验，包括：
    - 使用Nodeselector节点选择调度器
    - 使用Pod亲和性和反亲和性
    - 创建调度优先级Priority对象
- 实训任务
  - 请灵活使用本章课程及实验手册中学到的知识，按照2.14.4章节完成实训任务

版权所有© 2019 华为技术有限公司

本章总结

- 本章主要介绍了kubernetes的资源调度机制，包括：
  - Kubernetes资源管理和Eviction机制
    - Hard模式和Soft模式
  - kubernetes默认调度器scheduler
    - 节点预选、节点优选、节点选定
  - kubernetes常用调度策略
  - kubernetes调度优先级和抢占机制

版权所有© 2019 华为技术有限公司


谢 谢
www.huawei.com