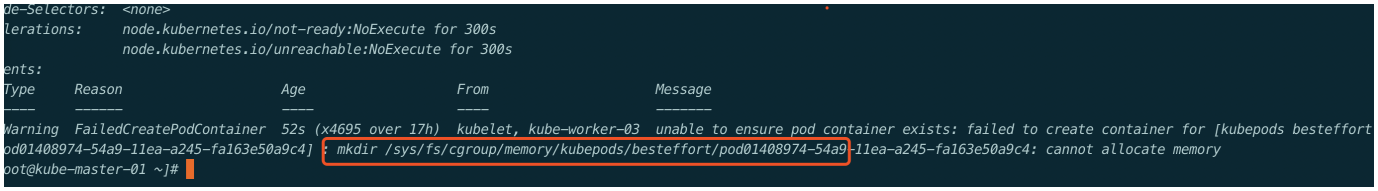
Cgoup泄漏问题修复

1. 问题

k8s集群中创建pod时报错一下内容：



这个时候，到节点上尝试创建几十个memory cgroup（以root权限执行 `for i inseq 1 20 ;do mkdir /sys/fs/cgroup/memory/${i}; done`），就会碰到失败：

```
mkdir: cannot create directory '/sys/fs/cgroup/memory/8': No space left on device
```

2. 复现问题

2.1 环境

	版本
系统	CentOS-7.6.1810
内核	3.10.0-957.27.2.el7.x86_64
docker	18.09.0
kubelet	v1.16.4

2.2 复现步骤

对于 cgroup memory 报 no space left on device ，是由于 cgroup memory 存在 64k（65535 个）大小的限制。
采用下面的测试方式，可以发现在删除 pod 后，会出现 cgroup memory 遗漏的问题。该测试方法通过留空 99 个 系统 cgroup memory 位置，来判断引起问题的原因是由于 pod container 导致的

1）填满系统 cgroup memory

```
# uname -r
3.10.0-514.10.2.el7.x86_64
# kubelet --version
Kubernetes 1.9.0
# mkdir /sys/fs/cgroup/memory/test
# for i in `seq 1 65535`;do mkdir /sys/fs/cgroup/memory/test/test-${i};
done
# cat /proc/cgroups |grep memory
memory  11      65535   1
```

把系统 cgroup memory 填到 65535 个。

2）腾空 99 个 cgroup memory

```
# for i in `seq 1 100`;do rmdir /sys/fs/cgroup/memory/test/test-${i}
2>/dev/null 1>&2; done
# mkdir /sys/fs/cgroup/memory/stress/
# for i in `seq 1 100`;do mkdir /sys/fs/cgroup/memory/test/test-${i}; done

mkdir: cannot create directory ‘/sys/fs/cgroup/memory/test/test-100’: No
space left on device
# for i in `seq 1 100`;do rmdir /sys/fs/cgroup/memory/test/test-${i}; done

# cat /proc/cgroups |grep memory
memory  11      65436   1
```

在写入第 100 个的时候提示无法写入，证明写入了 99 个。

3）创建一个 pod 到这个 node 上，查看占用的 cgroup memory 情况
每创建一个 pod ，会占用 3 个 cgroup memory 目录：

```
# ll
/sys/fs/cgroup/memory/kubepods/pod0f6c3c27-3186-11e8-afd3-fa163ecf2dce/
total 0
drwxr-xr-x 2 root root 0 Mar 27 14:14
6d1af9898c7f8d58066d0edb52e4d548d5a27e3c0d138775e9a3ddfa2b16ac2b
drwxr-xr-x 2 root root 0 Mar 27 14:14
8a65cb234767a02e130c162e8d5f4a0a92e345bfef6b4b664b39e7d035c63d1
```

这时再次创建 100 个 cgroup memory ，因为 pod 占用了 3 个，会出现 4 个无法成功：

```
# for i in `seq 1 100`;do mkdir /sys/fs/cgroup/memory/test/test-${i}; done

mkdir: cannot create directory ‘/sys/fs/cgroup/memory/test/test-97’: No
space left on device &lt;-- 3 directory used by pod
mkdir: cannot create directory ‘/sys/fs/cgroup/memory/test/test-98’: No
space left on device
mkdir: cannot create directory ‘/sys/fs/cgroup/memory/test/test-99’: No
space left on device
mkdir: cannot create directory ‘/sys/fs/cgroup/memory/test/test-100’: No
space left on device
# cat /proc/cgroups
memory  11      65439   1
```

写入到的 cgroup memory 增加到 65439 个。

4）删掉测试 pod ，看看 3 个占用的 cgroup memory 是否有释放
看到的结果：

```
# cat /proc/cgroups
memory  11      65436   1
# for i in `seq 1 100`;do mkdir /sys/fs/cgroup/memory/test/test-${i}; done

mkdir: cannot create directory ‘/sys/fs/cgroup/memory/test/test-97’: No
space left on device
mkdir: cannot create directory ‘/sys/fs/cgroup/memory/test/test-98’: No
space left on device
mkdir: cannot create directory ‘/sys/fs/cgroup/memory/test/test-99’: No
space left on device
mkdir: cannot create directory ‘/sys/fs/cgroup/memory/test/test-100’: No
space left on device
```

可以看到，虽然 cgroup memory 减少到 65536 ，似乎 3 个位置释放了。但实际上测试结果发现，并不能写入，结果还是 pod 占用时的无法写入 97-100 。
这就说明，cgroup memory 数量减少，但被 pod container 占用的空间没有释放。

反复验证后，发现随着 pod 发布和变更的增加，该问题会越来越严重，直到把整台机器的 cgroup memory 用完。

3. 问题原因

由于cgroup kernel memory 特性被激活，导致删除容器后，并未释放空间，仍有对 cgroup memory 的占用，所以导致了此问题的发生

关于 cgroup kernel memory，在 kernel-doc 中有如下描述：

```
# vim /usr/share/doc/kernel-doc-3.10.0/Documentation/cgroups/memory.txt
2.7 Kernel Memory Extension (CONFIG_MEMCG_KMEM)
With the Kernel memory extension, the Memory Controller is able to limit
the amount of kernel memory used by the system. Kernel memory is
fundamentally
different than user memory, since it can't be swapped out, which makes it
possible to DoS the system by consuming too much of this precious resource.
Kernel memory won't be accounted at all until limit on a group is set. This
allows for existing setups to continue working without disruption. The
limit
cannot be set if the cgroup have children, or if there are already tasks in
the
cgroup. Attempting to set the limit under those conditions will return
-EBUSY.
When use_hierarchy == 1 and a group is accounted, its children will
automatically be accounted regardless of their limit value.
After a group is first limited, it will be kept being accounted until it
is removed. The memory limitation itself, can of course be removed by
writing
-1 to memory.kmem.limit_in_bytes. In this case, kmem will be accounted, but
not
limited.
```

这是一个 cgroup memory 的扩展，用于限制对 kernel memory 的使用。但该特性在老于 4.0 版本中是个实验特性，若使用 docker run 运行，就会提示：

```
# docker run -d --name test001 --kernel-memory 100M
registry.vclound.com:5000/hyphenwang/sshdserver:v1
WARNING: You specified a kernel memory limit on a kernel older than 4.0.
Kernel memory limits are experimental on older kernels, it won't work as
expected and can cause your system to be unstable.
# cat
/sys/fs/cgroup/memory/docker/eceb6dfba2c64a783f33bd5e54cecb32d5e64647439b4
932468650257ea06206/memory.kmem.limit_in_bytes
104857600
```

4. 修复方案

4.1 方法一：重新编译runc、kubelet禁用kmem accounting

这里我们采用方法1:在 kubelet 和 docker 上都将 kmem account 功能关闭，步骤如下:

1 . kubelet 需要重新编译，不同的版本有不同的方式。

如果 kubelet 版本是 v1.14 及以上，则可以通过在编译 kubelet 的时候加上 Build Tags 来关闭 kmem account:

```
$ git clone --branch v1.14.1 --single-branch --depth 1
[https://github.com/kubernetes/kubernetes](https://github.com/kubernetes/k
ubernetes)
$ cd kubernetes
$ KUBE_GIT_VERSION=v1.14.1 ./build/run.sh make kubelet
GOFLAGS="-tags=nokmem"
```

kubelet v1.13 kubelet Build Tags kubelet

首先下载 Kubernetes 代码：

```
$ git clone --branch v1.12.8 --single-branch --depth 1
https://github.com/kubernetes/kubernetes
$ cd kubernetes
 kmem account
func EnableKernelMemoryAccounting(path string) error {
    return nil
}

func setKernelMemory(path string, kernelMemoryLimit int64) error {
    return nil
}
```

之后重新编译 kubelet：

```
$ KUBE_GIT_VERSION=v1.12.8 ./build/run.sh make kubelet
```

编译好的 kubelet 在 ./_output/dockerized/bin/$GOOS/$GOARCH/kubelet 中。

2 . 同时需要升级 docker-ce 到 18.09.1 以上，此版本 docker 已经将 runc 的 kmem account 功能关闭。或者手动编译runc

手动编译runc方法

a.先查看runc版本

```
[root@test-k8s-1 kubernetes]# runc -v
runc version 1.0.0-rc5
commit: 4fc53a81fb7c994640722ac585fa9ca548971871
spec: 1.0.0
```

然后查看github（https://github.com/opencontainers/runc/tree/master），1.0.0-rc5版本，发现并没更新nokemem功能，此功能经查看github，发现是从1.0.0-rc6及以上更新了此功能

Build Tag	Feature	Dependency
seccomp	Syscall filtering	libseccomp
selinux	selinux process and mount labeling	
apparmor	apparmor profile support	
ambient	ambient capability support	kernel 4.3
nokmem	disable kernel memory account	

b、下载runc源码

```
# mkdir -p /data/Documents/src/github.com/opencontainers/
# cd /data/Documents/src/github.com/opencontainers/
# git clone https://github.com/opencontainers/runc
# cd runc/
# git checkout v1.0.0-rc9  # v1.0.0-rc9 tag
```

c、编译，seccomp选项默认都会带上

```
apt-get install -y libseccomp-dev  pkg-config
make BUILDTAGS='seccomp nokmem'
```

runc可执行文件在当前目录，更多编译选项查看README.md

d、查看runc版本

```
# ./runc -v
runc version 1.0.0-rc9
commit: d736ef14f0288d6993a1845745d6756cfc9ddd5a
spec: 1.0.1-dev
```

e、替换此版本至服务器即可

3 . 最后需要重启机器。

验证方法是查看新创建的 pod 的所有 container 已关闭 kmem，如果为下面结果则已关闭：

```
$ cat
/sys/fs/cgroup/memory/kubepods/burstable/pod<pod-uid>/<container-id>/memor
y.kmem.slabinfo
cat: memory.kmem.slabinfo: Input/output error
```

4.1 方法二：内核禁用cgroup kmem(Kernel 3.10.0-1062.4.1.el7.x86_64支持)

步骤如下:

1. 因使用离线方法升级内核，所以先在https://centos.pkgs.org/7/centos-updates-x86_64/ 下载对应版本内核。

已经下载完成，并上传至172.16.56.143:/opt/cechealth/files/7/x86_64/kernel_update-3.10.0-1062.4.1.tar.gz，解压后执行update.sh即可

2. 查看当前系统上的所有可用内核启动项

```
 UEFI Legacy
[root@localhost ~]# awk -F\' '$1=="menuentry " {print i++ " : " $2}'
/etc/grub2.cfg
: CentOS Linux (4.4.206-1.el7.elrepo.x86_64) 7 (Core)
: CentOS Linux (3.10.0-862.el7.x86_64) 7 (Core)
: CentOS Linux (0-rescue-395aa029d0cca260e8304b6bf0846236) 7 (Core)
UEFI Legacy
[root@localhost ~]#  awk -F\' '$1=="menuentry " {print i++ " : " $2}'
/boot/efi/EFI/centos/grub.cfg
: CentOS Linux (4.4.205-1.el7.elrepo.x86_64) 7 (Core)
: CentOS Linux (3.10.0-693.el7.x86_64) 7 (Core)
: CentOS Linux (0-rescue-e2c56988fd1743ab9d8902c5dd4d4cc1) 7 (Core)
#
```

3. 修改默认启动项

首先使用如下命令查看默认启动项

```
grub2-editenv list
```

```
[root@kube-master-02 ~]# grub2-editenv list
saved_entry=CentOS Linux (3.10.0-1062.4.1.el7.x86_64) 7 (Core)
[root@kube-master-02 ~]#
```

如已经为上图所示显示为升级的内核，即可不用修改，直接reboot服务器即可

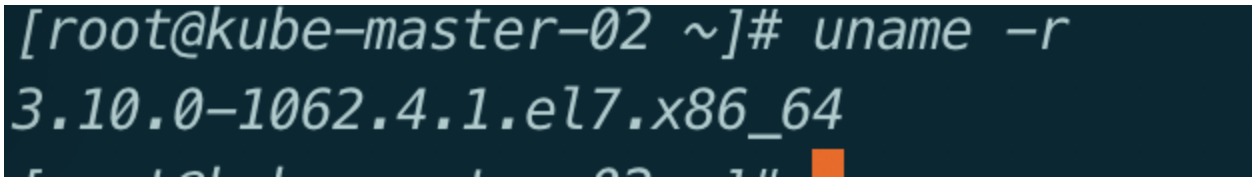
如果不是最新内核，那么执行如下命令修改默认启动项

```
grub2-set-default "CentOS Linux (3.10.0-1062.4.1.el7.x86_64) 7 (Core)"
#
grub2-editenv list
```

```
reboot
```

4. 重启之后查看服务器内核是否已经为最新

```
uname -r
```

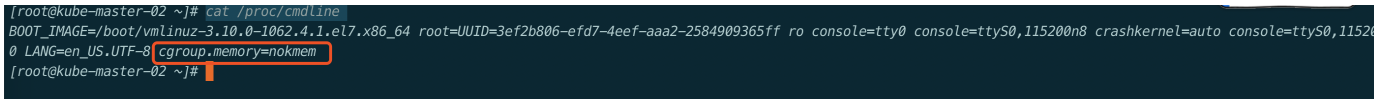


5.nokmem

```
grubby --args=cgroup.memory=nokmem --update-kernel
/boot/vmlinuz-3.10.0-1062.4.1.el7.x86_64
```

reboot

```
cat /proc/cmdline
```



6. 验证新生成的pod是否已关闭kmem

验证方法是查看新创建的 pod 的所有 container 已关闭 kmem，如果为下面结果则已关闭：

```
$ cat
/sys/fs/cgroup/memory/kubepods/burstable/pod<pod-uid>/<container-id>/memor
y.kmem.slabinfo
cat: memory.kmem.slabinfo: Input/output error
```