왜 쿠버네티스는 systemd로 cgroup을 관리하려고 할까요?

This document provides who want to know about cgroupfs in k8s.

- Email: pagaia@hotmail.com, geunwoo.j.shim@gmail.com

- Github: https://github.com/sysnet4admin, https://github.com/gnu-gnu

1. 배경 지식

프로세스에 할당하는 CPU, 메모리, 네트워크 대역폭 등과 같은 자원을 개별적으로 제한하기 위한 시스템 커널의 기능을 cgroup 이라고 합니다. 컨테이너 런타임은 이 기술을 사용하에 컨테이너에 자원을 할당합니다. cgroup을 관리하기 위한 기술은 크게 **cgroupfs**와 **systemd**가 있습니다. cgroup이 관리할 수 있는 자원의 컨트롤러는파일 시스템의 형태로 관리되고 있습니다.

[그림 1] /sys/fs/cgroup 정보

```
[root@m-k8s ~]# ll /sys/fs/cgroup
total 0
drwxr-xr-x. 5 root root  0 Aug 24 13:19 blkio
lrwxrwxrwx. 1 root root 11 Aug 24 13:19 cpu -> cpu,cpuacct
lrwxrwxrwx. 1 root root 11 Aug 24 13:19 cpuacct -> cpu,cpuacct
drwxr-xr-x. 5 root root  0 Aug 24 13:19 cpu,cpuacct
drwxr-xr-x. 3 root root  0 Aug 24 13:19 cpuset
drwxr-xr-x. 5 root root  0 Aug 24 13:19 devices
drwxr-xr-x. 3 root root  0 Aug 24 13:19 freezer
drwxr-xr-x. 3 root root  0 Aug 24 13:19 hugetlb
drwxr-xr-x. 5 root root  0 Aug 24 13:19 memory
lrwxrwxrwx. 1 root root 16 Aug 24 13:19 net_cls -> net_cls,net_prio
drwxr-xr-x. 3 root root  0 Aug 24 13:19 net_cls,net_prio
lrwxrwxrwx. 1 root root 16 Aug 24 13:19 net_prio -> net_cls,net_prio
drwxr-xr-x. 3 root root  0 Aug 24 13:19 perf_event
drwxr-xr-x. 5 root root  0 Aug 24 13:19 pids
drwxr-xr-x. 5 root root  0 Aug 24 13:19 systemd
```

이 때 이와 관련된 정보를 파일 시스템의 형태로 변환시켜 주는 매니저가 **cgroupfs** 입니다. cgroupfs는 /sys/fs/cgroup 경로의 하위에 이 정보를 마운트하여 관리하고 있습니다.

컨테이너에서 어떤 매니저를 사용하고 있는지는 다음의 명령으로 확인할 수 있습니다.

[그림 2] k8s 1.22 버전에서 cgroup 정보를 확인

```
[root@m-k8s ~]# kubelet --version
Kubernetes v1.22.0
[root@m-k8s ~]# docker info | grep Cgroup -F2
 userxattr: false
Logging Driver: json-file
Cgroup Driver: systemd
Cgroup Version: 1
Plugins:
 Volume: local
```

2. systemd와 cgroupfs의 차이점

systemd와 cgroupfs는 모두 cgroup을 관리하기 위해 동작할 수 있지만 관리에 대한 접근법에는 다소 차이가 있습니다.

cgroupfs 가 cgroup(v1) 디렉터리 하위에서 리소스를 직접 매핑하는 구조를 가집니다.

[그림 3] tree 명령어로 확인한 cgroup 디렉터리 구조

```
[root@m-k8s ~]# tree /sys/fs/cgroup
/sys/fs/cgroup
|-- blkio
|   |-- blkio.io_merged
|   |-- blkio.io_merged_recursive
|   |-- blkio.io_queued
[중략]
|   |-- cgroup.procs
|   |-- cgroup.sane_behavior
|   |-- kubepods.slice
|   |   |-- blkio.io_merged
[생략]
```

반면 **systemd**는 프로세스가 사용하는 자원을 관리하기 위해 slice > scope > service 단위의 계층 구조를 만들어 각 단위에 자원을 할당합니다. 이러한 계층 구조는 **systemd-cgls** 명령어가 표현해주는 구조 혹은 /sys/fs/cgroup/systemd 하위의 파일 시스템의 계층 구조를 통해서 확인할 수 있습니다. 이 때 systemd에 의해서 관리되는 cgroup과 관련된 정보는 /sys/fs/cgroup 하위의 자원의 컨트롤러(cpu, memory)등의 하위에서 slice의 이름으로 된 파일 시스템으로 표현됩니다.

[그림 4] systemd-cgls 명령어로 확인된 systemd의 계층적 구조

```
[root@m-k8s ~]# systemd-cgls
├─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 22
├─kubepods.slice
│ ├─kubepods-besteffort.slice
│ │ ├─kubepods-besteffort-podcad84c68_0c85_491a_8fe9_8a3d0e6220be.slice
│                    │                    │                    ├─docker-
37d280b0f630ed9be113f365000a25c5c8dddec549bee9f193906ff751e6cdc8.scope
│ │ │ │ └─3197 /speaker --port=7472 --config=metallb
│                    │                    │                    └─docker-
5c3eec595a326e26ae03e10261232b3fd6a9c80cc05a953f8a9f86d342f50f23.scope
│ │ │   └─2961 /pause
│ │ ├─kubepods-besteffort-podf028c909_cb9b_400e_9590_a3b2dec495f4.slice
│ │ │ ├─docker-
[생략]
```

3. cgroup에 대한 쿠버네티스와 도커 선호도 차이는 왜?

이론적으로는 직접 cgroup을 마운트해서 사용하는 cgroupfs를 이용하거나 systemd와 같이 계층적인 구조로 만들어 접근하거나 **결과적으로는 동일**해야 합니다.
그럼에도 쿠버네티스에서는 systemd를 선호하고 도커에서는 cgroupfs를 왜 선호하는지 알아보았지만, 정확한 결론을 도출하진 못했습니다.
다만 쿠버네티스 공식 문서에서는 도커 18.03 + 커널 4.17.17 이하의 RHEL 에서는 cgroupfs 에서 메모리 leaking issue가 발생하여 systemd로 변경하라고 권고하는 것이 있으며, 도커에서는 cgroupfs를 cgroup v1에서만 사용하고 cgroup v2에서는 systemd로 변경할 계획이 있음을 밝히고 있습니다. 그리고 카카오 엔터프라이즈에서는 2개의 차이는 경로 관리 정책의 차이 일뿐 지금 당장은 시스템 성능에 영향을 주는 것은 아니라고 판단하여 cgroupfs 으로 진행하기로 하였다고 밝힌바 있습니다.
이를 종합적으로 본다면, 결과적으로는 cgroup을 구조화 하는 **방법론적인 차이**가 있는 수준으로 추정해 볼 수 있습니다.

4. 그런데 왜 쿠버네티스 v1.22에는 영향이 있을까?

쿠버네티스 v1.22로 올라가면서 기존에는 경고 수준으로 처리하던 cgroup driver를 특별히 정해진 설정 없다면, **default**를 **systemd**로 설정하도록 코드를 변경하였습니다.

> **Note:** In v1.22, if the user is not setting the cgroupDriver field
>
> under KubeletConfiguration, kubeadm will default it to systemd.

이에 따라 Docker에서 기본 값으로 가지고 있던 **cgroupfs**와 일치하지 않는 문제가 발생하게 되었습니다.

[코드 1] kubernetes/pkg/kubelet/dockershim/docker_service.go

```
76        // The expiration time of version cache.
77        versionCacheTTL = 60 * time.Second
78
79        defaultCgroupDriver = "cgroupfs" # 기본 값
80
81        // TODO: https://github.com/kubernetes/kubernetes/pull/31169 provides
experimental
82        // defaulting of host user namespace that may be enabled when the docker daemon
83        // is using remapped UIDs.
```

```
84        // Dockershim should provide detection support for a remapping environment .
85        // This should be included in the feature proposal.  Defaulting may still occur
according
86        // to kubelet behavior and system settings in addition to any API flags that may be
introduced.
```

v1.22에서 문제가 발생하는 이유는 다음과 같이 v1.22 이전에는 기본 값에 대해서 경고만 처리하는 수준에 그쳤다면, v1.22부터는 기본 값이 없는 경우(null) systemd로 넣도록 코드를 수정했기 때문입니다.

[코드 2] v1.22 버전의 kubernetes/cmd/kubeadm/app/componentconfigs/kubelet.go

```
198       // We cannot show a warning for RotateCertificates==false and we must hardcode it
to true.
199       // There is no way to determine if the user has set this or not, given the field is a
non-pointer.
200       kc.config.RotateCertificates = kubeletRotateCertificates
201
202       if len(kc.config.CgroupDriver) == 0 {
203             klog.V(1).Infof("the value of KubeletConfiguration.cgroupDriver is empty;
setting it to %q", constants.CgroupDriverSystemd)
204             kc.config.CgroupDriver = constants.CgroupDriverSystemd
205       }
```

[코드 3] v1.21 버전의 kubernetes/cmd/kubeadm/app/componentconfigs/kubelet.go

```
202       // We cannot show a warning for RotateCertificates==false and we must hardcode it
to true.
203       // There is no way to determine if the user has set this or not, given the field is a
non-pointer.
204       kc.config.RotateCertificates = kubeletRotateCertificates
205
206       // TODO: Conditionally set CgroupDriver to either `systemd` or `cgroupfs` for CRI
other than Docker
207       if nodeRegOpts.CRISocket == constants.DefaultDockerCRISocket {
208             driver, err := kubeadmutil.GetCgroupDriverDocker(utilsexec.New())
209             if err != nil {
210                   klog.Warningf("cannot automatically set CgroupDriver when starting the
Kubelet: %v", err)
211             } else {
212                   // if we can parse the right cgroup driver from docker info,
213                   // we should always override CgroupDriver here no matter user
specifies this value explicitly or not
214                   if kc.config.CgroupDriver != "" && kc.config.CgroupDriver != driver {
215                         klog.Warningf("detected %q as the Docker cgroup driver, the
provided value %q in %q will be overrided", driver, kc.config.CgroupDriver, kind)
216                   }
```

```
217                     kc.config.CgroupDriver = driver
218             }
219     }
```

5. 결론(TL; DR)

쿠버네티스 v1.22부터는 **systemd**를 사용하도록 설정 변경하는 것이 필요해 보이지만, 이전 버전의 사용자들은 현재 상태를 유지하고, 추후 업그레이드 시에 systemd로 이전을 고려해 보는 것이 좋을 것 같습니다. 이는 cgroup v2에서는 docker 마저도 systemd를 사용하도록 권고하기 때문에 업계에서 이미 cgroup에 대한 관리를 systemd로 맞추려고 하는 경향에 따른 것입니다. 쿠버네티스 또한 이런 흐름에 따라 **systemd**를 **v1.22**부터 **기본 값**으로 정한 것으로 보여집니다.

Note: 이는 개인들의 의견이며, 회사 및 기관을 대표하는 의견이 아님을 밝힙니다.

참고 자료:

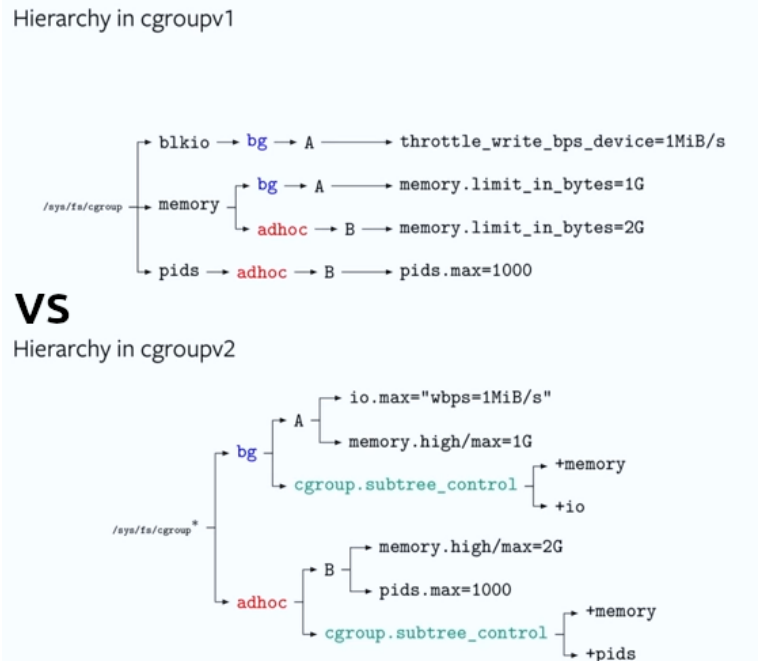
1) 테스트 환경

현재 환경의 CentOS 릴리즈 버전과 커널 버전은 다음과 같습니다.

```
[root@m-k8s ~]# cat /etc/redhat-release
CentOS Linux release 7.8.2003 (Core)
[root@m-k8s ~]# uname -r
3.10.0-1127.19.1.el7.x86_64
```

2) cgroup v1과 v2의 주요 차이

cgroup v2에서는 하나의 통일된 리소스 하위로 마운트하는 구조로 변경되었습니다.



출처: https://medium.com/some-tldrs/tldr-understanding-the-new-control-groups-api-by-rami-rosen-980df476f633

Reference:

1. **Cgroup Driver 선택하기**
   https://tech.kakao.com/2020/06/29/cgroup-driver/

2. **Changing the cgroup driver to systemd on Red Hat Enterprise Linux**
   https://www.ibm.com/support/knowledgecenter/SSBS6K_3.1.1/troubleshoot/docker_cgroup.html

3. **Options for the runtime**
   https://docs.docker.com/engine/reference/commandline/dockerd/#options-for-the-runtime

4. **Configuring a cgroup driver**
   https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/configure-cgroup-driver/

5. **Managing cgroups with systemd**
   https://www.redhat.com/sysadmin/cgroups-part-four

6. **default the "cgroupDriver" setting of the kubelet to "systemd"**
   https://github.com/kubernetes/kubeadm/issues/2376

See ya!

Hoon Jo & gnu.