

# ZNS를 이용한 리눅스 컨테이너 I/O isolation 기법

팀명: 백견이 불여일타

201724419 김동욱

201724596 채기중

201924445 김지원

지도교수: 안 성 용

부산대학교 전기컴퓨터공학부 정보컴퓨터공학전공  
School of Electrical and Computer Engineering, Computer Engineering Major  
Pusan National University

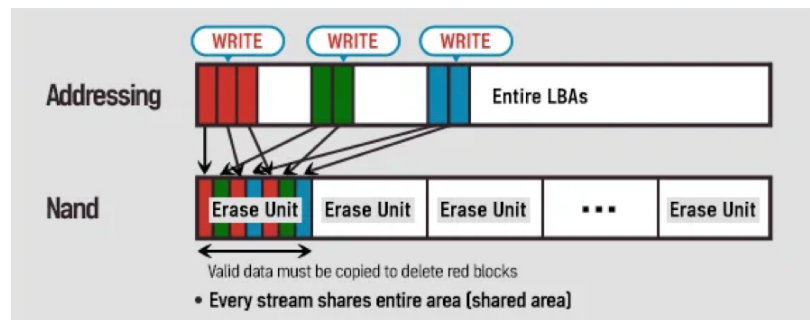
# 목차

<b>1. 과제 배경 및 목표</b>	3
1-1. 과제 배경	3
1-2. 과제 목표	4
<b>2. 세부 과제 내용</b>	5
2-1. 과제 내용	5
2-1-1. Linux Kernel 분석	5
2-1-2. ZNS I/O 과정 분석	5
2-1-3. 자원 분배 정책 개발	5
2-2. 개발 환경 및 사용 기술	6
2-2-1. Linux Container	6
2-2-2. Cgroup	7
2-2-3. Libcgroup	8
2-2-4. Cscope	8
<b>3. 현실적 제약 사항</b>	8
<b>4. 개발 일정 및 역할 분담</b>	9
4-1. 개발 일정	9
4-2. 역할분담	9

# 1. 과제 배경 및 목표

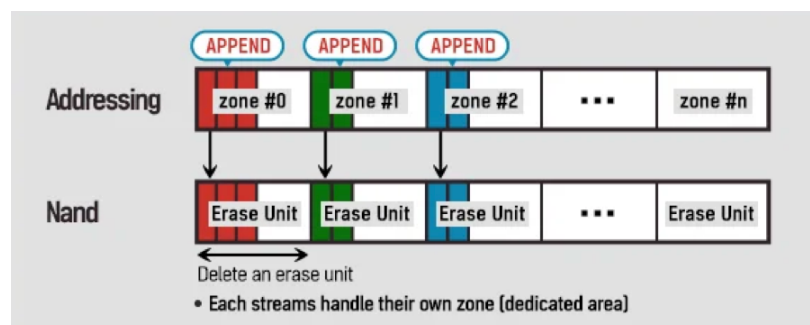
## 1-1. 과제 배경

현재 널리 사용되고 있는 저장장치인 SSD는 데이터 Random Write 방식을 사용한다. 논리적으로 원하는 영역에 데이터를 저장할지라도 물리적으로는 여러 개의 소프트웨어에서 생성되는 데이터들은 Nand block에 순차적으로 저장된다. Nand flash의 특성상 유효한데이터와 Garbage 영역이 혼재하기 때문에 저장 공간을 효율적으로 사용하기 어렵다. 이를 해결하기 위해서는 Garbage Collection 작업이 필요하다. 이는 유효한 자원들을 모아서 다른 Nand block에 써서 저장공간을 확보하는 작업이고 이를 위해 읽기/쓰기 과정에서 현재 작업을 중단하므로 SSD의 성능저하가 발생한다.



[그림 1. 기존 SSD의 저장 방식]

이에 비해 ZNS(Zoned Namespace) 기술은 용도와 사용 주기가 동일한 데이터를 정해진 구역인 Zone에 순차적으로 저장하고 Zone 단위로 지우기 때문에 Garbage Collection으로 인한 추가적인 I/O가 발생하지 않는다. ZNS는 성능 저하가 없고 효율적으로 저장공간을 활용하기 때문에 서버에 여러 운영체제와 프로세스를 처리하는 현대의 데이터센터에 적합하다.



[그림 2. ZNS SSD의 저장 방식]

이러한 ZNS의 특성을 클라우드(Cloud)기술의 기반이 되는 Linux Container(리눅스 컨테이너)와 접목시키고자 한다.

리눅스 컨테이너란 실행에 필요한 모든 파일(라이브러리 및 종속항목)을 포함하여 런타임 환경에서 애플리케이션을 패키지와 분리하는 기술이다. 리눅스 컨테이너는 호스트 운영체제에 의존하여 가상머신보다 가볍고 일관성 있게 애플리케이션 환경을 구축할 수 있는 장점이 있다. 이러한 장점 덕분에 많은 기업과 기관은 컨테이너를 실무에서 사용하고 있다.

그런데 리눅스 컨테이너에 ZNS를 바로 적용시키기 힘들다. Zone이라는 개념이 리눅스 컨테이너에서 자원을 할당하는 기존방식과는 다르기 때문이다. 따라서 본 팀은 ZNS의 Zone을 할당하고 읽고 쓰는 과정에서 컨테이너 간의 간섭을 최소화하여 할당하는 컨테이너 별 Zone 분리 할당 정책 개발하고자 한다.

## 1-2. 과제 목표

본 과제는 각 Linux Container에 ZNS의 Zone을 할당하여 컨테이너 간 성능 간섭을 최소화하는 기법을 개발하는 것을 목표로 한다.

1. Linux Kernel 분석(Cgroup, Block Layer, ZNS Device Driver)
2. ZNS I/O 과정 분석
3. 자원 분배 정책 개발

## 2. 세부 과제 내용

### 2-1. 과제 내용

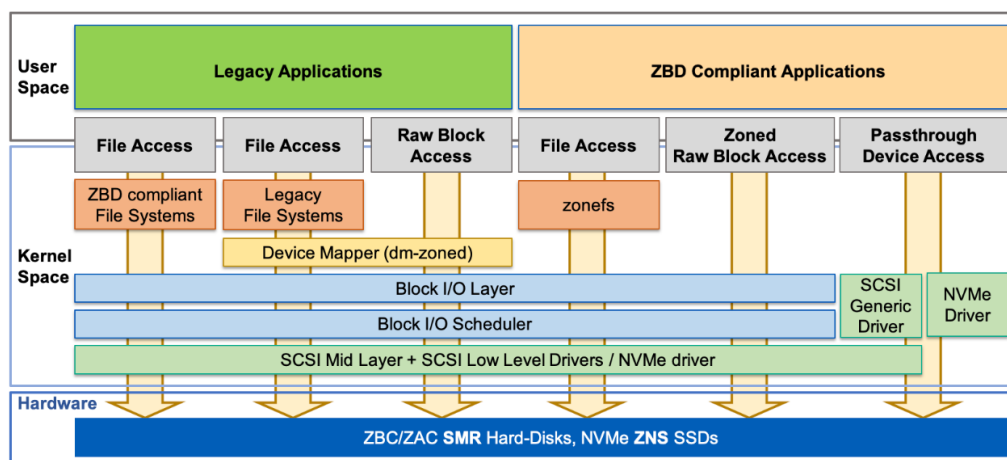
#### 2-1-1. Linux Kernel 분석

Linux Kernel은 cgroup, Block Layer, Device Driver 등의 기능을 통해 I/O를 관리한다. Cgroup은 각 프로세스 별 자원 분배를 담당하고, Block Layer는 block devices에서 수행되는 input/output operations을 관리하며, Device Driver는 커널 영역에서 이러한 I/O가 이루어지는 하드웨어들을 제어한다. Cscope를 이용해 커널 코드를 분석하고 Libcgroup을 이용해 Cgroup의 동작 원리와 조작법을 익힌다. Block Layer 분석을 통해 File System을 ZNS와 연결하기 위해서 어떤 Layer를 조작해야 하는지 알아본다. Device driver의 분석 및 수정을 통해 사용되는 Zone이 실제 ZNS 하드웨어 상에 Mapping되는 위치를 알 수 있다.

#### 2-1-2. ZNS I/O 과정 분석

QEMU를 이용해 구축한 에뮬레이터 환경 내에서 Container를 생성하면 해당 Container는 자동으로 가상 ZNS를 저장 장치로 인식한다. 따라서 Container 내에서 I/O를 요청한 후 Ftrace 프레임 워크를 통해 그 과정을 추적하면 해당 I/O가 ZNS에 도달하기까지 호출되는 함수와 경로를 알 수 있다. 하지만 이러한 I/O는 File System이 Zone에 대한 개념을 인식하지 못한 채 이루어지기 때문에 I/O 과정에서 사용되는 함수와 경로를 파악한 후 ZNS의 특성을 이용할 수 있도록 한다.

#### 2-1-3. 자원 분배 정책 개발



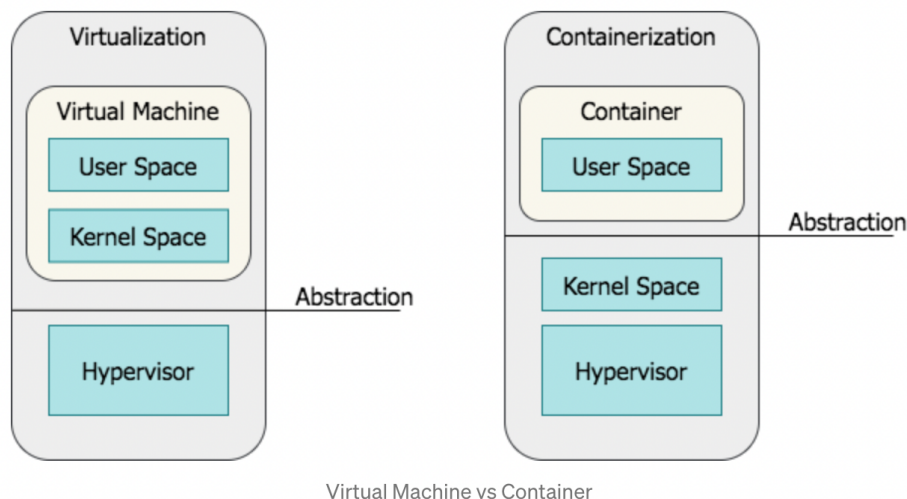
[그림 3. Linux zoned block device support overview]

일반적인 SSD에서는 cgroup이 용량 단위로 자원을 요청하는데, ZNS를 이용한 효율적인 I/O를 위해서는 용량 단위를 Zone 단위로 바꾸어 Mapping해주는 과정이 필요하다. 그림 3에서 Legacy Applications가 현재 사용되는 SSD I/O를 수행하는 영역으로, 새로운 Block Layer를 추가하거나 Device Mapper를 수정하여 SSD Mapping을 ZNS Mapping으로 변환할 수 있다.

그림 3의 ZBD Compliant Applications 영역은 ZNS를 이용한 File System인 zonefs가 이미 구현되어 있다. 이 zonefs가 Legacy Application처럼 용량 단위의 자원 분배 요청을 처리하고 싶은 경우에는 zonefs에 cgroup 정보를 추가하여 구현할 수 있다.

## 2-2. 개발 환경 및 사용 기술

### 2-2-1. Linux Container



[그림 4. 가상화 vs 컨테이너]

단일 리눅스 시스템에 동작하고 있는 프로세스를 격리시켜 각 프로세스마다 독자적인 리눅스 시스템 환경을 구축하는 것을 리눅스 컨테이너라고 한다. 이를 가르켜 OS 수준에서의 가상화라고 한다. 일반적으로 가상화는 Hypervisor라는 논리적 플랫폼을 이용해 하나의 HOST(원래 본체에 설치된 운영체제)위에 여러개의 Guest OS(구동하고자 하는 운영체제)를 구동하는 기술이다.

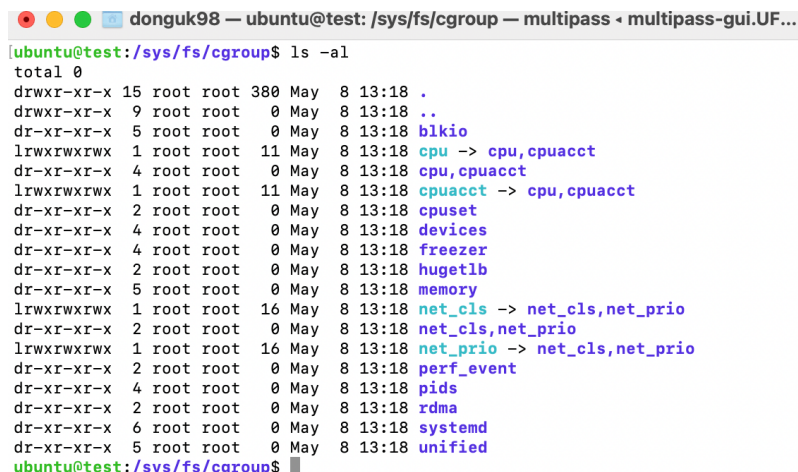
이와 달리 리눅스 컨테이너는 하나의 리눅스 시스템에서 프로세스들을 격리시켜 독자적인 시스템 환경을 구축한다. 리눅스 컨테이너는 단일 컨트롤 호스트 상에서 여러 개의 고립된 리눅스 시스템(컨테이너)들을 실행하기 위한 운영 시스템 레벨 가상화 방법이다.

리눅스 커널은 cgroup을 절충하여 가상화 머신을 시작할 필요 없이 자원(CPU, 메모리, 블록 I/O, Network)을 할당하고 cgroup은 애플리케이션 입장에서 프로세스 트리, 네트워크, 사용자 ID, 마운트된 파일 시스템 등의 운영 환경을 완전히 고립시키기 위해 namespace isolation을 제공한다. 즉, 리눅스 컨테이너는 cgroups과 namespace를 결합하여 애플리케이션을 위한 환경을 제공한다.

### 2-2-2. Cgroup

Cgroup은 프로세스와 마찬가지로 계층적으로 구성되어 있다. 하위 cgroup은 부모 cgroup 속성의 일부를 상속하도록 되어 있으며, 여러 개의 서로 다른 cgroup 계층이 시스템에 동시에 존재할 수 있다. 각 계층은 하나 이상의 subsystem(서브시스템)에 연결되는데, 서브시스템이란 CPU 시간 또는 메모리와 같은 자원을 조절하는 시스템이다.

Cgroup은 file system형태로 관리된다. /sys/fs/cgroup의 경로로 들어가보면 그림 3과 같은 cgroup의 구조가 directory와 file의 형태로 만들어져 있는 것을 확인할 수 있다. 이를 cgroupfs라 하며, Linux 시스템 상에서 cgroup 제어를 위해 탄생한 특수 File System이다. 해당 경로 아래에 directory를 생성, 삭제하거나 file 내부 내용을 변경하여 cgroup을 제어할 수 있다.



```

donguk98 — ubuntu@test: /sys/fs/cgroup — multipass • multipass-gui.UF...
ubuntu@test:/sys/fs/cgroup$ ls -al
total 0
drwxr-xr-x 15 root root 380 May  8 13:18 .
drwxr-xr-x  9 root root  0 May  8 13:18 ..
dr-xr-xr-x  5 root root  0 May  8 13:18 blkio
lrwxrwxrwx  1 root root 11 May  8 13:18 cpu -> cpu,cpuacct
dr-xr-xr-x  4 root root  0 May  8 13:18 cpu,cpuacct
lrwxrwxrwx  1 root root 11 May  8 13:18 cpuacct -> cpu,cpuacct
dr-xr-xr-x  2 root root  0 May  8 13:18 cpuset
dr-xr-xr-x  4 root root  0 May  8 13:18 devices
dr-xr-xr-x  4 root root  0 May  8 13:18 freezer
dr-xr-xr-x  2 root root  0 May  8 13:18 hugetlb
dr-xr-xr-x  5 root root  0 May  8 13:18 memory
lrwxrwxrwx  1 root root 16 May  8 13:18 net_cls -> net_cls,net_prio
dr-xr-xr-x  2 root root  0 May  8 13:18 net_cls,net_prio
lrwxrwxrwx  1 root root 16 May  8 13:18 net_prio -> net_cls,net_prio
dr-xr-xr-x  2 root root  0 May  8 13:18 perf_event
dr-xr-xr-x  4 root root  0 May  8 13:18 pids
dr-xr-xr-x  2 root root  0 May  8 13:18 rdma
dr-xr-xr-x  6 root root  0 May  8 13:18 systemd
dr-xr-xr-x  5 root root  0 May  8 13:18 unified
ubuntu@test:/sys/fs/cgroup$

```

[그림 5. /sys/fs/cgroup 경로 내부 파일]

Cgroupfs 내부에 Directory를 만들면 새로운 cgroup을 생성할 수 있다. 원하는 subsystem directory 안에 새로운 Directory를 생성하면 자동으로 부모 directory의 것을 상속받아 다수의 file들이 내부에 생성된다. 이렇게 생성된 file들 중 task 파일 안에 원하는 프로세스의 PID들을 적어주면 해당 프로세스를 포함하여 task에 속해있는 프로세스들의 시스템 자원을 제한할 수 있다.

### 2-2-3. Libcgroup

Libcgroup 라이브러리는 여러 가지 cgroup 관련 명령과 man 페이지를 포함하며 프로세스를 단순화하고 기능을 확장하는 역할을 한다. Libcgroup 내부 cgconfig 서비스를 이용하면 계층 구조를 생성하고 서브시스템을 연결하여 계층 내 cgroup을 편리하게 관리할 수 있다. Libcgroup은 /etc/cgconfig.conf 파일을 바탕으로 cgroup을 생성한다. 실제로 자원 분배의 역할을 수행하는 부분은 <controller> 영역으로, 조작을 원하는 subsystem과 그것들의 parameter들을 적어준다. 이 때, 자식 cgroup은 부모 cgroup의 자원 제한을 물려받기 때문에 부모 그룹의 제한을 초과하여 할당할 수 없다.

### 2-2-4. Cscope

소스 코드를 분석하는 데에 사용되는 Unix 기반 개발자 도구이다. Cscope가 지원하는 검색 요소들은 다음과 같다.

- All references to a symbol
- Global definitions
- Functions called by a function
- Functions calling a function
- Text string
- Regular expression pattern
- A file
- Files including a file

## 3. 현실적 제약 사항

실제 구현된 ZNS SSD는 사용할 수 없는 관계로 본 과제는 QEMU를 통한 ZNS 에뮬레이터로 진행된다. 해당 에뮬레이터는 ZNS의 가상 이미지를 메모리에 위치시키는 방식으로 동작하기 때문에 실제 하드웨어와 성능 차이가 있을 수 있다.

즉, 하드웨어의 성능에 따라 예측값과 다른 결과가 나올 수 있음을 의미한다. 이를 해결하기 위해 적절한 지연시간을 모델링해 하드웨어 성능을 정확하게 예측할 수 있도록 한다.



## 4. 개발 일정 및 역할 분담

### 4-1. 개발 일정

	5월	6월					7월				8월				9월				
주차	4	1	2	3	4	5	1	2	3	4	1	2	3	4	1	2	3	4	5
개발 환경 구성																			
리눅스 커널 분석																			
컨테이너 별 zone 분리 할당 정책 개발																			
중간 보고서 작성																			
성능 비교 및 분석 1																			
디버깅																			
성능 비교 및 분석 2																			
최종 보고서작성																			

### 4-2. 역할분담

학번	이름	구성원별 역할
201724419	김동욱	<ul style="list-style-type: none"> <li>- 리눅스 커널 코드 분석을 위한 cscope 사용법 숙지</li> <li>- 리눅스 커널 Device Driver 분석</li> <li>- 기존 리눅스 컨테이너의 cgroup file system 분석</li> <li>- 컨테이너 존 분리 할당 정책 개발</li> <li>- 리눅스 컨테이너 성능 간섭 최소화 기법 개발</li> </ul>
201724596	채기중	<ul style="list-style-type: none"> <li>- 리눅스 커널 코드 분석을 위한 cscope 사용법 숙지</li> <li>- ZNS를 이용한 znsfs Block I/O PATH 분석</li> <li>- ZNS SSD I/O 분석</li> <li>- 리눅스 컨테이너 성능 간섭 최소화 기법 개발</li> <li>- 컨테이너 I/O 성능 비교 분석 주도</li> </ul>
201924445	김지원	<ul style="list-style-type: none"> <li>- 리눅스 커널 코드 분석을 위한 cscope 사용법 숙지</li> <li>- cgroup 코드를 다루는 Libcgroup 라이브러리 설치</li> <li>- 리눅스 커널 Device Driver 분석</li> <li>- 리눅스 커널 Block Layer I/O 분석</li> <li>- 리눅스 컨테이너 성능 간섭 최소화 기법 개발</li> </ul>

## Reference

[1] Red Hat - 자원 관리 가이드

([https://access.redhat.com/documentation/ko-kr/red\\_hat\\_enterprise\\_linux/6/html/resource\\_management\\_guide/index](https://access.redhat.com/documentation/ko-kr/red_hat_enterprise_linux/6/html/resource_management_guide/index))

[2] Linux cgroups에 대해 알아보자 1,2

(<https://torch.vision/2019/06/18/cgroups-1.html>, <https://torch.vision/2019/06/20/cgroups-2.html>)

[3] 삼성전자 차세대 기업 서버용 SSD 출시

(<https://news.samsung.com/kr/%EC%82%BC%EC%84%B1%EC%A0%84%EC%9E%90-%EC%B0%A8%EC%84%B8%EB%8C%80-%EA%B8%B0%EC%97%85-%EC%84%9C%EB%B2%84%EC%9A%A9-zns-ssd-%EC%B6%9C%EC%8B%9C>)

[4] [ZNS] QEMU를 이용한 Zoned Namespace 개발 환경 구축

(<https://blackinkgj.github.io/qemu-zns/>)

[5] Scope.sourceforge

(<http://cscope.sourceforge.net/>)

[6] Red Hat - Linux 컨테이너란?

(<http://cscope.sourceforge.net/>)

[7] Kakao Tech - Cgroup Driver 선택하기

(<https://tech.kakao.com/2020/06/29/cgroup-driver/>)

[8] SK하이닉스 뉴스룸 - [궁금한 반도체 WHY] ZNS SSD, 기존 SSD와 무엇이 다를까?

(<https://news.skhyunix.co.kr/post/zns-ssd-existing-ssd-and>)

[9] [커널분석] - Architecture별 분석을 위한 설정(ctags&cscope)

(<https://sonseungha.tistory.com/458>)

[10] 클라우드 환경에서 고성능 저장장치를 위한 동적 대역폭 분배 기법

(<https://www.kci.go.kr/kciportal/ci/sereArticleSearch/ciSereArtiView.kci?sereArticleSearchBean.artid=ART002602292>)

[11] Linux Zoned Storage Support Overview

(<https://zonedstorage.io/docs/linux/overview>)