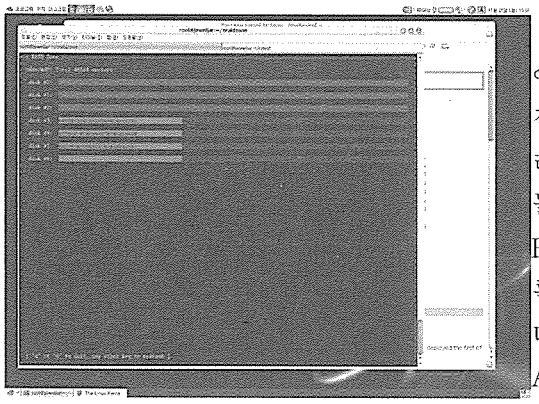
충남대학교 "RAID Zone"

물리적디스크추가로동적인용량확장을 지원하는리눅스SoftwareRAID모듈



충남대학교 컴퓨터 공학 전공에서 개발한 RAID Zone은 물리적 디스크를 추가하여 논리적인 레거시 RAID array의 총 용량을 늘리되, Reconstruction이나 Reorganization이 발생하지 않도록 만들어, 보다 유연하고 보다 더 질 높은 고 가용성 (High Availability)을 성취할 수 있는 Linux용 Software RAID 모듈이다. 일반적인 기존의 RAID array 에서는 Array의 용량이 다 찰 경우 관리자가 필요에 의해 디스크를 추가하게 되면 잠시 동안 서비스가 중지된다. 물리적 디스크를 새로 추가하면 기존에 디스크에 쌓여있던 모든 데이터 중 일부를 늘어난 디스크 위치로 이동하는 재구성(Reorganization)현상이 발생하기 때문이다. 이 경우 다시금 데이터가 차곡차곡 정리 된다는 장점은 있으나 기존의 용량이 크면 클수록 디스크를 재구성 하는 시간이 기하급수적으로 커지게 되며 또한 필요에 의해 더하는 디스크의 개수와 용량이 커질수록 데이터 재분배의 시간도 길어지게 된다.

RAID Zone의 핵심 기술은 RAID로 묶여진 디스크들과 새로이 추가되는 디스크들을 별도의 독립적인 영역(ZONE)으로 구분하기 위한 ZONE Number(Zone의 인덱스), The number of disks(Zone에 포함된 디스크의 개수), First sector(Zone의 논리적인 첫 번째 섹터), Last sector(Zone의 논리적인 마지막 섹터), Physical first sector(Zone의 물리적 첫 번째 섹터), Used sectors(데이터를 기록한 Zone의 물리적인 섹터 수), Virtual Storage Disk(가상디스크)등의 메타 데이터를 효율적으로 관리하는데 있다.

동 상

RAID Zone

1. 작품명 : RAID Zone

2. 제작자 : 충남대학교 컴퓨터 공학

대표자 : 정 진 욱 (충남대 컴퓨터공학과 4학년)
개발자 : 정 진 욱 (충남대 컴퓨터공학과 4학년)
남 양 현 (충남대 컴퓨터공학과 3학년)
최 하 루 (충남대 컴퓨터공학과 2학년)
김 지 욱 (충남대 컴퓨터공학과 2학년)
주 소 : (305-764) 대전시 유성구 궁동 충남대학교 컴퓨터공학
전 화 : 042-821-6651
팩 스 : 042-822-4997
e-mail : jeweljar@hanmail.net

3. S/W 요약설명

충남대학교 컴퓨터 공학 전공에서 개발한 본 RAID Zone 이라는 작품은 RAID 기술로 묶여있는 논리적인 하나의 Storage에 하나, 혹은 둘 이상의 물리적인 Disk를 동적으로 추가하여 논리적인 Storage의 총 용량을 늘리되, 기존 RAID 5의 규칙을 유지하면서도 Reconstruction(혹은 Reorganization)이 발생하지 않도록 하여, 보다 질 높은 High Availability를 추구할 수 있는 Software RAID 기술이다.

3.1 개발 배경

일반적으로 RAID로 구성한 논리적 디스크 Array의 용량이 부족하게 될 경우 관리자가 필요에 의해 물리적 디스크를 추가하게 되면 잠시 동안 서비스가 중지되고, 기존에 디스크에 쌓여있던 모든

데이터 중 일부를 늘어난 디스크 위치로 이동하는 재구성(Reorganization)현상이 발생한다. 그리고 재구성으로 인해 추가된 디스크 개수만큼 데이터들이 균등하게 기록되어지면 흩어져있던 패리티 비트를 재계산하고 그 이후에서야 Write & Read 명령을 수행하게 된다. 그러나 이런 작업은 기존의 RAID Array 용량이 크면 클수록 디스크를 재구성 하는 시간이 기하급수적으로 커지게 되며 또한 필요에 의해 더하는 디스크의 개수와 용량이 커질수록 또한 데이터 재분배의 시간이 길어지게 된다. 더욱이 HA 서버 제품군이라면 많은 사용자들이 동시에 사용할 것이고 또한 고용량 데이터들의 보관으로 인해 서버내의 데이터의 누적 용량 또한 빠르게 늘어날 것이다. 이러한 환경에서의 Reorganization이란 거의 불가능에 가까워진다.

따라서, 용량 확보를 위해 새로운 디스크를 추가할 경우 데이터 디스크의 재구성이 발생하지 않고 순서대로 차곡차곡 RAID를 구성해 나가며 서버로서의 서비스를 지속적으로 수행할 수 있도록 RAID 기능을 보완하는 기술이 필요하게 되었다.

3.2 시스템 개요

본 RAID Zone 기술은, 기존의 RAID로 묶여진 디스크들과 새로이 추가되는 디스크들을 별도의 독립적인 영역(ZONE)으로 구분하기 위한 ZONE Number(Zone의 인덱스), The number of disks(Zone에 포함된 디스크의 개수), First sector(Zone의 논리적인 첫 번째 섹터), Last sector(Zone의 논리적인 마지막 섹터), Physical first sector(Zone의 물리적 첫 번째 섹터), Used sectors(데이터를 기록한 Zone의 물리적인 섹터 수), Virtual Storage Disk(가상디스크)등의 메타 데이터를 효율적으로 관리하여 물리적 디스크를 추가하는 순간에도 서비스의 중단이 발생하지 않는 새로운 RAID array 기술이다.

3.3 시스템 특징

현재 Linux의 Software RAID에서는 물리적인 디스크를 추가하게 되면 추가되는 디스크는 기존의 RAID array에 참여하지 못하고 Spare Disk로 인식이 된 후 Sync를 요청한 이후에야 Spare Disk를 추가된 디스크로 인식하고 그 부분으로 기존의 데이터가

일부 옮겨지면서 패리티 비트가 재계산 되는 데이터의 전체적인 재구성(Reorganization)이 발생한다. 그러나 본 RAID ZONE에서는 디스크 추가 시 전체적인 데이터의 재구성이 없이 추가되는 디스크는 새로운 RAID Array로서 기존의 RAID Array의 마지막 섹터 이후로 디스크의 섹터 공간이 linear하게 연결된다. 이런 새로운 공간을 Zone이라고 명명하고, 그와 관련된 메타데이터들을 Zone Table이라는 내부 구조체로 유지한다. 그리고 이 Zone Table은 RAID Array의 Superblock에 기록하여 RAID Array를 잠시 shutdown했다가 다시 startup해도 Zone 정보가 여전히 유지된다. 또한, 개발된 코드는 완전히 새로운 것이 아닌 이미 구성되어 있는 Linux Kernel의 RAID 관련 소스 코드를 수정 및 추가하여 기존의 구축된 시스템 안에서도 업그레이드/패치를 통해 손쉽게 본 프로그램을 설치할 수 있다.

3.4 개발 효과

■ 효율성

- 최근에는 메인보드에서 자동적으로 하드디스크가 여러 개일 경우 자동적으로 RAID 방식을 지원하는 기능이 추가 되어 손쉽게 유저들도 RAID 구성을 할 수 있으며 이러한 방법이 가장 널리 보편화 되어 있다. 즉 RAID Controler에 대한 추가 비용 없이 간단한 초기 세팅을 통해 RAID가 구성된 시스템을 만날 수 있게 된 것이다.

  그러나 메인보드에 내장된 레이드 시스템은 간단한 RAID 기능만을 추가하였기에 좀 더 빠르고 안정적인 성능을 원하는 전문 사용자의 경우는 따로 비용을 들여 RAID를 구성하도록 도와주는 RAID Controler를 추가로 구입하여 장착하게 되었다. 과거에 RAID Controler는 상당한 부피와 고가의 가격, 많은 전력 소모로 인해 특수한 제품에만 달려있는 특수한 장비였다. 그러나 최근에 제작되는 카드형 RAID Controler는 독립적으로 작동하는 서버용 RAID Controler에 못지않는 성능과 여러 우수한 경쟁업체들로 인해 낮아진 가격으로 손쉽게 구입할 수 있는 장비가 되었다.

  하드웨어 RAID 장치의 높은 가격과 성능 사이에서 고민 중인

사용자들은 가격대 성능비가 훌륭한 Software RAID를 많이 사용하고 있다.

- 이러한 RAID를 개선하여, 3.2 시스템 개요에서 언급한 메타데이터, 즉 Zone Table은 최소한의 필요한 정보만 유지한다. 따라서 Superblock에 기록되는 정보도 최소화된다.
- 그리고 Zone Table은 RAID Array가 startup될 때 메인 메모리에 로드되기 때문에, 사용자 프로세스가 접근하려는 논리적 섹터를 물리적 섹터로 변환하기 위한 계산을 최대한 빠른 시간 안에 완료할 수 있도록 작성되었다.

■ **차별성**

- 기존의 RAID5에는 Disk Array에 묶여있는 데이터의 용량이 부족할 경우 서비스 중지 없이 새로운 디스크를 추가하지 못하는 단점이 존재한다. 이것은 클러스터링 시스템에서 지향하는 고가용성과 반대되는 것이다. 이러한 단점은 서버로서의 중요한 문제로 발생한다. 가령 서버를 운영함에 있어 서비스를 중지함은 회사의 이미지, 손해 비용 등의 문제로 직결된다. 따라서 서비스를 운영하는 서버의 경우 대부분은 고가용성을 지향하는 서버가 목표로 운영된다. 고가용성의 방법이 될 수 있는 것 중 하나가 RAID5이나 위에서 언급했듯 RAID에도 문제가 존재한다.
  이러한 문제를 해결하기 위한 방법은 적은 메타 데이터를 사용함으로써 기존의 디스크 사용 영역을 유지시킴과 동시에 새로운 디스크 영역을 추가할 수 있는 방법이 필요하게 된다. 여기서의 메타 데이터라는 것은 지금까지 기록했던 데이터 정보들과 디스크 정보들을 의미한다. 최소한의 이런 정보를 이용해 데이터의 재구성 없는 디스크 추가 기능이 가능하다. 이를 이용한 방법 중 하나가 ZONE 개념이다. 데이터 디스크의 재구성 없는 디스크 추가를 위해서는 RAID5로 묶인 디스크에 새로운 디스크를 더한다면 이는 각각의 독립적인 RAID5로 보는 개념을 도입하였다. 즉 기존의 디스크를 ZONE1이라고 한다면 새로 추가된 디스크 영역은 ZONE2로 구성되어 각각의 독립된 RAID5를 구축 할 수 있는 아이디어이다. 이러한 RAID ZONE 개념은 그동안 Linux 안의 RAID의 중요한 문제 중 하나였던 실시간 디스크 추가 시 데이터의 재구성(Reorganization)을 막아주는 역할을 하게 되었

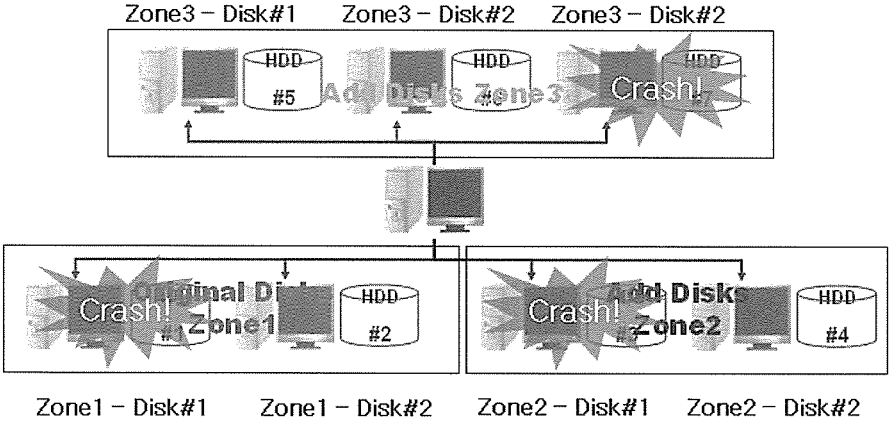
다. 더욱 자세하게 말하자면 디스크를 새로이 추가할 경우 재구성이 일어나는 경우 보다 기존의 데이터를 백업하여 다른 곳으로 옮긴 후 새로이 추가된 디스크를 RAID로 다시 묶고 그 안에 백업한 데이터를 다시 덮어쓰는 번거로움을 해소하게 되었다.

이러한 migration 방식은 상당히 많은 시간을 허비하는 반면 RAID ZONE 개념으로 구축한 본 프로그램에서는 실시간으로 하드를 추가하고 OS에게 해당 RAID5에 디스크가 추가된 것을 인식하게 한다면 기존의 디스크 RAID5영역과는 독립적으로 새로이 추가된 디스크를 RAID5로 묶어 순서대로 데이터를 기록한다.

- RAID ZONE의 개념을 추가하여 디스크 1개 이상인 다수의 디스크를 추가할 경우 RAID를 구성하는 것은 기존의 디스크 안에 있는 Super Block의 내용을 약간 수정하여 기능을 구현케 하였지만 만약 단 한 개의 디스크만을 추가할 경우 RAID를 구성한다는 것은 쉽지가 않다. 하나의 디스크를 추가할 경우 Parity Bit 와 Data Bit를 각각 써 놓을 곳의 위치 지정이 쉽지 않기 때문이다. 한 개의 디스크 안에 파티션을 나누어 각각을 기록한다 하더라도 만약에 해당 되는 디스크가 불량이 생긴다면 이는 parity bit와 data bit 둘 모두를 읽어올 수가 없어 데이터 복구가 불가능하기 때문이다. 그러나 일반 소비자의 경우 두 개의 디스크가 아닌 하나의 디스크만을 써서 추가적인 용량확보를 원하는 경우가 종종 있다. 게다가 현재까지 나오는 메인 보드의 확장 한계크기 등 개인용 PC에서 한번 구축한 RAID에 다수의 디스크를 구매하여 확장하는 경우는 적다고 할 수 있다. 늘 언제나 RAID5를 구성하기 위해서 다수의 디스크를 구입하게 된다면 많은 부담이 따를 것이다. 그리고 위에 설명했듯이 RAID의 특성상 근본적으로 하나의 디스크만으로는 RAID5의 구축이 불가능한 것이 사실이다. 그러나 본 프로그램에서는 기존의 구축된 RAID ZONE 영역에서 새로 추가한 디스크 용량만큼의 잔여 공간이 남는다면 그 잔여공간을 이용하여 가상의 디스크를 만들어 그 부분과 추가한 디스크 하나를 RAID로 묶는 새로운 RAID5 구성 방법을 구현해 내었다. 이는 RAID를 구성할 때는 꼭 1개 이상인 다수의 디스크를 구입해야 한다는 비용적 측면을 해결한 아이디어 인 것이다. 비록 기존에 구성된 RAID ZONE의 잔여용

량이 추가되는 디스크의 용량보다 같거나 커야한다는 전제조건이 있긴 하지만 기존에 구성된 RAID ZONE의 용량이 크면 클수록 더욱 잔여용량의 크기는 커지기에 하나의 디스크 정도는 충분히 RAID ZONE으로 연결 할 수 있다.

- 기존의 RAID5는 다수의 디스크를 데이터 부분과 패리티 부분으로 나누어 데이터를 저장한 뒤 구성된 RAID 디스크 중 한 개의 디스크가 장애가 생길 경우 다른 디스크의 패리티를 이용 자료를 복구하는 것이 특징이라 설명하였다. 그런데 이런 RAID5에서 만약 1개가 아닌 다수의 디스크에 장애가 생길 경우 자료를 복구하지 못한다는 단점이 있다. 패리티 비트로는 한 개의 디스크만은 복구해도 다수의 디스크에 동시에 장애가 생기면 데이터 손실이 너무 커 어느 것을 복구할지 모르는 것이다. 그러나 RAID ZONE을 구현한 본 프로그램은 다수의 독립적인 RAID ZONE 안에서 1개 이상의 디스크에서 장애가 발생하지 않는 다면 동시에 여러 개의 디스크에 장애가 발생해도 복구할 수 있다는 장점이 있다.



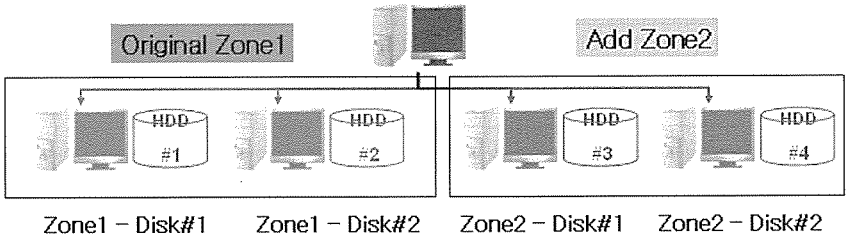
[그림] RAID ZONE에서 1개 이상의 디스크 장애 발생

그림과 같이 각각의 RAID ZONE에서 한 개의 디스크들이 장애가 발생했다. 그러나 실질적으로는 총 3개의 디스크 장애가 생긴 것이다. 기존의 RAID5는 만약 위와 같은 상황이 발생하였다면 총 7개의 디스크 중에서 3개의 디스크 불량으로 인해 결국 데이터를 복구 하지 못한다.

그러나 RAID ZONE이 각각 독립적인 RAID5를 구성하고 있는 상태에서는 위와 같은 상황 시에도 해당 ZONE안에 있는 다른 데이터 디스크들과 패리티 디스크들로부터 손실된 디스크의 데이터를 다시금 복구 할 수가 있다. 즉, 3개의 RAID ZONE을 각기 다른 RAID 디스크로 처리하여 서로의 데이터가 간섭을 받지 않기에 각각의 RAID ZONE은 서로에게 영향이 없는 것이다.

이와 같은 구성은 기존 한 개의 디스크 장애 가능성을 생각한 RAID5의 단점을 상당수 보완해 주는 기능을 하게 된다. 비록 한 RAID ZONE 안에서 2개 이상의 디스크가 동시에 장애가 발생하는 경우는 원래 RAID5의 태생적인 한계이므로 극복할 수 없지만, 새로 추가되는 디스크들 가운데 불량이 있을 가능성이 있다면 기존의 Original RAID ZONE과 구별되게 Add RAID ZONE을 구성함으로서 중간 중간 분류하여 2개 이상의 디스크가 동시에 장애가 발생하더라도 서비스의 중지 및 데이터의 손실이 없는 HA한 서버를 구축할 수 있는 것이다.

3.5 시스템 구성 및 주요기능



< RAID ZONE의 구성 형태의 예제 >

위의 그림과 같이 RAID ZONE은 처음에 HDD1과 HDD2가 RAID5 형식으로 구성되어 있다.(이것을 Original ZONE이라 한다) 그 상태서 새로운 하드 디스크인 HDD3, HDD4가 추가되어 새로운 RAID로서 참여하게 된다면 기존의 RAID5에서는 총 4개의 HDD를 가진 거대한 하나의 RAID 디스크가 만들어져야 하나 RAID ZONE에서는 뒤에 추가로 더해진 두 개의 디스크는 따로 독립적인 RAID5로 구

성이 된다.(이것을 ADD ZONE이라 한다)

즉 실질적으로는 총 4개의 하드 디스크가 연결된 상태이나 컴퓨터 내부에서는 2개의 RAID5가 구성되어 각각의 하드는 분리되어 각자의 RAID ZONE에 포함된다.

3.5.1 Adding 2 or more disks

우선 세 개의 디스크가 RAID5로 묶여져 있는 시스템에 예로 두개의 디스크를 추가했을 경우에 대해 설명하자면, 기존의 RAID5로 묶여있는 디스크 3개를 ZONE1로 두고 새로 추가된 디스크 2개를 ZONE2로 둔다. 즉 , ZONE1과 ZONE2를 각각의 RAID5를 이룬 구성으로 보면 된다.

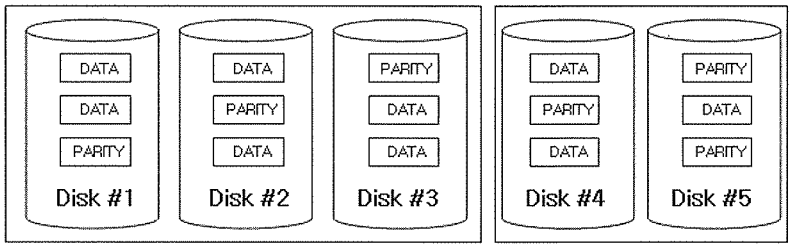
< 물리적 디스크 두개를 추가한 이후의 Zone Table >

Zone	# of disks	First sector	Last sector	Phys first sector	Used sectors
1	3	0	20224	0	4000
2	2	20224	30336	0	8576

위의 표는 ZONE1의 디스크 수는 3개 이고, 새로 추가한 디스크 수는 2개로 2개 이상의 디스크를 추가했을 경우에 만들어진 ZONE Table이다. Phys first sector를 보면 알 수 있듯이 각각의 존은 독립적인 RAID5 구성을 이룬 것을 알 수 있다. 조금 더 자세히 설명하자면 File System Level에서 보기에는 두 개의 ZONE들은 하나의 디스크로 인식 되어 Disk I/O 작업을 수행 한다. 이는 Frist sector와 Last sector를 보면 알 수 있다. 즉 첫 번째 논리적 섹터는 0이고, 디스크의 마지막 논리적 섹터는 30336인 하나의 디스크로 인식한다.

또한 디스크의 개수가 3개이더라도 2개 디스크의 용량만 실질적인 데이터 비트가 들어가고 나머지 한 개의 디스크 용량은 패리티 비트가 차지하고 있다. 위 표에서 보면 한 개의 물리적 디스크 섹터 개수는 10112 개로서 실질적으로 2개의 데이터 디스크가 쓰이므로 20224가 되는 것이다. 또한 ZONE2의 경우는 한 개의 디스크 크기만이 실질적인

데이터 비트로 쓰이므로 20224+10112 = 30336 개로 표시된다. 참고로 주의해야 할 점은 밑의 그림 4와 같이 RAID5 특성상 한 개의 패리티 디스크가 따로 모여 디스크를 구성하는 것이 아니라 패리티 비트는 각각의 디스크에 나누어져 분산 되 저장된다는 것이다.



< Disk 2개 Add시 ZONE의 형태 >

그러나 이러한 방법에도 문제가 있다. 앞에서도 이야기 했듯이 두 개 이상의 디스크를 추가 시에는 각각의 독립적인 영역으로 구성하면 되었으나 하나의 디스크를 추가한다면 예외적인 상황이 발생한다.

3.5.2 Adding just ONE disk

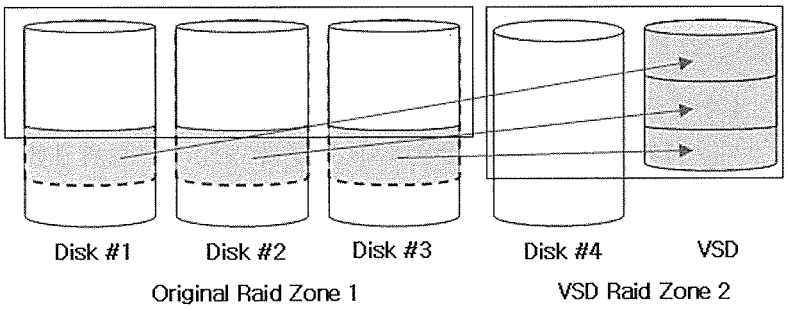
하나의 디스크가 추가된다면 Parity Bit를 어떻게 저장해야 하는지 고민할 필요가 있다. 하나의 Stripe 안에 Parity Bit Index를 두지 못한다면 이는 RAID5의 기본 코드마저 모두 바꿔야 하는 문제가 있다.

예를 들면 기존에 3개의 디스크가 있었고(이것은 ZONE1), 하나의 디스크를 추가했다고 가정하고 이것을 ZONE2로 두자. 이 때 ZONE2(Disk 하나)에 쓰이는 Data의 Parity Bit를 저장을 어디다 해야 하는지에 문제점이 발생한다는 것이다. 그렇다고 하나의 디스크를 반으로 나누어 한쪽은 패리티 데이터를 넣고 한쪽에 실질적 데이터를 넣는다고 한다면 해당 디스크가 물리적으로 문제가 발생하게 된다면 그 디스크 안의 모든 데이터는 사라지고 만다. 그 RAID5 특징을 그대로 살리면서 새로운 디스크를 추가하는 영역을 독립적인 영역으로 둘 수 있는 방법이 필요한 것이다. 이에 대한 해결방안에

서 두 가지 부분을 고민을 해야 할 필요가 있다.

첫째, Parity Bit Disk를 어떻게 둘 것인가.
디스크가 고장 났을 경우에도 복구가 가능하기 위해선 Parity를 Data 영역과 분리시킬 필요가 있다. 즉 하나의 디스크 안에 Data와 Parity를 쓸 수 없다는 결론에 이른다. 데이터의 Parity 정보를 분리시켜 저장해야 한다면 결국 어디일까? 답은 기존의 Disk들에 Parity를 쓰면 이에 대한 문제는 해결 할 수 있다. 즉 하나의 디스크를 추가했다면 그 전에 있던 디스크(위의 예에서는 ZONE1의 3개의 디스크가 되겠다)에 마지막 썼던 sector다음으로 Parity Bit를 쓰면 된다.

둘째, 기존의 RAID5의 Feature들을 어떻게 살릴 것인가.
RAID5 특징이란 추가했던 디스크 하나가 고장이 나도 혹은 기존의 있던 디스크 하나가 고장이 나도 서비스와 복구에 문제가 없어야 한다는 것이다. 좀 더 자세히 말한다면 하나의 Stripe에 Data Disk Index와 Parity Disk Index의 정보가 있어야 함을 의미하기도 한다. 정답은 하나의 디스크를 추가할 경우 마치 두 개의 디스크를 추가한 것과 같은 효과를 준다면 이 또한 해결이 가능하다.
즉 새로운 하드 디스크를 가상으로 만들어내는 것이다. 바로 기존에 만들어 놓은 RAID ZONE1에서 아직 데이터가 기록되지 않는 잔여 용량을 계산하여 추가된 디스크와 똑같은 크기로 떼어내어(물론 실질적으로 분리시키는 것이 아니라 가상으로 그 크기 부분을 계산하여 하나의 디스크로 보이게 한다) 가상의 장치. 즉 디스크로 만들게 된다면 이는 한 개의 하드 추가만으로도 두 개의 하드로 인식하여 새로운 RAID ZONE을 구성하게 되는 것이다.
기록 되지 않는 부분의 섹터를 모아 하나의 가상된 장치로 만드는 것. 우리는 이것을 VSD (Virtual Storage Disk)라고 명명했고, 이것이 물리적 디스크 하나를 추가할 때의 핵심적인 기술이다.



< VSD 구성 형태 >

위의 그림에서 기존 Original RAID ZONE에 하얀색 부분은 이미 Data bit와 Parity bit가 저장되어 있는 부분이다. 그리고 분홍색 부분은 바로 새로이 Add된 디스크의 용량의 크기(노란색으로 표시된 디스크)를 나누어 계산하여 하나의 가상된 장치(디스크) 즉, VSD로 구성된 부분이다. 새로이 Add된 Disk #4와 VSD는 새로운 RAID ZONE 2을 구성하게 된다. 물론 VSD에 쓰이는 데이터는 실질적으로 물리적 Disk #1,2,3에 나누어 저장되고 있다. 이러한 VSD를 이용하게 되면 하나의 Stripe에 Data Disk Index와 Parity Bit Disk Index 모두 저장해야 한다는 RAID5의 특징을 그대로 유지할 수 있게 되는 것이다. 그리고 마지막으로, 하늘색 부분의 데이터와 새로이 Add된 Disk #4의 남은 부분은 디스크 개수가 4개이므로 RAID ZONE3로 구성될 수 있다. (그림에서 표시는 제외하였다.)

< Add One Disk >

Zone	# of disks	First sector	Last sector	Phys first sector	Used sectors
1	3	0	4000	0	4000
2	1+1(VSD)	4000	6000	0	0
3	4	6000	10000	6000	0

위의 표는 기존의 3개의 디스크를 RAID5로 묶은 ZONE1에 같은 용량인 빈 한 개의 디스크를 추가할 경우 가상 저장 디스크(VSD)의 생

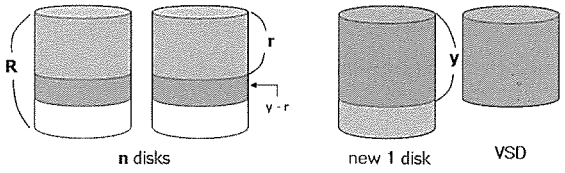
성된 ZONE Table을 보여주는 것이다. ZONE1의 디스크 3개중 새로 추가된 디스크 Disk #4와 RAID5를 구성할 수 있는 만큼의 용량을 기존 ZONE1의 Disk #1,2,3에서 1/3씩 떼어 내어 또 하나의 가상 디스크 하나를 만들어 내어 추가된 Disk #4와 합쳐져서 다시 RAID5를 구성하여, 기존 ZONE1의 마지막 섹터 이후에 추가된 RAID ZONE2의 섹터 공간을 linear하게 연결하게 된다. 기존 ZONE 1의 Disk #1,2,3의 나머지 부분과 새로 추가된 Disk #4에서 RAID5에 속한 영역을 제외한 부분을 다시 RAID5로 만들어 ZONE3로 만들고 이 역시 ZONE2의 마지막 섹터 이후에 linear하게 연결한다. 이렇게 생성된 각각의 RAID-5 ZONE을 모두 묶어서 커다란 하나의 디스크처럼 ZONE Table을 구성하게 된다. 참고로, ZONE Table에서 보는 바와 같이 ZONE1과 ZONE2는 기존 실제 디스크를 묶었던 것이기 때문에 Physical Sector는 0이 되게 되지만, ZONE3는 VSD를 만들고 남는 영역을 사용하였기 때문에 실제적인 Physical Sector는 0이 될 수 없다.

이와 같은 방법에도 한 가지 문제가 있다면 추가하려는 디스크의 크기가 그 전 ZONE의 디스크 크기 이상이 되어야 하는 상황이 발생한다. 그 이유에 대해서는 다음 섹터 계산에 대해서 자세히 설명 되어 있다.

- **How to Divide Sector Space**

위의 그림 <VSD의 구성형태>를 다시 살펴보면, 기존의 영역과 새로운 디스크 용량은 새로운 ZONE으로 그리고 나머지 Disk영역은(기존 디스크의 파랑색 부분과 새로운 디스크의 노랑색 부분)은 새로운 ZONE으로 처리 된다. 만약 새로 추가되는 디스크의 크기가 기존 디스크보다 작다면 ZONE은 몇 개로 나누어져야 하는가에 대한 문제가 남게 된다. 적어도 3개의 존으로 더 나누어지는 현상이 나타난다. 따라서 새로 추가되는 디스크의 용량은 적어도 기존의 디스크 용량과 같거나 그보다 커야 문제없음을 알 수 있다. (현재 시스템은 만약 잔여 용량이 기존의 디스크 용량보다 작다면 RAID5를 구성할 수 없으므로 디스크 추가가 불가능하다는 메시지를 출력한다.)

그렇다면 이제 새로 추가될 ZONE에 필요한 용량을 계산을 알아보자.



[그림 2-4] VSD

r = 사용중인 용량

y = 새로운 ZONE의 용량

(y - r) * n = y 이므로 정리하면 **y = r * n / (n - 1)**

R은 기존에 구성된 RAID안에 묶여 있는 디스크 한 개의 크기이며 또한 새로이 추가되는 디스크의 크기이다. 그리고 r은 슈퍼 블럭에 기록되어 있는 현재 디스크에 쓰인 용량의 크기이다.

y는 현재 우리가 구성하려는 VSD와 같은 크기의 실질적인 디스크이다. 각 디스크의 잔여 공간 중 일부를 모아 VSD를 만들어야 하기에 y의 크기는 n개의 디스크에 각각 (y-r) 만큼의 용량을 요구하게 된다. 즉 y = n*(y-r) 인 것이다.

이식을 y에 대한 식으로 재정리하게 되면 y = r*n/(n-1)이 된다. 우리는 각 디스크에 써진 용량 r과 기존에 RAID로 구성된 디스크의 개수 n을 알고 있기에 새로이 구성된 VSD의 크기를 계산할 수 있다.

이와 같이 디스크 수와 현재까지 사용한 용량을 알면 새로운 ZONE에 필요한 용량에 관한 공식을 얻을 수 있다.

4. 개발 단계별 기간 및 투입 인원수

기간	Project Schedule	비고
7월	1. Project 개발의 필요성을 느끼고 팀원 구성 2. 회의를 통하여 구체적인 개발 방향 및 목표 설정 - 블록 레벨에서 하나의 디스크 관리가 가능한 디바이스 드라이버 개발 목표 설정 3. 기존의 Hardware 및 Software Raid 구성 및 비교 테스트 4. ENBD 설치 및 테스트 5. Linux Block Device Driver Study 6. Object Orinted Programming in C Study 7. 작업환경 구축 및 프로그램 Design, 개발 분업 결정	
8월	1. SZIT 논문 Study 2. RAID ZONE 개념 도입 시도 3. RAID5 특성을 적용하고, 디스크 추가 기능 구현 설정 4. Hard Coding으로 Zone-table 구현 및 테스트 성공 5. Zonte-table 동적 구성 구현 6. Virtual Disk Storage 개념 도입 7. Coding 및 Debugging, SuperBlock 부분 수정 8. Filesystem 수정에 대한 필요성 인식	
9월	1. Filesystem 역할 해결 (ext2 파일시스템은 resizing이 가능) 2. 9월 한 달 동안에 걸친 필드 테스트 2. 프로그램 최종 테스트 및 Bench-Marking 3. Monitoring Program 개발 4. Project 문서 작성	

5. 사용 언어

개발언어 : C

빠르고 이식성이 강한 C를 선택하였다. 리눅스 환경에서 이루어질 것을 고려, 리눅스 커널 프로그래밍을 해야 하는 점, 기존의 RAID5에 새로운 기능을 추가하려고 했던 상황 등을 고려해서 C언어로 개발하였고, 그 외 User-level에서 디버그 및 모니터링을 할 수 있는 간단한 프로그램 모두 C에서 작성되었다. 그 외에 테스트를 위한 코드들은 대부분 shell-script로 되어 있다.

6. 사용 시스템

구분	이름	비고
사용모델	IBM PC	**3대**
CPU	Pentium 3 - 800	
RAM	256MB	
O S	Linux (Fedora) 2.6.12.2	
Network Card	10/100M LanCard	**4포트 허브 사용**
Compiler	gcc-4.0.0-8	

커널 버전은 기존 패키지 버전에서 개발 당시의 최신 버전인 2.6.12로 업그레이드 한 후 작업하였다. 따라서 RAIDZone-0.1.0은 Kernel 2.6.12 이상의 시스템에서 설치되기를 권장한다.

(2.6 이하 버전의 커널에서는 작동을 보장하지 않는다. 2.4 버전 및 그이하의 커널 과 2.6버전 이상의 커널에는 많은 변화가 있었다. 2.6버전의 커널에 최적화되어 있으므로 2.6버전 이하의 커널에서는 작동을 권하지 않는 바이다. 2.4 버전으로의 back-porting은 아직 고려하지 않고 있다.)