Logical Volume Manager HOWTO

bert hubert <ahu@ds9a.nl> Richard Allen <ra@ra.is>

도운이

정강훈

skyeyes (at) nextware.co.kr

skyeyes (at) soback.kornet.net

서성용

pooh (at) kldp.org

Version 0.0.2 $Date: 2002/07/15 01:55:00 $
V0.11, 6 June 1997

Linux LVM에 관한 간단한 HOWTO 문서.

1. 소개

독자 여러분을 환영한다.

이 문서는 LVM이란 무엇이고, 어떻게 작동하고, 여러분의 생활을 쉽게 할수 있도록 LVM을 사용할수 있는 방법에 관해 여러분에게 알려주는데 도움을 주기 위해 쓰여졌다. 현재 LVM FAQ와 German HOWTO도 있지만, 이 문서는 기존 문서와는 다른 면에서 쓰여 졌다. 이 문서는 매우 간단한 'HOWTO' 인 반면, 또한 이해도 줄 수 있다.(그러길 바란다.)

나는 Linux Logical Volume Manager 저자가 아님을 명백히 밝힌다. 나는 개발한 사람들을 많이 존경하며, 그들과 상호 협력하길 바란다.

매우 이상하겠만, 나는 LVM의 개발자들을 알지 못한다. 나는 이러한 상황이 곧 바뀌길 바란다. 개발자들의 기분이나 입장을 고려하지 못한 점에 대해서 미리 사과한다.

1.1. 권리 포기& 라이센스

1.2. 사전 지식(Prior knowledge)

그렇게 많이 필요하지 않다. 만일 여러분이 Linux를 설치한 경험이 있고 filesystem(fdisk/mkfs)을 만들었다면, 여러분은 모두 설정해야 된다. 루트로써 작업할때는 항상 신중 해라. 잘못된 명령어나 장치 파일들에 대한 어떠한 작업들은 존재하는 데이타를 손상시킬수 있다.

만일 여러분이 HP/UX LVM을 설정하는 방법을 안다면, 여러분은 이미 거의 한거나 마찬가지다. 리눅스는 HP 실행과 거의 같다.

1.3. 관리할때 주의할 점

이 문서에 관해 주의해야 할 몇 가지가 있다. 내가 이 문서의 대부분을 썼지만, 나는 실제로 이러한 방법으로 이 문서를 유지하길 원하지 않는다. 나는 Open Source에 대한 지지자이며, 여러분들의 feedback, 갱신, 패치등을 원한다. 오타나 에러들에 관해 나에게 알리는걸 주저하지 마라.

만일 여러분이 섹션을 유지하는데 더 좋은 자격이 있거나 새로운 섹션의 저자이고 유지할수 있다면, 그렇게 하길 환영한다. 이 HOWTO의 SGML문서는 CVS로 이용할 수 있다. 나는 이 작업이 상호 협동적인 프로젝트가 되길 바란다.

이러한 목적으로, 여러분은 FIXME의 많은 주의 사항들을 발견할 것이다. 패치들은 항상 환영한다. 여러분이 FIXME를 발견하는 곳이 어디든, 여러분은 여러분이 잘 알지 못하는 분야를 다루고 있다는 것을 알아야 한다. 이것은 그밖의 곳에 에러가 없다는것을 말하는 것은 아니다. 단지 주의하라는 것이다. 만일 여러분이 유효한 것을 가지고 있다면, 우리가 알수 있도록 해라. 그러면 나는 FIXME의 주의사항에서 제거할 것이다.

1.4. CVS 접근 & updates 하기

이 HOWTO 문서의 공식적인 위치는 http://www.ds9a.nl/lvm-howto/이다.

우리는 지금 anonymous CVS 접근을 이용할수 있다. 이것은 여러분에게 쉽게 이 HOWTO 문서의 최근 버전을 얻고 변경 사항들을 제공하기 위함이다.

여러분이 CVS를 통해 HOWTO의 복사본을 원한다면, 다음과 같이 해라.:

```
$ export CVSROOT=:pserver:anon@outpost.ds9a.nl:/var/cvsroot
$ cvs login
CVS password: [enter 'cvs' (without 's)]
$ cvs co lvm-howto
cvs server: Updating lvm-howto
U lvm-howto/lvm-howto.sgml
```

만일 여러분이 에러나 추가하길 원하는 것이 있다면, 로컬에서 수정하고 "cvs diff -u" 실행하고, 그 결과물을 우리에게 보내줘라.

Makefile은 여러분이 postscript, dvi, pdf, html과 text를 만드는 것을 도울 수 있는 것들을 제공한다. 여러분은 모든 문서 형식을 가지기 위해서는 sgml-tools, ghostscript, tetex를 설치할 필요가 있다.

1.5. 이 문서의 구조

우리는 처음에 해야 할 작업들에 관한 기본적인 사항들을 설명할 것이다. 그러나 우리는 이해에 도움을 줄수 있는 예제들도 포함 할 것이다.

2. LVM이란 무엇인가?

전통적으로, 파티션 크기는 고정적이다. 이것은 시스템 설치자는 "나는 이 파티션에 얼마나 많은 데이타를 저장할 것이다"가 아니라 "나는 이 파티션에 얼마 이상을 저장할 것이다"라는 질문을 고려할 것을 요구한다. 사용자가 파티션 공간을 모두 사용하였을때, 보통은 파티션을 다시 잡든지 (전체 운영 시스템의 reload를 의미한다.) 심볼릭 링크 같은 방법으로 해결을 한다.

파티션은 물리 디스크의 연속된 블럭들이다라는 생각은 계속 바뀌었다. 대부분의 유닉스 시스템들은 물리 디스크를 몇몇 단위(unit)들로 나눌수 있는 능력을 가지고 있다. 다중 드라이브에서 저장 단위들은 "논리적인 volume"으로 모아지고, 이것들이 파티션으로 할당된다. 추가적으로, 단위(unit)들은 원하는 공간처럼 파티션에서 추가되거나 제거될 수 있다.

이것이 논리적인 볼륨 관리자(LVM)의 기본적인 생각이다.

예를 들어, 여러분이 1G 디스크를 가지고 있고, 600MB를 사용할수 있는 "/home" 파티션을 만든다고 하자. 그리고 여러분이 할당한 공간을 모두 사용하였는데 "/home"에서 1G를 사용할 필요가 있다고 가정하자. 파티션의 예전 개념을 사용하면, 여러분은 1GB의 다른 드라이브를 가지길 원할 것이다. 그리고 여러분은 디스크를 추가하고, 새로운 "/home"을 만들고 현재 존재하는 데이타를 복사할 것이다.

그러나, LVM 설정으로, 여러분은 단순히 400MB(또는 더) 디스크를 추가할 수 있고, 저장 단위(unit)들을 "/home" 파티션에 추가할 수 있다. 다른 툴들은 지금의 파일 시스템을 재 조정할 수 있도록 허용하며, 여러분이 더 커다란 파티션 크기로 재 조정할 수 있고 원래의 비지니스로 돌아갈수 있다.

매우 특별한 경우로써, LVM은 이동할수 없는 타켓의 백업을 만들수 있도록 자체적인 "snapshots"을 만들수도 있다. 우리는 이러한 흥미로운 가능성으로 돌아가서, 이것은 다른 많은 실제 어플리케이션을 가진다.

다음 섹션에서 우리는 LVM의 기초를 설명하고 LVM이 사용하는 여러 추상적 개념에 대해서도 설명한다.

3. 기본 원리

여러분에게 겁을 주기 위해서가 아니라, LVM은 여러분의 파일 시스템을 위험하게 하지 않도록 하기 위한 용어에서 왔다.

다소, 밑바닥 부터 시작하자.

물리적 미디어

우리가 단순히 하드 디스크나 파티션을 가정하였다 할지라도, 여러분은 어림 잡아서 '물리적' 이라는 말을 이해해야 한다. 예를 들어, /dev/hda, /dev/hda6, /dev/sda. 여러분은 블럭 장치의 연속적인 블럭 수들을 바꿀수 있다.

물리적 볼륨(Volume) (PV)

PV는 단지 여기에 추가된 관리 데이타를 가지는 물리적 미디어이다. -- 일단 여기에 추가하면, LVM은 이것을 소유한 것처럼 인식한다.

물리적 확장(PE)

물리적 확장(Physical Extents)은 메가 바이트 크기를 가지는 큰 블럭같은 것이다. PEs는 할당될 수 있다.

볼륨 그룹(Volume Group)

VG는 물리적 확장의 수(여러 물리적 볼륨이나 하드 드라이브가 기본인)로 이루어 진다. 이것을 여러 하드 드라이브(예를 들어, /dev/hda 와 /dev/sda)로 이루어져 있는 것 같은 VG로 생각할수 있지만, 이것은 이들 하드 드라이브가 제공하는 PE들을 포함한다고 말하는 것이 더 정확하다.

>From this Volume Group, PEs can be assigned to a ...

논리적 볼륨(LV)

우리는 마지막으로 갖는 것이 있다. 논리적 볼륨은 모든 작업의 결과이며 우리는 정보를 여기에 저장한다. 이것은 파티션에 대한 생각과 동일한 것이다.

정규 파티션처럼, 논리적 볼륨은 전형적으로 만들어 진다.

파일 시스템

이 파일 시스템은 여러분이 원하는 모든 것이다.: 표준 ext2, ReiserFS, NWFS, XFS, JFX, NTFS 등등. 리눅스 커널에서, 정규 파티션과 논리적 볼륨사이에는 아무런 차이가 없다.

나는 여러분이 쉽게 이것을 볼수 있도록 하기 위해 ASCII 챠트로 만들었다.

```
물리적 확장을 포함한, 물리적 볼륨:

  +-----[ Physical Volume ]------+
  | PE | PE | PE | PE | PE | PE  |
  +------------------------------+

6개의 물리적 확장과 2개의 물리적 볼륨(PVs)를 포함한 볼륨 그룹:

  +------[ Volume Group ]-----------------+
  |  +--[PV]--------+  +--[PV]---------+  |
  |  | PE | PE | PE |  | PE | PE | PE  |  |
  |  +--------------+  +---------------+  |
  +---------------------------------------+

우리는 여기에 더 추가 확장을 하였다.:

  +------[ Volume Group ]-----------------+
  |  +--[PV]--------+  +--[PV]---------+  |
  |  | PE | PE | PE |  | PE | PE | PE  |  |
  |  +--+---+---+---+  +-+----+----+---+  |
  |     |   |   | +-----/     |    |      |
  |     |   |   | |           |    |      |
  |   +-+---+---+-+      +----+----+--+   |
  |   |  Logical  |      |  Logical   |   |
  |   |  Volume   |      |   Volume   |   |
  |   |           |      |            |   |
  |   |  /home    |      |    /var    |   |
  |   +-----------+      +------------+   |
  +---------------------------------------+
```

이것은 두 디스크에 걸친 두 파일 시스템을 우리에게 보여준다. /home 파일 시스템은 4개의 물리적 확장을, /var 파일 시스템은 2개의 물리적 확장을 포함한다.

bert hubert는 더욱 시각적으로 LVM을 보여주기 위해 툴 (http://ds9a.nl/lvm-viewer) 을 만들었다. screenshot (http://ds9a.nl/lvm-howto/screenshot.gif)도 있다. ASCII 차트보다 더 좋게 보인다.

3.1. 보여주기& 말하기

이 부분은 이해하기 어렵다. 그래서 논리적 볼륨을 만드는 예제에 주석을 달았다. 이 예제를 콘솔에 복사하지 마라. 왜냐하면 만일 여러분의 컴퓨터가 /dev/hda3와 /dev/hdb2를 사용하지 않는다면, 여러분의 데이타를 파괴하기 때문이다.

의문스럽다면, 위의 ASCIIgram을 봐라.

여러분은 /dev/hda3와 /dev/hdb2 파티션 타입을 0x8e, 즉 'Linux LVM'으로 설정해야 한다. fdisk의 버전이 이 타입을 아직 알지 못해 'Unknown'으로 나오는지 확인해라.:

```
# fdisk /dev/hda

Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 623 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1         2     16033+  83  Linux
/dev/hda2             3       600   4803435   83  Linux
/dev/hda3           601       607     56227+  83  Linux
/dev/hda4           608       614     56227+  83  Linux

Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): 8e

Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 623 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1         2     16033+  83  Linux
/dev/hda2             3       600   4803435   83  Linux
/dev/hda3           601       607     56227+  8e  Unknown
/dev/hda4           608       614     56227+  83  Linux

Command (m for help): w
```

우리는 /dev/hdb2도 했지만, 여기서는 보여주지 않았다. 이것은 LVM이 여러분의 설정을 잃은 것들을 재구성하기 위해 필요하다.

지금, 이것이 필요하지는 않지만, 몇몇 컴퓨터는 여기서 재부팅을 요구하기도 한다. 그래서 만일 다음 예제가 제대로 작동하지 않는다면, 재 부팅해라.

그리고, 우리는 다음처럼 물리적 볼륨을 만든다.:

```
# pvcreate  /dev/hda3
pvcreate -- physical volume "/dev/hda3" successfully created
# pvcreate  /dev/hdb2
pvcreate -- physical volume "/dev/hdb2" successfully created
```

그리고, 우리는 이들 두개의 PVs를 'test'라 불리는 볼륨 그룹에 추가한다:

```
# vgcreate test /dev/hdb2 /dev/hda3
vgcreate -- INFO: using default physical extent size 4 MB
vgcreate -- INFO: maximum logical volume size is 255.99 Gigabyte
vgcreate -- doing automatic backup of volume group "test"
vgcreate -- volume group "test" successfully created and activated
```

그래서, 우리는 빈 볼륨 그룹을 가지게 되며, 이제 비트(bit)를 검사하도록 하자.

```
# vgdisplay -v test
--- Volume group ---
VG Name               test
VG Access             read/write
VG Status             available/resizable
VG #                  0
MAX LV                256
Cur LV                0
Open LV               0
MAX LV Size           255.99 GB
Max PV                256
Cur PV                2
Act PV                2
VG Size               184 MB
PE Size               4 MB
Total PE              46
Alloc PE / Size       0 / 0
Free  PE / Size       46 / 184 MB

--- No logical volumes defined in test ---


--- Physical volumes ---
PV Name (#)           /dev/hda3 (2)
PV Status             available / allocatable
Total PE / Free PE    13 / 13

PV Name (#)           /dev/hdb2 (1)
PV Status             available / allocatable
Total PE / Free PE    33 / 33
```

여기에 있는 많은 데이타들 - 이 데이타중 대부분은 지금 이해해야 한다. 우리는 여기에 정의된 어떠한 논리적 볼륨도 없어서 이걸 치료해야 한다. 우리는 볼륨 그룹 'test'에 'HOWTO'라 불리는 50 메가 바이트 볼륨을 만들도록 한다.:

```
# lvcreate -L 50M -n HOWTO test
lvcreate -- rounding up size to physical extent boundary "52 MB"
lvcreate -- doing automatic backup of "test"
lvcreate -- logical volume "/dev/test/HOWTO" successfully created
```

자, 여기서 파일 시스템을 만들도록 하자.

```
# mke2fs /dev/test/HOWTO
mke2fs 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
13328 inodes, 53248 blocks
2662 blocks (5.00%) reserved for the super user
First data block=1
7 block groups
8192 blocks per group, 8192 fragments per group
1904 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
# mount /dev/test/HOWTO /mnt
# ls /mnt
lost+found
```

다 했다. 이제 볼륨 그룹을 다시 보도록 해라. 왜냐하면, 지금쯤 비트(bit)가 채워져야 하기 때문이다.

```
# vgdisplay test -v
--- Volume group ---
VG Name               test
VG Access             read/write
VG Status             available/resizable
VG #                  0
MAX LV                256
Cur LV                1
Open LV               1
MAX LV Size           255.99 GB
Max PV                256
Cur PV                2
Act PV                2
VG Size               184 MB
PE Size               4 MB
Total PE              46
Alloc PE / Size       13 / 52 MB
Free  PE / Size       33 / 132 MB

--- Logical volume ---
LV Name               /dev/test/HOWTO
VG Name               test
```

```
LV Write Access        read/write
LV Status              available
LV #                   1
# open                 1
LV Size                52 MB
Current LE             13
Allocated LE           13
Allocation             next free
Read ahead sectors     120
Block device           58:0


--- Physical volumes ---
PV Name (#)            /dev/hda3 (2)
PV Status              available / allocatable
Total PE / Free PE     13 / 13

PV Name (#)            /dev/hdb2 (1)
PV Status              available / allocatable
Total PE / Free PE     33 / 20
```

자, 됐다. /dev/hda3는 완전히 사용되지는 않았지만, /dev/hdb2는 13개의 물리적 확장을 사용하고 있다.

3.2. 활성화(Active)와 비 활성화(Inactive): 커널 영역과 유저 영역

모든 운영체제처럼, Linux도 두 부분으로 나누어져 있다.:커널 영역과 유저 영역. 유저 영역은 가끔 userland라 불리며, 이것은 'Userland'를 위한 좋은 이름이기도 하다.

논리적 볼륨 관리를 포함하는 복구, 생성과 수정과 같은 것들은 유저 영역에서 행해 지며, 그리고 커널과 통신한다. 일단 볼륨 그룹과 논리적인 볼륨이 커널에 보고되면, '활성화'라 불린다. 어떤 변화들은 엔터티(entity)가 활성화될때만 이루어지며, 어떤 것들은 비 활성화되었을 때 이루어진다.

4. 필요 조건

LVM을 이용할수 있는 커널 범위가 넓다. Linux 2.4에서, LVM은 완전히 통합되었다. 커널 2.3.47 이후에서, LVM은 메인 커널로 통합되는 과정에 있다.

4.1. Kernel

4.1.1. Linux 2.4

이 버전은 여러분이 필요한 모든것을 포함하고 있다. 대부분의 배포판은 LVM을 모듈로써 가지고 릴리즈되었다. 만일 여러분이 컴파일할때, 여러분의 블럭 장치들을 선택할때 LVM 옵션을 알리면 된다.

4.1.2. Linux 2.3.99.*

이 버전의 커널이 안정화되면, 이 섹션은 없어질 것이다.

우리가 이 문서를 쓸때, Linux 2.3.99pre5가 최신 버전이며 이 버전에서 LVM을 작동시키기 위해서는 패치가 필요하다.

Linux 2.3.99pre3에서, 두 패치가 릴리즈되었다.:

패치는 linux-kernel로 포스팅되었고, 여기서 (http://ds9a.nl/lvm-howto/2.3.99pre3) 이용할수 있다.

Andrea Arcangeli는 이 패치를 향상시키고, 향상된 패치 (http://ds9a.nl/lvm-howto/andrea.patch)에 적용하였으며, 이 패치는 2.3.99pre3 LVM 패치에 적용되어야 한다.

Linux 2.3.99pre5에서, bert hubert는 두 패치를 하나로 하고 2.3.99pre5에 포팅했다. Patch (http://ds9a.nl/lvm-howto/2.3.99-pre5.lvm.patch). 주의하면서 사용해라.

prepatch에 대한 prerelease인 2.3.99pre6-1는 최초로 완전한 LVM을 지원한다. 이 버전도 여전히 Andreas 패치가 적용되지 않았지만, 곧 릴리즈 될 가장 앞선 순위에 있다.

2.3.99pre4-ac1 는 기본적으로 LVM 패치가 되어 있으며, 작동한다. 그렇지만 Andreas 패치는 포함하지 않는다.

4.1.3. Linux 2.2

FIXME: 이 부분을 채워라.

4.1.4. Linux 2.3

FIXME: 이 부분을 채워라.

4.2. Userspace

여러분은 LVM 사이트 (http://lvm.msede.com/lvm)에서 필요한 툴들을 이용할수 있다. glibc2.2 시스템에서 이 툴들을 컴파일할려면 패치가 필요하며, Debian 2.2에서는 패치를 해도 에러가 발생한다.

5. 파일 시스템 늘리기

여러분은 제공되는 스크립트로 이것을 할수 있으며, 필요하다면 직접 손으로 할수도 있다.

5.1. e2fsadm

만일 여러분의 볼륨 그룹에 공간(room)이 있고 ext2 파일 시스템을 사용한다면, 여러분은 이 툴들을 사용할수 있다.

`e2fsadm` 명령어는 상업적인 resize2fs 툴을 사용한다. 이게 좋은 소프트웨어라고 느끼지만, 범용적이지는 않다.

여러분이 FSF의 `ext2resize` 명령어를 사용하길 원한다면, 여러분은 `e2fsadm`를 알려 줄 필요가 있다.:

```
# export E2FSADM_RESIZE_CMD=ext2resize
# export E2FSADM_RESIZE_OPTS=""
```

나머지는 쉽다. `e2fsadm`는 다른 LVM 명령어들과 많이 비슷하다.:

```
# e2fsadm /dev/test/HOWTO -L+50M
e2fsadm -- correcting size 102 MB to physical extent boundary 104 MB
e2fsck 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/test/HOWTO: 11/25688 files (0.0% non-contiguous), 3263/102400 blocks
lvextend -- extending logical volume "/dev/test/howto" to 104 MB
lvextend -- doing automatic backup of volume group "test"
lvextend -- logical volume "/dev/test/HOWTO" successfully extended

ext2_resize_fs
ext2_grow_fs
ext2_block_relocate
ext2_block_relocate_grow
ext2_grow_group
ext2_add_group
ext2_add_group
ext2_add_group
ext2_add_group
ext2_add_group
ext2_add_group
direct hits 4096 indirect hits 0 misses 1
e2fsadm --
ext2fs in logical volume "/dev/test/HOWTO" successfully extended to 104 MB
```

5.2. 논리적 볼륨 늘리기

`e2fsadm` 명령어는 이 부분을 다룬다. 그러나, 이것을 하는 방법을 이해하는게 유용하다.:

만일 여러분이 볼륨 그룹안에 공간(room)을 가지고 있다면, 이것은 한 라이너(liner)이다.

```
# lvextend -L+12M /dev/test/HOWTO
lvextend -- rounding size to physical extent boundary
lvextend -- extending logical volume "/dev/test/HOWTO" to 116 MB
lvextend -- doing automatic backup of volume group "test"
lvextend -- logical volume "/dev/test/HOWTO" successfully extended
```

5.3. 볼륨 그룹 늘리기

이것은 vgextend 유틸리티로 하며 쉽다. 여러분은 우선, 물리적 볼륨을 만들 필요가 있다. 이것은 `pvcreate` 유틸리티로 할수 있다. 이 툴로, 여러분은 어떤 블럭 장치를 물리적 볼륨으로 바꿀수 있다.

이것을 한뒤, `vgextend`가 나머지를 한다:

```
# pvcreate /dev/sda1
pvcreate -- physical volume "/dev/sda1" successfully created
# vgextend webgroup /dev/sda1
vgextend -- INFO: maximum logical volume size is 255.99 Gigabyte
vgextend -- doing automatic backup of volume group "webgroup"
vgextend -- volume group "webgroup" successfully extended
```

이것을 하기 위해, 볼륨 그룹은 활성화 될 필요가 있다는 점에 주의해라. 여러분은 'vgchange -a y webgroup'을 실행하여 이것을 할수 있다.

5.4. 파일 시스템 늘리기

만일 여러분이 매뉴얼대로 이것을 하길 원한다면, 여기에는 몇가지 방법이 있다.

5.4.1. ext2 오프라인시 ext2resize 사용하기

오프 라인이라는 것은, 여러분이 파일 시스템을 늘리는 작업을 하기 위해 파일 시스템을 언마운트하는 것을 의미한다. 파일 시스템과 데이타는 이 작업동안은 이용할수 없다. 만일 루트 크기나 다른 중요한 파티션의 크기를 확장한다면, 여러분은 다른 부트 미디어를 사용해야 한다.

ext2resize 툴은 GNU ftp 사이트에서 이용할수 있지만, 대부분의 배포판은 패키지로서 제공한다. 문법은 매우 명확하다.:

```
# ext2resize /dev/HOWTO/small 40000
```

40000은 파일 시스템을 늘리거나 줄여야하는 블럭 수이다.

5.4.2. ext2 on-line

FIXME: 이 부분을 채워라.

6. 디스크 교체하기

이것은 LVM 의 장점중의 하나이다. 한번 디스크에서 에러가 발견되기 시작하면, 자료를 이동시켜야 할 적절한 시기이다. LVM 을 이용하면 이것은 매우 쉽다. 먼저 확실한 교체 예제를 들도록 보도록 하는데, 이것은 당신이 적어도 당신이 교체하고 하는 것만큼의 용량을 가진 디스크를 시스템에 추가하는 것이다.

자료를 이동하기 위해서는, Volume Group 의 Physical Extents 를 다른 디스크로 이동하는데, 보다 정확하게 말하면, 다른 Physical Volume 으로 이동하는 것이다. 이것을 위해서 LVM 은 `pvmove` 유틸리티를 제공한다.

우리의 의심스런 디스크는 /dev/hda1 이고 그것을 /dev/sdb3 로 교체하려고 한다고 가정하자. 먼저 /dev/sdb3 를 /dev/hda1 을 포함하는 Volume Group 에 추가한다.

이것을 하기 전에 이 볼륨 그룹에 있는 어떠한 파일시스템이라도 언마운트 하는 것이 좋을 것 같다. 풀 백업 또한 손해보지는 않을 것이다.

FIXME: 이것이 필요할까?

그리고 나서 `pvmove`을 실행한다. 가장 간단한 사용법에서는 단지 제거하고자 하는 디스크만을 언급한다. 다음과 같다:

```
# pvmove /dev/hda1
pvmove -- moving physical extents in active volume group "test1"
pvmove -- WARNING: moving of active logical volumes may cause data loss!
pvmove -- do you want to continue? [y/n] y
pvmove -- doing automatic backup of volume group "test1"
pvmove -- 12 extents of physical volume "/dev/hda1" successfully moved
```

이 경고를 주의하기 바란다. 또한, 적어도 어떤 커널이나 LVM 버전은 이 명령과 문제가 있는 것으로 보인다. 필자는 2.3.99pre6-2 로 테스트했고, 동작은 했지만, 경고를 받았다.

이제 더이상 /dev/hda1 이 Physical Extents 를 갖고 있는 않으므로, 그것을 볼륨 그룹에서 제거할 수 있다.

```
# vgreduce test1 /dev/hda1
vgreduce -- doing automatic backup of volume group "test1"
vgreduce -- volume group "test1" successfully reduced by physical volume:
vgreduce -- /dev/hda1
```

FIXME: 몇가지에 대해 명확히 할 필요가 있다. 볼륨 그룹이 활성화되어야 하는가? 언제 데이터를 잃게 될까?

6.1. 너무 늦었을 때

만약 디스크가 경고 없이 고장났고 물리적 확장(PE) 를 다른 물리적 볼륨(PV) 로 옮길 수 없다면, 문제가 생긴 PV 에 있는 논리적 볼륨(LV)이 미러되고 있지 않는 한은 자료를 잃게 될 것이다. 취해야 할 조치의 정확한 방법은 문제가 생긴 PV 를 동일하거나 적어도 같은 크기의 파티션으로 교체하는 것이다.

/etc/lvmconf 디렉토리에는 디스크들을 물리적 볼륨(PV) 으로 만드는 LVM 자료와 스트럭쳐들과 물리적 볼륨이 어느 볼륨 그룹에 속해 있는지, 볼륨 그룹에는 어떤 논리적 볼륨이 있는지에 대한 백업을 담고 있다.

고장난 디스크를 교체하고 난 후에는 `vgcfgrestore` 명령어를 사용하여 LVM 자료를 새로운 PV 에 복구할 수 있다. 이것은 볼륨 그룹과 그것의 모든 정보를 복구하지만, 논리적 볼륨에 있던 자료들은 복구하지 않는다. 이것이 대부분의 LVM 명령들이 변화가 생길때 자동으로 LVM 자료를 백업하는 이유이다.

7. 완벽한(consistent) 백업을 위한 스냅샷 만들기

이것은 보다 믿을수 없는 기능중의 하나이다. 당신에게는 많은 작업을 하고 있는 바쁜 서버가 있다고 하자. 유용한 백업을 위해서는, 많은 프로그램들을 셧다운 해야 하는데, 그렇지 않으면 자료가 백업당시와 변동된 상태로 끝나기 때문이다.

표준적인 예제는 파일을 /tmp 에서 /root 로 옮기는 것인데, /root 는 첫번째로 백업되는 곳이다. /root 가 읽혀졌을때, 파일은 아직 거기에 있지 않다. /tmp 가 백업될때는, 그 파일은 그곳에 없다.

또다른 예로는 데이터베이스나 디렉토리를 저장하는 것이 있다. 우리가 완전한 셧다운을 할 시간을 어플리케이션에 주지 않는 한은, 파일이 사용가능한 상태에 있는지 확인할 수 있는 단서가 없다.

또다른 문제가 생길 수도 있다. 우리는 어플리케이션을 셧다운하고, 백업을 하고, 어플리케이션을 다시 시작한다. 이것은 백업이 단 몇분만에 된다면 괜찮지만, 만약 여러 시간이 걸리거나 얼마나 오래 걸릴지 확신할수 조차 없다면 정말로 골치가 아프다.

LVM 은 이에 대한 해결책이다.

LVM 을 이용하여 논리적 볼륨에 대한 즉각적인 스냅샷 사진을 찍고, 그것을 마운트해서 그에 대한 백업을 만들 수 있다.

이렇게 해보자:

```
# mount /dev/test/HOWTO /mnt
# echo > /mnt/a.test.file
```

```
# ls /mnt/
a.test.file  lost+found
# ls -l /mnt/
total 13
-rw-r--r--    1 root     root            1 Apr  2 00:28 a.test.file
drwxr-xr-x    2 root     root        12288 Apr  2 00:28 lost+found
```

좋아, 이제 작업할 것이 생겼다. 스냅샷을 만들어보자:

```
# lvcreate --size 16m --snapshot --name snap /dev/test/HOWTO
lvcreate --
WARNING: all snapshots will be disabled if more than 16 MB are changed
lvcreate -- INFO: using default snapshot chunk size of 64 KB
lvcreate -- doing automatic backup of "test"
lvcreate -- logical volume "/dev/test/HOWTO" successfully created
```

'--size' 파라미터는 나중에 더 자세히 다루겠다. 스냅샷을 마운트하자:

```
# mount /dev/test/snap /snap
# ls /snap
total 13
-rw-r--r--    1 root     root            1 Apr  2 00:28 a.test.file
drwxr-xr-x    2 root     root        12288 Apr  2 00:28 lost+found
```

이제 원본으로부터 a.test.file 을 지우고, 스냅샷에 여전히 그것이 있는지 확인해보자:

```
# rm /mnt/a.test.file
# ls /snap
total 13
-rw-r--r--    1 root     root            1 Apr  2 00:28 a.test.file
drwxr-xr-x    2 root     root        12288 Apr  2 00:28 lost+found
```

놀라운 일이군!

7.1. 그것이 어떻게 작동하는가?

우리가 '--size' 파라미터를 설정해야 했음을 기억하는가? 실제로 일어나는 것은 'snap' 볼륨이 모든 블럭들의 사본, 혹은 LVM 이 그것들을 부르는 이름인 'chunks' 를 가지기를 필요로 하는 것인데, 이것은 원본에서 변경된 것이다.

우리가 a.test.file 을 삭제했을때, 그것의 inode 가 삭제되었다. 이것은 64KB를 'dirty' 상태로 표기하게 만들고 - 원본 자료의 사본은 'snap' 볼륨에 쓰여졌다. 이 경우에 우리는 스냅샷에 16MB 를 할당했고, 그래서 만약 16MB 이상의 'chunks' 가 수정되었다면, 스냅샷은 비활성화 될 것이다.

스냅샷 파티션에 대한 정확한 크기를 결정하기 위해서는, 프라이머리 LV 의 사용 경향과 스냅샷에 활성화될 시간의 길이에 기반해서 추측해야 한다. 예를 들어, 아무도 시스템을 사용하지 않는 한밤중에 하는 한시간짜리 백업은 공간을 거의 필요로 하지 않을 것이다.

스냅샷이 persistent 하지 않다는 점에 유의하라. 만약 LVM 을 unload 하거나 리부트한다면, 그것들은 사라지고, 다시 만들어져야 한다.

8. 여분(redundancy)과 성능

성능상의 이유로, 여러개의 디스크에 'stripe' 로 자료를 분산해두는 것이 가능 하다. 이것은 블럭 1이 물리적 볼륨 A 에, 블럭 2가 물리적 볼륨 B 에 있고, 다시 블럭 3이 물리적 볼륨 A 에 있음을 의미한다. 또한 2개 이상의 디스크에 스트라이 프 할 수도 있다.

이러한 배열은 더 많은 디스크 대역폭을 이용가능함을 의미한다. 또한 보다 많은 'spindles' 가 포함된다. 뒤에서 더 자세히 다루겠다.

성능을 향상시킬 뿐 아니라, 자료의 사본을 여러개의 디스크에 보관하는 것도 가능 하다. 이것은 미러링(mirroring) 이라고 불린다. 현재, LVM 자체에서는 이것을 지원하지 않지만, 미러링을 할 수 있는 방법은 있다.

8.1. 왜 스트라이프인가?

디스크 성능은 적어도 세가지 요소에 의해 영향을 받는다. 가장 명백한 것은 디스크에서 자료가 순차적으로 읽혀지거나 쓰여지는 속도이다. 이것은 SCSI/IDE 버스에서 그것에 물려있는 단일 디스크에서 파일을 읽거나 쓸때 제한 요소이다.

그 후에는 디스크로 이용가능한 대역폭이 있다. 한개의 SCSI 버스에 7개의 디스크 가 있다면, 이것은 디스크 자체의 쓰기 속도보다 작을수도 있다. 만약 충분한 메모리를 사용한다면, 이러한 병목점이 문제가 되는것을 막을수도 있다.

그리고 레이턴시도 있다. as the saying goes, 레이턴시는 언제나 나쁜 소식이다. 그리고 더 안좋은 것은, 레이턴시를 낮추기 위해서는 더 많은 돈을 쓸 수도 없다 는 것이다. 오늘날 대부분의 디스크들은 7ms 정도의 레이턴시를 갖는 것으로 보인 다. 그 뒤에는 SCSI 레이턴시도 있는데, 25ms 정도가 된다.

FIXME: 최근의 수치들이 필요하다!

이것은 무엇을 의미할까? 합쳐진 레이턴시가 전형적인 경우엔 30ms 근방이 될 것이라는 것이다. 그래서 초당 33 번 정도의 디스크 작업만을 수행할 수 밖에 없다. 만약 초당 수천번 이상의 쿼리를 할 수 있기를 원하지만, 거대한 캐쉬를 갖고 있지 못할 경우라면, 당신은 매우 운이 없는 것이다.

만약 병렬로 동작하는 여러개의 디스크나, 'spindles' 를 갖고 있다면, 동시에 여러개의 명령어를 실행시킬 수 있는데, 이것은 훌륭하게 레이턴시 문제를 피해가는 방법이다. 어떤 어플리케이션들은, 거대한 뉴스 서버와 같은 것들은, 스트라이핑이나 다른 IO 현명함(smartness) 없이는 더이상 동작할 수 없다.

이것이 스트라이핑이 하는 것이다. 만약 당신의 버스가 그것에 도달한다면 (if your bus is up to it), 순차적인 읽기와 쓰기조차도 빨라질 수 있다.

8.2. 사용해서 안되는 경우는

추가의 방법이 수반되지 않는 스트라이핑은 실패 확률을, '비트 당'으로 증가 시킨다. 만약 당신의 디스크중 어떤 것이라도 고장난다면, 전체의 논리적 볼륨이 사라져버린다. 만약 단순히 데이터를 연결하기(concatenate)만 한다면, 파일 시스템의 일부분만을 잃게 될 것이다.

궁극의 선택은 미러되는 스트라이프이다.

FIXME: 미러되는 스트라이프를 LVM 과 md 로 만들라

8.3. LVM 자체 스트라이핑

스트라이프 설정을 지정하는 것은 lvcreate 로 논리적 볼륨을 생성할때 완료된다. 그중에는 두가지 관련있는 파라미터가 있다. -i 를 이용해 LVM 이 얼마나 많은 물리적 볼륨을 분산시켜 사용해야 하는지를 지시할 수 있다. 스트라이핑은 실제로 bit-by-bit 기반으로 행해지지는 않으며, 블럭상에서 일어난다. -I 로는 킬로바이트 단위로 설정할 수 있다. 이것은 2의 거듭제곱 형태가 되어야 함과, 가장 조잡한 뭉치회는 128Kbyte 임을 유의하라.

예제:

```
# lvcreate -n stripedlv -i 2 -I 64 mygroup -L 20M
lvcreate -- rounding 20480 KB to stripe boundary size 24576 KB / 6 PE
lvcreate -- doing automatic backup of "mygroup"
lvcreate -- logical volume "/dev/mygroup/stripedlv" successfully created
```

8.3.1. 성능상의 유의점

만약 같은 디스크에서 2개 이상의 파티션에 대해 스트라이프를 한다면, 성능 '이득' 은 음이 될 수도 있다 - 그렇게 하지 않도록 유의하라. 하나의 IDE 버스에 연결된 두개의 디스크로 스트라이핑을 하는 것 역시 쓸모없는 것으로 보인다 - 내가 기억 하는 것 이상으로 IDE 가 발전해오지 않은 한은 그렇다.

FIXME: 지금도 여전히 그럴까?

오래된 마더보드들은 두개의 IDE 버스를 갖고 있을 것인데, 두번째 버스는 느린 씨디롬 드라이브를 사용하는데 할당되었을 것이다. 여러가지의 툴을 이용해서 벤치마크를 수행할 수 있는데, 가장 주목할만한 것은 'Bonnie' 이다. ReiseFS 개발자들은 Bonnie++ (http://www.coker.com.au/bonnie++/) 를 발표했는데 성능 자료를 측정하기 위해 사용할 수 있다.

8.4. Hardware RAID

많은 하이엔드 인텔 x86 서버들은 하드웨어 RAID 컨트롤러를 갖고 있다. 그것들의 대부분은 적어도 2개의 독립적인 SCSI 채널을 갖고 있다. 다행히도, 이것들은 LVM 에 거의 관계가 없다. Linux 가 그러한 컨트롤러에 관한 것을 알 수 있기 전에 관리자는 raid 컨트롤러 자체 안에서 논리적 드라이브를 결정해야 한다. 예를 들어 [SCSI 채널 A에 있는 두개의 디스크를 스트라이프로 묶어서, 채널 B 에 있는 두개의 디스크에 그것들을 미러할 수 있다. 이것은 전형적인 성능과 데이터 안정성 을 최대화하는 전형적인 레이드 0/1 설정이다. 이렇게 설정된 시스템에서 리눅스가 부팅될 때 리눅스는 레이드 컨트롤러에 있는 오직 하나의 디스크만을 ’볼’ 수 있으 며, 이 디스크는 레이드 0/1 스트라이프셋에서 네개의 디스크를 포함하고 있는 논리적 드라이브이다. 이것은, LVM 에 관련해서는, 머신에 오직 하나의 디스크만 존재하며, LVM 에서도 역시 그렇게 사용됨을 의미한다. 만약 디스크중의 하나가 고장 나더라도, LVM 은 알지도 못할 것이다. 관리자가 디스크를 교체할 때(심지어 핫스왑 하드웨어를 가진 것을 즉시(on the fly) 교체하더라도), LVM 은 그것을 알지 못할 것이고, 컨트롤러가 미러된 자료를 재동기화(resync) 하고, 모든것이 좋은 상태로 돌아올 것이다. 이것은 대부분의 사람들이 한걸음 뒤로 물러서서 "그렇다면 이 레이드 컨트롤러를 이용하여 LVM 이 나에게 어떤 도움이 될까요?" 라고 물어보는 것이다. 간단한 답변은, 대부분의 경우에, 당신이 레이드 컨트롤러에서 논리적 드라이브를 정의한 이후에는, 더 이상의 디스크를 그 드라이브에 추가할 수 없다는 것이다. 그래서 만약 당신이 공간 요구량을 잘못 계산하거나 단지 더 많은 공간을 추가로 필요로 할 뿐이라면, 이미 존재하는 스트라이프셋에 새로운 디스크나 디스크의 집합 을 추가할 수 없다. 이것은 당신이 컨트롤러에서 새로운 레이드 스트라이프셋을 생성 해야 하고, 그 후에는 LVM 을 이용해서 단순히 LVM 논리 볼륨을 확장할 수 있으며, 따라서 빈틈없이 레이드 컨트롤러에 있는 두개의 스트라이프셋을 모두 확장하는 것을 의미한다.

FIXME: 이 주제에 대해 더 많은것이 필요한가?

8.5. Linux software RAID

리눅스 2.4에는 매우 훌륭한 레이드가 있다. 리눅스 2.2에서는 기본값으로, Alan Cox 에 의해 릴리즈 된 것에는, 잘 고려되지 않는 이전의 레이드 버전을 특징이다. 2.2 가 여전히 오래된 릴리즈를 특징으로 삼고 있는 이유는 커널 개발자들이 안정버전에서 사용자영역(userland) 업데이트를 필요로 하는 변화를 원하지 않기 때문이다.

Red Hat, Madrake, SuSE 를 포함한 대부분의 사람들은, 그것을 훨씬 좋아보이는 0.90 버전으로 교체하기로 결정했다.

우리는 여기서 오직 0.90 버전만을 다룰 것이다.

FIXME: 이것에 대해 더 많이 써야 한다

9. 상세 설명

9.1. 컴퓨터간에 LVM 디스크 옮기기

이 모든 새로운 기법에서는, 한 머신에서 다른 머신으로 디스크를 옮기는 것과 같은 간단한 작업들이 까다로울 수 있다. 예전에는, LVM 사용자들은 오직 디스크를 새 머신에 장착하고 파일시스템을 마운트하기만 하면 되었다. LVM 에는 그것에 약간 더 해 해줘야 할것이 있다. LVM 스트럭쳐들은 디스크들과 /etc/lvmconf 디렉토 리에 모두 저장되므로 한개의 디스크 혹은 볼륨 그룹을 포함하는 디스크들의 집합을 이동하기 위해서 해야 하는 일은 오직 VG 가 속한 기계가 그것을 놓치지 않을 것인지 확인하는 것 뿐이다. 이것은 `vgexport` 명령을 통해 할 수 있다. `vgexport` 는 단순히 /etc/lvmconf 에서 VG 에 대한 스트럭쳐를 제거할 뿐이며, 디스크에 있는 것은 아무것도 바꾸지 않는다. 새로운 머신에 디스크가 장착되면, (그것들이 같은 ID 를 가질 필요는 없다) 해줘야 할 유일한 일은 /etc/lvmconf 를 갱신하는 것이다. 그것은 `vgimport` 를 통해 할 수 있다.

예제:

#1: 번 머신에서

```
vgchange -a n vg01
vgexport vg01
```

#2: 번 머신에서

```
vgimport vg01 /dev/sda1 /dev/sdb1
vgchange -a y vg01
```

볼륨 그룹에 대해 같은 이름을 사용할 필요는 없음에 주목하라. 만약 vgimport 명령이 설정 백업을 저장하지 않았다면 설정 파일을 저장하기 위해서는 `vgcfgbackup` 명령을 사용하라.

9.2. /etc/lvmtab 과 /etc/lvmtab.d 를 재설정한다

FIXME: 보다 좋은 방법에 대해 써야 한다

10. 더 읽을거리

LVM site (http://lvm.msede.com/lvm/)

주 LVM 리소스를 이용가능가능한 곳

German LVM HOWTO (http://litefaden.com/lite00/lvm/)

만약 당신이 독일어를 읽을 수 있다면, 이곳엔 이미 많은 정보가 있을 것이다.

Translation of the German HOWTO (ftp://linux.msede.com/howto/)

Peter.Wuestefeld@resnova.de 는 독일어 HOWTO 를 영어로 번역하고 있다. 그들이 곧 거기에 많은 시간을 투자할 것으로 보인다. 만약 당신이 우리의 HOWTO가 의심되거나 무엇인가가 빠져있다고 생각된다면, 그들의 시도를 사용해보라.

HP/UX Managing Disks Guide (http://docs.hp.com/cgi-bin/omcgi/omdoc?action=getcon=7425)

리눅스 LVM 은 HP/UX 구현과 가장 정확하게 비슷한 것이므로, HP 의 문서 역시 우리에게 매우 유용할 것이다. 아주 좋은 자료이다.

11. 고마운 분들

우리는 이 HOWTO 를 작성하는데 도움을 준 모두를 언급하고 싶다. 여기에는 업데이트, 수정사항 혹은 기고를 보내준 사람들 뿐 아니라, 우리가 이 주제를 이해하는데 도움을 준 사람들도 포함된다.

- Axel Boldt <axel@uni-paderborn.de>
- Sean Reifschneider <jafo@tummy.com>
- Alexander Talos <at@atat.at>
- Eric Maryniak <e.maryniak@pobox.com>