The NetBSD Logical Volume Manager

Adam Hamsik

The NetBSD Foundation

haad@NetBSD.org

Abstract

LVM is a method of allocating disk space on a disk storage devices. Which is more flexible than conventional ones. Logical Volume Manager can usually stripe, mirror or otherwise combine disk partitions to bigger virtual partitions which can be easily moved, resized or manipulated in different ways while in use. Volume Management is one form of disk storage virtualization used in Operating Systems.

The NetBSD LVM has two parts user land tools and a kernel driver. Kernel driver is called device-mapper. User land part is based on Linux lvm tools developed by a community managed by Redhat inc.

The Device-mapper driver can create virtual disk devices according to device table loaded to it. This table specifies which devices are used as a backend, on which offset on particular device virtual device starts. Device-mapper configuration is not persistent and must be loaded to kernel after each reboot by lvm the tools.

Table of Contents

1. Introduction

Until today many types of standard partitioning schemes were developed such as:

- MBR/PC BIOS partition table
- BSD disklabel
- GPT

BSD disklabel is used in BSD systems for historical reasons for a long time, but it has very bad shortcomings. Limitation of maximum disk size to 2Tera bytes is probably most significant one. Every administrator knows that it is very unpleasant to resize MBR or disklabel partition while is this partition in use (I'm not sure if it is possible).

Logical Volume Management was designed for use on storage arrays exported with SAN to serves. It was implemented and used in many commercial unixes such as IBM Aix, Sun Solaris and HP HP-UX.

From Open source systems only FreeBSD and Linux supported Volume Management.

In these days were standard home developed server can have greater than 4Tb of disk storage and much more disks than just one, even low cost servers need to support some sort of Volume Management.

2. Background

2.1. Volume Management Design

Most Volume Managers share same basic design. Most of them have some sort of Physical Volumes called PVs which can be hard disks, hard disk partitions, LUN's located on external SAN storage. PV is created as linear sequence of smaller chunks called Physical Extents PE's. Some of Volume Manager implementations support variable sized PE's examples of them are Veritas Volume Manager or Aix Volume Manager [1]. Other implementations in Linux and HP-UX have PEs of a uniform size.

Every PE is usually mapped to one Logical Extent LE. In cases where striping or mirroring on Volume group level is used one LE is mapped to multiple PEs. In mirroring mode one LE is mapped on two PEs like disk blocks in raid1.

LEs are usually grouped in to bigger pools called Volume Groups VGs. The pooled LE's can be concatenated together to create virtual partitions called Logical Volumes or LVs. LVs act like a raw disk block devices on which file system can be created, they can be mounted or exported as iSCSI disk.

LVs can be grown by adding new LEs from VG pool to them. During shrunk LEs are returned into the VG pool. Concatenated LEs in LV doesn't need to be contiguous, they can be paced on multiple PVs. This allows to grown LV without having to move already used LEs to different PEs. Every PV must belong to only one Volume Group and can't be shared between multiple VGs.

Changing LV size but doesn't automatically resize file system which was created on top of used LV. File system must be resized later, online resize able file system should be used on LV, because it does not interrupt applications during their work.

VGs can be grown by adding new PVs to them, LEs can be later moved from one PV in a Volume Group to another. Most Volume managers can perform this move online while is LV in production.

Many Volume Managers supports block level snapshots. Snapshots are implemented by using Copy on Write approach for each LE. Snapshotted LE is copied out just before data are written to it.

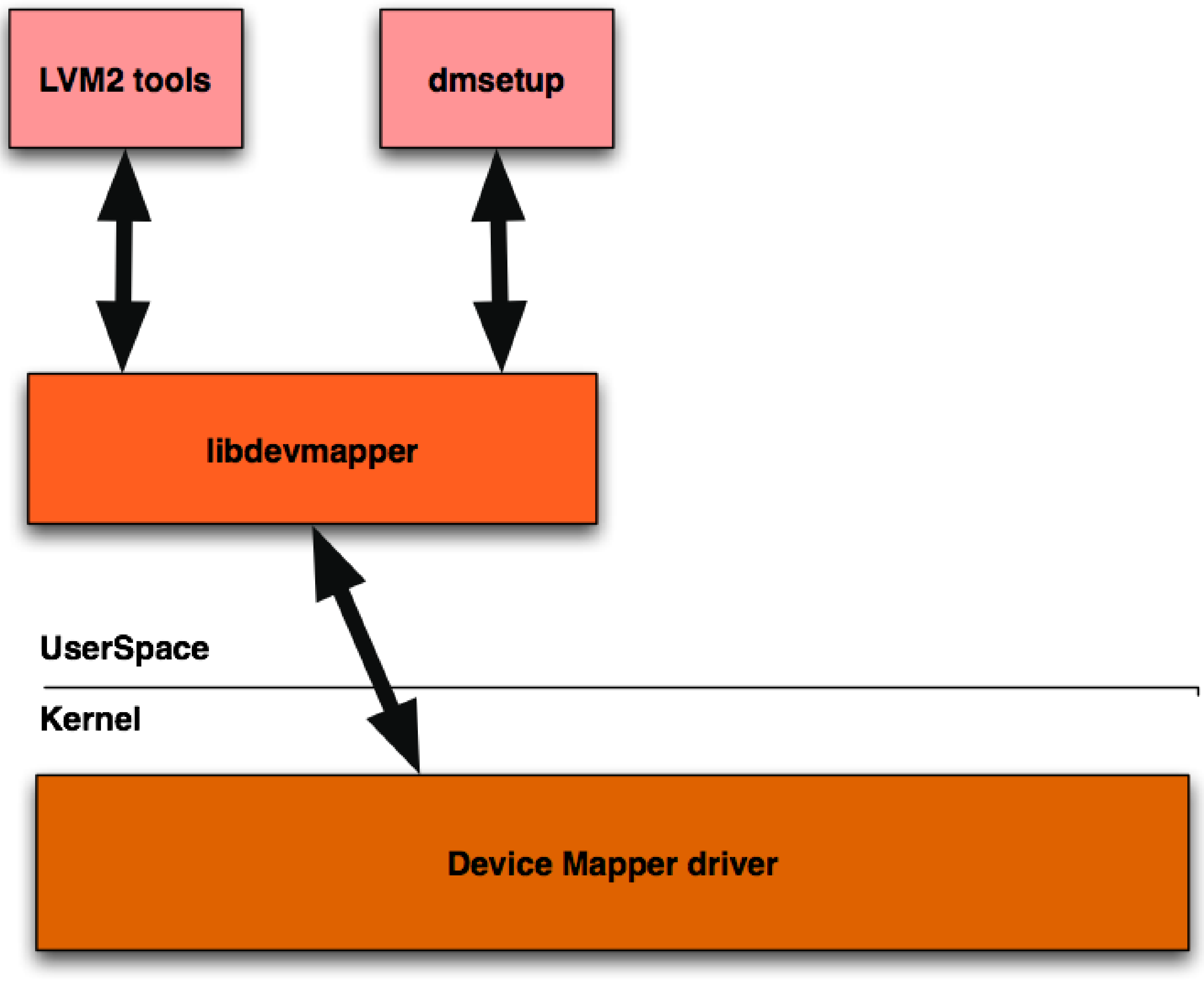
2.2. Volume Manager Features

2.2.1. Linux Volume Manager

The Linux Volume Manager was implemented in two parts user land tools and kernel device driver called device- mapper. Linux LVM supports all listed features such as snapshotting and creating mirrored, striped, linear LVs. LVM in Linux has fixed size LE with 4Mb in size.

Linux LVM was designed with the smallest amount of kernel code needed as possible. Almost whole LVM was implemented in user space and device-mapper driver only maps contents of virtual block device to real one.

Figure 1. LVM architecture



There is a metadata header on start of every Physical Volume device which contains whole LVM configuration. This header contains defined PVs and LVs, their UUIDs and allocation maps which maps LEs to PEs.

Pros

- Easier implementation
- Easier clustering support because only user space code is involved in it.

Cons

- Hard to place / on LVM need to use init ramdisk to load LVM configuration to system with lvm tools.
- Hard to implement some kernel/bootloader LVM detection.

In Linux LVM only LVs are defined in kernel, everything else is defined only in user space and managed by it. Device-mapper driver uses /dev/mapper/control for communication with lvm tools. LVM tools sends data packets to kernel with ioctl commands to request any actions from device-mapper. Example action can be request list of devices defined in driver or create a new device.

Linux lvm tools was designed to be very similar to HP-UX lvm tools [6], which uses same command names and parameters.

Figure 2. Device-mapper interfaces

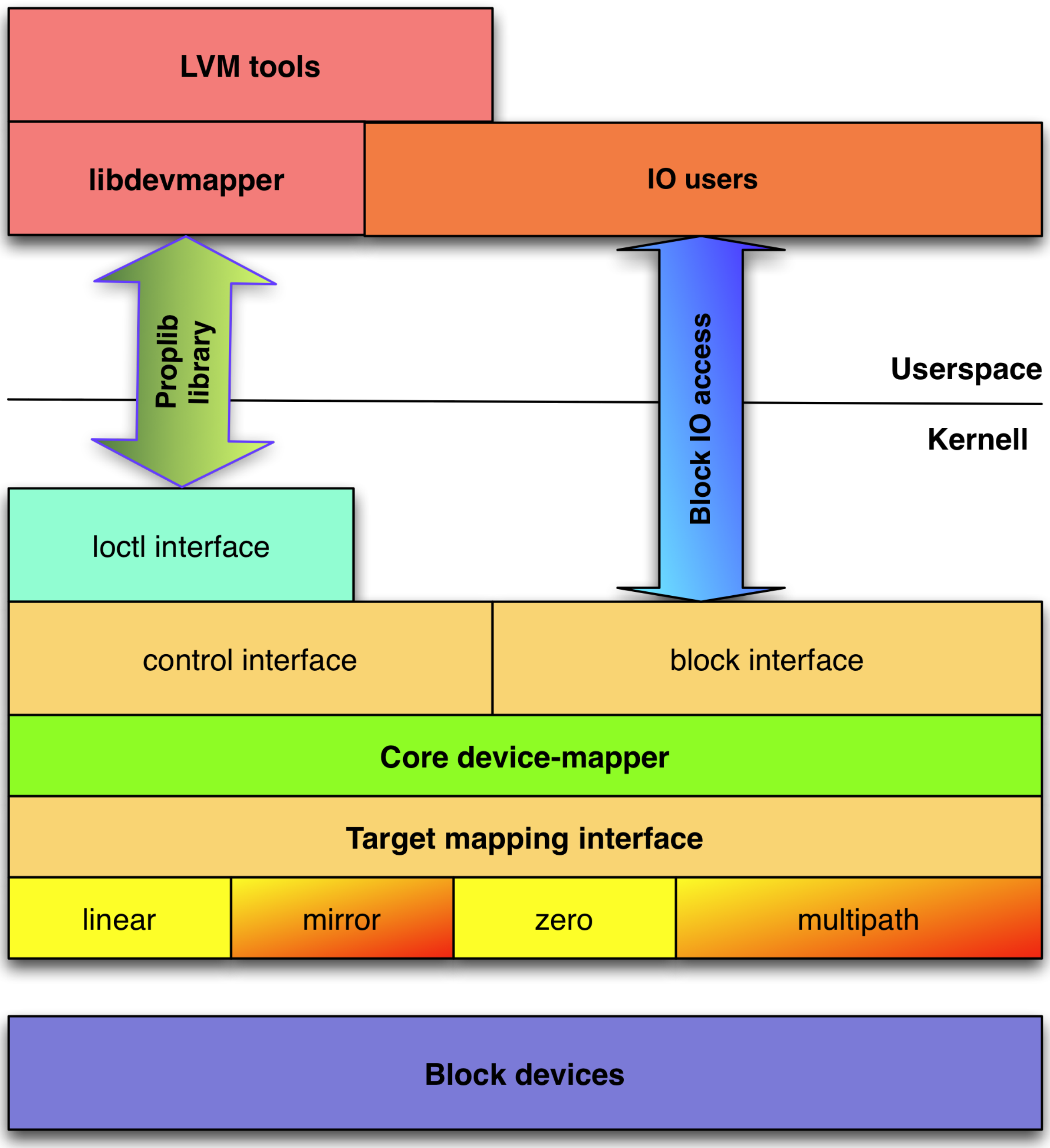
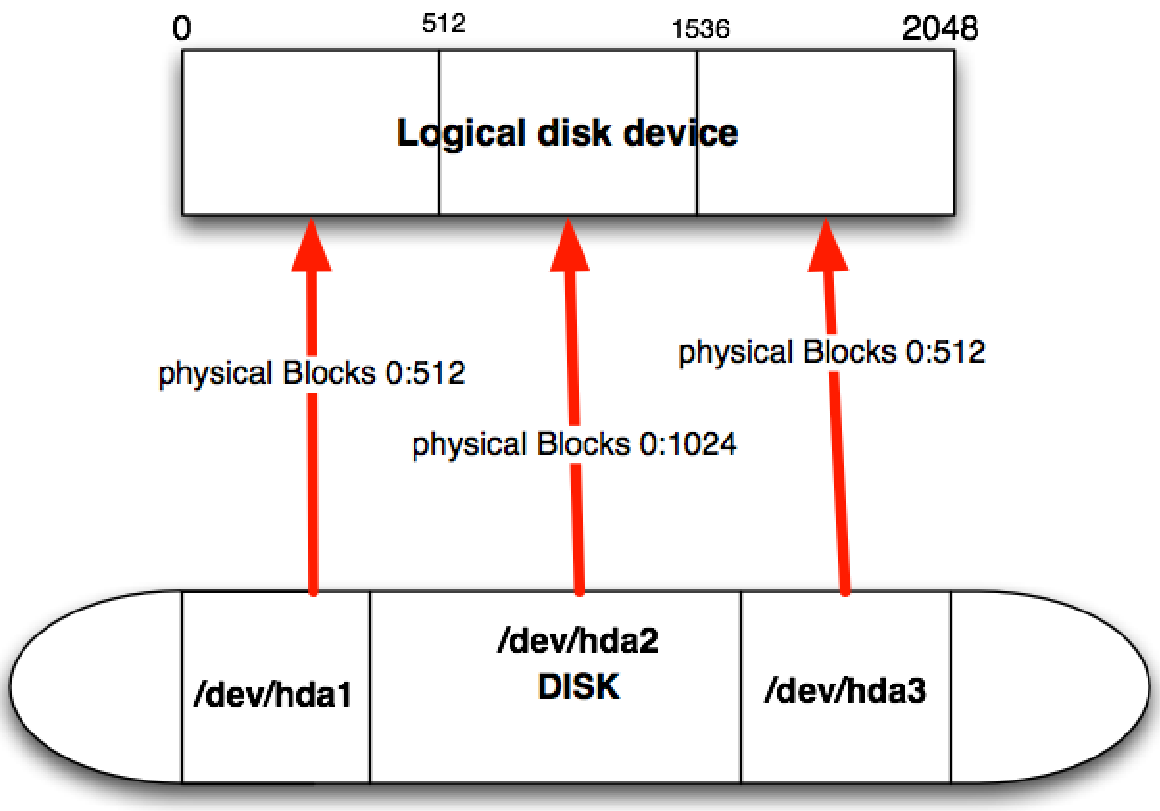


Figure 3. Device-mapper targets



2.2.2. ZFS Volume Manager

The ZFS Volume Manager is tightly integrated with file system. ZFS consist of two major parts the first one is ZVOL and the second is ZPL. ZVOL layer is Logical Volume Manager for ZFS. ZPL means ZFS Posix Layer and it exports POSIX compatible APIs to ZFS users. It can creates Logical Volumes called zvols and export them as a block devices.

ZFS Volume Manager is on the other hand completely implemented in a kernel. It allows to use Logical Volumes for / disk if boot loader is able to load ZFS module and configuration.

The ZFS ZVOL subsystem allows creation of several types of ZPOOLs such as:

- Linear
- Mirror
- Stripe
- Raidz

  Sun implementation of raid5 like redundancy scheme, but without raid5 write hole problem [4].
- Raidz2

  Sun implementation of raid6 like redundancy scheme, but without raid6 write hole problem [4].

3. The NetBSD LVM implementation

The NetBSD LVM implementation consists from two major parts.

- GPL licensed user land code ported from Linux
- BSD licensed kernel code written from scratch

3.1. The NetBSD lvm tools

LVM tools has three parts lvm commands, lvm library and libdevmapper library which is responsible for kernel user space communication. Libdevmapper implements it's own protocol for communication with device-mapper driver. This protocol uses one buffer which is sent to kernel with different types of ioctl commands. Buffer can contain from several structures with *`struct dm_ioctl`* as starting point and with other possible entries starting at *`dm_ioctl::data_start`*

To avoid using GPL licensed headers in kernel source tree we designed and implemented new ioctl communication channel. This channel is based on proplib library which is used to send/receive proplib dictionaries from kernel [8].

Proplib library provides abstract interface for manipulating property lists[7]. Property lists knows these object types number, string, data and boolean. Structure is provided by two other types dictionary and array. Dictionary is name/value pair based array.

The NetBSD lvm tools port contains changes mostly in 2 files `libdm-nbsd-iface.c` and `libdm_netbsd.c`. For NetBSD we translate dm_ioctl structure to the dictionary and back before and after every ioctl. With this approach `libdevmapper` internals stay same as in Linux.

Figure 4. Example proplib message sent to kernel

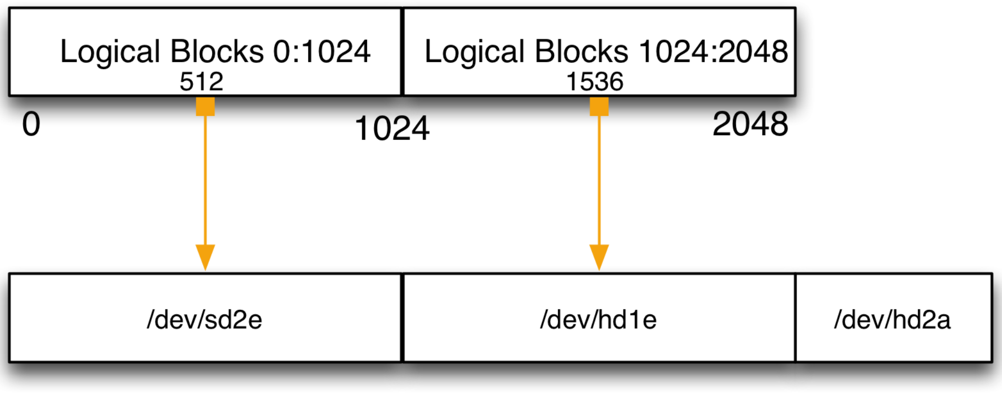
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://
www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>cmd_data</key>
        <array/>
        <key>command</key>
        <string>version</string>
        <key>event_nr</key>
        <integer>0x0</integer>
        <key>flags</key>
        <integer>0x4</integer>
        <key>version</key>
        <array>
                <integer>0x4</integer>
                <integer>0x0</integer>
                <integer>0x0</integer>
        </array>
</dict>
</plist>
```

Other required changes was in lvmlib because NetBSD has raw devices which are not known in Linux world.

3.2. The device-mapper driver

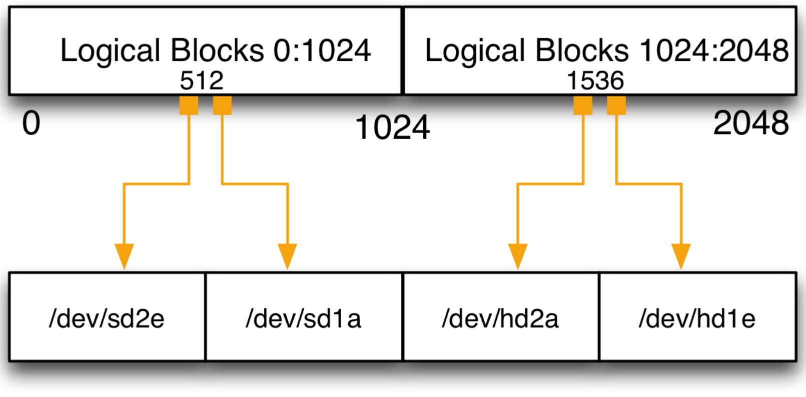
Device-mapper driver currently implements 2 types of target mapping linear and striped

Figure 5. Linear block device



Linear target concatenates disk space from physical volumes together.

Figure 6. Striped block device



Stripe target stripes disk block writes to multiple physical volumes to improve IO throughput of Logical Volumes.

Device-mapper driver was designed to be very effective in a SMP environment. There are 2 ways how are users using dm devices ioctl administrating interface and doing IO operations on a dm device. NetBSD device-mapper was designed to block IO operations only in a case where it is really needed, otherwise ioctl operations has very low impact on IO operations.

3.3. NetBSD Use cases

- iSCSI server
- XEN Server
- Network NAS device

The LVM subsystem can be used for several useful tasks. It can help administrators to manage their XEN based NetBSD servers. Where every domU can be placed on it's own LV. iSCSI server is another small use case example. iSCSI LUNs can be placed on a LVs and exported with iscsi target to any client as network disk.

4. Future Development

There are still missing features in the NetBSD LVM now. The most significant parts are:

- Mirror target
- Snapshot target
- Clustering support
- DRBD like target

4.1. Mirror Target

This mapping can be used for creating RAID1 like devices with device-mapper. It is also used during pvmove command, which creates mirror between current and next PV where LEs are going to be moved and mirror them until all PEs are moved.

Mirror target need support for creating log devices which can be used for keeping parity between two mirrored devices.

4.2. Snapshot target

This mapping creates Copy on Write usable block level snapshots. Snapshots can be mounted and used as normal block devices.

Current snapshot implementation in Linux is insufficient for current use cases and there is some work ongoing to replace snapshot target with new approach derived from zumastor storage server [3].

New snapshot target should allow unlimited number of snapshots to be created without any performance impact, which is really high with current implementation.

4.3. Clustering Support

Cluster LVM is used for managing access to shared disk storage exported to multiple servers. CLVM manage access to LVM metadata located on each PV used by LVM. This is needed to ensure that only one server from a cluster will manipulate with LVM metadata in time [5].

Distributed Lock Manager library is needed to get CLVM working on the NetBSD. To be able to use all benefits of clustered LVM, cluster file system is needed. Cluster file system cares about access to files located on it.

4.4. DRBD like target

The DRBD means Distributed Raid Block Device and can be used for a simple high availability clusters, with two or more nodes. Distributed raid block device target would be useful. DRBD allows creation of block device which is mirrored to another node/nodes through network connection.

DRBD creates persistent connection between cluster nodes and manages access to disks. There are two possible options used single master and multiple master. With single master only one node can write to drbd disk, with multi master mode anyone can write to disk and drbd takes care of synchronisation on block level.

5. Conclusion

The NetBSD LVM support was commit to HEAD branch in a December of 2008 after one year development. LVM will be part of next NetBSD major release 6.0. Device-mapper driver interface should be stable and will not change for any already implemented targets.

Bibliography

[1] Laurent Vanel. Ronald van der Knaap. *AIX Logical Volume Manager, from A to Z: Introduction and Concepts* *[http://www.redbooks.ibm.com/redbooks/pdfs/sg245432.pdf]* . IBM corp.. January 2000.

[2] Heinz Mauelshagen. Matthew O'Keefe. *http://www.redhat.com/magazine/009jul05/features/lvm2* . RedHat inc.. July 2005.

[3] Daniel Phillips. Robert Nelson. Jane Chiu. *The Zumastor Linux Storage Server* *[http://zumastor.googlecode.com/files/zumastor.whitepaper.pdf]* . . February 11, 2007.

[4] Jeff Bonwick. *RAID-Z* *[http://blogs.sun.com/bonwick/entry/raid_z]* . Sun corp.. November 18, 2005.

[5] *CLVM introduction* *[http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Cluster_Logical_Volume_Manager/LVM_Cluster_Overview.html]* . RedHat Inc.. 2009.

[6] . *HP-UX System Administrator's Guide: Logical Volume Management* *[http://www.docs.hp.com/en/5992-4589/5992-4589.pdf]* . Hewlett-Packard Development Company, L.P.. 2008.

[7] Apple inc . *Apple property list dtd* *[http://www.apple.com/DTDs/PropertyList-1.0.dtd]* . . .

[8] Jason R. Thorpe. *Proplib manual page.* *[http://netbsd.gw.com/cgi-bin/man-cgi?proplib++NetBSD-current]* . . NetBSD 4.0.