LVM2 - storage virtualization

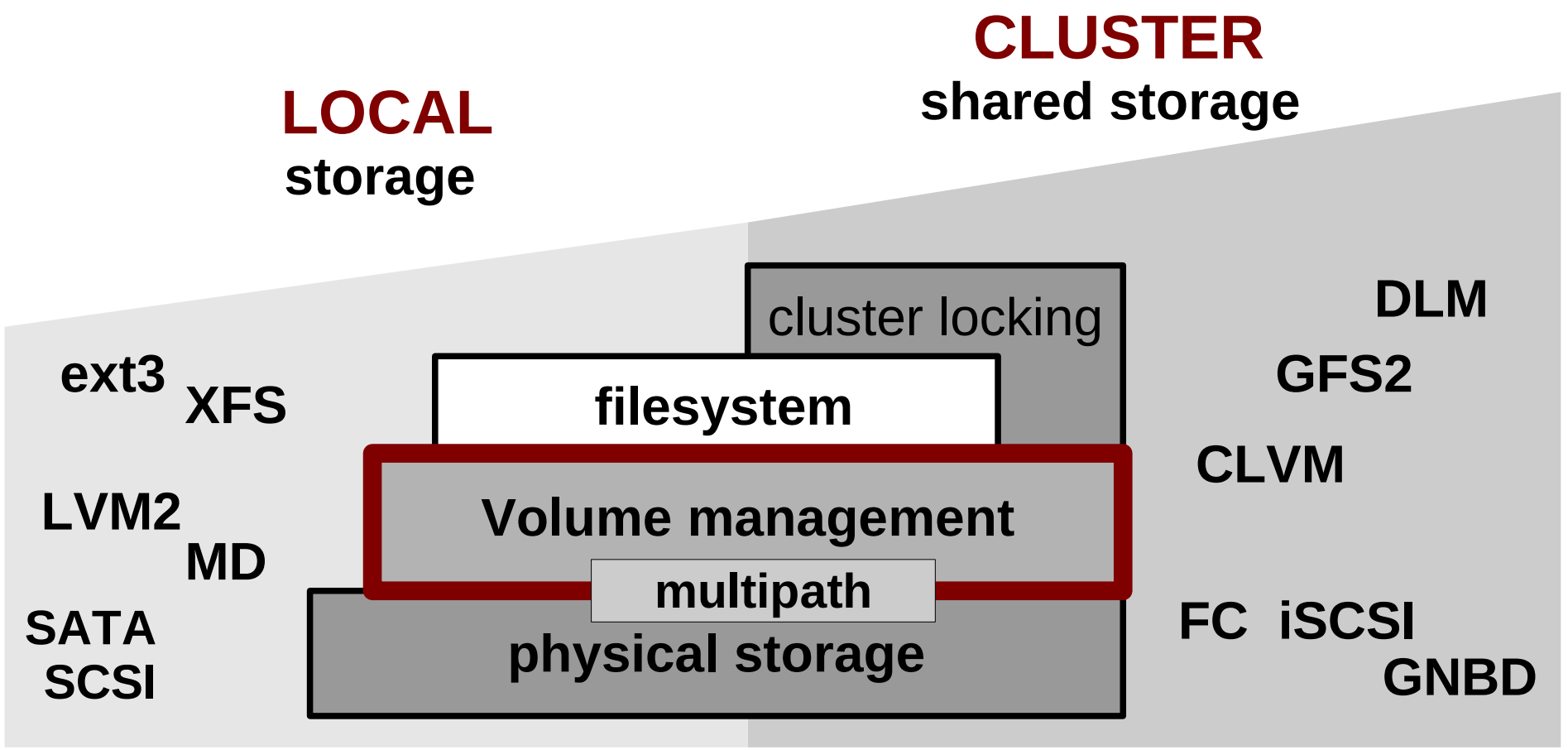
Milan Brož
mbroz@redhat.com

Why storage virtualization ?

- **Abstraction** over physical storage
- **Simplify management** of storage resources
- **High-Availability** (on the fly) operation
- **Flexibility** - resize, relocation, optimize, ...
- **Scalability** - no physical devices restriction
- **Security -** RAID, online snapshots
- **Reduce operation costs**

Storage virtualization in Linux

technology overview



LVM2 – Logical volume manager

- **Layer of abstraction** over the physical storage
- Provides **Logical Volumes** used by filesystems (or directly by applications)
- Logical Volume is **block device**
- Logical Volume ~ similar to disk partition
- Hides physical hardware configuration

Linux LVM

- **LVM1 – only in Linux kernel 2.4**
  major part of functionality in kernel, design inspired by HP-UX LVM
- **LVM2 - more generic concept**
  LVM1 commands compatible, can use LVM1 metadata
  - **KERNEL driver (Device mapper)**
  - **USERSPACE toolset (LVM2)**

MD versus DM/LVM2

- MD is multi-disk kernel layer
  - it can create new block devices from existing discs
  - linear, raid 0,1,4,5,6
  - very stable, management through mdadm utility
- DM (device mapper) is kernel driver
  - Generic block device mapping utility, you can combine (almost) all possible targets and create new block device
- LVM2 is generic volume management
  - you can combine physical discs, MD and DM devices into volumes

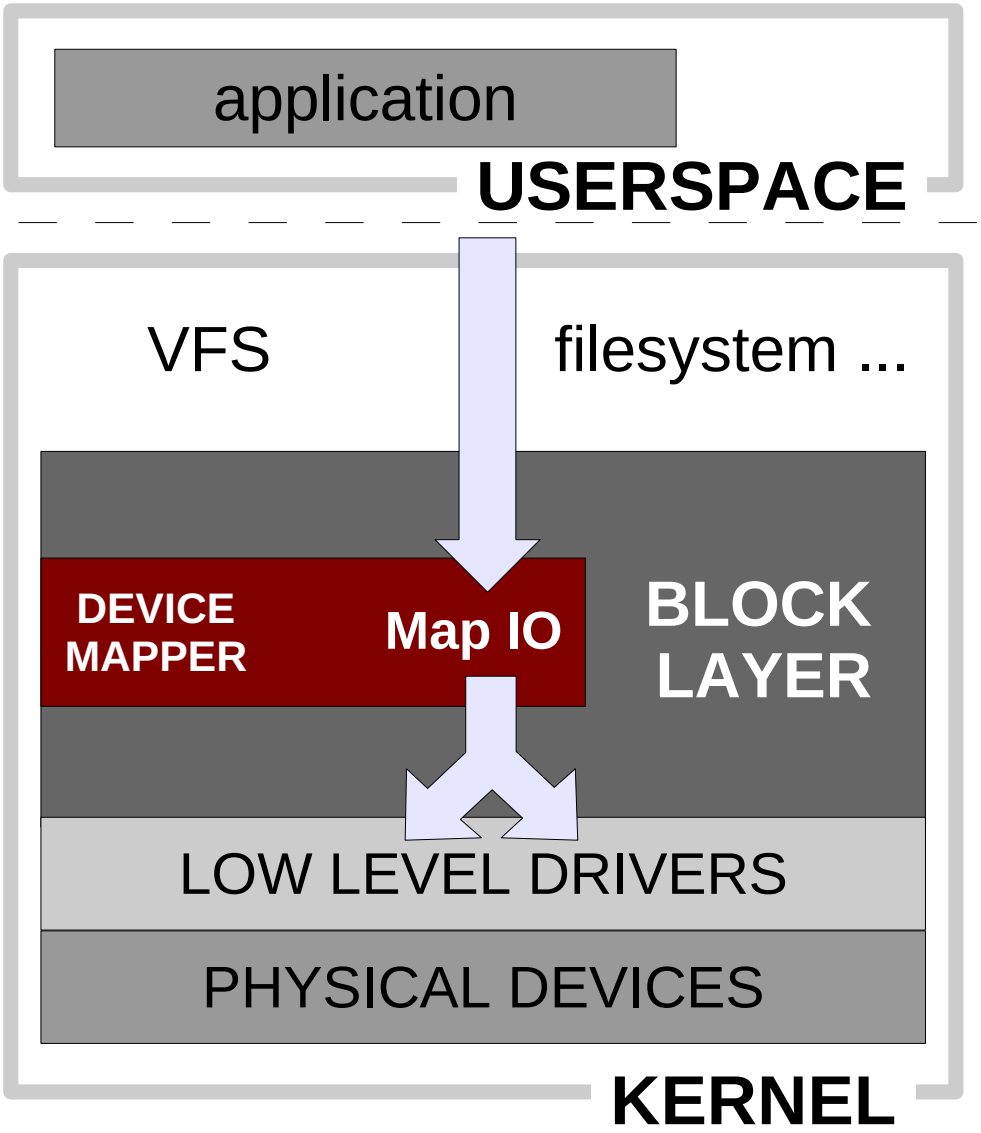
Device mapper ...

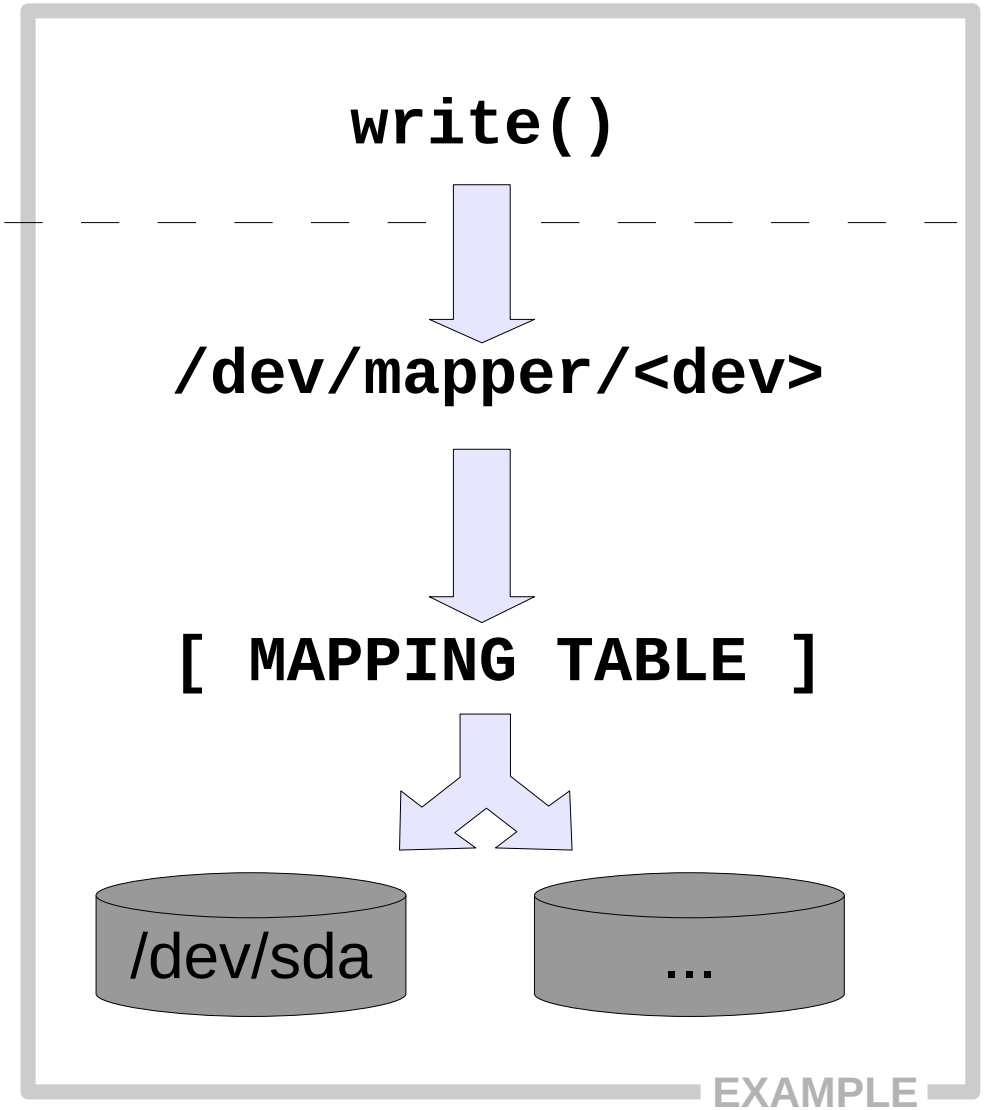
- modular Linux 2.6 **kernel driver**
- framework for constructing new block devices
  and mapping them to existing block devices
- **managed through API** (IOCTL interface)
- libdevmapper, **dmsetup** command utility

DM knows **nothing** about

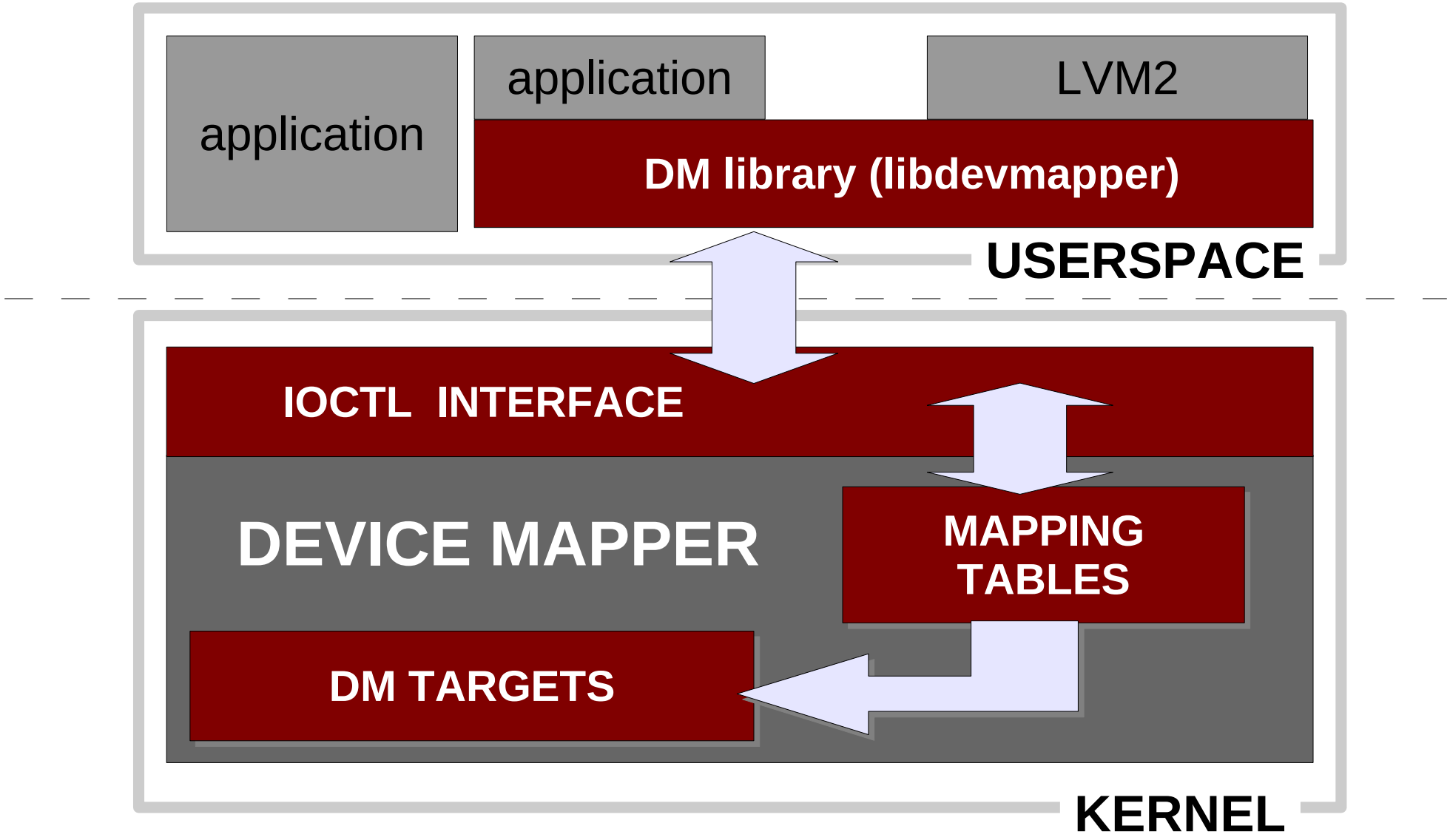
- **LVM** (logical volumes, volume groups)
  - ► managed by userspace tools (LVM2, EVMS, ...)
- **partitions, filesystems**
  - ► managed by userspace tools (fdisk, mkfs, mount, ...)

Device mapper – mapped device access





Device mapper – control interface



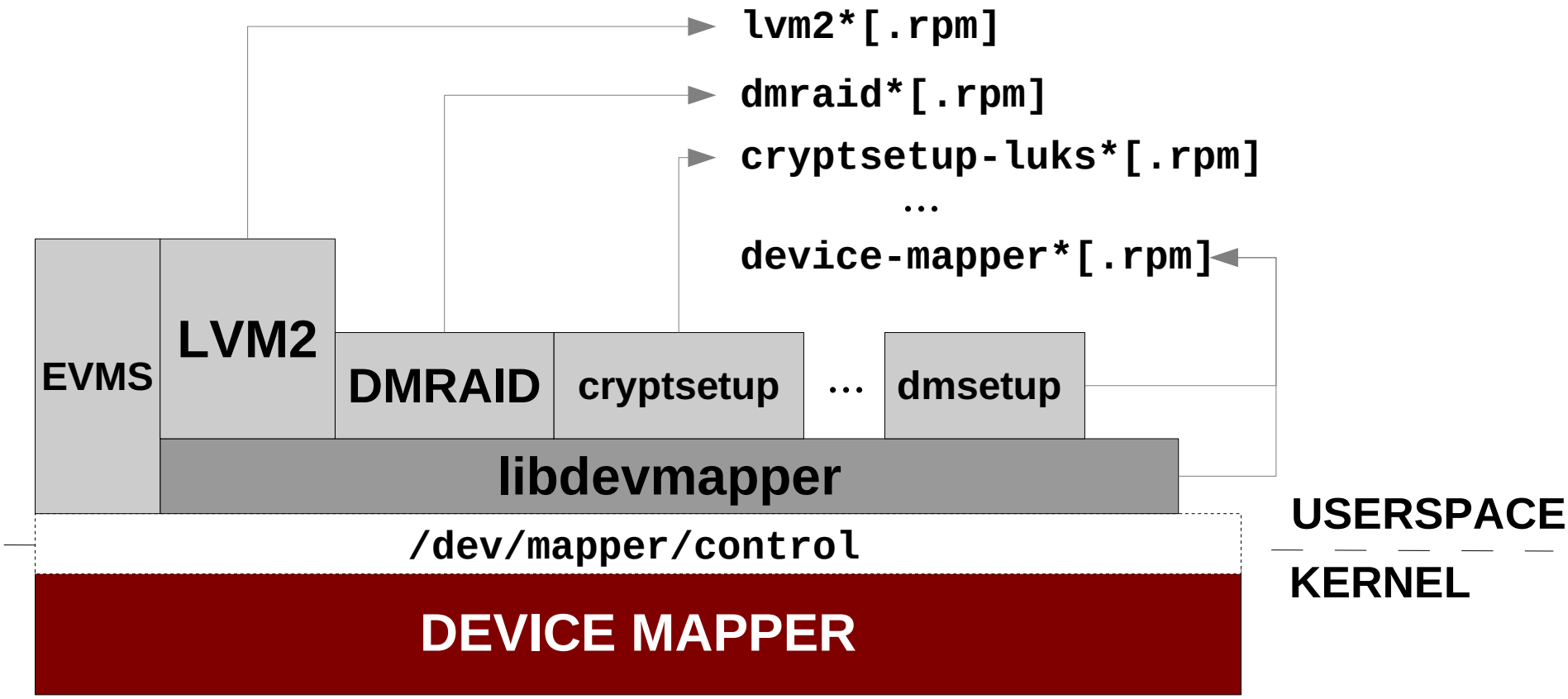
Device mapper - TARGETS

- **linear** – maps continuous range of another block device
- **striped** *(~RAID0)* – striping across devices
- **mirror** *(~RAID1)* – mirroring devices
- **crypt** – encrypt data using CryptoAPI
- **snapshot** – online snapshots of block device
- **multipath** – access to multipath devices (misc. hw handlers)
- **zero,error** – test targets
- **truecrypt**

...

- **raid45** *(~RAID4,5)* – raid (with dedicated) parity
- **loop –** stack device over another or over file
- **delay, throttle, rwsplit, flakey** – test targets

Device mapper – applications



Low-level DMSETUP EXAMPLE – striped target

```
dmsetup create strip
 0  819200  striped 2 128 /dev/sdb1 0 /dev/sdc1 0
```

device
start sector
chunk size
num.stripes
target
*sectors 400MB = 400*1024*2 sectors*
start sector [sector = 512 bytes]

dmsetup table
ls -l /dev/mapper/

mke2fs /dev/mapper/strip
mount /dev/mapper/strip /mnt/tst

df -h /mnt/tst

umount /mnt/tst
dmsetup remove strip

LVM2 – Logical Volume Manager

(abstraction over Device Mapper)

3 LAYERS

PHYSICAL VOLUMES – PV
storage devices ~ DISKS

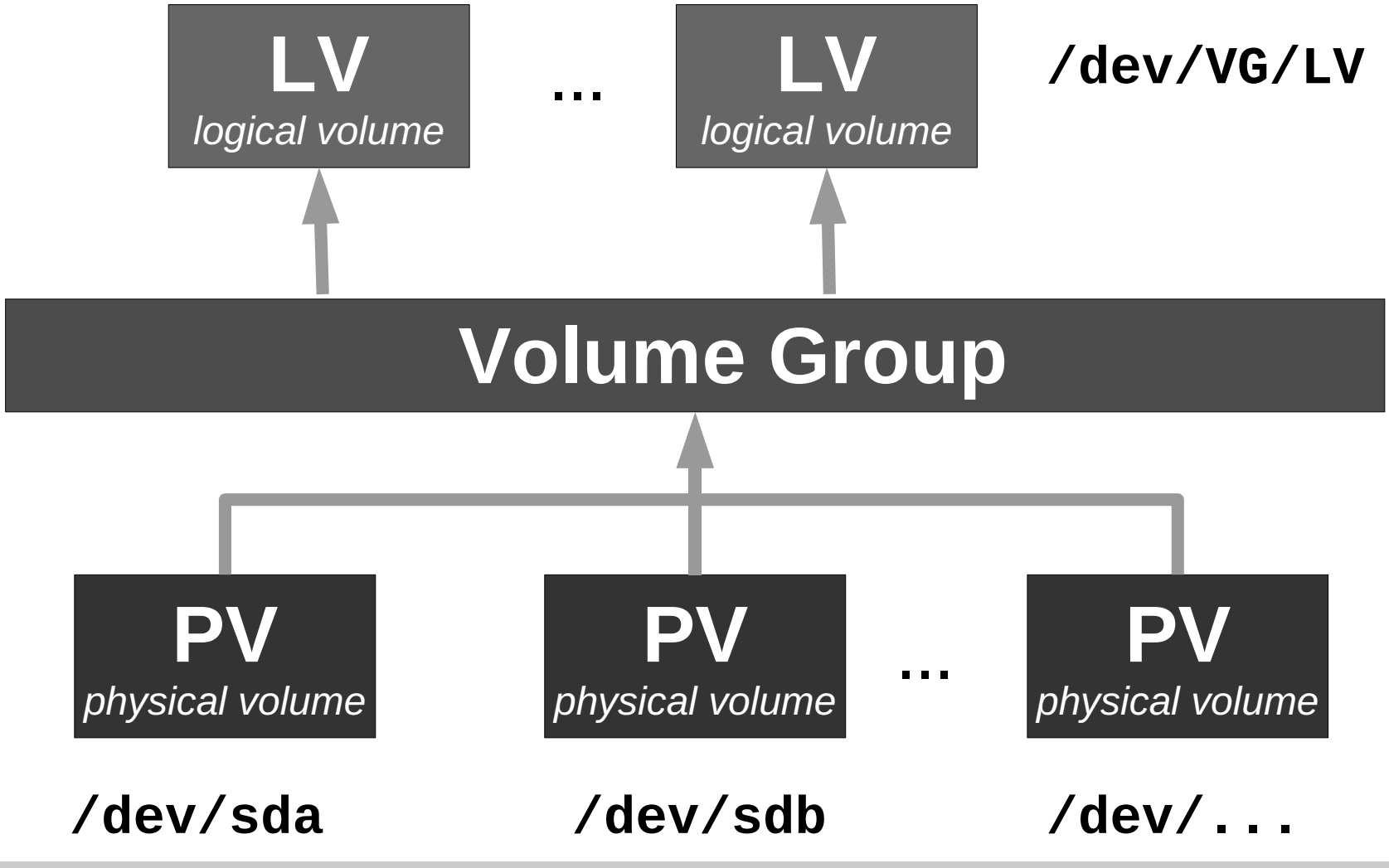
grouped in

VOLUME GROUPS – VG
logical disks

VG's capacity allocated to

LOGICAL VOLUMES – LV
logical partitions ~ PARTITIONS

LVM2 – components schema



LVM2 – commands

- command prefix → **pv***, **vg***, **lv***
- **lvm –** one binary file (symlinks), interactive shell, list – **lvm help**
  - [pv,vg,lv] ***s**[how] [*display], ***scan**, ***create**, ***remove**, ***change**
  - [vg,lv] ***reduce**, ***extend**, ***rename**
  - [pv,lv] ***resize**
  - **pvmove**
  - **vgmerge, vgsplit**
  - **lvconvert**
  - **lvmdiskscan**
- [vg] ***cfgbackup/restore, *import/export, *convert, *ck, *mknodes**

LVM2 – METADATA

- **Where is the mapping stored ?**
  - pointers to data in the beginning of PVs after disk label
  - text format – plain ASCII
  - backups in /etc/lvm
  - objects have unique id – UUID

  - fail → recovery can be very complicated

  INCLUDE METADATA IN DAILY BACKUPS !

LVM2 EXAMPLE – METADATA

```
creation_time = ...
...
vg_strip { id = "xxxxxx-xxxx-xxxx-xxxx-xxxx-xxxx-xxxxxx"
        ...
        physical_volumes {
            pv0 {    id = "xxxxxx-xxxx-xxxx-xxxx-xxxx-xxxx-xxxxxx"
                     device = "/dev/sdb1"  # Hint only
                     ...
                     pe_start = 384
                     pe_count = 50   # 200 Megabytes }
            pv1 { ... }
        }
        logical_volumes {
            lv_strip {   id = "xxxxxx-xxxx-xxxx-xxxx-xxxx-xxxx-xxxxxx"
                         ...
                         segment1 {
                             start_extent = 0
                             extent_count = 100 # 400 Megabytes
                             type = "striped"
                             stripe_count = 2
                             stripe_size = 128  # 64 Kilobytes
                             stripes = [  "pv0", 0,
                                          "pv1", 0 ]
                         }
            }
        }
}
```

LVM2 EXAMPLE – striped target

fdisk /dev/sd[bc] → create sdb1,sdc1 of type 0x8e [LVM]

pvcreate /dev/sdb1 /dev/sdc1 – create PV

vgcreate vg_strip /dev/sdb1 /dev/sdc1 – VG

pvs

vgs

dmsetup table → no entry yet (creating metadata only)!

lvcreate -L 400M -i 2 -n lv_strip vg_strip – LV

dmsetup table → see reserved space for metadata

ls -l /dev/mapper

ls -l /dev/vg_strip/lv_strip

mke2fs /dev/vg_strip/lv_strip – make fs and mount

mount /dev/vg_strip/lv_strip /mnt/tst

...

umount /mnt/tst

lvremove vg_strip lv_strip – destroy LV, VG, PV

vgremove vg_strip

pvremove /dev/sdb1 /dev/sdc1

PVMOVE – move data between PV

- **move arbitrary physical extents**
- **online operation** – using kernel kcopyd
- operation can be restarted after crash from last **checkpoint (stored in metadata)**
- using temporary **dm-mirror**
- pvmove starts operation and monitors move
- moving root volume – possible locking issues

LVM2 EXAMPLE – PVMOVE

/dev/sdc1 → /dev/sdd1

pvcreate /dev/sdd1 – add new device

lvs -o +devices

pvs

vgextend vg_strip /dev/sdd1

pvmove /dev/sdc1 /dev/sdd1 – move

pvs

vgreduce vg_strip /dev/sdc1 – remove old device

pvremove /dev/sdc1

Useful commands

- **pvmove --abort** – abort current move
- **pvmove src:PE-PE dst:PE-PE** – specify phys. extents
- **pvmove** – (no parameters) – continue last operation

LVM2 snapshots

- **block layer snapshots** (fs independent)
- **multiple** and **writable** snapshots
- using COW (**Copy On Write**) store
- COW on disk (special LV)

  space for changed blocks (**smaller than LV**)
  Note: overfilled snapshot is invalidated !

- no LVM2 support for incremental snapshots yet

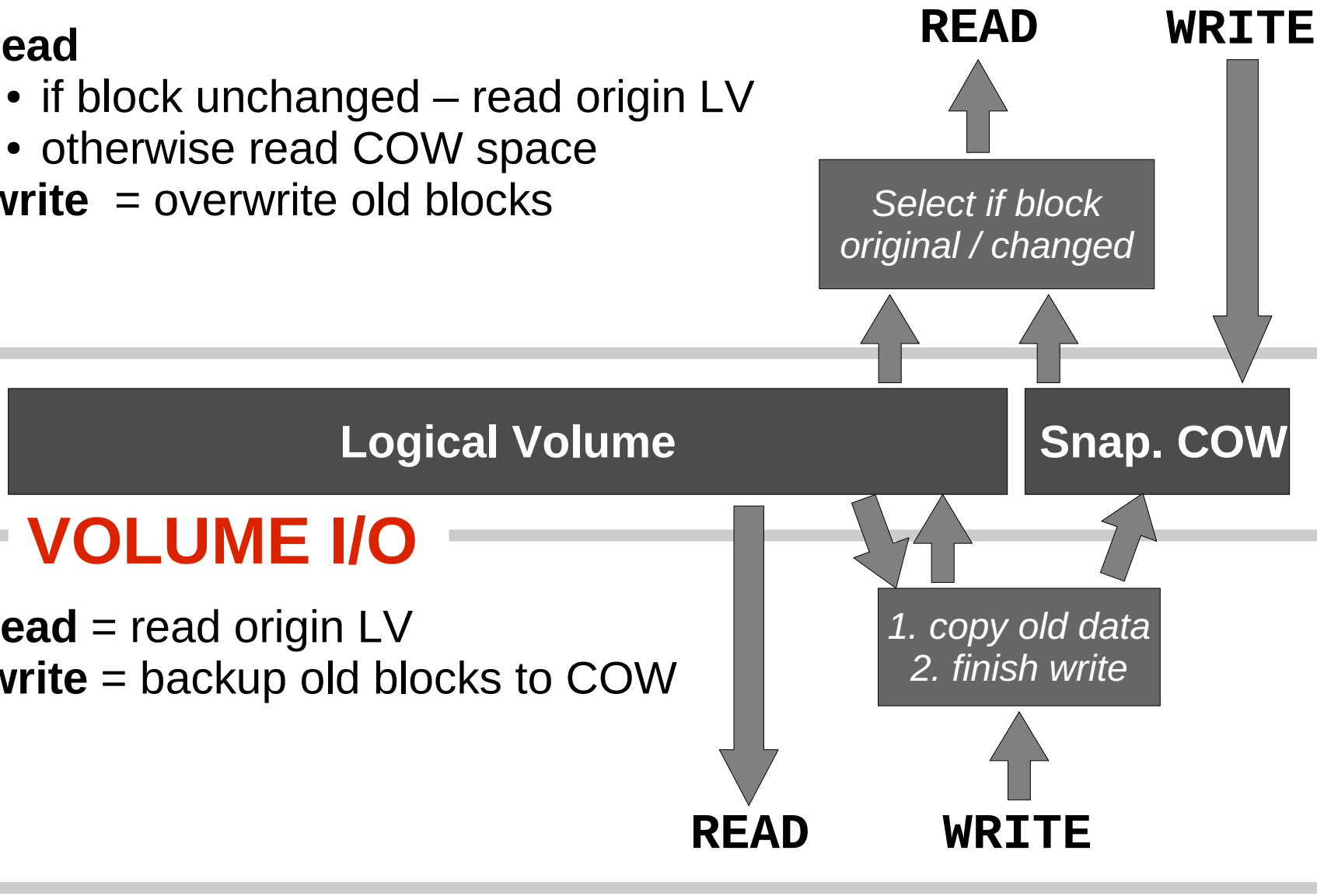
  multiple snapshots of the same LV = huge performance loss

- no snapshot merging yet

SNAPSHOT I/O

read
- if block unchanged – read origin LV
- otherwise read COW space

write = overwrite old blocks



VOLUME I/O

read = read origin LV
write = backup old blocks to COW

LVM2 EXAMPLE – snaphots

`vgextend vg_strip /dev/sdc1` – extend VG (for snapshot)

Create snapot of <cow_size> on lv_strip LV

```
lvcreate -s -L<size> -n lv_snap vg_strip/lv_strip
```

```
mount /dev/vg_strip/lv_snap ...
```

`lvs` – show volumes

- /dev/vg_strip/lv_srip – original volume
- /dev/vg_strip/lv_snap – writable snapshot of lv_strip

RESIZING logical volume

- Order of steps is important – risk losing data !

- **Growing volume**

  (1. add storage device to volume group)
  2. extend logical volume
  3. resize filesystem (online)

- **Shrinking volume**

  1. reduce filesystem (if possible, offline)
  2. reduce logical volume
  (3. remove device from volume group)

LVM2 EXAMPLE – resize volume with ext3 fs

shrink volume

```
pvs --units k
resize2fs -p /dev/vg_strip/lv_strip <size>
lvresize -L <size> /dev/vg_strip/lv_strip
```

mount /dev/vg_strip/lv_strip ...

grow volume

```
lvresize -l +100%FREE /dev/vg_strip/lv_strip
lvs
df -h
resize2fs -p /dev/vg_strip/lv_strip
df -h
```

DMRAID utility

- mapping tool for disk arrays created by **sw raid** controllers
  - ATARAID – cheap vendor sw RAID (SiL, Highpoint, Promise, VIA, ...) using BIOS extension (typically including disk management utility) and sw drivers providing RAID functionality)
- **on-disk metadata → dm target mappings**
  - linear, JBOD (just bunch of disks)
  - RAID 0,1
  - RAID 4,5 (dm target in development)

DMRAID – basic commands

`dmraid -r [...]` – list all discovered devices

device names, metadata driver, set name, status, etc.

`dmraid -a [y|n] [name]` – activate/deactivate disk sets

create new dm block device based on dicovered metadata on disk (in /dev/mapper/<drivername>_<devicename>

`dmraid -s [...]` – show raid sets properties

RAID level, number od devices, chunch size, etc

`dmraid -l [...]` – list all known metadata formats

which dmraid can process
*you need generic **dm raid kernel** modules too (not md raid456 !)*

DMRAID raid-45 target

- new kernel target for device-mapper based raid
- needed for **dmraid RAID5 sets** (and in future for cluster raid5)
- dm-raid45 target in development, not yet in stable kernel
- note: second implementation of raid5 in kernel

  (the first is stable md raid456 – not usable for ATARAID/dmraid)

kpartx – a partition mapping tool

- create devices (parititions) from partition tables
- parse partitions mapped through device mapper or loop device

kpartx - example

`kpartx -l device` – **list partitions on device**

`kpartx -a device` – **add partitions mappings**
`kpartx -d device` – **delate partitions mappings**

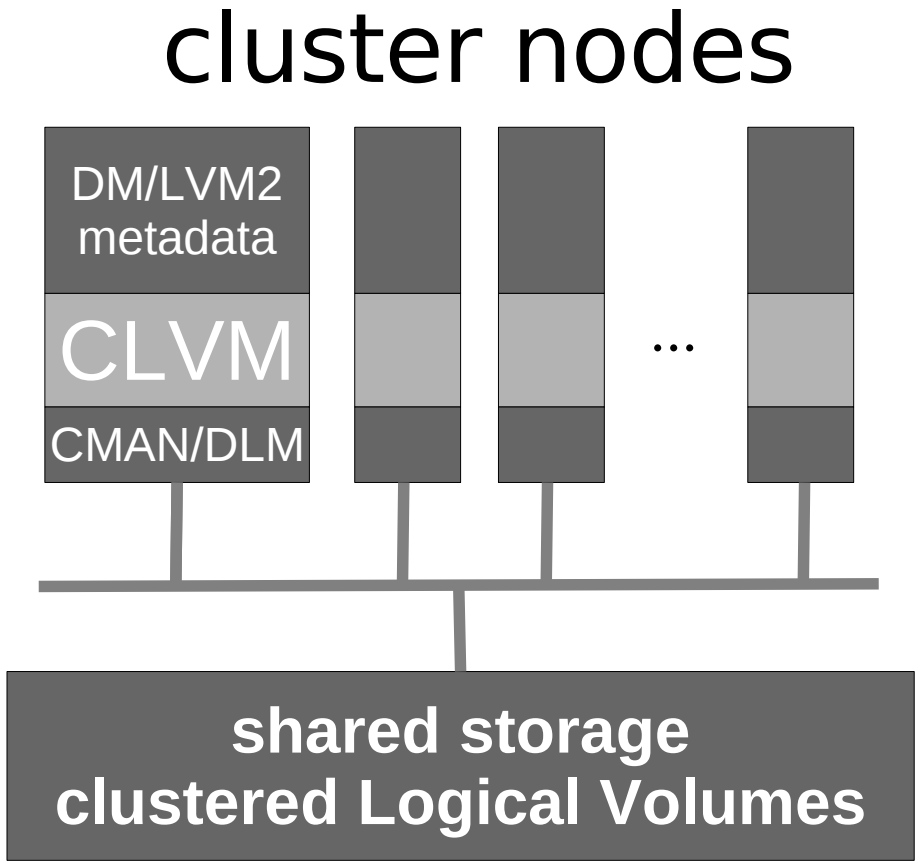
map partitions from whole disk image in file
losetup /dev/loop0 file
kpartx -a /dev/loop0
and partitions are accessible in /dev/mapper/loop0p1,p2,...

Cluster LVM (CLVM)

- **Extends LVM2 to provide cluster-wide Logical Volume management**

- Provides locking infrastructure for proper **volume activation** and **metadata management**

- **clvmd daemon**
  need DLM (distributed lock manager) and functional cluster management (CMAN, ...)



Clustered storage requirements

- **Shared storage** (block device)
  - **SAN** directly connected to nodes (FC)
  - **iSCSI**
  - **GNBD** (Global Network Block Device)
- Volume management – **CLVM**
- cluster-aware filesystem e.g. **GFS**

Links

Device mapper	http://sources.redhat.com/dm/
LVM2	http://sources.redhat.com/lvm2/
LVM wiki	http://sources.redhat.com/lvm2/wiki
Cluster suite	http://sources.redhat.com/cluster/
dmraid	http://people.redhat.com/heinzm/sw/dmraid/
multipath	http://christophe.varoqui.free.fr/wiki/
dm-crypt	http://www.saout.de/misc/dm-crypt/
EVMS	http://evms.sourceforge.net/