Device mapper

(kernel part of LVM2 volume management)

Milan Brož

mbroz@redhat.com

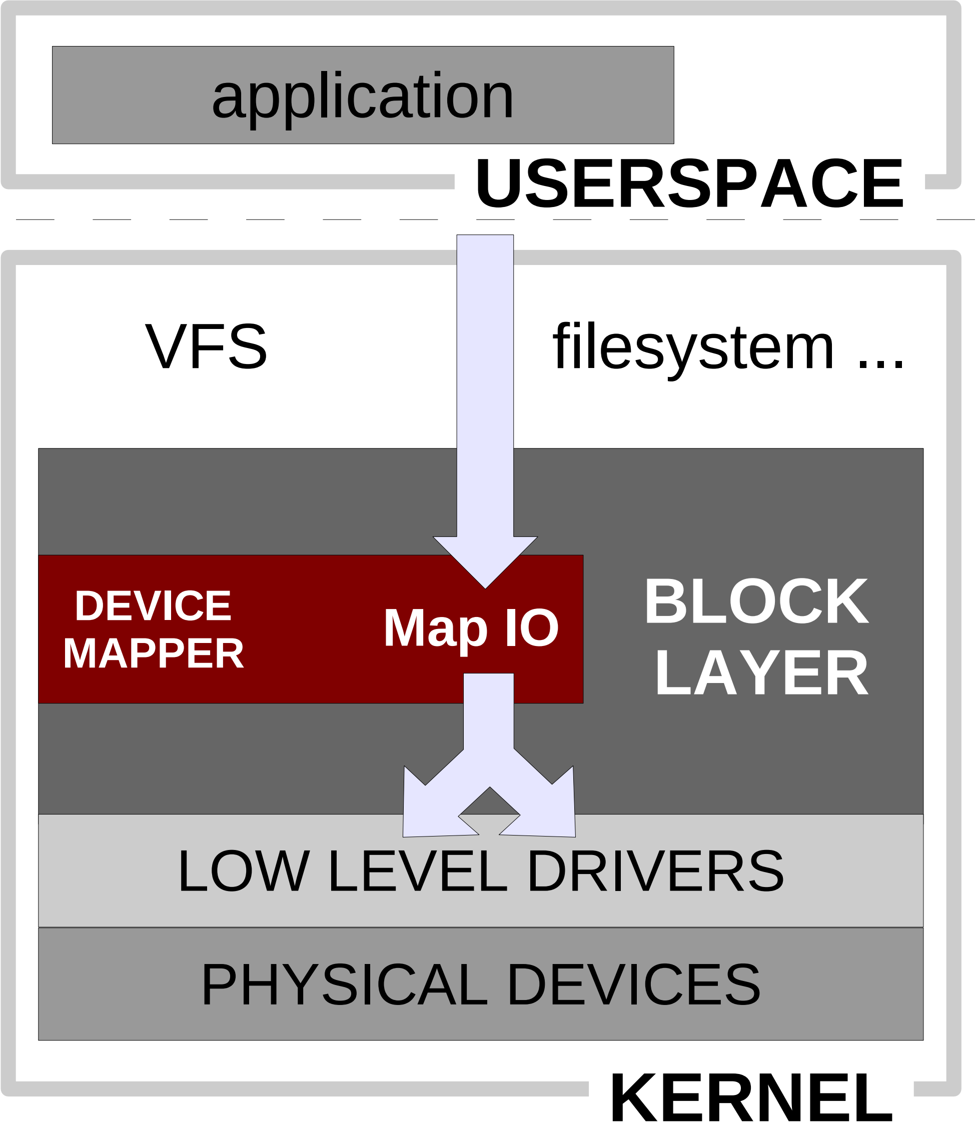
Device mapper ...

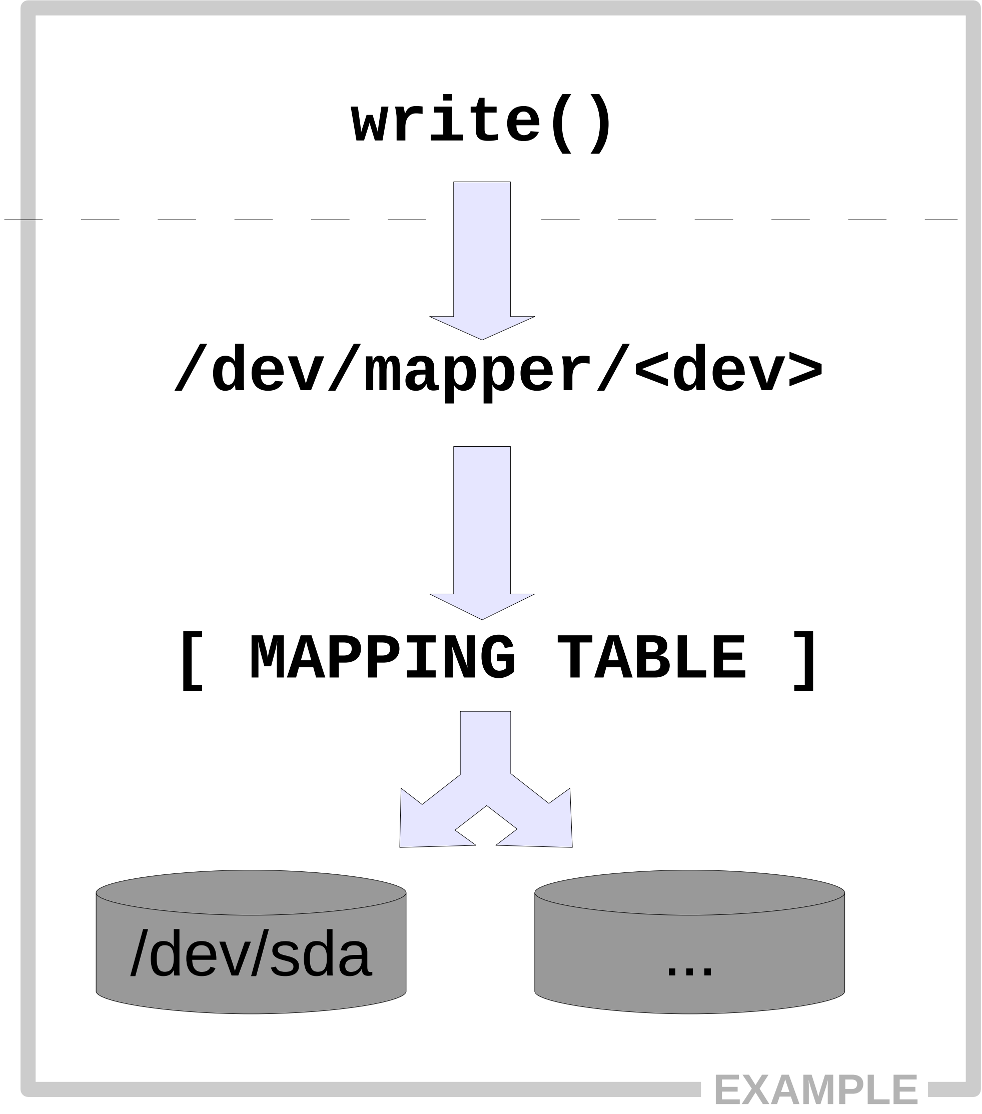
- modular Linux 2.6 **kernel driver**
- framework for constructing new block devices and mapping them to existing block devices
- **managed through API** (IOCTL interface)
- libdevmapper, **dmsetup** command utility

DM knows **nothing** about

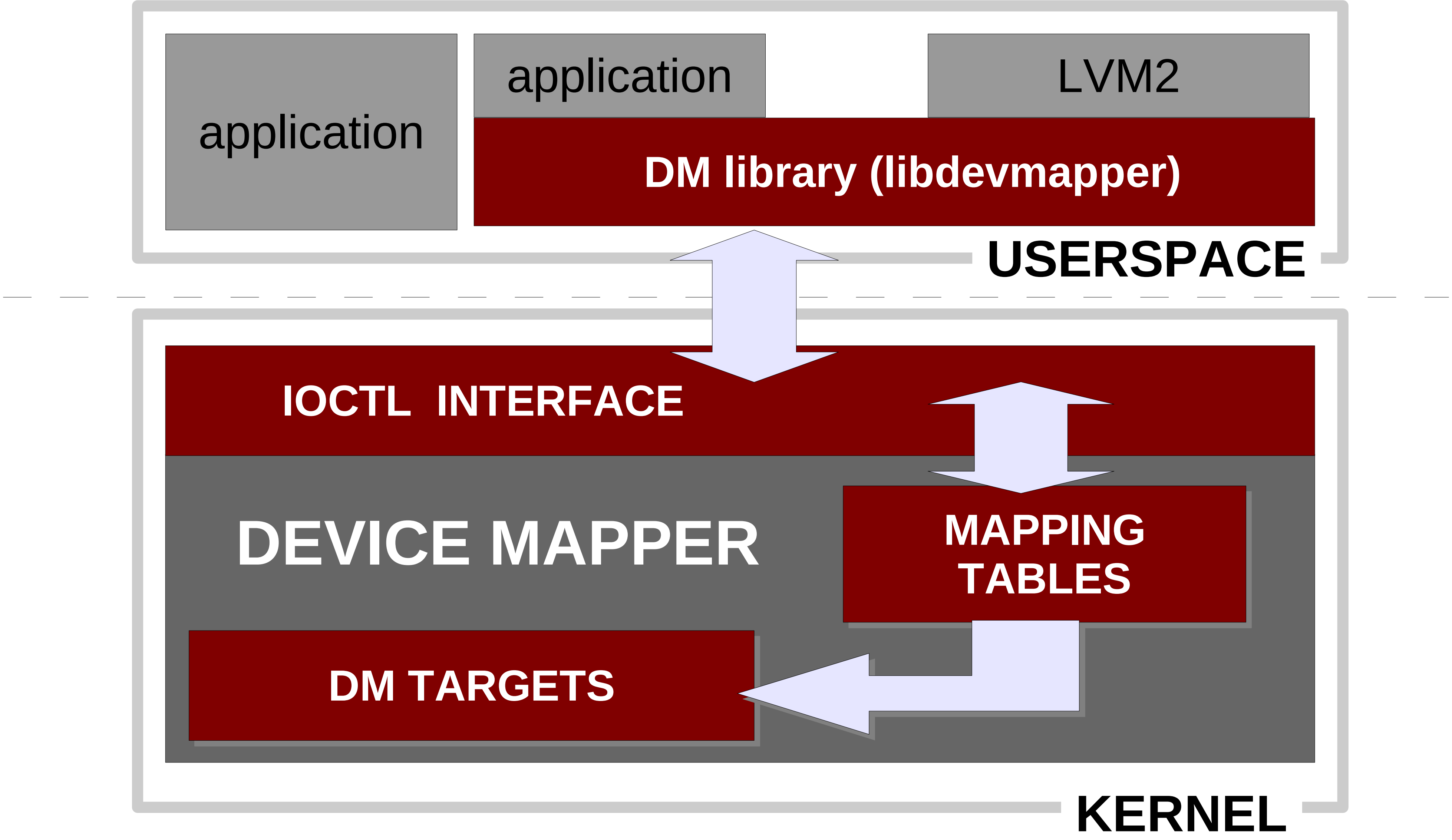
- **LVM** (logical volumes, volume groups)
  - ► managed by userspace tools (LVM2, EVMS, ...)
- **partitions, filesystems**
  - ► managed by userspace tools (fdisk, mkfs, mount, ...)

Device mapper – mapped device access





Device mapper – control interface



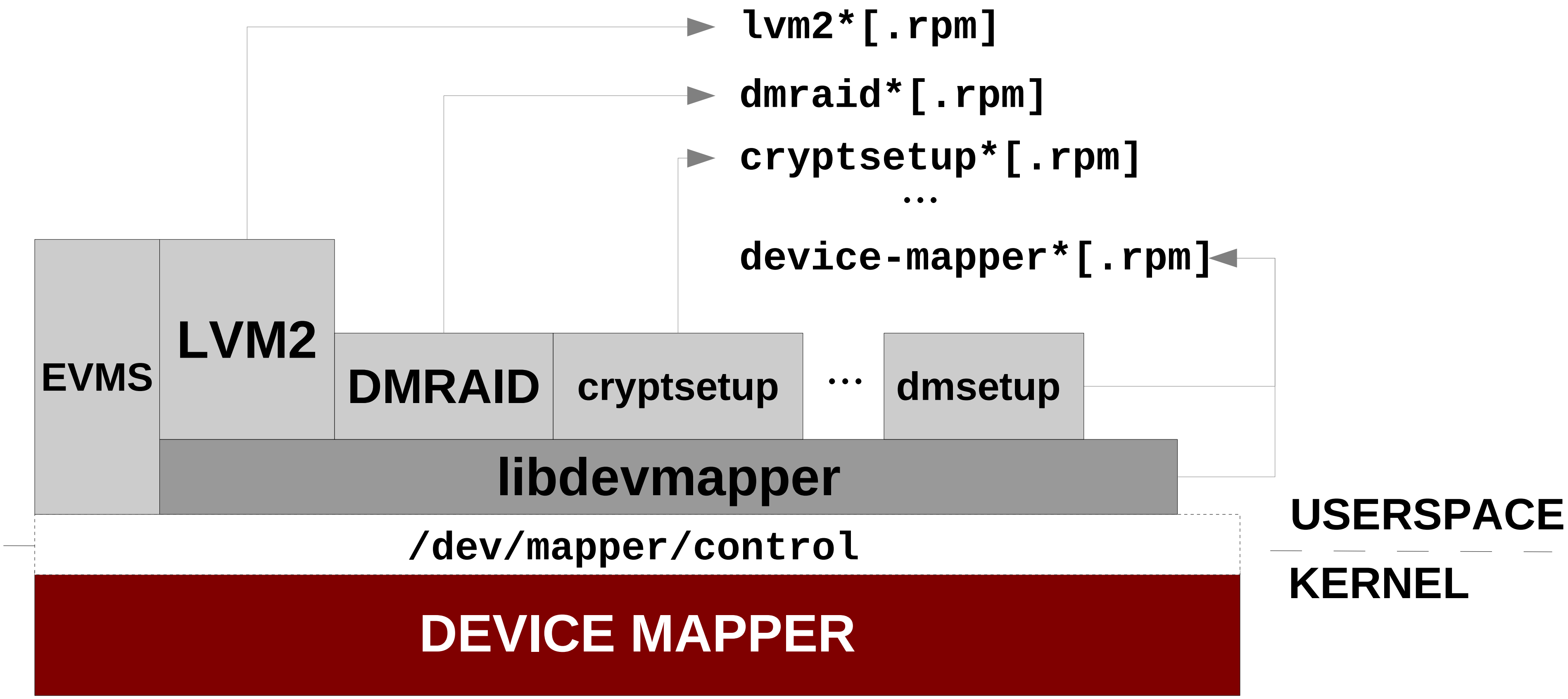
Device mapper - TARGETS

- **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,delay** – test and special targets
- **truecrypt**

...

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

Device mapper – applications



lvm2*[.rpm]

dmraid*[.rpm]

cryptsetup*[.rpm]

...

device-mapper*[.rpm]

EVMS

LVM2

DMRAID

cryptsetup

...

dmsetup

libdevmapper

USERSPACE

/dev/mapper/control

KERNEL

DEVICE MAPPER

Simulate disk fail – 100MB disk with bad 9th sector

Create new disk and map 9th sector to error target

```
dmsetup create bad_disk
   0  8       linear /dev/sdb1 0
   8  1       error
   9  204791  linear /dev/sdb1 9
```

Set readahead to 0, check block device size

```
blockdev --setra 0 /dev/mapper/bad_disk
blockdev --getsz /dev/mapper/bad_disk
```

DD should fail on 9th sector...

```
dd if=/dev/mapper/bad_disk of=/dev/null bs=1k count=4
```

ok

```
dd if=/dev/mapper/bad_disk of=/dev/null bs=1k count=5
```

io error

DM: Creating fake 8TB disk

Create disk where first 32k is writable (real device) and rest is mapped to zero target
(read returns 0, writes are succesfully discharged)

```
dmsetup create fake_disk
   0  64 linear /dev/sdb1 0
   64  17179869120 zero
```

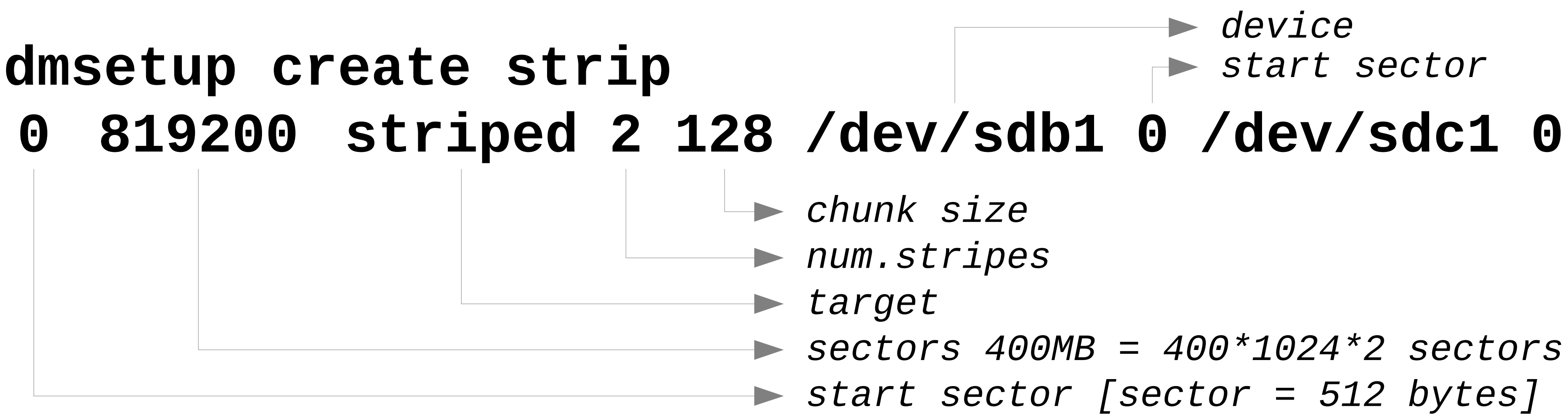
```
blockdev --getsize64 /dev/mapper/fake_disk
```

DM & LVM2: Creating fake 8TB disk ...

Another extension of this concept is create LVM snapshot over fake disk and use snapshot volume – then you can even write into arbitrary part of device (but not to overfill maximal snapshot size).

```
dmsetup create pv_8tb <<EOF    fake disk
0 1024 linear /dev/sdb1 0
1024 17179868160 zero
EOF
(Note: filter /dev/sdb1 in lvm.conf to not scan for PV directly !)

pvcreate /dev/mapper/pv_8tb    use as LVM physical volume
pvcreate /dev/sdh1    physical volume for snapshot

vgcreate vg_8tb /dev/mapper/pv_8tb
lvcreate -n lv_fake_8tb -l 100%FREE  vg_8tb /dev/mapper/pv_8tb

vgextend vg_8tb /dev/sdh1
lvcreate -n lv_real_8tb -s -l 100%FREE vg_8tb/lv_fake_8tb /dev/sdh1

   here you can use /dev/mapper/lv_real_8tb

vgchange -a n vg_8tb    clean everything
lvremove vg_8tb/lv_real_8tb
lvremove vg_8tb/lv_fake_8tb
vgremove vg_8tb
pvremove /dev/mapper/pv_8tb /dev/sdh1

dmsetup remove pv_8tb
```

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
```

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,...
```