



# Disk Management Theory



# Unit Objectives

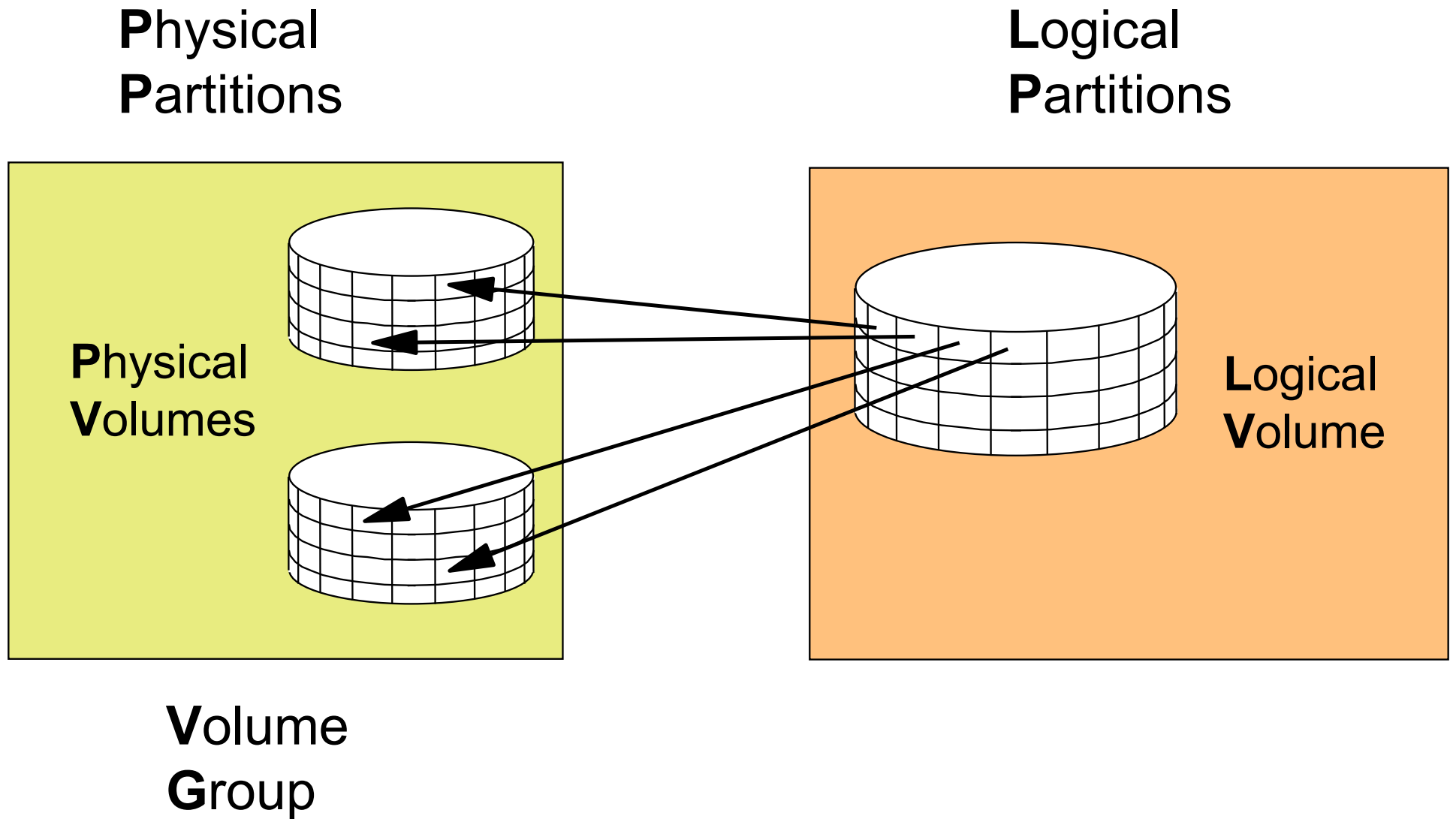
---

After completing this unit, you should be able to:

- Explain where LVM information is stored
- Solve ODM-related LVM problems
- Set up mirroring appropriate to your needs
- Describe the quorum mechanism
- Explain the physical volume states used by the LVM

# LVM Terms

---



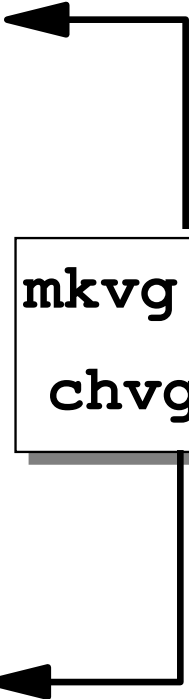
# Volume Group Limits

- Normal Volume Groups (**mkvg**)

Number of disks:	Max. number of partitions/disk:
1	32512
2	16256
4	8128
8	4064
16	2032
<b>32</b>	<b>1016</b>

- Big Volume Groups (**mkvg -B** or **chvg -B**)

Number of disks:	Max. number of partitions/disk:
1	130048
2	65024
4	32512
8	16256
16	8128
32	4064
64	2032
<b>128</b>	<b>1016</b>



```
mkvg -t
chvg -t
```

# Scalable Volume Groups

---

- Introduced in AIX 5L V5.3
- Support 1024 disks per volume group.
- Support 4096 logical volumes per volume group.
- Maximum number of PPs is VG instead of PV dependent.
- LV control information is kept in the VGDA.
- No need to set the maximum values at creation time; the initial settings can always be increased at a later date.

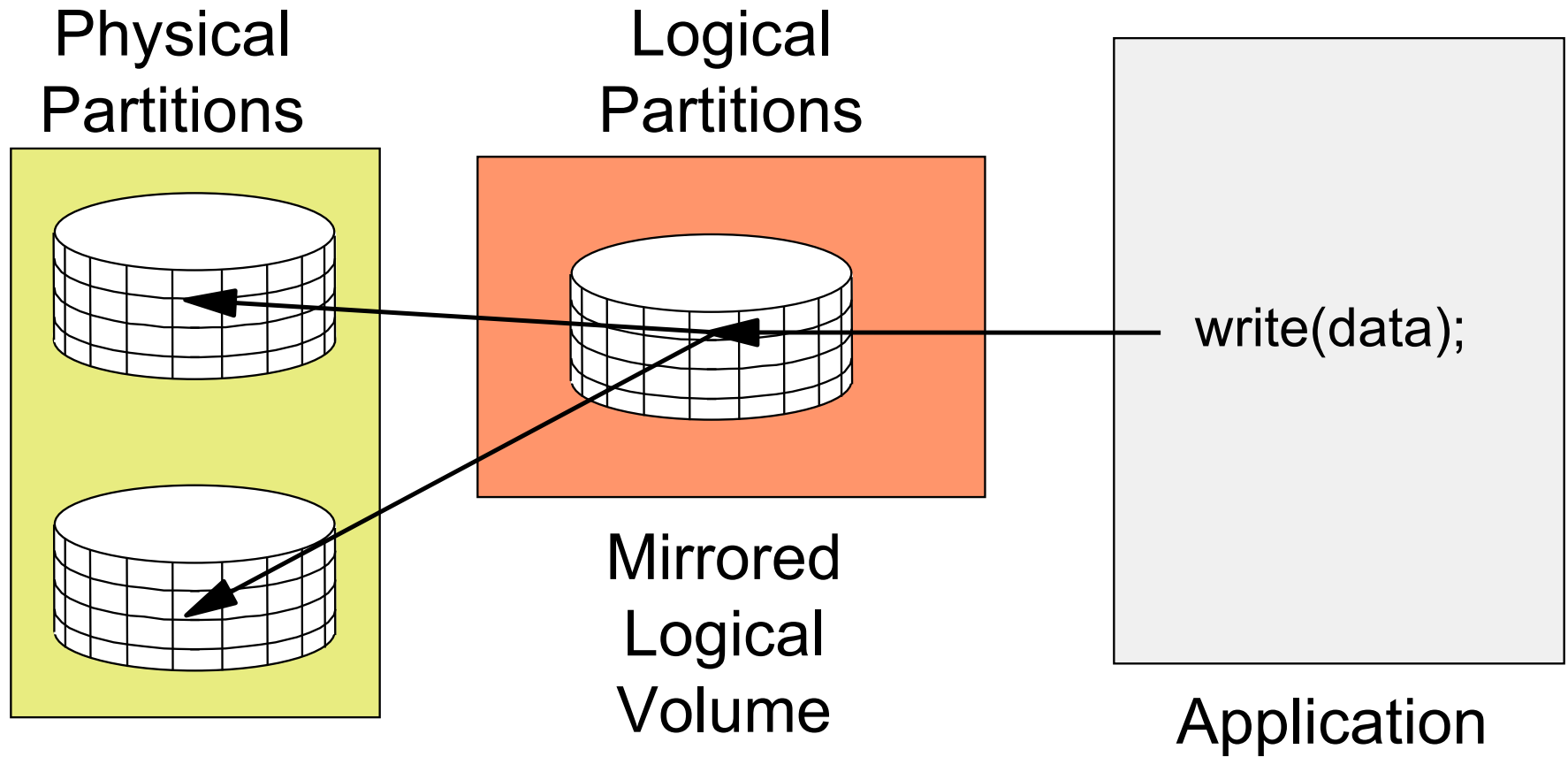
# Configuration Limits for Volume Groups

---

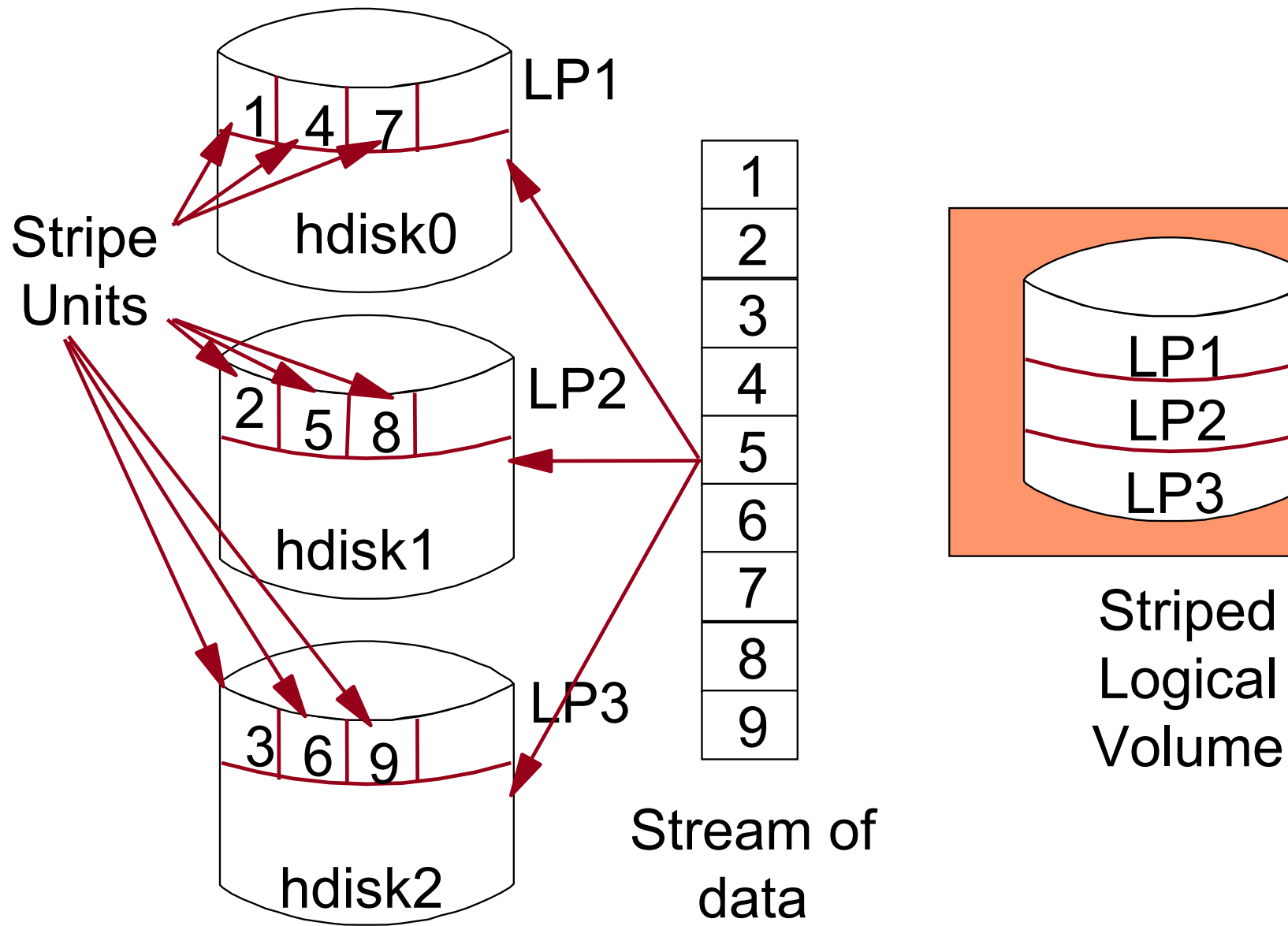
<b>VG Type</b>	<b>Maximum PVs</b>	<b>Maximum LVs</b>	<b>Maximum PPs per VG</b>	<b>Maximum PP size</b>
Normal VG	32	256	32512 (1016*32)	1 GB
Big VG	128	512	130048 (1016*128)	1 GB
Scalable VG	1024	4096	2097152	128 GB

# Mirroring

---

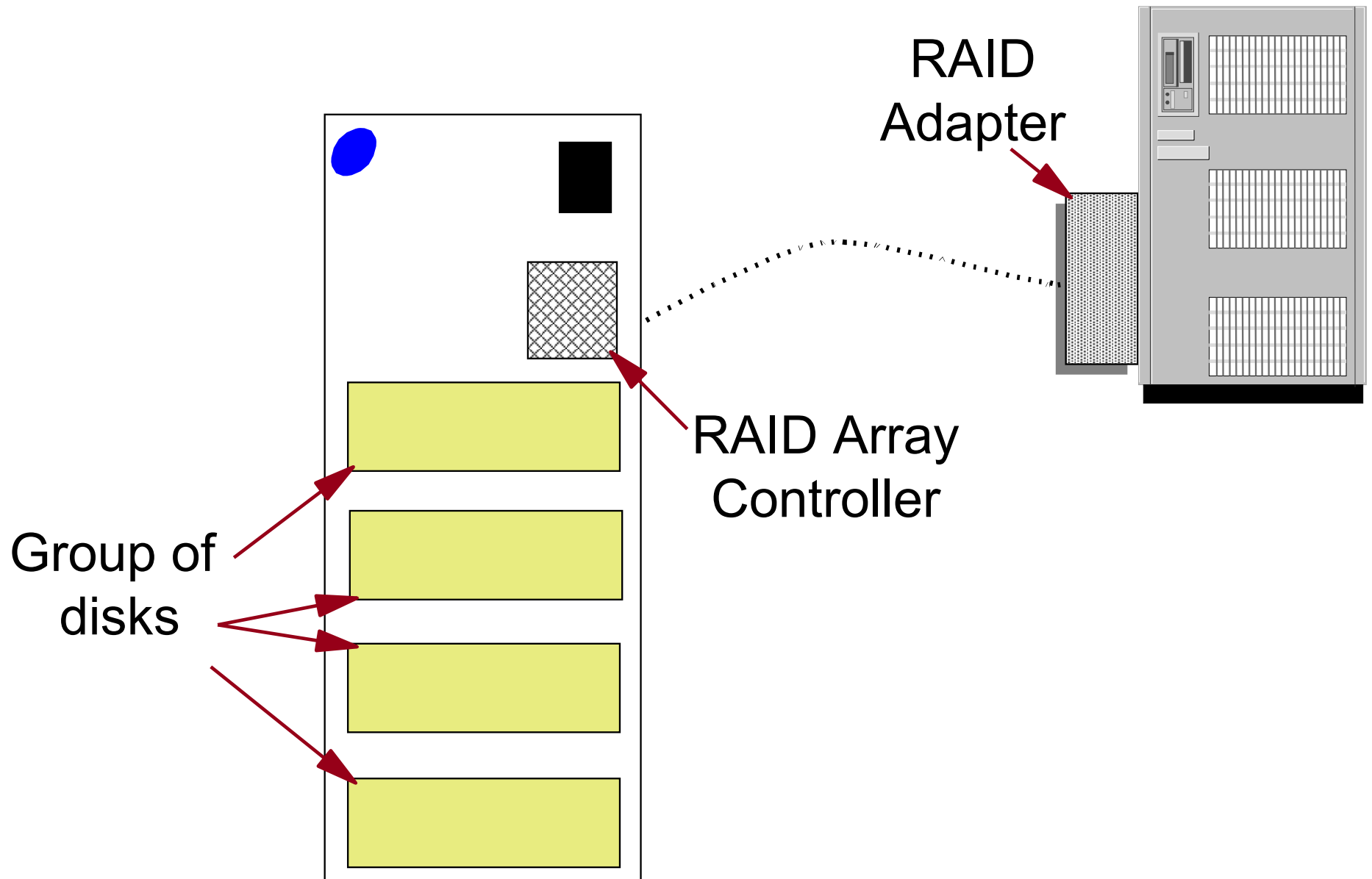


# Striping



# Mirroring and Striping with RAID

RAID = **R**edundant **A**rray of **I**ndependent **D**isks



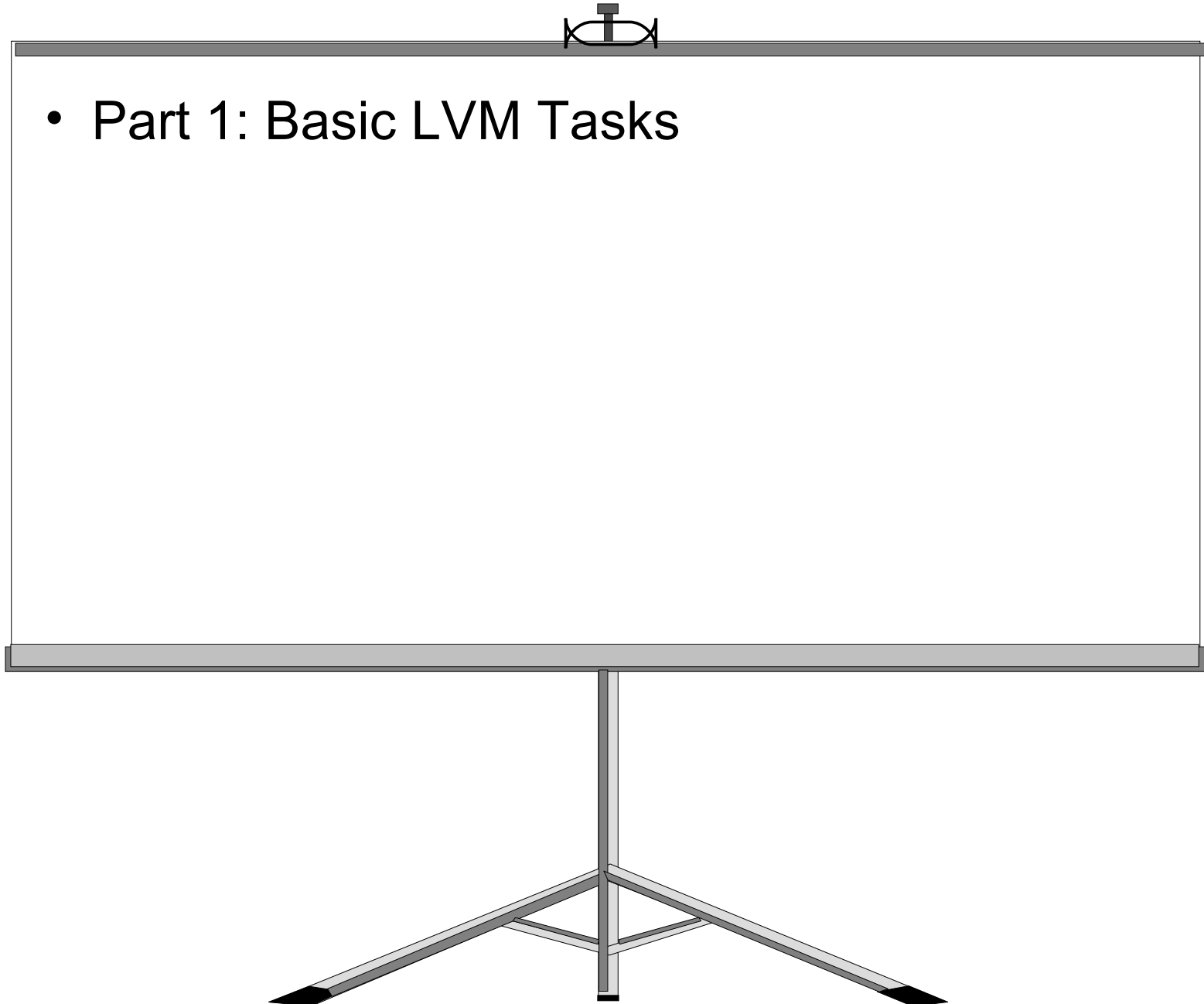
# RAID Levels You Should Know About

---

RAID Level	Implementation	Explanation
0	Striping	Data is split into blocks. These blocks are written to or read from a series of disks in parallel. No data redundancy.
1	Mirroring	Data is split into blocks and duplicate copies are kept on separate disks. If any disk in the array fails, the mirrored data can be used.
5	Striping with parity drives	Data is split into blocks that are striped across the disks. For each block, parity information is written that allows the reconstruction in case of a disk failure.

# Exercise 5: LVM Tasks and Problems (Part 1)

---



# LVM Identifiers

Goal: Unique worldwide identifiers for

- Volume groups
- Hard disks
- Logical volumes

```
# lsvg rootvg
... VG IDENTIFIER: 00c35ba000004c000000001157f54bf78

# lspv
hdisk0    00c35ba07b2e24f0    rootvg    active
...

# lslv hd4
LOGICAL VOLUME:      hd4    VOLUME GROUP: rootvg
LV IDENTIFIER: 00c35ba000004c000000001157f54bf78.4 ...
...

# uname -m
00C35BA04C00
```

The diagram illustrates the structure of LVM identifiers using red arrows:

- An arrow points from the text "32 bytes long" to the end of the VG IDENTIFIER (00c35ba000004c000000001157f54bf78).
- An arrow points from the text "32 bytes long (16 are shown)" to the end of the hdisk0 identifier (00c35ba07b2e24f0).
- An arrow points from the text "VGID.minor number" to the end of the LV IDENTIFIER (00c35ba000004c000000001157f54bf78.4).

# LVM Data on Disk Control Blocks

---

## Volume Group Descriptor Area (VGDA)

- Most important data structure of LVM
- Global to the volume group (same on each disk)
- One or two copies per disk

## Volume Group Status Area (VGSA)

- Tracks the state of mirrored copies
- One or two copies per disk

## Logical Volume Control Block (LVCB)

- Has historically occupied first 512 bytes of each logical volume
- Contains LV attributes (policies, number of copies)
- Should not be overwritten by applications using raw devices!

# LVM Data in the Operating System

---

## Object Data Manager (ODM)

- Physical volumes, volume groups, and logical volumes are represented as devices (customized devices)
- **CuDv, CuAt, CuDvDr, CuDep**

## AIX Files

- |                           |   |
|---------------------------|---|
| • <b>/etc/vg/vgVGID</b>   | Handle to the VGDA copy in memory   |
| • <b>/dev/hdiskX</b>      | Special file for a disk   |
| • <b>/dev/VGname</b>      | Special file for administrative access to a VG  |
| • <b>/dev/LVname</b>      | Special file for a logical volume   |
| • <b>/etc/filesystems</b> | Used by the <b>mount</b> command to associate LV name, file system log, and mount point |

# Contents of the VGDA

---

<b>Header Time Stamp</b>	<ul style="list-style-type: none"><li>• Updated when VG is changed</li></ul>
<b>Physical Volume List</b>	<ul style="list-style-type: none"><li>• PVIDs only (no PV names)</li><li>• VGDA count and PV state</li></ul>
<b>Logical Volume List</b>	<ul style="list-style-type: none"><li>• LVIDs and LV names</li><li>• Number of copies</li></ul>
<b>Physical Partition Map</b>	<ul style="list-style-type: none"><li>• Maps LPs to PPs</li></ul>
<b>Trailer Time Stamp</b>	<ul style="list-style-type: none"><li>• Must contain same value as header time stamp</li></ul>

# VGDA Example

```
# lqueryvg -p hdisk1 -At
```

Max LVs:256

PP Size:20

1:\_\_\_\_\_

Free PPs:12216

LV count:3

PV count:1

2:\_\_\_\_\_

3:\_\_\_\_\_

Total VGDA:2

4:\_\_\_\_\_

MAX PPs per PV:32768

MAX PVs:1024

Logical:

00c35ba000004c00000001157fcf6bdf.11v001

00c35ba000004c00000001157fcf6bdf.21v011

00c35ba000004c00000001157fcf6bdf.31v021

5:\_\_\_\_\_

Physical:00c35ba07fcf6b9320

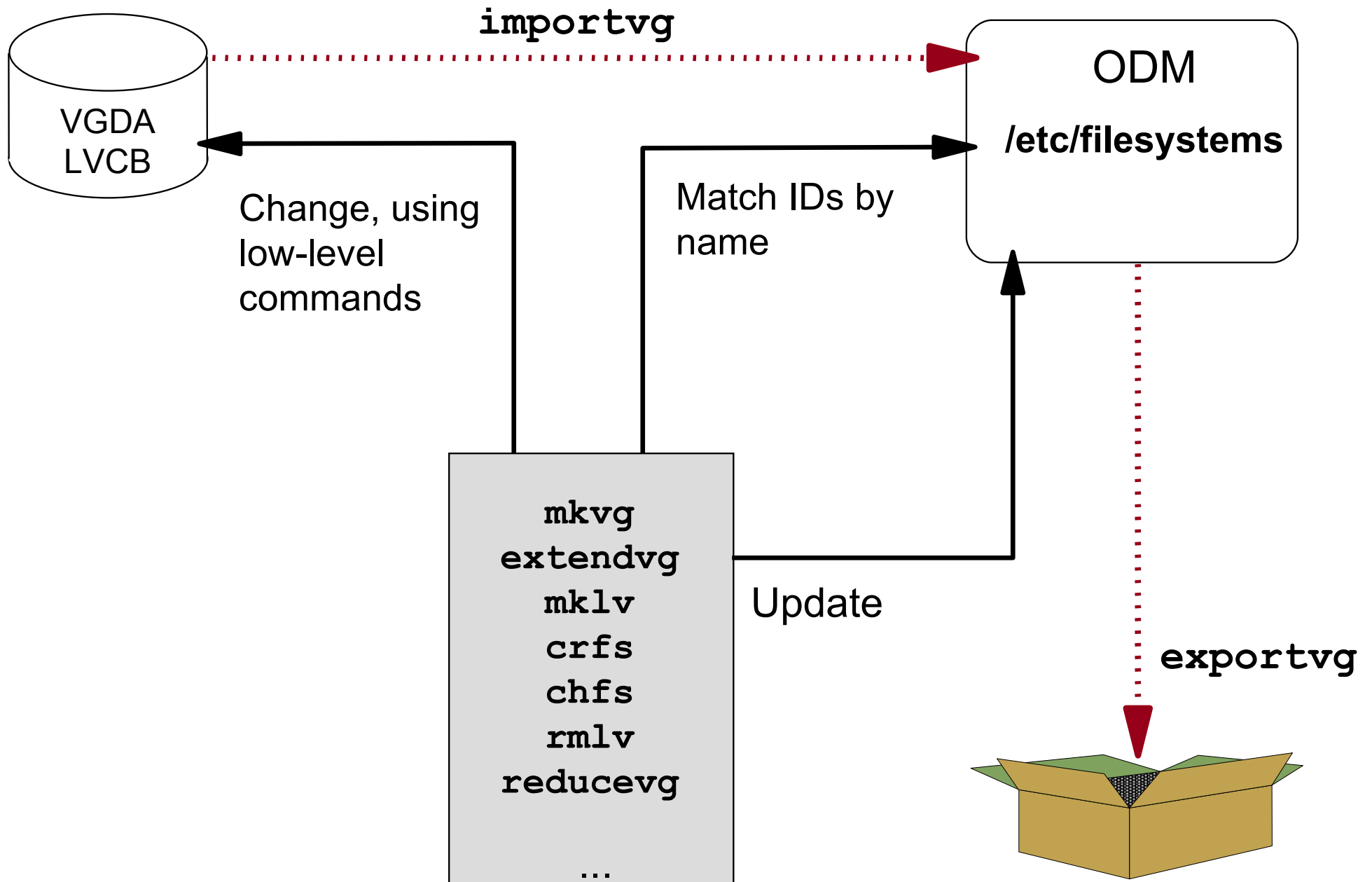
6:\_\_\_\_\_

7:\_\_\_\_\_

# The Logical Volume Control Block (LVCB)

```
# getlvcb -AT hd2
    AIX LVCB
    intrapolicy = c
    copies = 1
    interpolicy = m
    lvid = 00c35ba000004c00000001157f54bf78.5
    lvname = hd2
    label = /usr
    machine id = 35BA04C00
    number lps = 102
    relocatable = y
    strict = y
    stripe width = 0
    stripe size in exponent = 0
    type = jfs2
    upperbound = 32
    fs =
    time created   = Mon Oct  8 11:16:49 2007
    time modified  = Mon Oct  8 07:00:09 2007
```

# How LVM Interacts with ODM and VGDA



# ODM Entries for Physical Volumes (1 of 3)

```
# odmget -q "name like hdisk[02]" CuDv
```

CuDv:

```
name = "hdisk0"  
status = 1  
chgstatus = 2  
ddins = "scsidisk"  
location = ""  
parent = "vscsi0"  
connwhere = "8100000000000"  
PdDvLn = "disk/vscsi/vdisk"
```

CuDv:

```
name = "hdisk2"  
status = 1  
chgstatus = 0  
ddins = "scdisk"  
location = "01-08-01-8,0"  
parent = "scsi1"  
connwhere = "8,0"  
PdDvLn = "disk/scsi/scsd"
```

# ODM Entries for Physical Volumes (2 of 3)

---

```
# odmget -q "name=hdisk0 and attribute=pvid" CuAt
CuAt:
    name = "hdisk0"
    attribute = "pvid"
    value = "00c35ba07b2e24f000000000000000000000"
    type = "R"
    generic = "D"
    rep = "s"
    nls_index = 11
```

# ODM Entries for Physical Volumes (3 of 3)

```
# odmget -q "value3 like hdisk[03]" CuDvDr
```

```
CuDvDr:
```

```
    resource = "devno"  
    value1 = "17"  
    value2 = "0"  
    value3 = "hdisk0"
```

```
CuDvDr:
```

```
    resource = "devno"  
    value1 = "36"  
    value2 = "0"  
    value3 = "hdisk3"
```

```
# ls -l /dev/hdisk[03]
```

```
brw----- 1 root  system  17, 0 Oct 08 06:17 /dev/hdisk0  
brw----- 1 root  system  36, 0 Oct 08 09:19 /dev/hdisk3
```

# ODM Entries for Volume Groups (1 of 2)

```
# odmget -q "name=rootvg" CuDv
```

```
CuDv:
```

```
    name = "rootvg"
```

```
    status = 0
```

```
    chgstatus = 1
```

```
    ddins = ""
```

```
    location = ""
```

```
    parent = ""
```

```
    connwhere = ""
```

```
    PdDvLn = "logical_volume/vgsubclass/vgtype"
```

```
# odmget -q "name=rootvg" CuAt
```

```
CuAt:
```

```
    name = "rootvg"
```

```
    attribute = "vgserial_id"
```

```
    value = "00c35ba000004c000000001157f54bf78"
```

```
    type = "R"
```

```
    generic = "D"
```

```
    rep = "n"
```

```
    nls_index = 637
```

(output continues on next page)

## ODM Entries for Volume Groups (2 of 2)

```
# odmget -q "name=rootvg" CuAt
```

...

**CuAt:**

```
name = "rootvg"
attribute = "timestamp"
value = "470a1bc9243ed693"
type = "R"
generic = "DU"
rep = "s"
nls_index = 0
```

**CuAt :**

```
name = "rootvg"  
attribute = "pv"  
value = "00c35ba07b2e24f000000000000000000000"  
type = "R"  
generic = ""  
rep = "sl"  
nls index = 0
```

# ODM Entries for Logical Volumes (1 of 2)

```
# odmget -q "name=hd2" CuDv
```

```
CuDv:
```

```
    name = "hd2"
```

```
    status = 0
```

```
    chgstatus = 1
```

```
    ddins = ""
```

```
    location = ""
```

```
    parent = "rootvg"
```

```
    connwhere = ""
```

```
    PdDvLn = "logical_volume/lvsubclass/lvtype"
```

```
# odmget -q "name=hd2" CuAt
```

```
CuAt:
```

```
    name = "hd2"
```

```
    attribute = "lvserial_id"
```

```
    value = "00c35ba000004c000000001157f54bf78.5"
```

```
    type = "R"
```

```
    generic = "D"
```

```
    rep = "n"
```

```
    nls_index = 648
```

Other attributes include `intra`,  
`stripe_width`, `type`, etc.

# ODM Entries for Logical Volumes (2 of 2)

---

```
# odmget -q "value3=hd2" CuDvDr
CuDvDr:
```

```
    resource = "devno"
    value1 = "10"
    value2 = "5"
    value3 = "hd2"
```

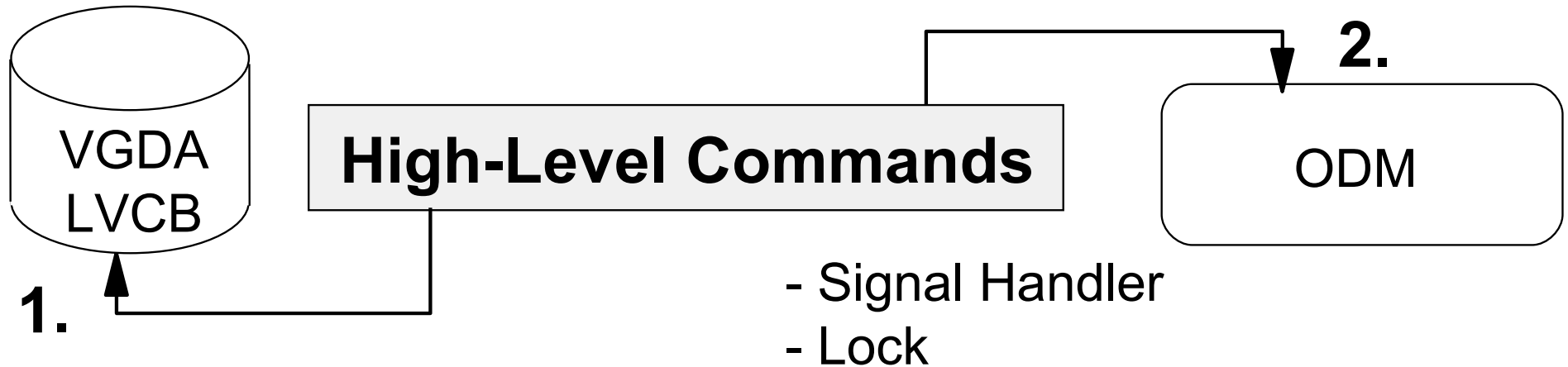
```
# ls -l /dev/hd2
```

```
brw----- 1 root system 10,5 08 Jan 06:56 /dev/hd2
```

```
# odmget -q "dependency=hd2" CuDep
CuDep:
```

```
    name = "rootvg"
    dependency = "hd2"
```

# ODM-Related LVM Problems



## What can cause problems ?

- **kill -9**, shutdown, system crash
- Improper use of low-level commands
- Hardware changes without or with wrong software actions
- Full root file system

# Fixing ODM Problems (1 of 2)

---

If the ODM problem is *not in the rootvg*, for example in volume group **homevg**, do the following:

```
# varyoffvg homevg  
  
# exportvg homevg  
  
# importvg -y homevg hdiskX
```

← Remove complete volume group from the ODM

↑ Import volume group and create new ODM objects

# Fixing ODM Problems (2 of 2)

---

If the ODM problem is in the **rootvg**, try using **rvgrecover**:

```
PV=hdisk0
VG=rootvg
cp /etc/objrepos/CuAt /etc/objrepos/CuAt.$$
cp /etc/objrepos/CuDep /etc/objrepos/CuDep.$$
cp /etc/objrepos/CuDv /etc/objrepos/CuDv.$$
cp /etc/objrepos/CuDvDr /etc/objrepos/CuDvDr.$$
lqueryvg -Lp $PV | awk '{print $2}' | while read LVname;
do
    odmdelete -q "name=$LVname" -o CuAt
    odmdelete -q "name=$LVname" -o CuDv
    odmdelete -q "value3=$LVname" -o CuDvDr
done
odmdelete -q "name=$VG" -o CuAt
odmdelete -q "parent=$VG" -o CuDv
odmdelete -q "name=$VG" -o CuDv
odmdelete -q "name=$VG" -o CuDep
odmdelete -q "dependency=$VG" -o CuDep
odmdelete -q "value1=10" -o CuDvDr
odmdelete -q "value3=$VG" -o CuDvDr
importvg -y $VG $PV # ignore lvaryoffvg errors
varyonvg $VG
```

- Uses **odmdelete** to “export” **rootvg**
- Uses **importvg** to import **rootvg**

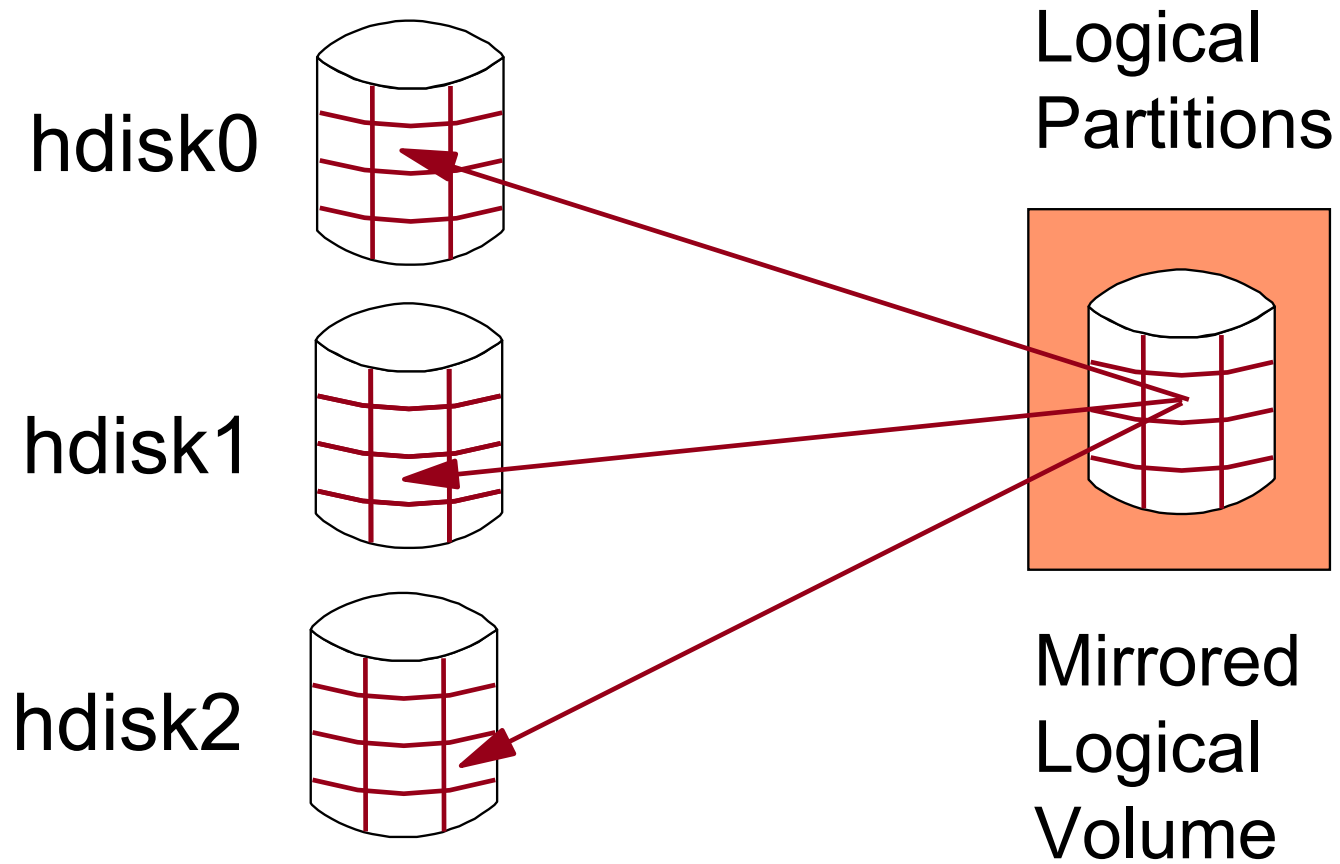
# Exercise 5: LVM Tasks and Problems (Part 2)

---



- Part 2: Analyze and Fix an LVM-related ODM Problem
- Part 2: Analyze and Fix an LVM-related ODM Problem Using **rvgrecover**

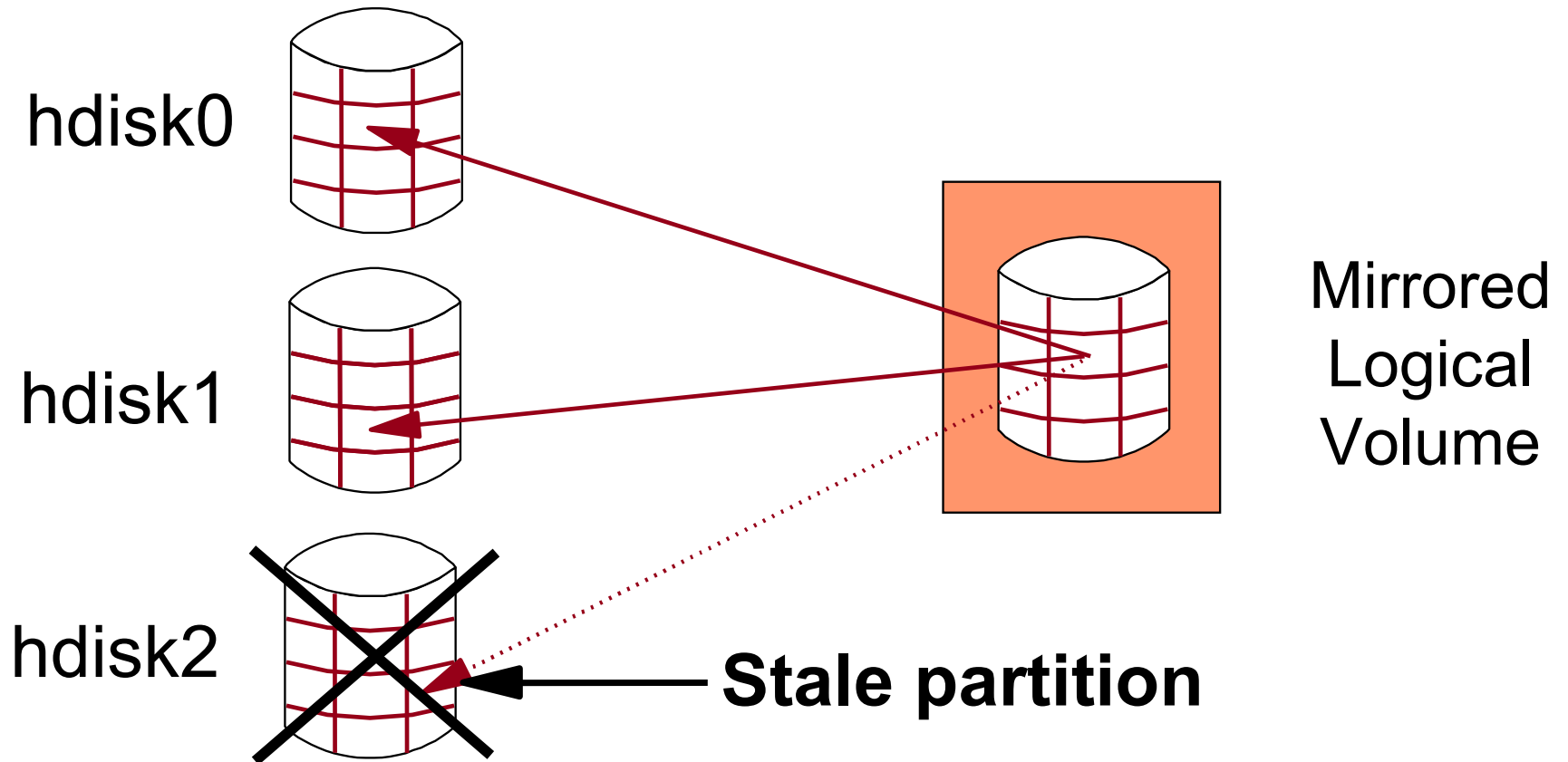
# Mirroring



**VGSA** →

LP:	PP1:	PP2:	PP3:
5	hdisk0, 5	hdisk1, 8	hdisk2, 9

# Stale Partitions



After repair of **hdisk2**:

- **varyonvg VGName** (calls **syncvg -v VGName**)
- Only stale partitions are updated

# Creating Mirrored LVs (smit mk1v)

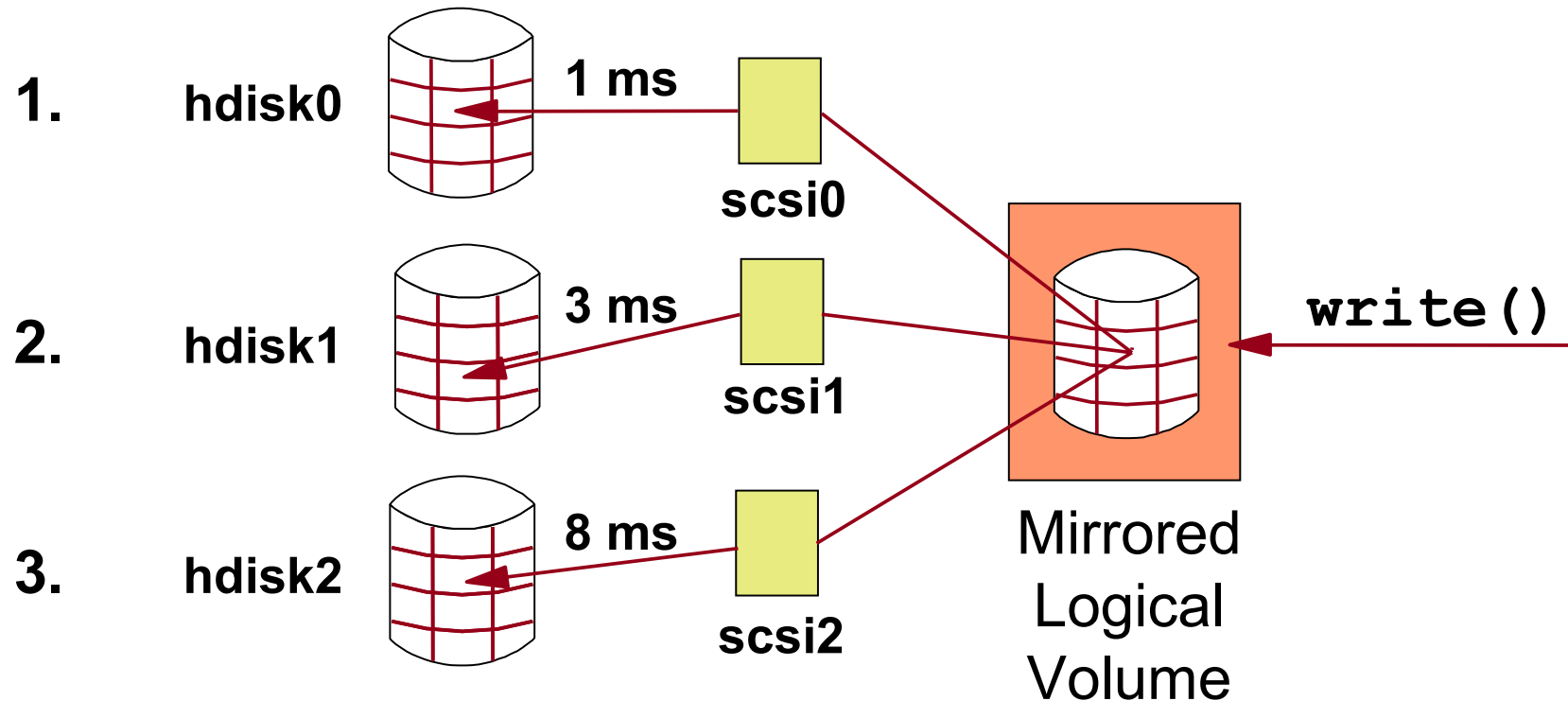
## Add a Logical Volume

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

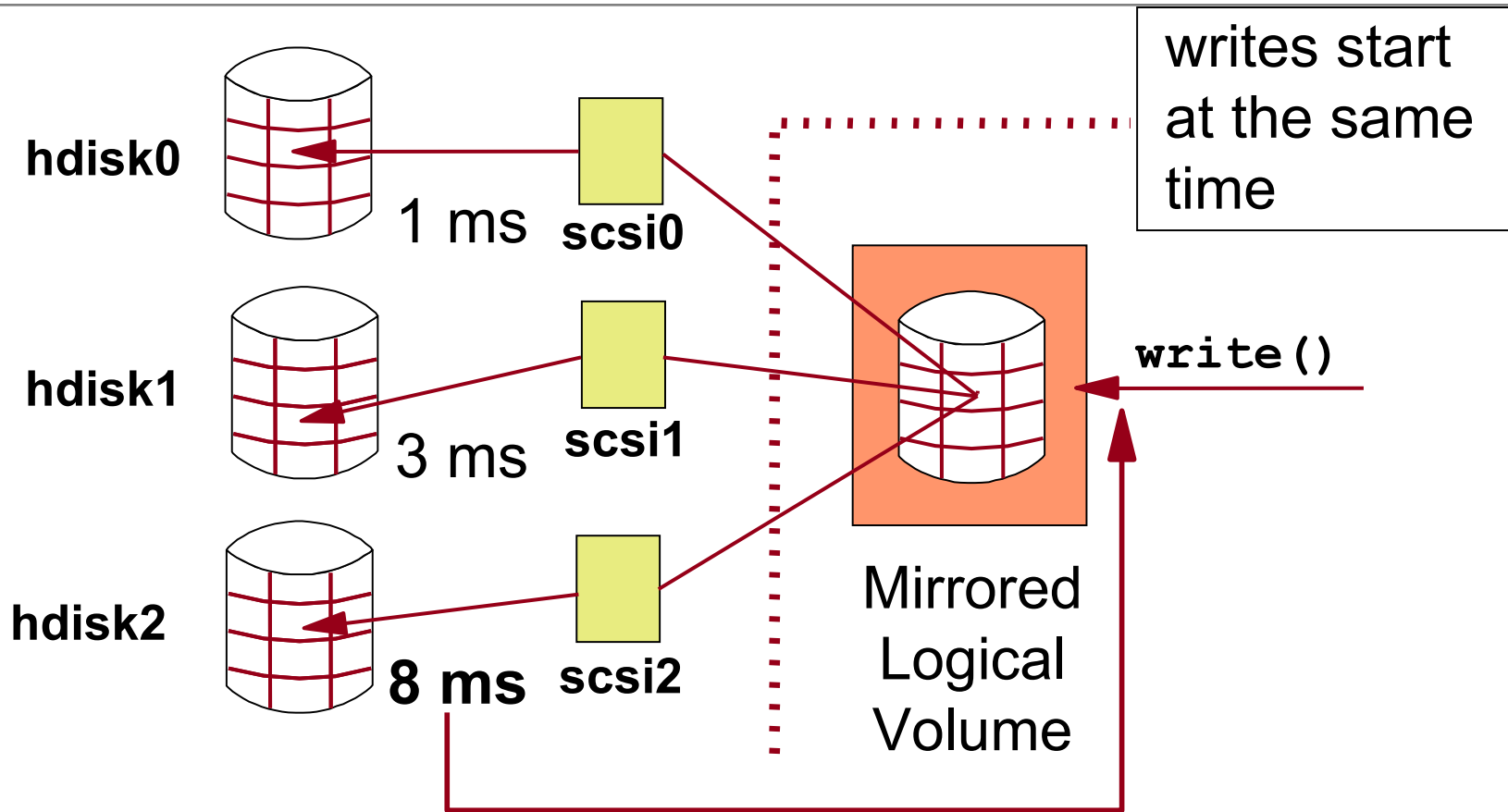
[TOP]	[Entry Fields]
Logical volume NAME	[lv01]
VOLUME GROUP name	rootvg
Number of LOGICAL PARTITIONS	[50]
PHYSICAL VOLUME names	[hdisk2 hdisk4]
Logical Volume TYPE	[]
POSITION on physical volume	edge
RANGE of physical volumes	minimum
MAXIMUM NUMBER of PHYSICAL VOLUMES to use for allocation	[]
Number of COPIES of each logical partition	[2]
Mirror Write Consistency?	active
Allocate each logical partition copy on a SEPARATE physical volume?	yes
...	
SCHEDULING POLICY for reading/writing logical partition copies	parallel

# Scheduling Policies: Sequential



- Second physical write operation is not started unless the first has completed successfully
- In case of a total disk failure, there is always a "good copy"
- Increases availability, but decreases performance
- In this example, the write operation takes 12 ms (1 + 3 + 8)

# Scheduling Policies: Parallel



- Write operations for physical partitions start at the same time: When the longest write (8 ms) finishes, the write operation is complete
- Improves performance (especially READ performance)

# Mirror Write Consistency (MWC)

---

## Problem:

- Parallel scheduling policy and ...
- ... system crashes *before the writes to all mirrors* have been completed
- Mirrors of the logical volume are in an *inconsistent* state

## Solution: Mirror Write Consistency (MWC)

- MWC information used to make logical partitions consistent again after reboot
- *Active* MWC uses separate area of each disk (outer edge area)
- Try to place logical volumes that use active MWC in the outer edge area

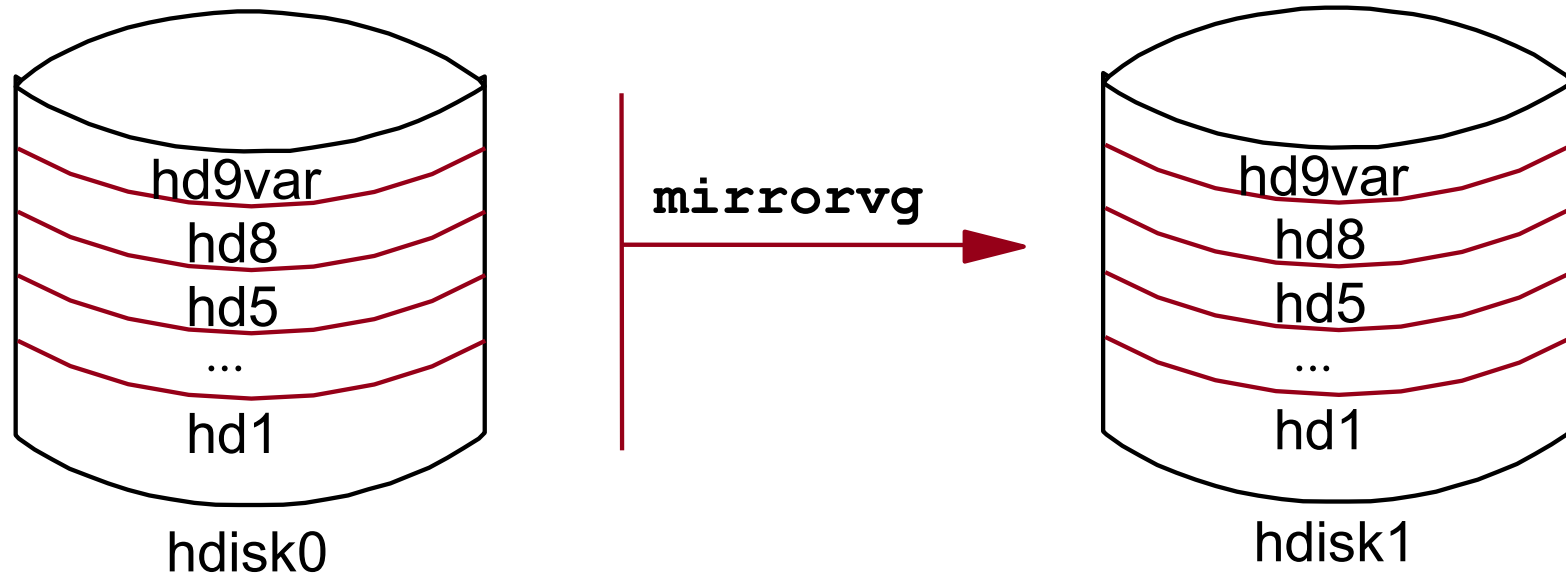
# Adding Mirrors to Existing LVs (mk1vcopy)

## Add Copies to a Logical Volume

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]
Logical volume NAME	[hd2]
NEW TOTAL number of logical partition copies	2
PHYSICAL VOLUME names	[hdisk1]
POSITION on physical volume	outer edge
RANGE of physical volumes	minimum
MAXIMUM NUMBER of PHYSICAL VOLUMES to use for allocation	[32]
Allocate each logical partition copy on a SEPARATE physical volume?	yes
File containing ALLOCATION MAP	[]
SYNCHRONIZE the data in the new logical partition copies?	no

# Mirroring rootvg



1. `extendvg`
2. `chvg -Qn`
3. `mirrorvg -s`
4. `syncvg -v`

5. `bosboot -a`
6. `bootlist`
7. `shutdown -Fr`
8. `bootinfo -b`

- Make a copy of all **rootvg** LVs using **mirrorvg** and place copies on the second disk
- Execute **bosboot** and change your **bootlist**

# Mirroring Volume Groups (mirrorvg)

## Mirror a Volume Group

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]
VOLUME GROUP name	rootvg
Mirror sync mode	[Foreground]
PHYSICAL VOLUME names	[hdisk1]
Number of COPIES of each logical partition	2
Keep Quorum Checking On?	no
Create Exact LV Mapping?	no

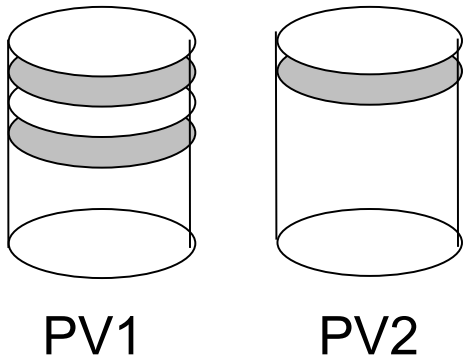
For rootvg, you need to execute:

- `bosboot`
- `bootlist -m normal ...`

# VGDA Count

---

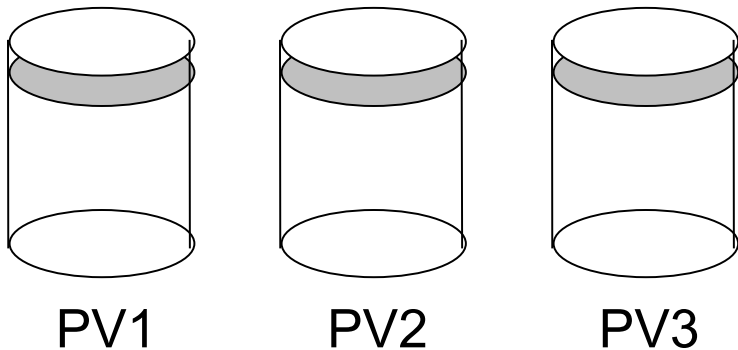
## Two-disk Volume Group



Loss of PV1: Only 33% VGDA's available  
**(No quorum)**

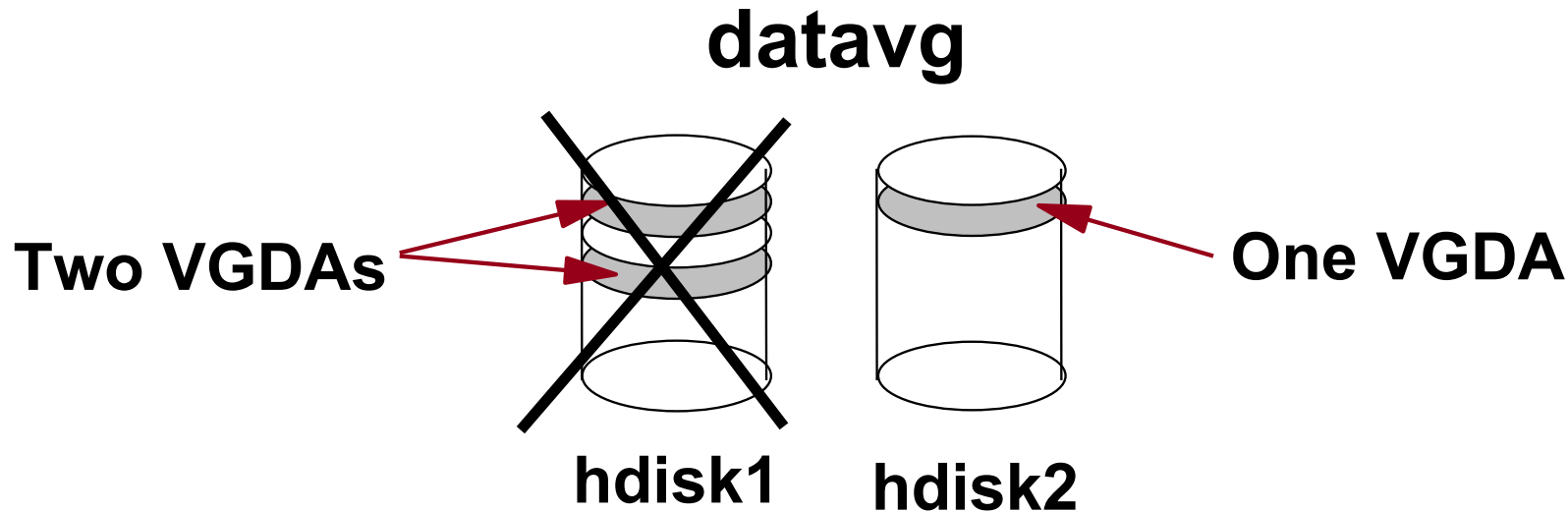
Loss of PV2: 66% of VGDA's available  
**(Quorum)**

## Three-disk Volume Group



Loss of 1 PV: 66% of VGDA's still available  
**(Quorum)**

# Quorum Not Available



**If hdisk1 fails, datavg has no quorum !**

VG not active

VG active

# varyonvg datavg

**FAILS !!!**

**Closed during operation:**

- No more access to LVs
- LVM\_SA\_QUORCLOSE in error log

# Nonquorum Volume Groups

---

With single mirroring, always disable the quorum:

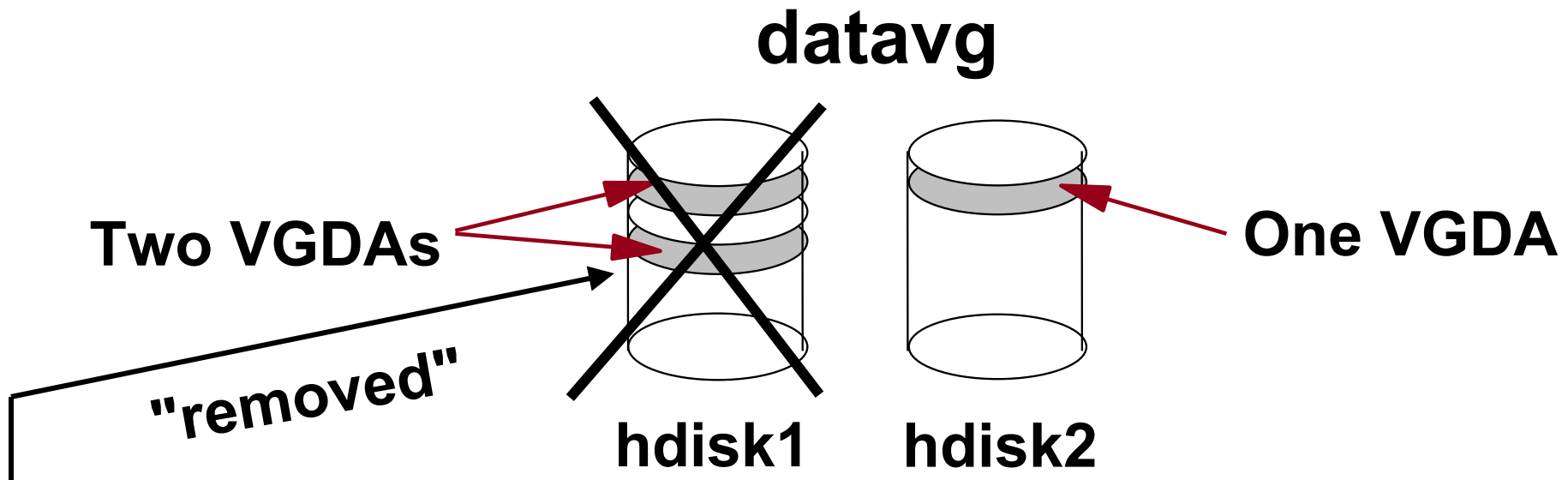
- `chvg -Qn datavg`
- `varyoffvg datavg`
- `varyonvg datavg`

Additional considerations for **rootvg**:

- `chvg -Qn rootvg`
- `bosboot -ad /dev/hdiskX`
- Reboot

- Turning off the quorum checking does not allow a normal **varyonvg** without a quorum
- It does prevents closing of the volume group when quorum is lost

# Forced Varyon (`varyonvg -f`)



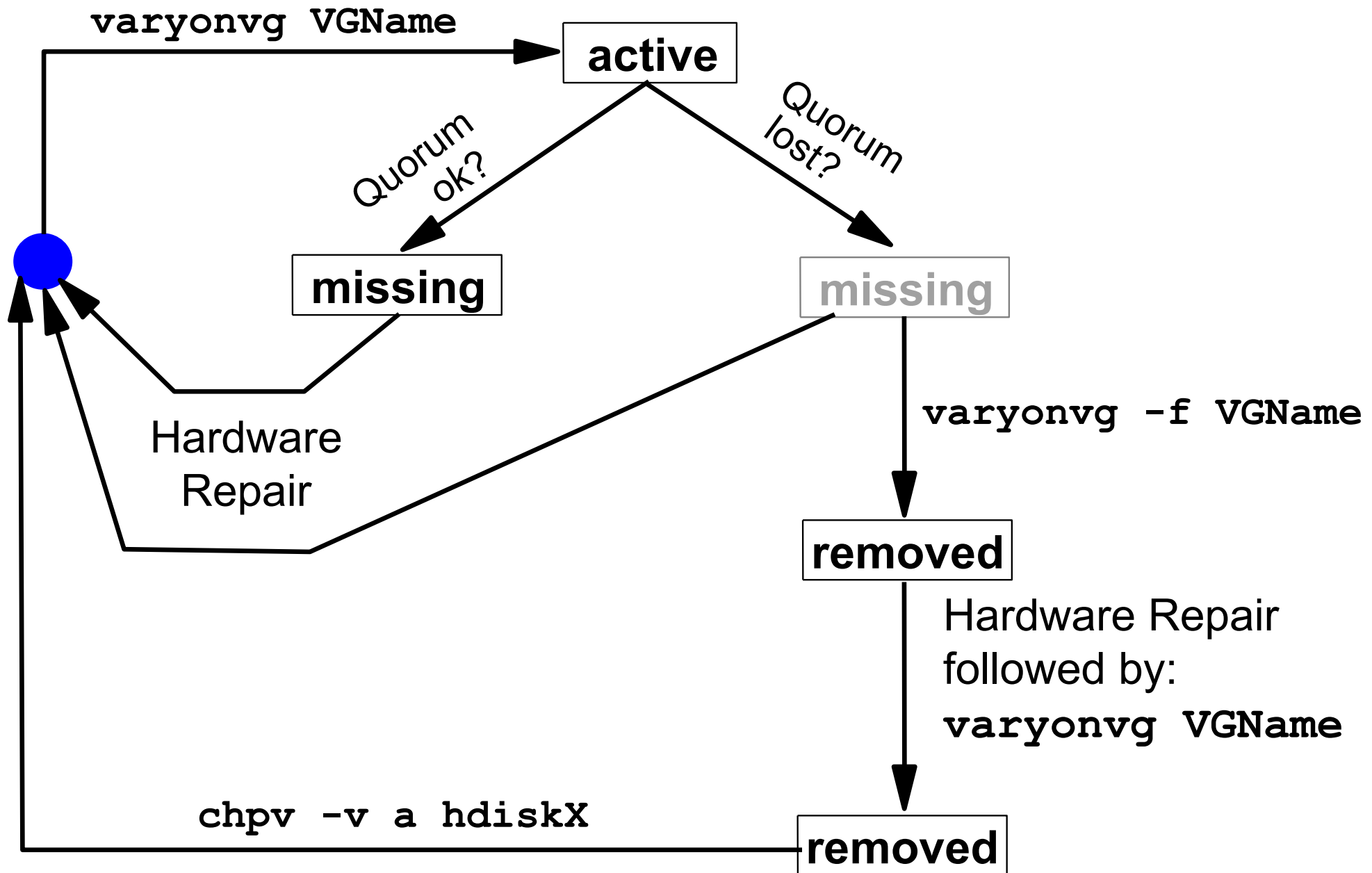
# `varyonvg datavg` *FAILS !!! (even when quorum disabled)*

Check the reason for the failure (cable, adapter, power),  
before doing the following ...

# `varyonvg -f datavg`

Failure accessing **hdisk1**. Set PV STATE to removed.  
Volume group **datavg** is varied on.

# Physical Volume States



# Checkpoint

---

1. (True or False) All LVM information is stored in the ODM.
2. (True or False) You detect that a physical volume **hdisk1** that is contained in your **rootvg** is missing in the ODM. This problem can be fixed by exporting and importing the **rootvg**.
3. (True or False) The LVM supports RAID-5 without separate hardware.

# Checkpoint Solutions

- (True or **False**) All LVM information is stored in the ODM. False. Information is also stored in other AIX files and in disk control blocks (like the VGDA and LVCB).
- (True or **False**) You detect that a physical volume **hdisk1** that is contained in your **rootvg** is missing in the ODM. This problem can be fixed by exporting and importing the **rootvg**. False. Use the **rvgrecover** script instead. This script creates a complete set of new rootvg ODM entries.
- (True or **False**) The LVM supports RAID-5 without separate hardware. False. LVM supports RAID-0, RAID-1, and RAID-10 without additional hardware.

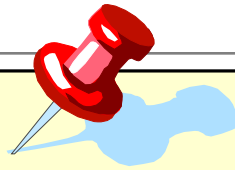
# Exercise 6: Mirroring rootvg

---



- Mirror and Unmirror the Complete **rootvg**

# Unit Summary



- The LVM information is held in a number of different places on the disk, including the ODM and the VGDA
- ODM related problems can be solved by:
  - `exportvg/importvg` (non-rootvg VGs)
  - `rvgrecover` (rootvg)
- Mirroring improves the availability of a system or a logical volume
- Striping improves the performance of a logical volume
- Quorum means that more than 50% of VGDA's must be available