Advanced Linux System Administration

Topic 7. File systems, advanced management

Pablo Abad Fidalgo
José Ángel Herrero Velasco

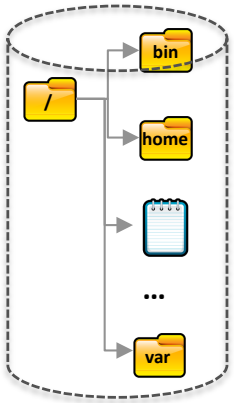
Departamento de Ingeniería Informática y Electrónica

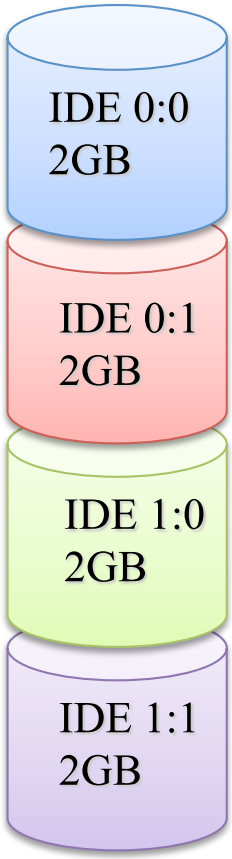
Index

- **Logical Volume Manager (LVM).**
- **Redundant Array of Inexpensive Disks (RAID).**
- **Backup.**

Logical Volume Manager (LVM)

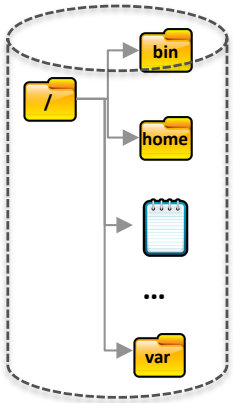
- My File System has a size of 4GB, but I only have 2GB Disks. Is there any solution?

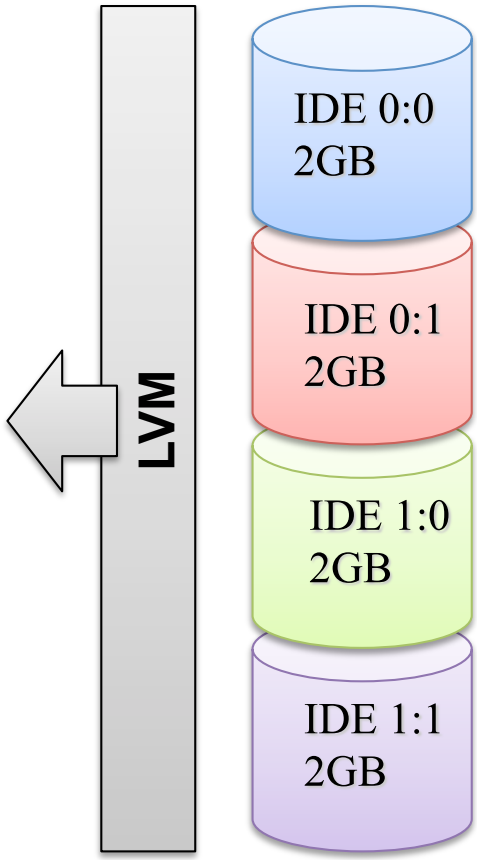




Logical Volume Manager (LVM)

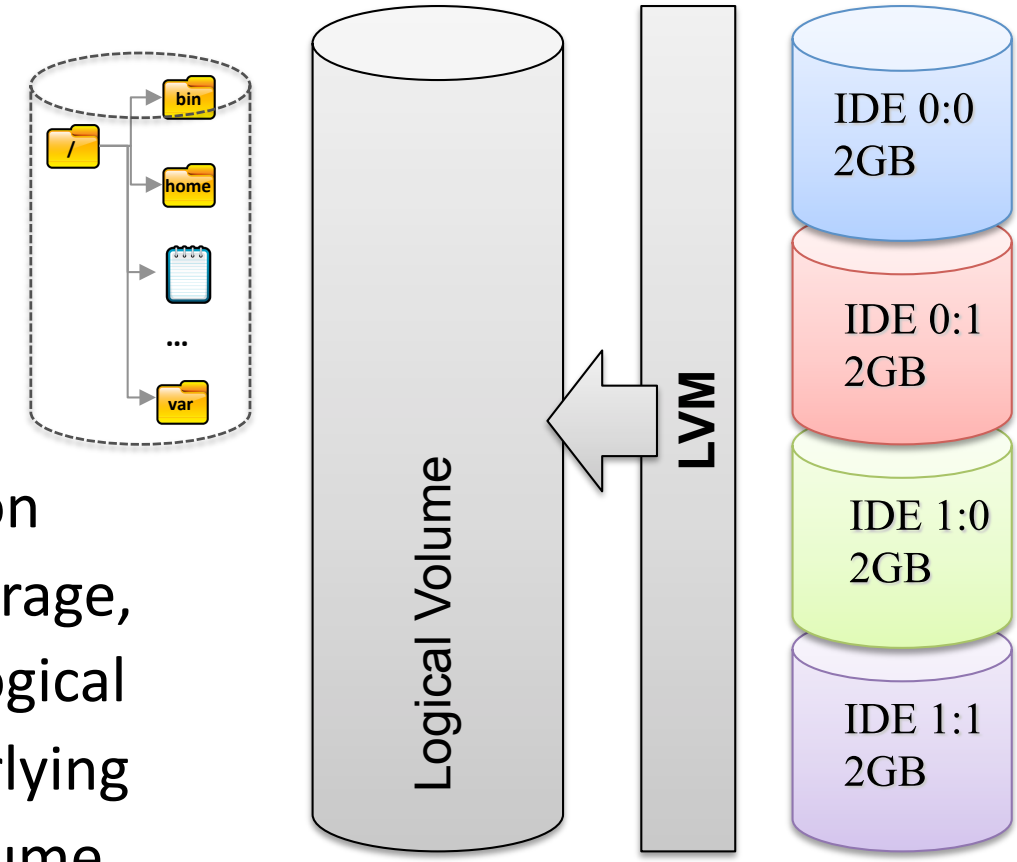
- My File System has a size of 4GB, but I only have 2GB Disks. Is there any solution?





Logical Volume Manager (LVM)

- My File System has a size of 4GB, but I only have 2GB Disks. Is there any solution?



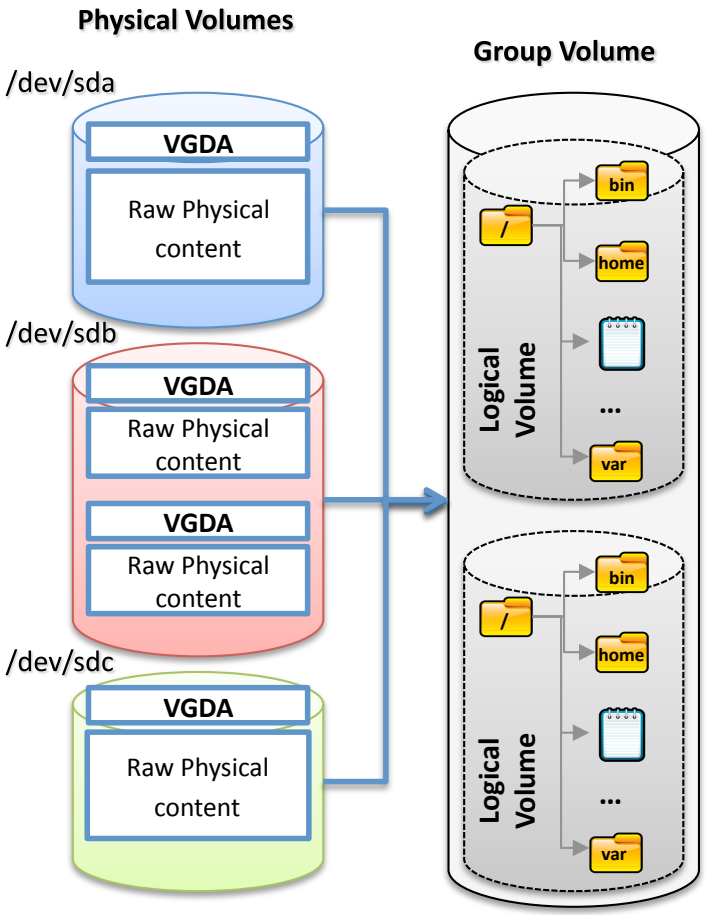
- **LVM:** creates an abstraction layer over the physical storage, allowing the creation of logical volumes("hide" the underlying HW, exposing a single Volume to the SW).

Logical Volume Manager (LVM)

- LVM Advantages:
  - **Flexible management of disk storage:** avoids the limitations imposed by disk physical size. A File System can be extended through multiple disks.
  - **Re-sizeable Storage:** logical volumes can be extended/reduced in a simple way. Some operations do not require File System umounting.
  - **On-line Data movement:** data con be moved between disks while these disks are in use. A disk can be replaced without interrupting system service.
  - **Taking "Snapshots":** eases the process of taking snapshots of devices (backup).

Logical Volume Manager (LVM)

- LVM Hierarchy:
  - Physical Volumes (PV):
    - Lowest level of LVM hierarchy.
    - Complete disk or partition.
    - Contains VGDA (Volume Group Descriptor Area) and the raw physical content.
  - Group Volumes (VG):
    - Equivalent to “super-disks”.
    - Built with one or more PVs:
      - More PVs can be added to the GV without modifying the previous ones.
  - Logical Volumes (LV):
    - Equivalent to “super-partitions”.
    - File Systems are created on a Logical Volume.



Logical Volume Manager (LVM)

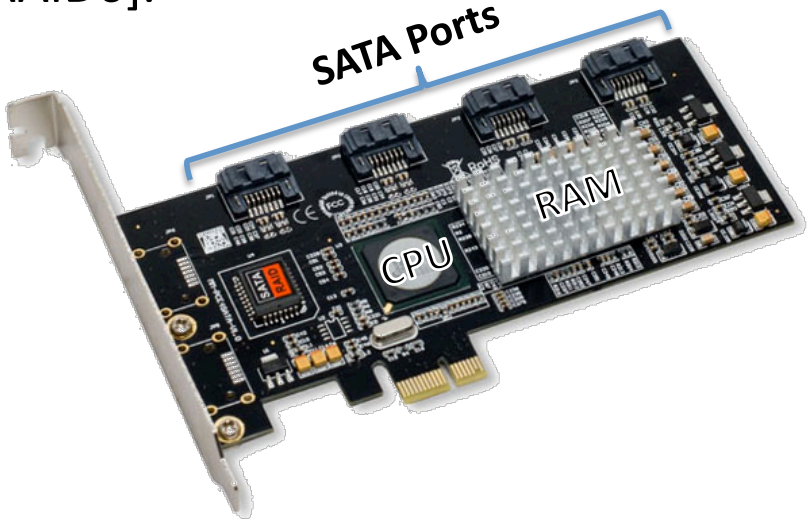
- LVM Administration:
  - Command **pvcreate:** creation of a Physical Volume:
    - Syntax: pvcreate [partition] (It is necessary to previously create a partition with fdisk).
  - Command **vgcreate:** creation of a Group Volume from multiple PVs:
    - Syntax: vgcreate [name-vol] [PVs]:
      - Example: vgcreate vg01 /dev/sdb /dev/sdc1 (group disk sdb and partition sdc1 in a GV in /dev/vg01).
  - Command **lvcreate:** creation of a Logical Volume:
    - Syntax: lvcreate [GV] –L[size] –n[name-vl]:
      - Example: lvcreate vg01 –L1000M –nvol1 (after this we can create the FS with mkfs).
  - Need more storage?:
    - Add a new Physical Volume to the Group Volume (**vgextend**).
    - Extend the Logical Volume to the larger Group Volume (**lvextend**).
    - Re-size the File System (resize2fs):
      - Can do this online !!! (…In contrast, reductions must be done offline).
    - We can also reduce VG and LV (**vgreduce**, **lvreduce**).

Index

- Logical Volume Manager (LVM).
- **Redundant Array of Inexpensive Disks (RAID).**
- Backup.

RAID (Redundant Array of Inexpensive Disks)

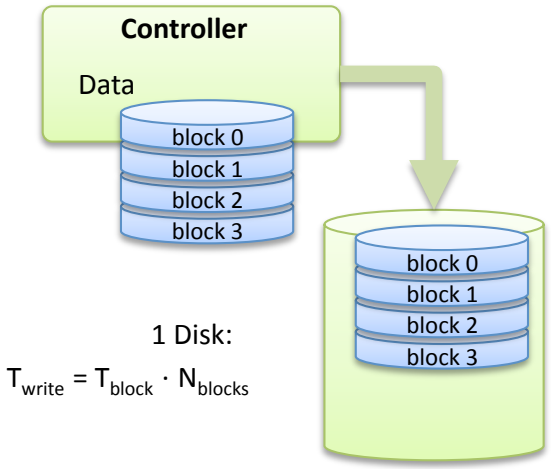
- Mechanism to provide reliability and performance in disks:
  - Make use of multiple disks to create the illusion of a disk with larger capacity, faster access and fault tolerance.
  - Transparent to the user and the OS.
  - Different configuration options (Reliability vs Performance vs Capacity) denoted as levels (standard) [RAID0 … RAID6].

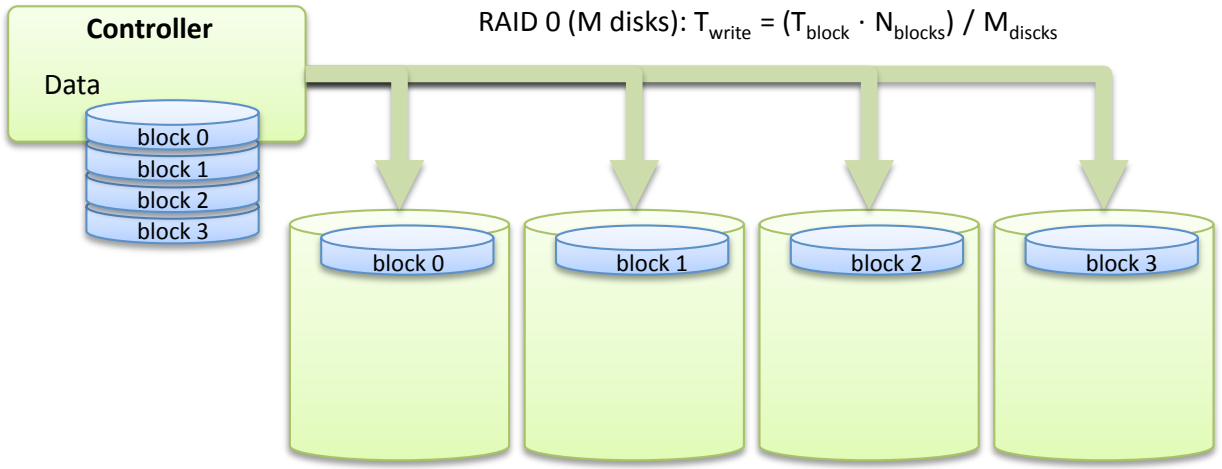


  - Can be implemented via HW or SW:
    - HW Implementation: high efficiency but also high cost:
      - RAID Controller: CPU + dedicated sw, RAM + non-volotile memory.
    - SW Implementation: efficient management of simplest RAID configs (0,1).

RAID (Redundant Array of Inexpensive Disks)

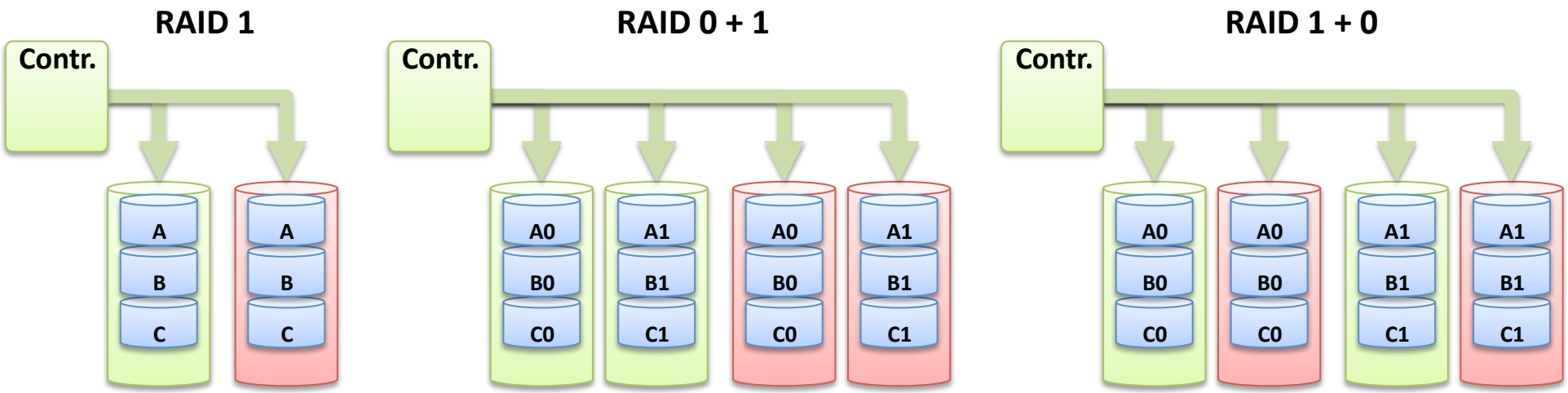
- **RAID 0** (striping):
  - Data are divided into segments (strips) and distributed among multiple disks:
    - Parallel access to disks.
  - **Performance:** improves read/write latency:
    - Speed increases as the number of disks grows (also depends on data size).
  - **Reliability:** no fault tolerance:
  - **Capacity:** 100% storage utilized (no redundancy).





RAID (Redundant Array of Inexpensive Disks)

- **RAID 1** (mirroring):
  - Employ a secondary disk to copy all data being modified.
  - **Performance:** low performance caused by writes (everything replicated).
  - **Reliability:** high redundancy, one disk can fail.
  - **Capacity:** 50% of total capacity available.



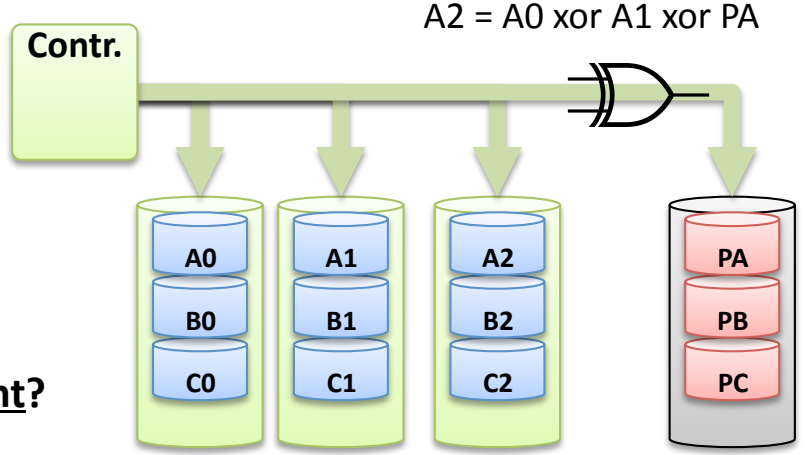
RAID (Redundant Array of Inexpensive Disks)

- **RAID 4** (striping + parity):
  - One disk stores information about the parity of the rest.
  - Block-level division (1 strip = 1 block). Can access disks individually.
  - **Performance:** high performance for reads. Bottleneck for writes.
  - **Reliability:** tolerance to 1 faulty disk.
  - **Capacity:** only 1 disk is not available.

PA = A0 xor A1 xor A2

If disk 2 fails:

A2 = A0 xor A1 xor PA



How is the new parity calculated after a write event?

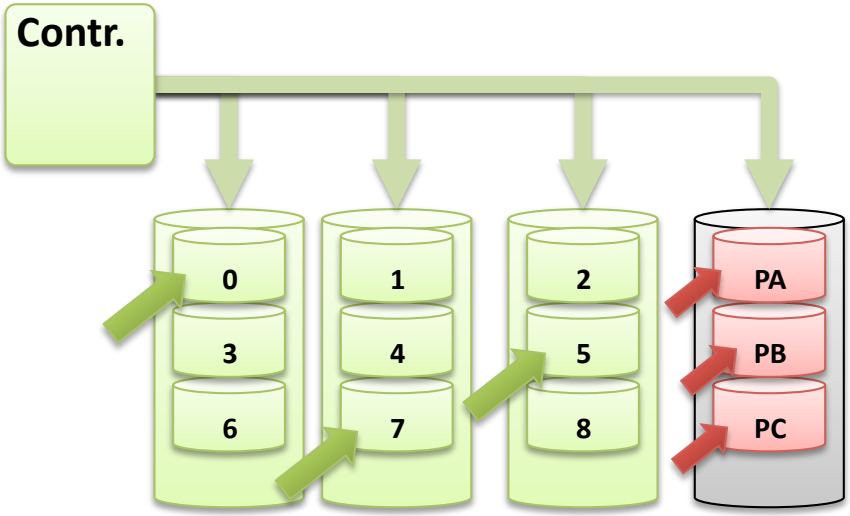
(Example: write in block B1):

Option 1: read the rest of blocks (B0, B2) and recalculate.

Option 2: read the content of B1 and PB and calculate: $PB_{new} = PB_{old}$ xor $B1_{new}$ xor $B1_{old}$

RAID (Redundant Array of Inexpensive Disks)

- Write problem in **RAID 4:**
  - Need to write in positions 0, 5, 7.



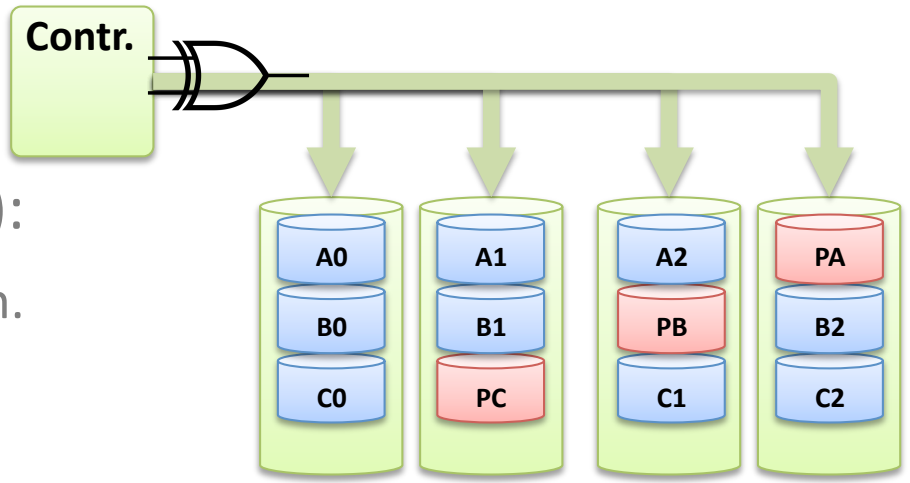
Write process:

1. Read the blocks 0, 5, 7 and PA, PB, PC.
2. Calculate the new value of PA, PB, PC.
3. Write new data blocks.
4. Write new parity blocks.

Serialized (writes in the same disk) -> Low performance.

RAID (Redundant Array of Inexpensive Disks)

- **RAID 5** (striping + distributed parity):
  - Parity information is distributed among all the disks.
  - Similarly to RAID 4, block-level division (1 strip = 1 block).
  - **Performance:** eliminates the writes bottleneck.
  - **Reliability:** tolerates 1 faulty disk.
  - **Capacity:** only 1 disk lost.



- RAID 6 (striping + double parity):
  - RAID 4 + double parity distribution.
  - Tolerates two faulty disks.
- RAID 2, RAID 3:
  - Parity control at a lower (than block) level.
  - Rarely employed.

RAID (Redundant Array of Inexpensive Disks)

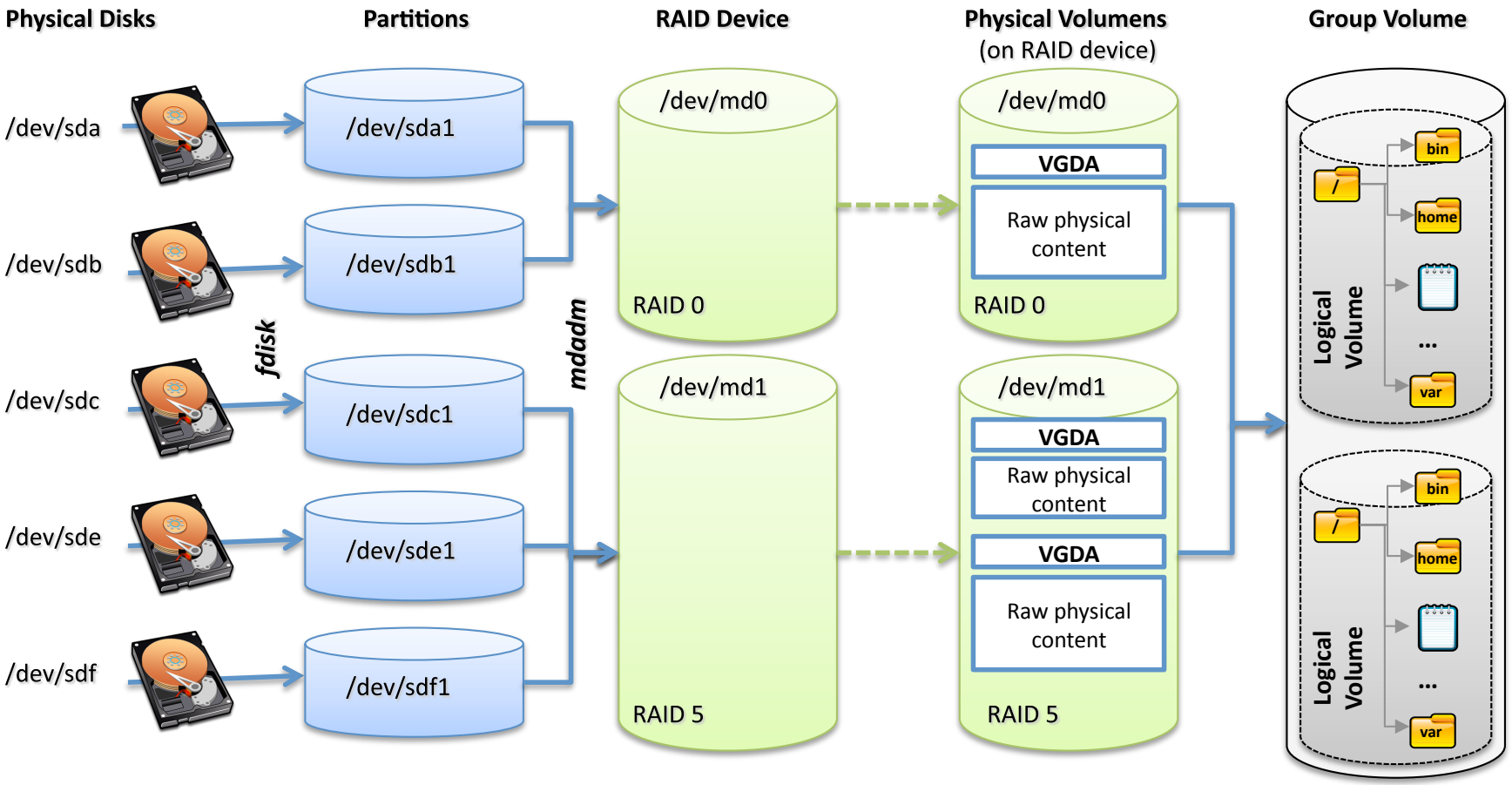
- RAID Administration, command **mdadm:**
  - **Creation** of a RAID device:
    - **# mdadm --create /dev/md0 --verbose --level=0 --raid-devices=2 /dev/sdb /dev/sdc2.**
    - First, disks must be partitioned (fdisk).
    - Creation process can be monitored: # cat /proc/mdstat.
    - Creates a RAID in /dev/md0. On it we can create a File System (or a LVM Physical Volume).
  - **Monitoring** the RAID system:
    - # cat /proc/mdstat.
    - # mdadm --monitor [options] /dev/md0.
  - **Elimination** (deactivation) of RAID:
    - "Stop" device: # mdadm --stop /dev/md0.
    - Clean previous information from a RAID disk: # mdadm --zero-superblock /dev/sdX.

RAID (Redundant Array of Inexpensive Disks)

- Procedure for a **disk failure:**
  - Assume a RAID5 system, still operative with a significant performance degradation.
  - Broken disk can be automatically **restored:**
    1. Eliminate broken disk from RAID: # mdadm /dev/md0 –r /dev/sdc1.
    2. Physically replace with another one (identical).
    3. Create the partitions as in the original: # fdisk /dev/sdc.
    4. Add it to the RAID device: # mdadm /dev/md0 –a /dev/sdc1.
    5. Monitor the reconstruction process: # cat /proc/mdstat.
  - We can simulate a disk failure:
    - # mdadm /dev/md0 –f /dev/sdc1.
    - All the process log information in /var/log/messages.

RAID (Redundant Array of Inexpensive Disks)

- Combination **RAID + LVM:**
  - RAID must be implemented below LVM.

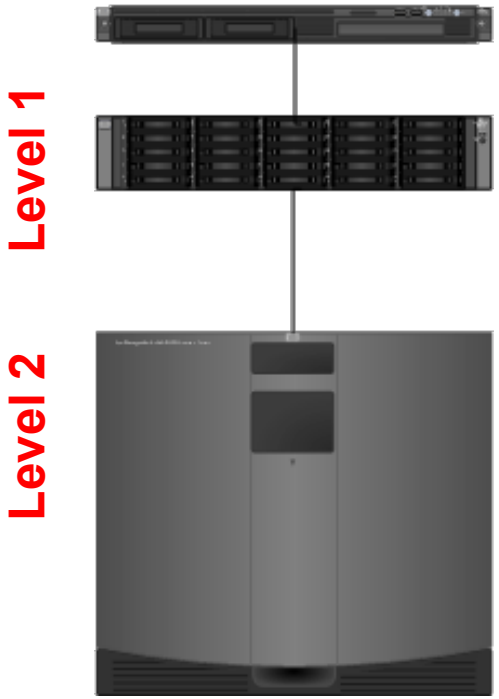


Index

- Logical Volume Manager (LVM).
- Redundant Array of Inexpensive Disks (RAID).
- **Backup.**

Backup

- RAID + journaling not enough to provide 100% availability.
- Essential: backup copies:
  - Solution for multiple unexpected events, both HW and SW.
  - Mainly “the users”.
- Performed with dedicated resources:
  - Hard Disks:
    - Exclusively dedicated to backup.
    - SAN Servers.
    - Disk hierarchy with decreasing performance.
  - Tapes (or other magnetic support):
    - LTO (Linear Tape-Open) (LTO-6 Ultrium):
      - 2.5TB capacity, 160MB/s transference.
    - Others: SAIT, AIT.



Backup

- **Backup Policy:** configured according to our requirements:
  - **What** do we need to store?:
    - Data from users/apps/system.
    - Select the critical parts of the system.
  - **When** do we want to backup?:
    - Do not overload systems with useless work.
    - Depends on the kind of utilization and the part of the file system.
    - Employ programming/automatization mechanisms (cron).
  - **Where** do we want to backup?:
    - Efficient labeling and organization of storage support (tapes).
  - **Always check that the backup has finished correctly (recuperation test).**

Backup

- Basic system tool: **dump/restore:**
  - Present in most UNIX/Linux systems.
  - Many advanced tools employ this as starting point.
  - Designed to work at File System level:
    - Can copy any kind of file (even devices).
    - Preserves permissions, property and timestamps of files.
    - “sparse” files managed correctly.
  - Backups are incremental (backup levels):
    - Only available for the whole File Systems.
    - Level 0: (FULL) copies all files from scratch.
    - Level 1: (INCREMENTAL) adds to the previous backup only modified files.
    - Level N: adds to the previous backup the files modified since the last time a “less than N” backup was performed.
    - The information about backup history is stored in /etc/dumpdates.

Backup

- Creation of backups with **dump** command:
  - Syntax: dump -<level> <options> -f [destination] [File system]:
    - Level: int from 0 (FULL) to 9.
    - Option –f: destination of backup file. Can be a device file (tape).
    - Option –u: update the file /etc/dumpdates after the backup.
    - Example: # dump -0u -f /dev/tape /.
- Recovery with **restore** command:
  - restore –C: compare the stored File system (from /).
  - restore –i: interactive operation with backup:
    - add/delete: files/dirs to the restoration list.
    - cd/ls/pwd: move through the backup FS (Files with * are in the restoration list).
    - extract: restore the files from the list.
  - restore –r: restore the whole file system:
    - # restore –r –f <backup_file> <destination>.

Backup

- Alternative tools (rudimentary):
  - Command tar (package):
    - Can understand devices without file system.
    - Can be completed with compression tools (bzip, zip).
  - Command dd:
    - # dd if=/dev/sda2 of=/dev/tape.
  - Command cp –a: optimal to replicate disk content (at file level).
- Advanced tools for distributed systems backup:
  - Data Protector (HP): many different plataforms, relatively cheap, can be integrated with HP OpenView.
  - Legato/Tivoli (IBM): expensive licensing.
  - Bacula: GNU alternative to non-free software.

rdump + rrestore + proper HW + scripting = enough