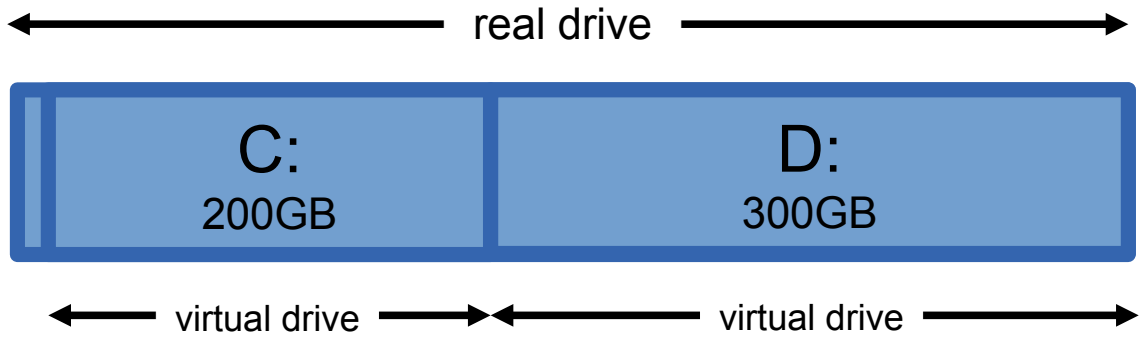
Logical Volume Management

Cloud and Virtualization Workshop

UNIVERSITY OF OREGON

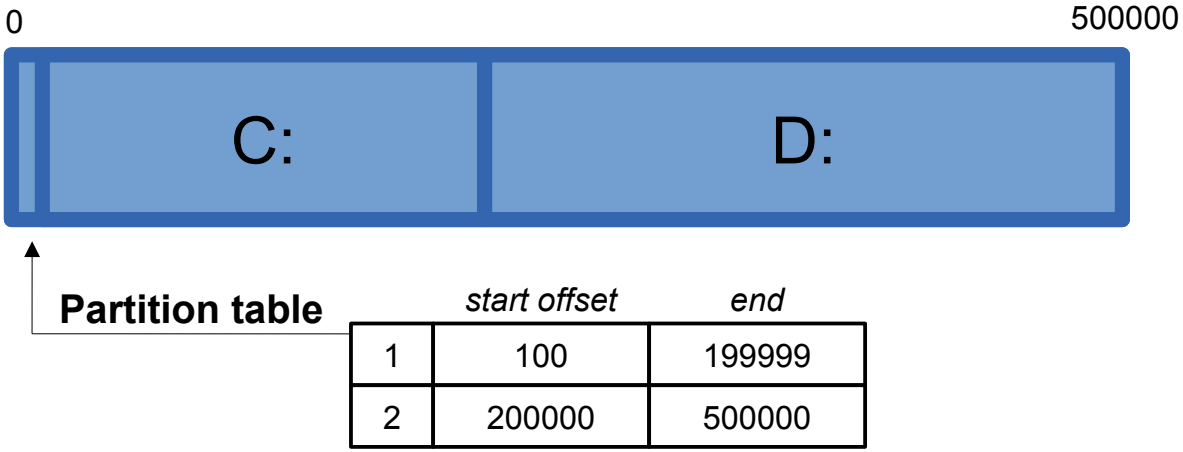
NSRC
Network Startup Resource Center

Partitioning: a familiar approach



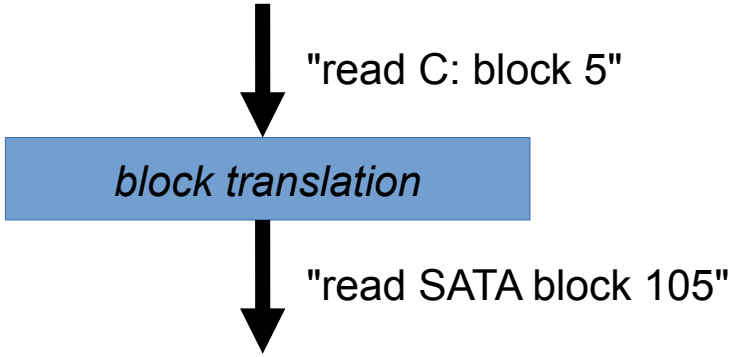
Accesses to virtual drives "C:" and "D:" are mapped to the real underlying drive

How does partitioning work?



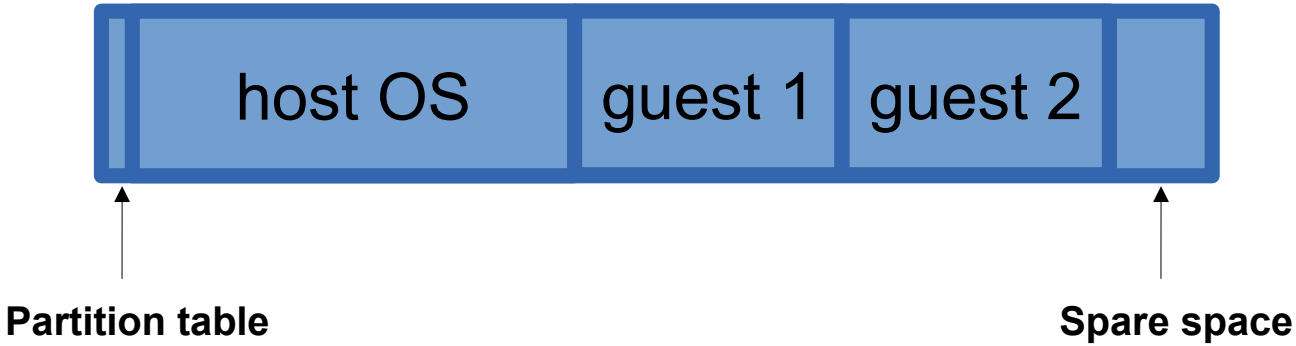
- **Partition table is an example of *metadata***
- **When the OS wants to access the Nth block, the real disk access is block (N+offset)**

Implementation: translation layer



- **Very simple and fast: just add offset**
- **Data is contiguous on disk**
- **Moving/resizing a partition can require copying all the data on the disk :-(**

Could we use partitions for VM storage?



Certainly possible, but:

- Partitions are a pain to manage / move
- Partitions cannot span across drives

Solution: a Logical Volume Manager

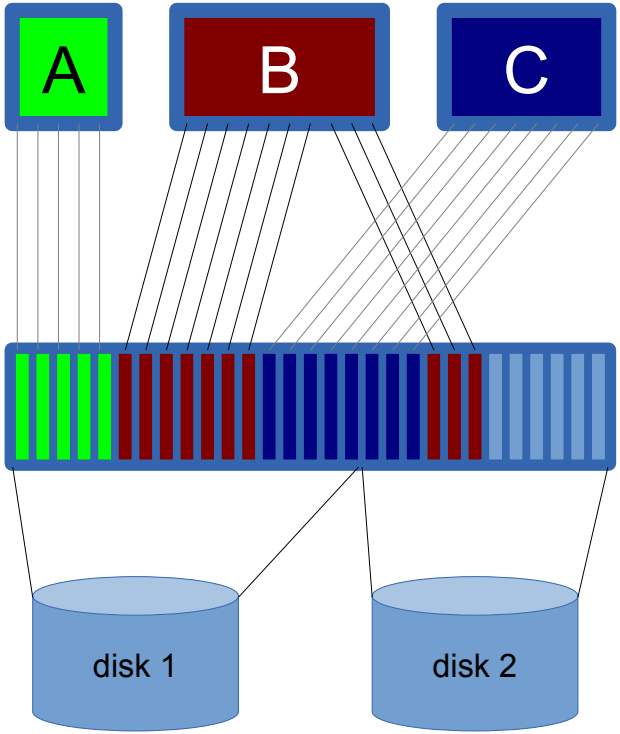
- Assemble volumes from smaller chunks of disk space
- Can grow (and shrink) volumes by adding and removing chunks, without moving other data around
- Can allocate chunks from multiple physical disks
- For Linux there are two main choices: LVM and ZFS

Aside: ZFS

- ZFS is a filesystem
- …and a volume manager (vdevs → zpools → zvols)
- …and a RAID system (mirror, raidz1/2/3, dRAID)
- …and does compression, checksumming, bitrot repair
- …and does snapshotting and incremental replication
- It's very powerful, and highly recommended
  - When used for VM block storage it can perform very well for some workloads, poorly for others. Tuning can sometimes help, e.g. *recordsize*
- For simplicity here we're going to use LVM

LVM Overview

Logical Volumes



Extent Mapping

Volume Group

Extents

Physical Volumes

About LVM

- LVM stores data on "physical volumes", which are block devices
- Physical volumes divided into "extents" – usually 4MiB
- Physical volumes combined into "volume groups"
- Logical volume is a collection of extents from one volume group
  - You can grow a logical volume by adding extents
  - When you remove a LV, its extents are freed and can be re-used
- Small metadata tables map volume offsets to extents, and help assemble the volume groups correctly

More about LVM

- LVM stores a small amount of metadata
  - small table of mappings from logical vols to extents
  - volume IDs to allow the physical volumes to be recognized and assembled into volume groups automatically
- Extent mapping is very quick
- No need to move any data when adding, removing or resizing volume groups
- Can add new physical volumes to a volume group, to grow it

Accessing logical volumes

- **Logical volumes appear as block devices**
  - `/dev/VOLGROUP/VOLUME  or`
  - `/dev/mapper/VOLGROUP-VOLUME`
- **CLI tools in the "lvm2" package, including:**
  - `pvs, pvscan   # list all physical vols`
  - `lvs, lvscan   # list all logical vols`
  - `lvdisplay     # more detail`
  - `lvcreate --size 1G --name foo myvg`
  - `lvextend --size +512M myvg/foo`
  - `lvremove myvg/foo`

LVM and snapshots

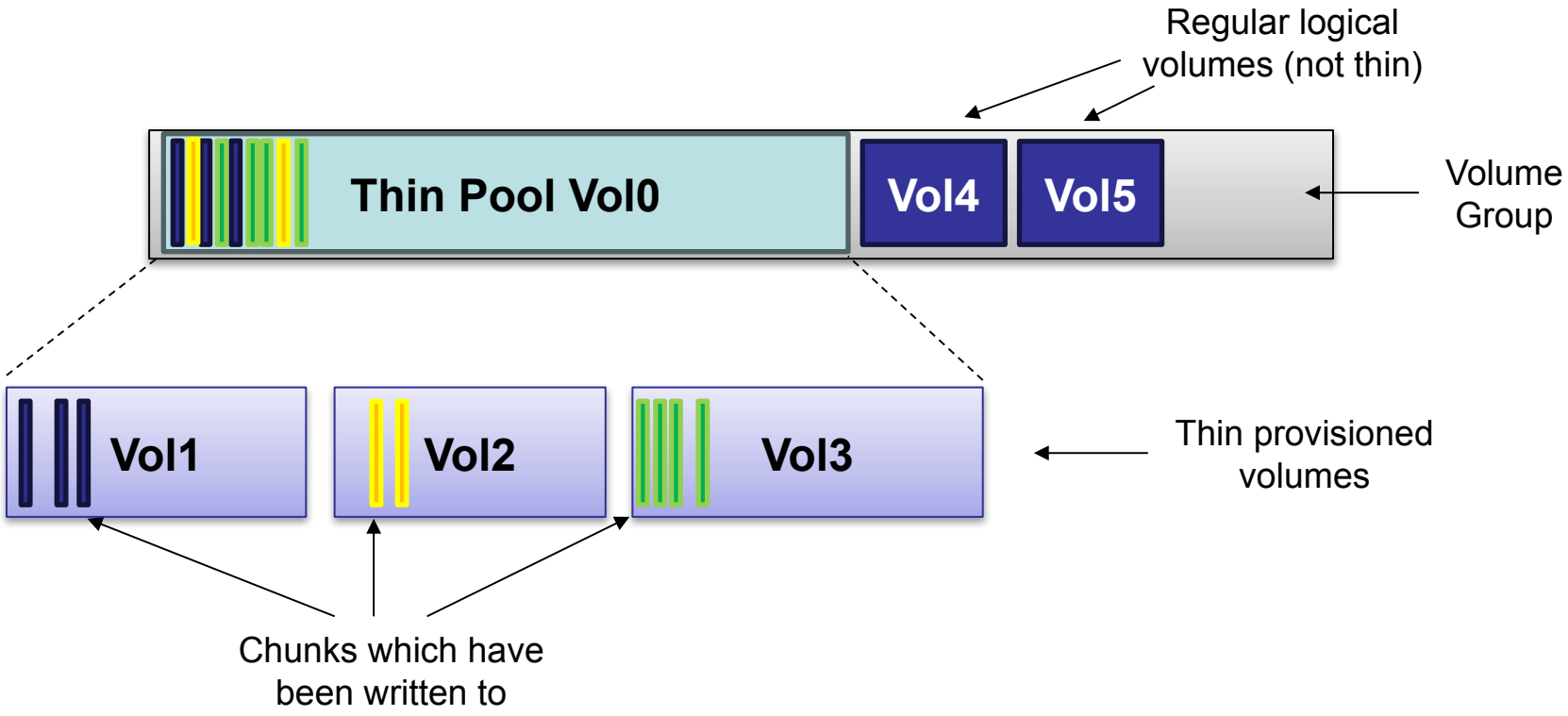
- By default, LVM allocates all the extents for a logical volume at the time you create it
  - You can't allocate more space than exists in the volume group
  - A safe, conservative approach
- LVM *can* create snapshots of a logical volume, but it's inefficient
  - If you modify any block in a snapshot, the whole 4MiB extent has to be duplicated. This is slow
  - For this reason, Proxmox* and Linstor forbid snapshots in regular LVM
- But there is another option which involves "thin provisioning"

* PVE v9 has a technology preview of LVM snapshots via "volume chains", layering LVM volumes – but it still requires thin provisioning to be efficient

LVM thin provisioning

- Space is allocated "lazily", the first time you write to it
- It's allocated in smaller, 64KiB chunks, which make snapshots much more efficient
- Space is allocated from within a "thin pool", which is a regular LVM volume
  - internally, a second small LVM volume holds metadata to track all the chunk allocations, and which snapshots they belong to (if any)

LVM thin provisioning



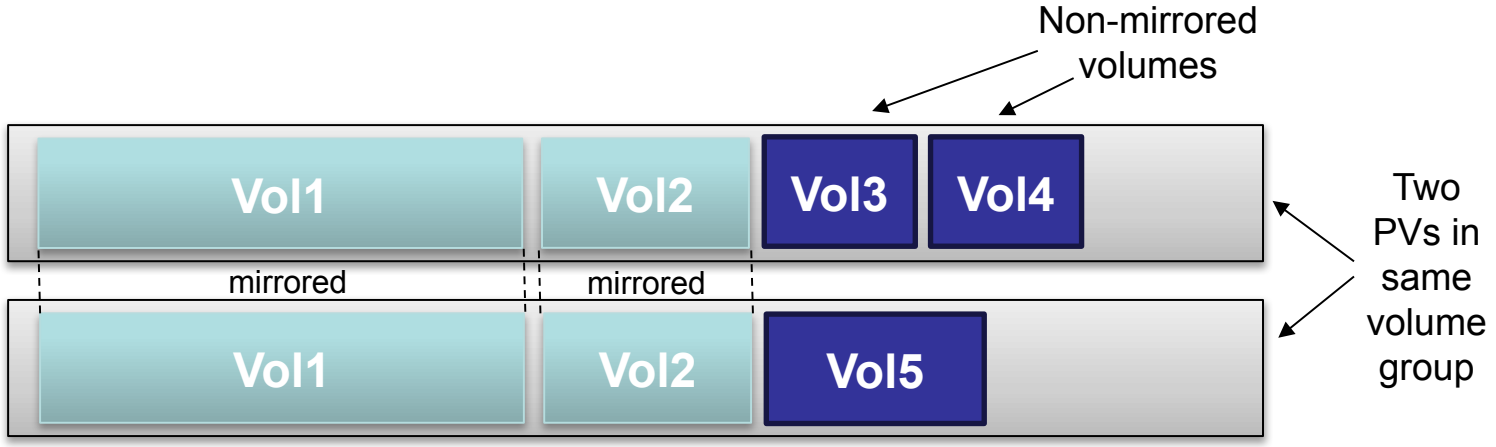
Points to note

- Vol1/2/3 are still Logical Volumes, but they belong to the thin pool
- Only the chunks which have been written to are allocated
- You are allowed to "overprovision", i.e. create volumes whose total size is bigger than the thin pool
  - Good, because it saves disk space, especially if you have many volumes which are not full
  - Bad, because of the risk of running out of space in the thin pool: writes will halt until you add space to the thin pool, or delete some data
- Growing a thin pool is easy. Shrinking it is very, very hard
- Remember, you need it for LVM snapshots in Linstor

LVM RAID (dmraid)

- LVM also has built-in features for RAID (mirroring, RAID5, RAID6…)
- You might not need it if you have RAID underneath LVM, or some other replication on top of LVM. But it's good to know it's available
- Mirroring is configured at the level of individual logical volumes
  - or an entire thin pool (since it is itself a logical volume)
- You can convert a plain volume to mirrored, and vice versa
  - except thin pools are best created as mirrored if you want them that way

LVM mirroring



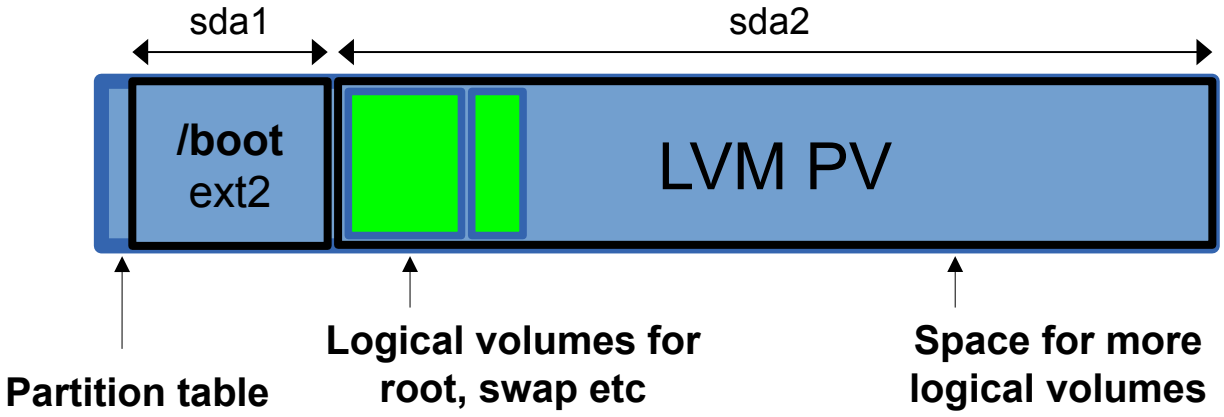
LVM Lab

Loose ends: Booting with LVM

Note on physical volumes

- An LVM "physical volume" need not be an entire disk
  - It can just be a partition
- Hence you can mix LVM and non-LVM on the same disk
- This is important if you don't have a separate boot disk; the partition table is needed for booting

Partitioning and LVM: old style

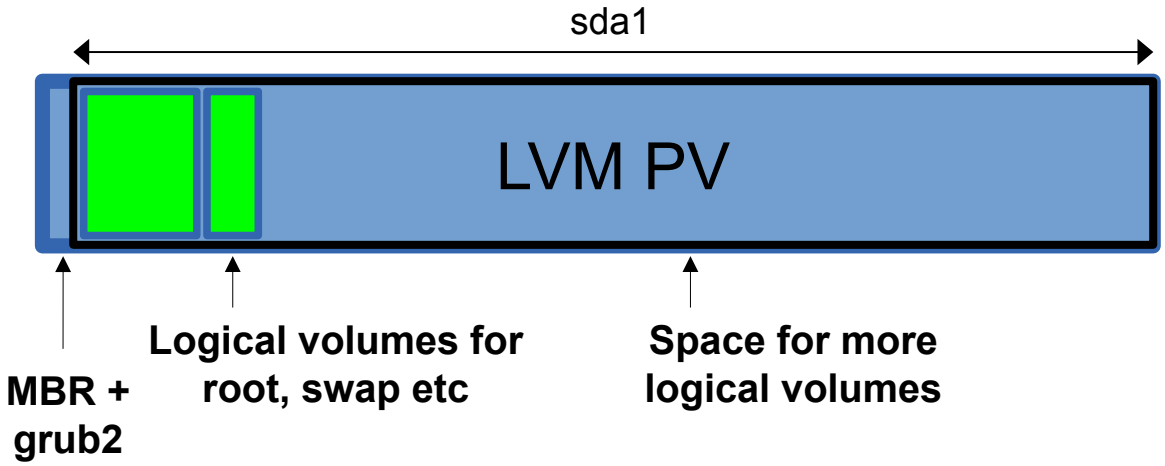


- **Partition table includes Master Boot Record**
- **sda1 (e.g. 1GB) partition for /boot filesystem (kernel)**
- **sda2 (rest of disk) is LVM physical volume**

Systems which boot using BIOS and grub2

- The kernel and initrd are stored under `/boot`
- For older versions of grub, this had to be a regular partition
  - Which limits size of /boot, and kernels are getting bigger…
- However, grub2 is able to read from inside LVM
- So newer systems don't need a /boot partition
- Still need a partition for LVM though
  - grub2 is installed in the MBR and the blocks following it
  - first partition should start at offset 2048

Partitioning and LVM (BIOS & grub2, no /boot)



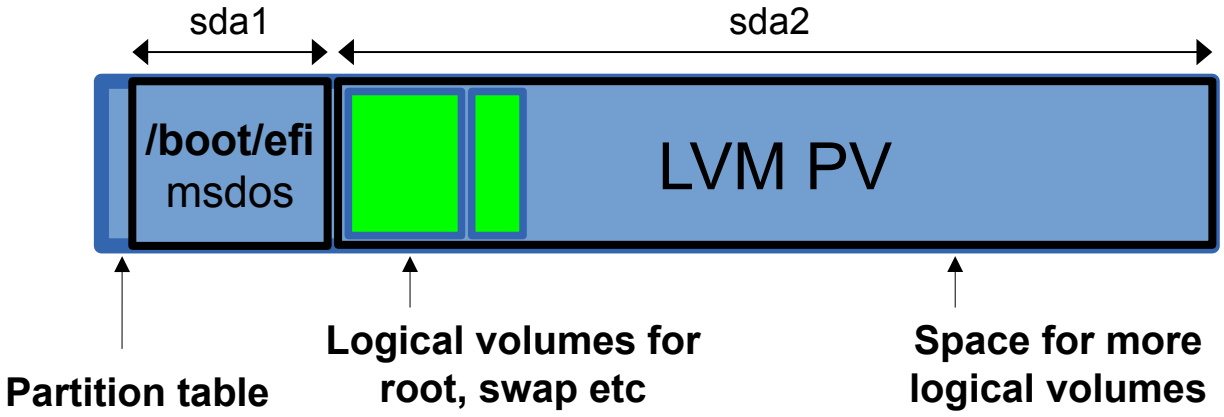
- **blocks 0-2047 for MBR and grub2**
- **sda1 is LVM's "physical volume"**

Systems which boot using UEFI

- Modern PCs need a separate partition for booting using UEFI firmware instead of BIOS
- This is MS-DOS formatted, and mounted at `/boot/efi`
- Small is fine (100-250MB) since it doesn't contain the kernel
- You may find some Linux installers still create separate partitions for `/boot` and `/boot/efi` by default

```
# lsblk
nvme0n1                     259:0    0  1.8T  0 disk
├─nvme0n1p1                 259:2    0    1G  0 part /boot/efi
├─nvme0n1p2                 259:3    0    2G  0 part /boot
└─nvme0n1p3                 259:4    0  1.8T  0 part
  └─ubuntu--vg-ubuntu--lv   253:0    0  100G  0 lvm  /
```

Partitioning and LVM (UEFI)



- **Partition table includes Master Boot Record**
- **sda1 (e.g. 100MB) partition for /boot/efi**
- **sda2 (rest of disk) is LVM physical volume**

Whole disk LVM

- You can choose to make the whole data disk be a physical volume (no partition table at all)
- If so, you need to find another way to boot
  - Separate bootable OS disk
  - Boot kernel from USB stick
  - Boot kernel over network (PXEboot)
- Simpler? You decide

Take care!

- Dealing with logical volumes is like dealing with raw partitions, with the same dangers
- Easy to write to the wrong volume device!
  - especially if LVs have auto-assigned, random-looking names
- Don't mount the same LV on the host and in a virtual machine, or in multiple VMs, or you'll corrupt the filesystem

Summary of LVM

- LVM breaks disk space into 4MiB extents
- Logical Volumes can be assembled out of any extents in a Volume Group
- A Volume Group can span multiple Physical Volumes
- Gives the speed of direct-to-disk access without the inflexibility of partitioning
- Thin pools permit overprovisioning and efficient volume snapshots, at the risk of running out of space