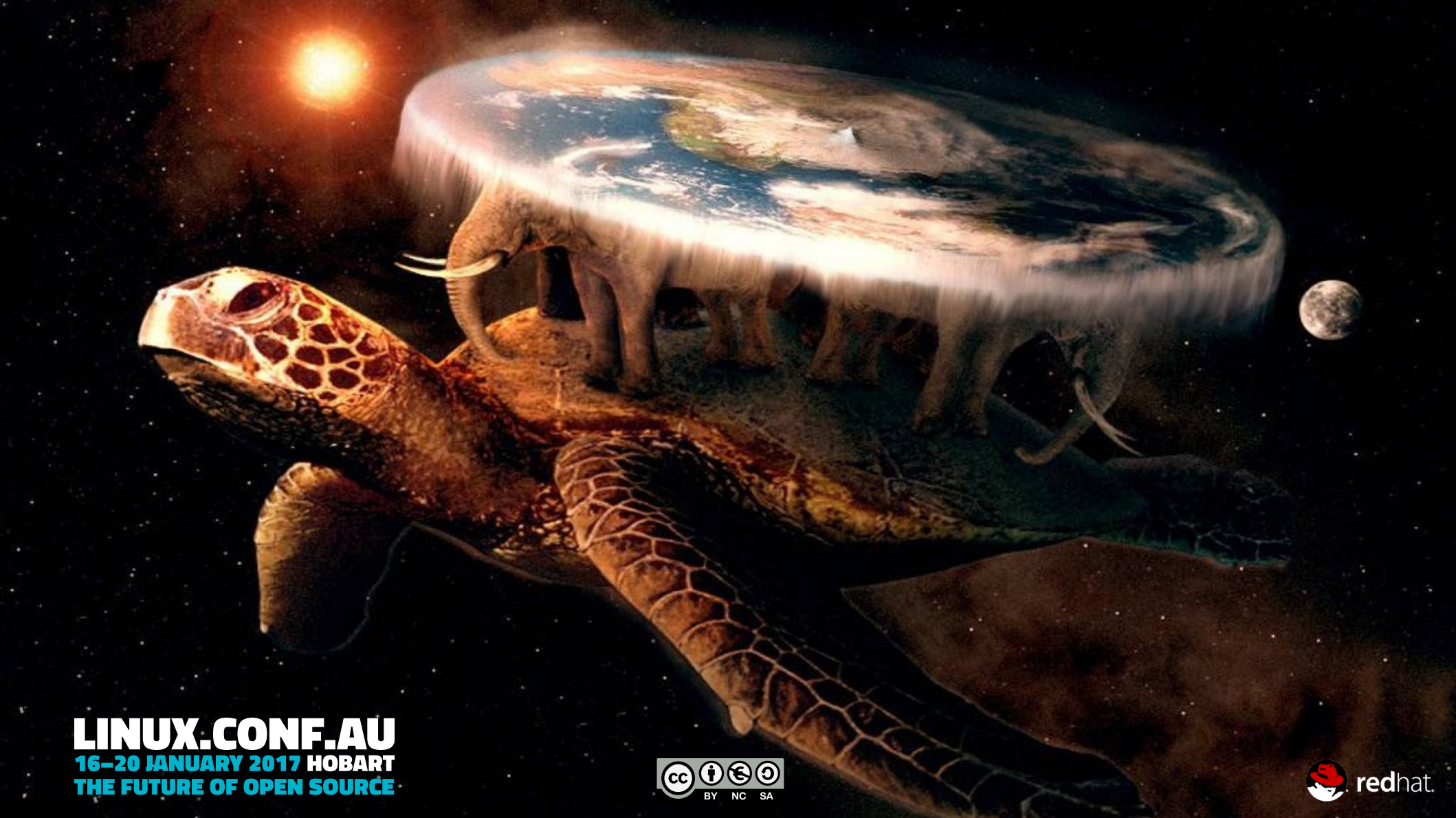

LINUX.CONF.AU
16–20 JANUARY 2017 HOBART
THE FUTURE OF OPEN SOURCE
Turtles all the Way Down
Image ©2013 CC-BY-NC-SA by Tony Gray
redhat.

Turtles all the Way Down

Misadventures with Trim and Thin

Steven Ellis
Senior Solution Architect – Red Hat NZ
sellis@redhat.com / @StevensHat

redhat.


LINUX.CONF.AU
16–20 JANUARY 2017 HOBART
THE FUTURE OF OPEN SOURCE
BY NC SA
redhat.

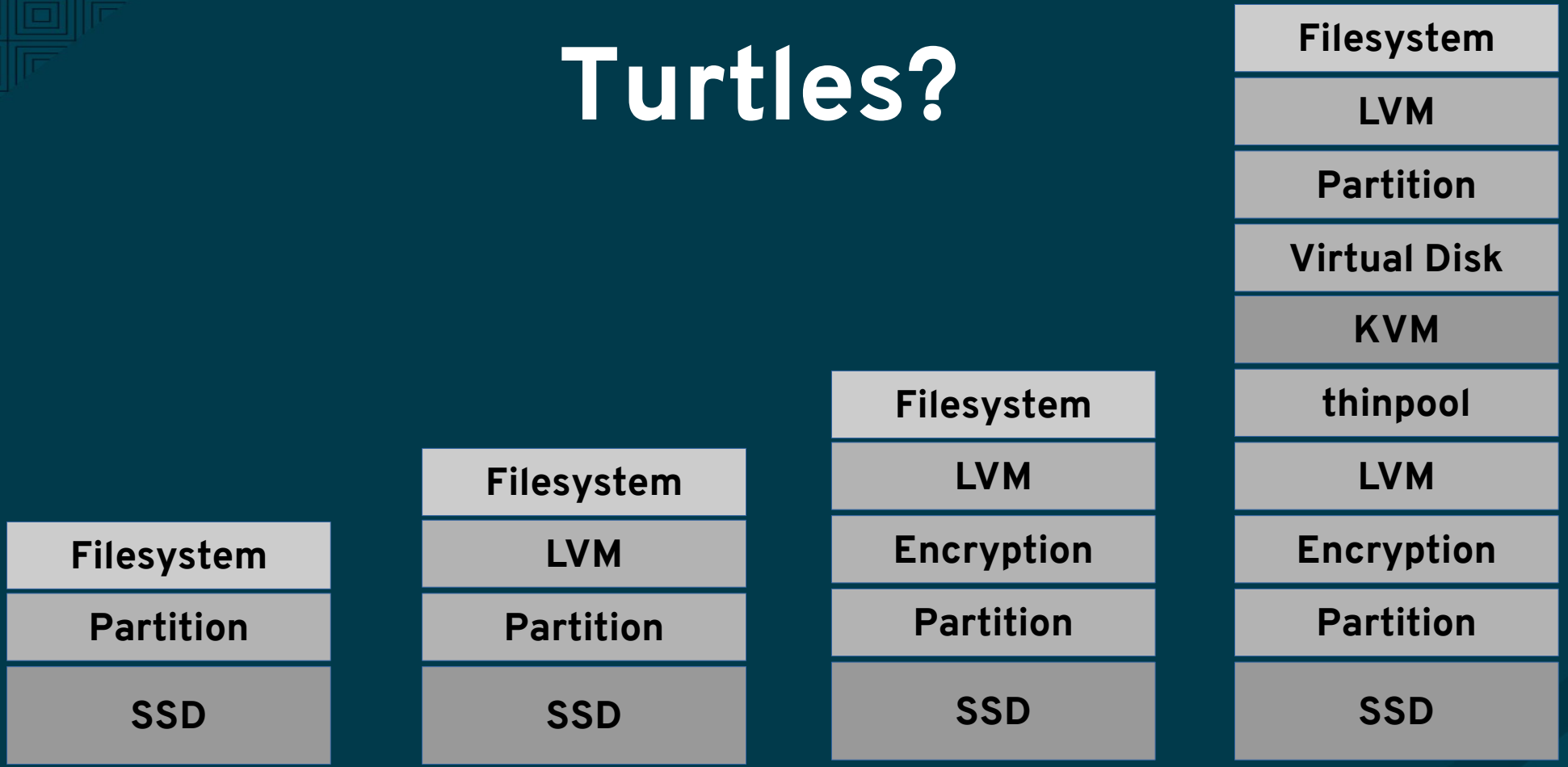
Agenda

- SSDs + Trim + Thin
- Turtle
- Turtles
- Virtual Turtles
- Migrating Turtles
- Nesting Turtles
- Cows
- Redundant Turtles
- KVM Serial Console

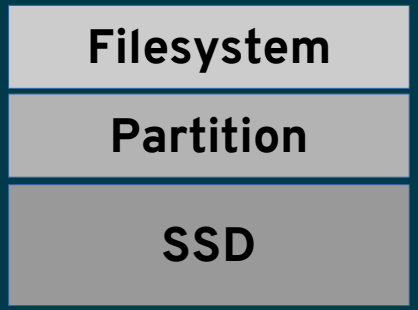
SSD + Trim / Unmap / Discard

- Critical to effective use of SSDs
- Lack of Trim can impact
  - Lifetime of disk
    - Blocks aren't released / re-allocated correctly
  - Performance
- OS + File-system need to support Trim
  - Delete requests are passed to SSD

Turtles?

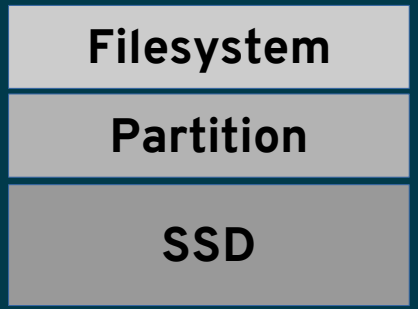


File-system Discard



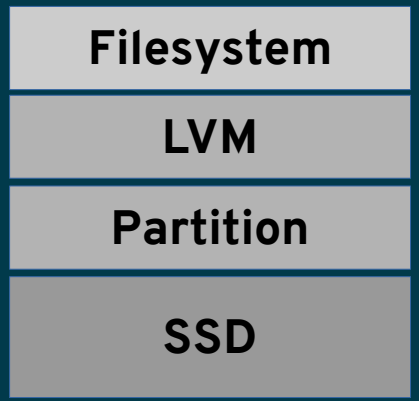
File-system Discard

- XFS / EXT4 / BRTFS / JFS
  - fstrim
  - periodic discard
    - /etc/fstab options discard
- SWAP understands discard
- F2FS for Android / IoT / Embedded
    - /etc/fstab options discard
- Confirm working
  - `sudo lsblk -o MOUNTPOINT,DISC-MAX,FSTYPE`



LVM Passthru

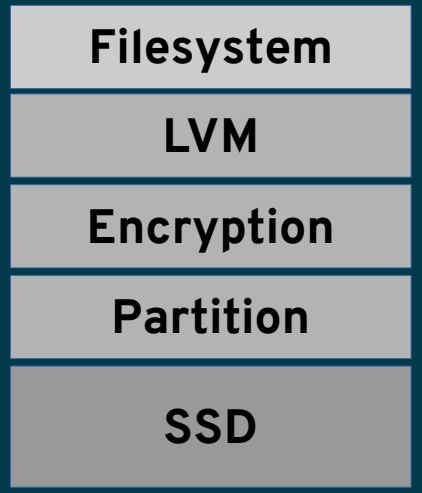
- Edit /etc/lvm/lvm.conf
  - issue_discards = 1



- Previous file-system tips apply

dmcrypt / Luks

- Edit /etc/crypttab
  - `luks-blah-blah-blah UUID=blah-blah-blah none luks,discard`
- Recreate boot disks
  - `dracut -force`
- Reference
  - `http://blog.christophersmart.com/2013/06/05/trim-on-lvm-on-luks-on-ssd`



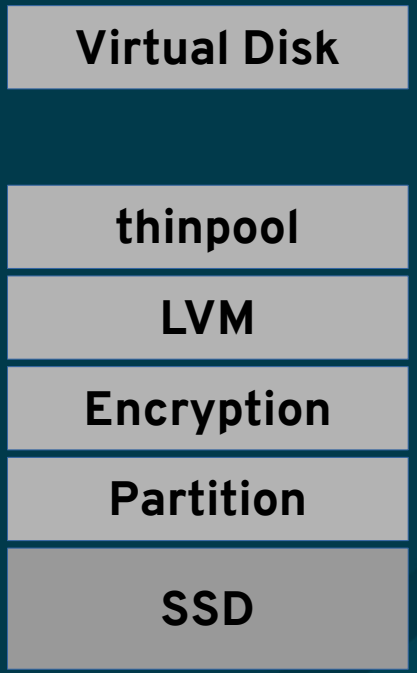
Thin LVM

Creating a thin pool on VG myVG

```
[root@t440s ~]# lvcreate -T -L 80G myVG/lv_thin
Logical volume "lv_thin" created.
```

New VM Disk Image

```
[root@t440s ~]# lvcreate -V40G -T myVG/lv_thin -n New_VM
Logical volume "New_VM" created.
```



Trim + libvirt

Check your machine type > 2.1

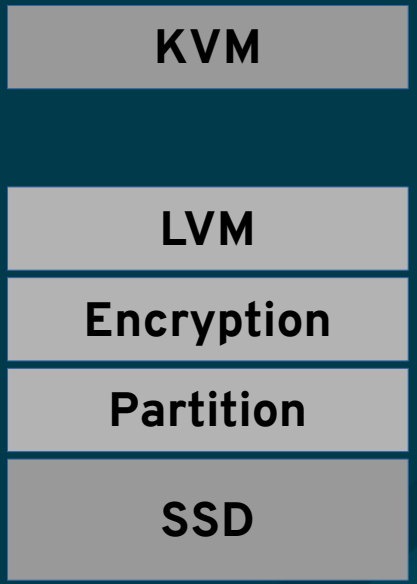
```
<type arch='x86_64' machine='pc-i440fx-2.4'>hvm</type>
```

Add discard to your hard disks

```
<driver name='qemu' type='raw' discard='unmap'/>
```

Add scsi controller model virtio-scsi

```
<controller type='scsi' index='0' model='virtio-scsi'>
```

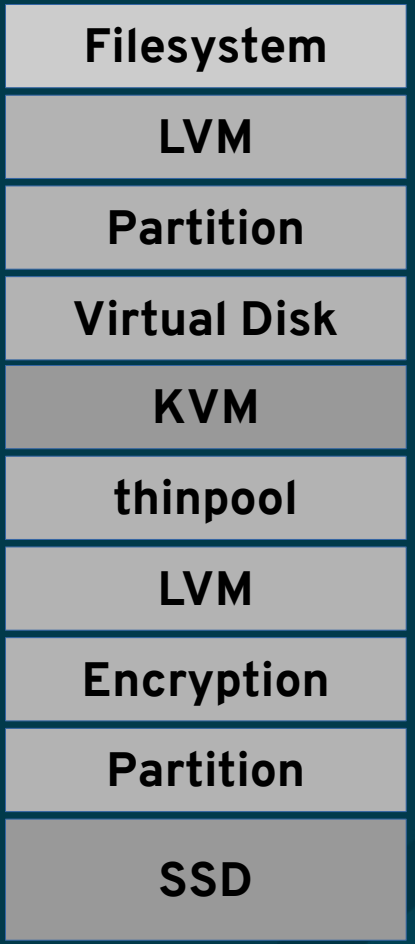


Trim in a VM

Treat like a physical environment

Same rules apply for

- fstrim vs discard
- lvm.conf
- Luks



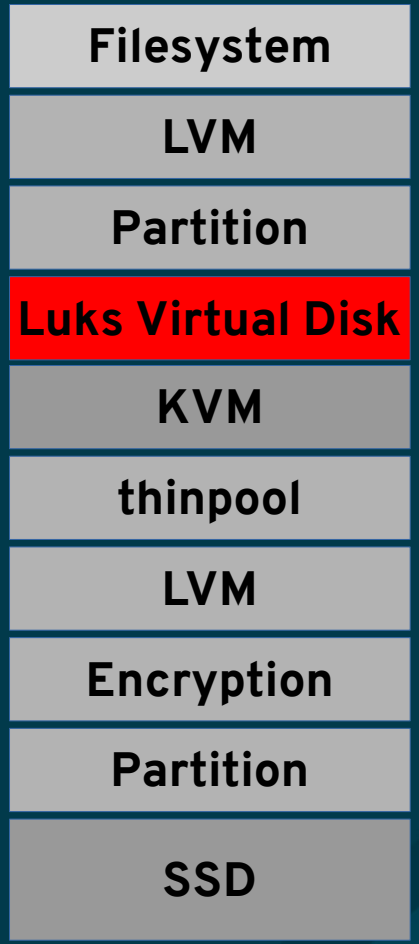
Encryption + Trim in a VM

Don’t..

Just Don’t..

Please please...

DO NOT DO IT.



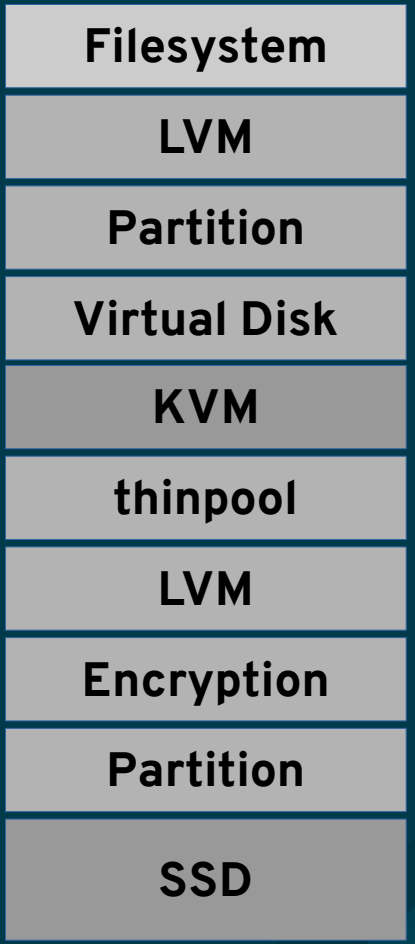
Are you Trim?

Boot and test your VM

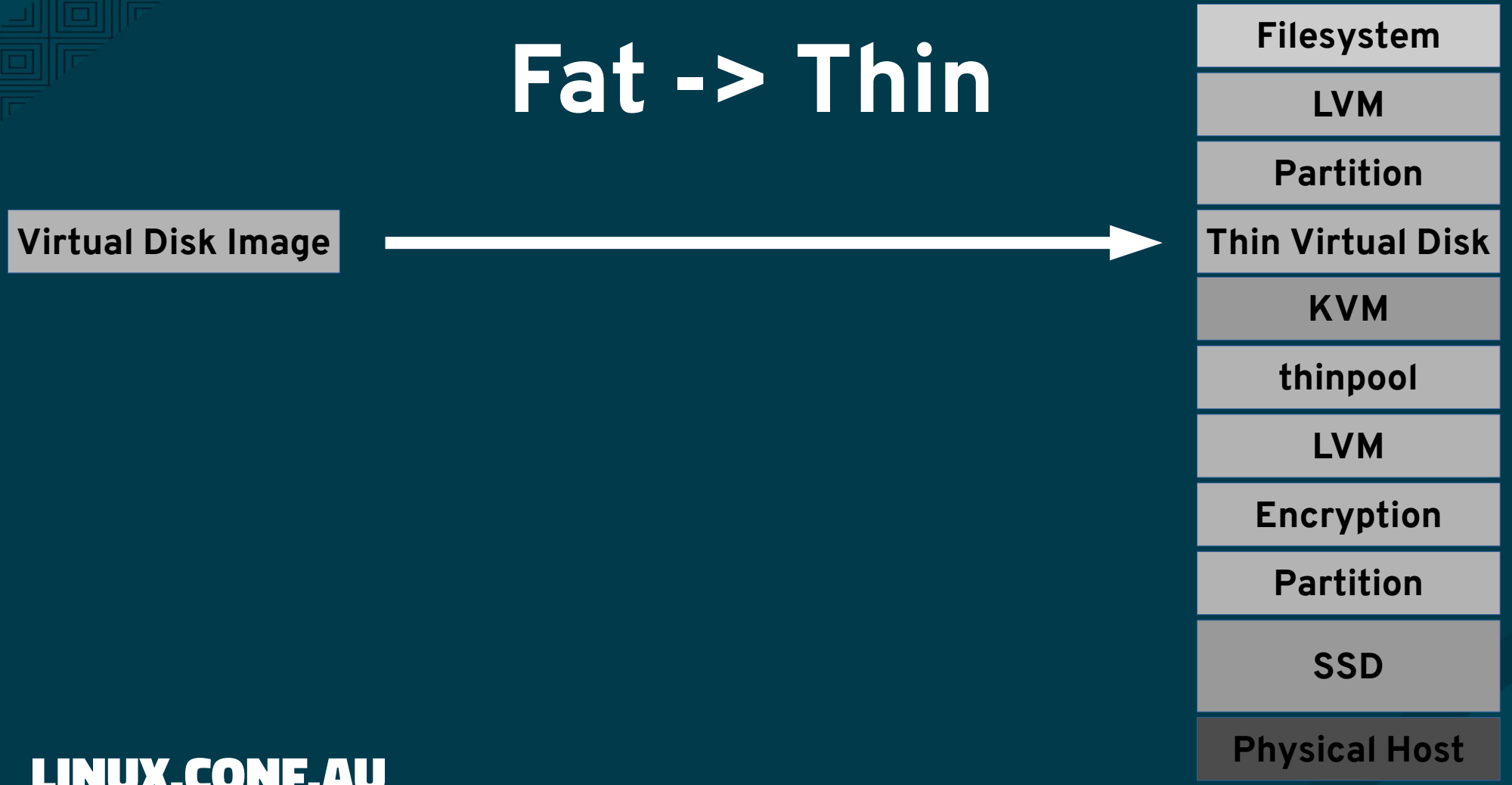
```
[root@testvm ~]# lsblk -o MOUNTPOINT,DISC-MAX,FSTYPE
MOUNTPOINT DISC-MAX FSTYPE
/boot            1G xfs
[SWAP]           1G swap
/                1G xfs
```

Run fstrim in your VM

```
[root@testvm ~]# fstrim -v /
/: 10.4 GiB (11197124608 bytes) trimmed
```



Fat -> Thin



Fat -> Thin
kpartx + fstrim

Virtual Disk Image

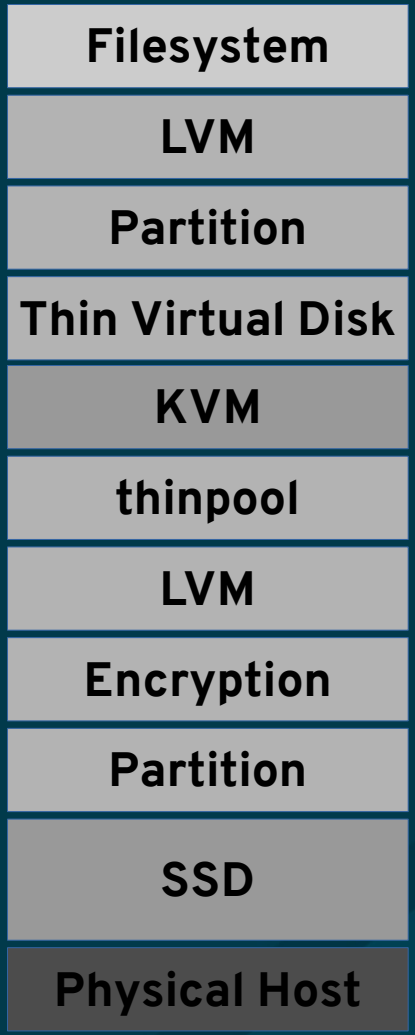
- Filesystem
- LVM
- Partition
- Thin Virtual Disk
- KVM
- thinpool
- LVM
- Encryption
- Partition
- SSD
- Physical Host

Painfully via kpartx + fstrim

```
# kpartx -a /dev/myVG/myVMDisk
# cd /mnt; mkdir volume
# mount /dev/mapper/myVG-myVMDisk_BASE1 /mnt/volume/
# fstrim -v volume
boot: 356.5 MiB (373850112 bytes) trimmed
# umount volume/
```

Fat -> Thin
kpartx + fstrim

Virtual Disk Image



Repeat for other filesystems

```
mount /dev/guestVG/root /mnt/volume/
# fstrim -v ./volume
./volume: 9.8 GiB (10481520640 bytes) trimmed
# umount volume
```

Fat -> Thin
kpartx + fstrim

Virtual Disk Image

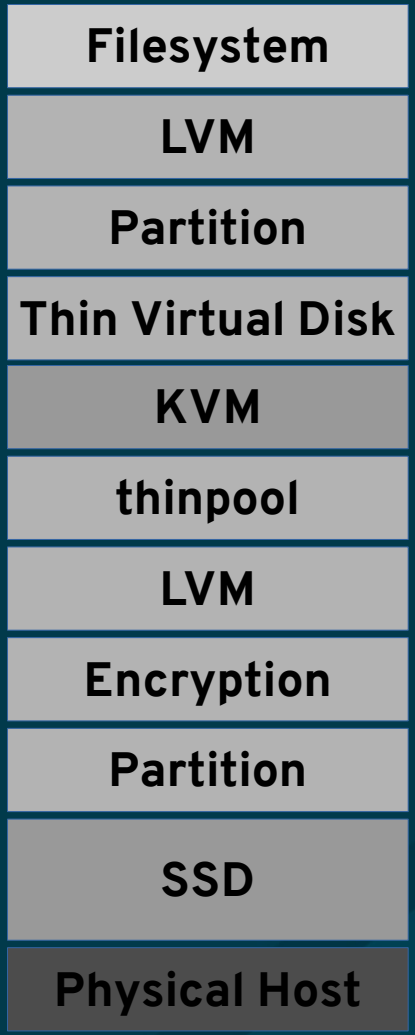
Filesystem
LVM
Partition
Thin Virtual Disk
KVM
thinpool
LVM
Encryption
Partition
SSD
Physical Host

Clean up VGs and kpartx /dev/mapper entries

```
# vgchange -a n guestVG

0 logical volume(s) in volume group "guestVG" now active

# kpartx -d /dev/mapper/myVG-myVMDisk_BASE
```

Fat -> Thin virt-sparsify

Virtual Disk Image



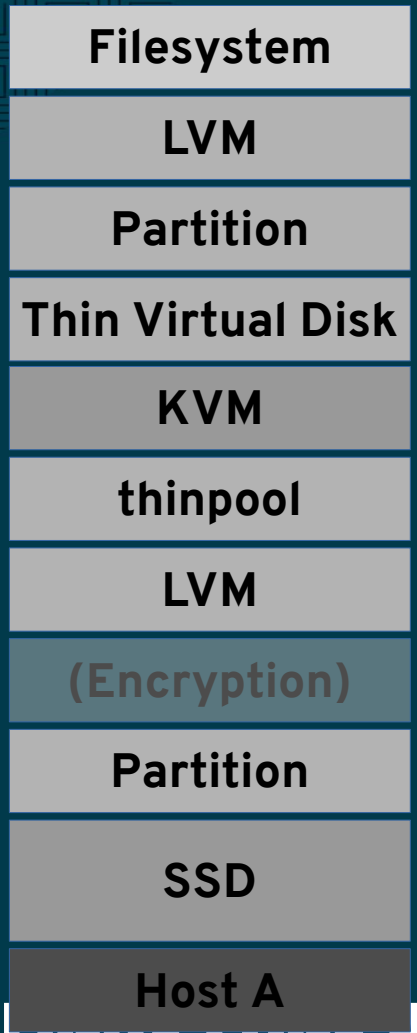
Existing FAT LVM

```
# dd if=myVMDisk.img of=/dev/myVG/myVMDisk
# virt-sparsify --in-place /dev/myVG/myVMDisk
```

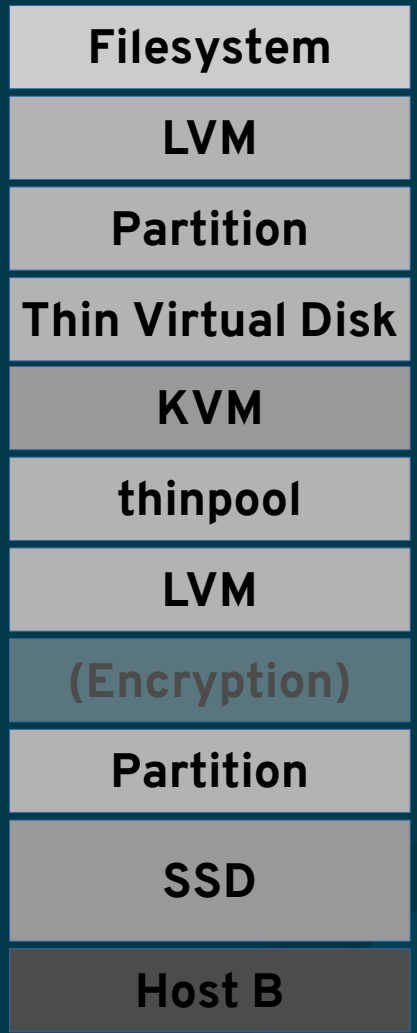
Fat image ->Thin LVM

```
# virt-sparsify myVMDisk.img /dev/myVG/myVMDisk
# lvs /dev/myVG/myVMDisk
```

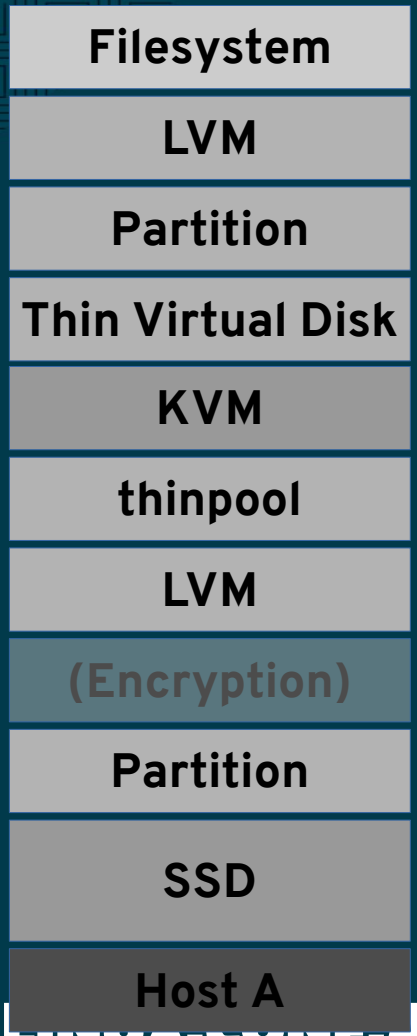
Thin -> Thin



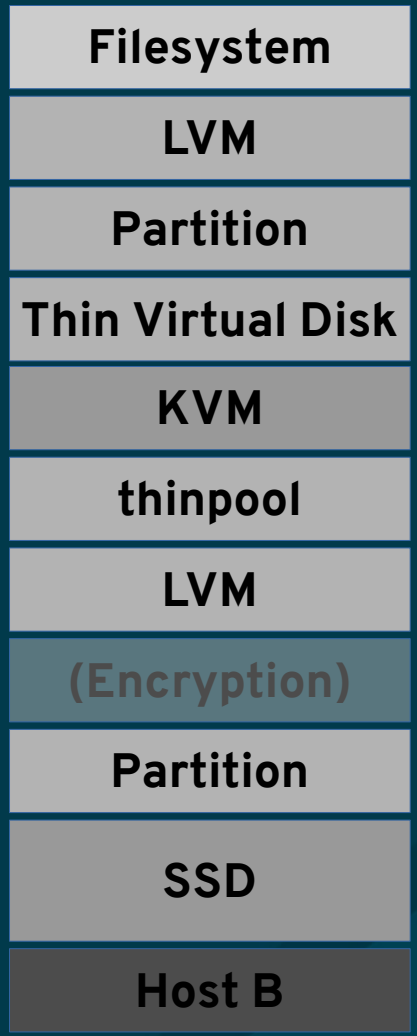
- Painful
- Sparse
- With Compression



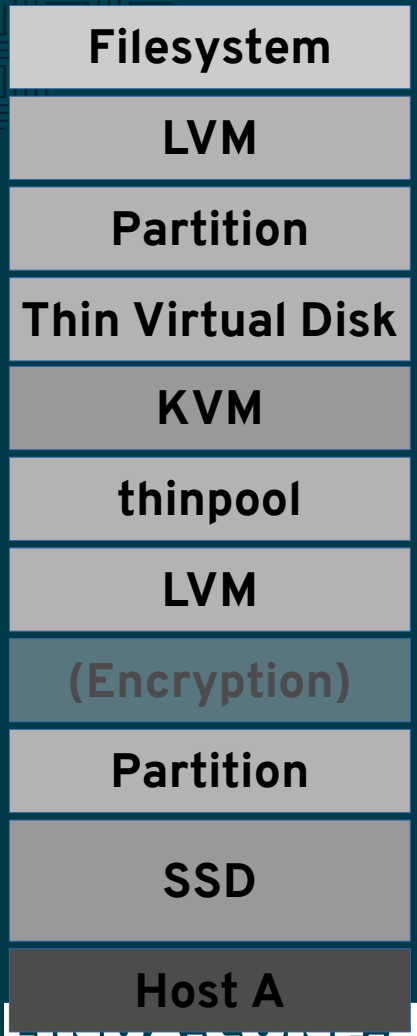
Thin -> Thin



- Painful
  - dd via ssh or nc
    - Results in a Fat LV
  - Use kpartx/fstrim or virt-sparsify at destination

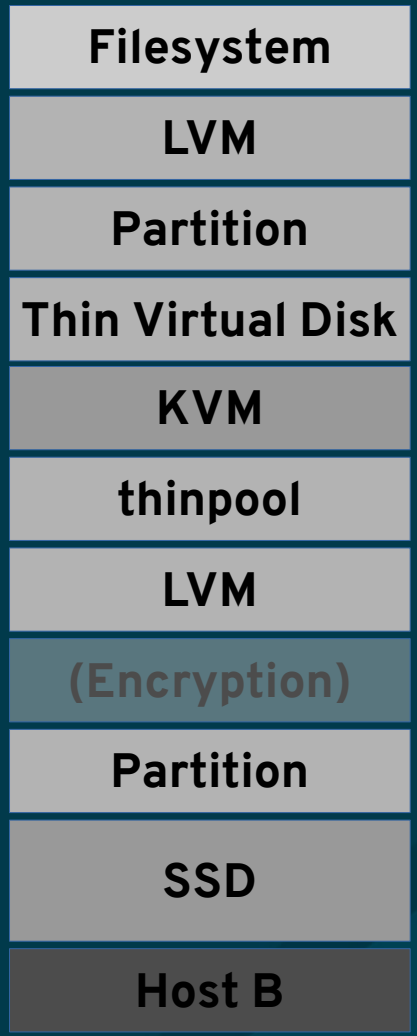


Thin -> Thin

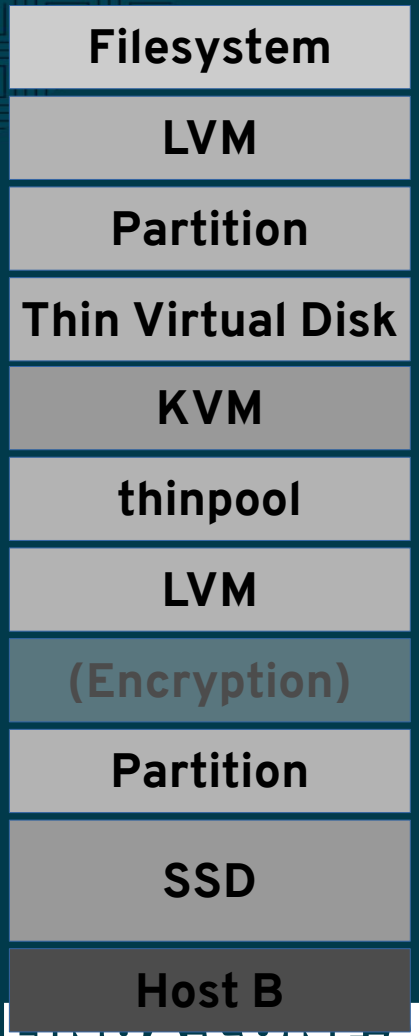


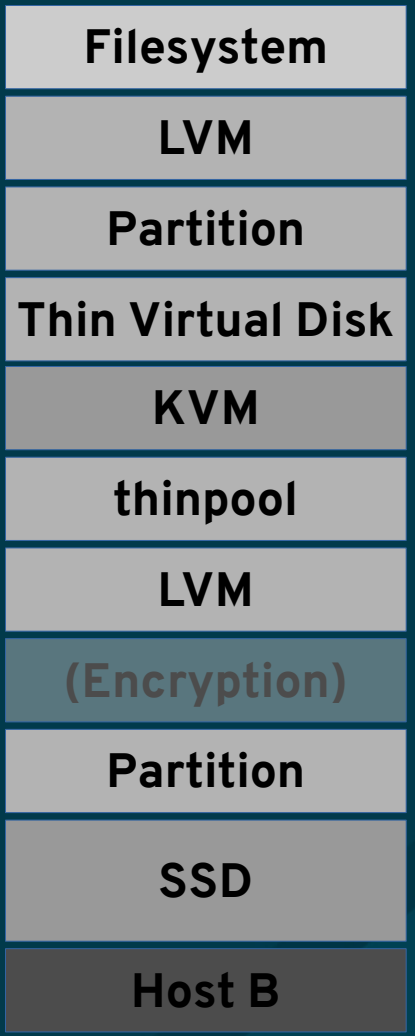
Sparse

```
[root@hostb]# nc -l 11900 | \
 dd of=/dev/HostbVG/guestVG bs=16M conv=sparse

[root@hosta]# dd if=/dev/HostaVG/guestVG \
  bs=16M | nc hostb 11900

[root@hostb]# virt-sparsify -in-place \
  /dev/HostbVG/guestVG
```



Thin -> Thin



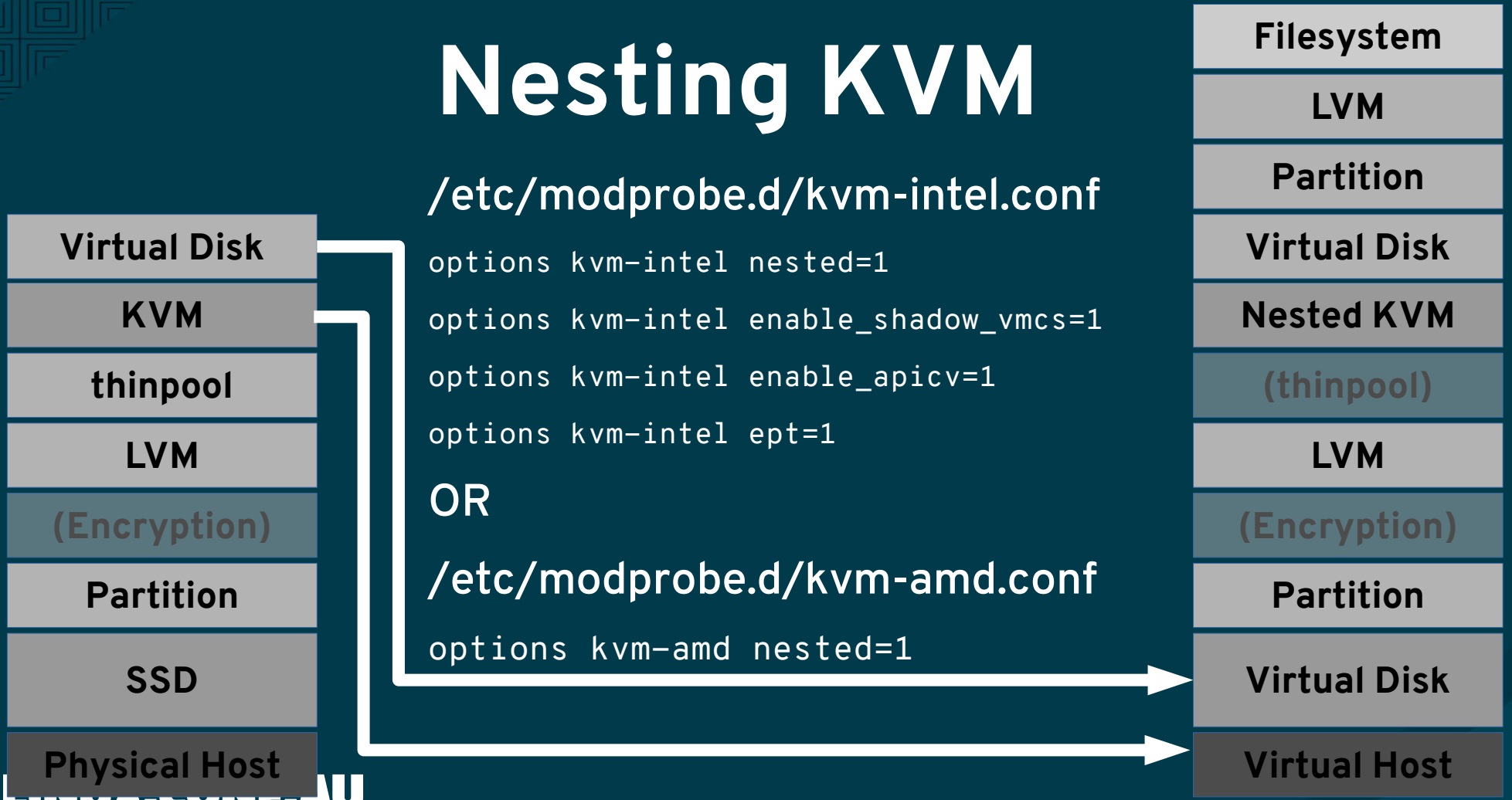


Sparse + lzop

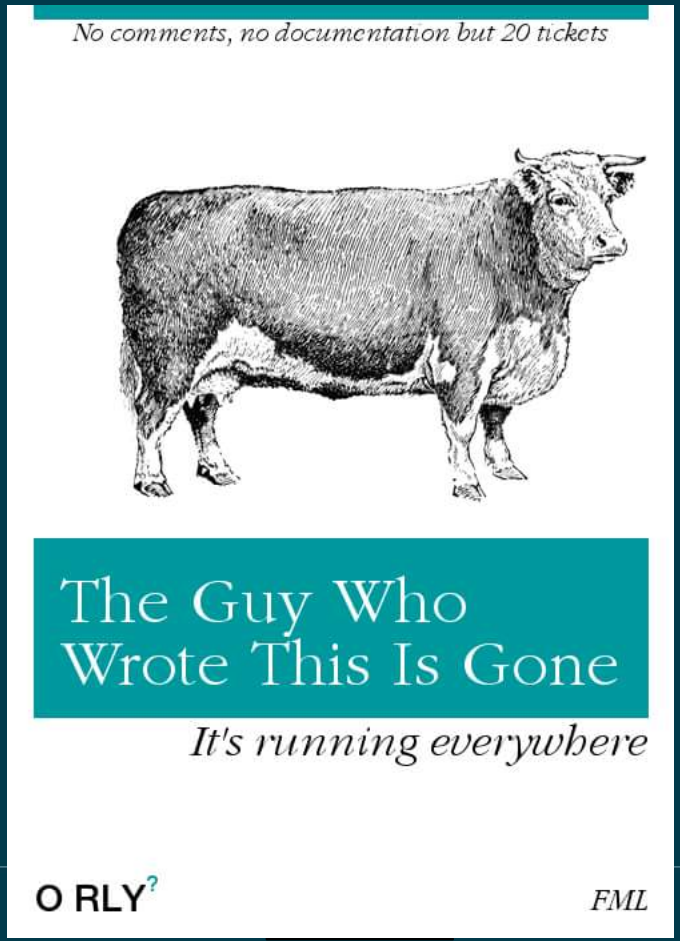
```
[root@hostb]# nc -l 11900 | lzop -d -c - \
 dd of=/dev/HostbVG/guestVG bs=16M conv=sparse

[root@hosta]# dd if=/dev/HostaVG/guestVG \
  bs=16M | lzop --fast - | nc hostb 11900

[root@hostb]# virt-sparsify -in-place \
  /dev/HostbVG/guestVG
```

Nesting Turtles?

Nesting KVM



But I like (q)Cows?



Trim a qcow2 VM

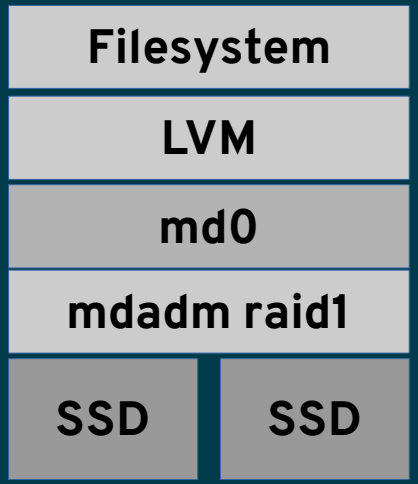
Same rules as we covered KVM and LVM

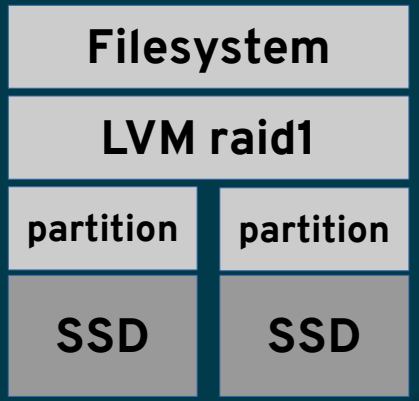
- KVM
  - Machine type / unmap / virtio-scsi
- Inside the VM
  - fstrim vs discard
  - lvm.conf
- virt-sparsify
  - `virt-sparsify disk.raw --convert qcow2 disk.qcow2`

Raid and Trim?

Two key software raid technologies

- mdadm
- lvm raid

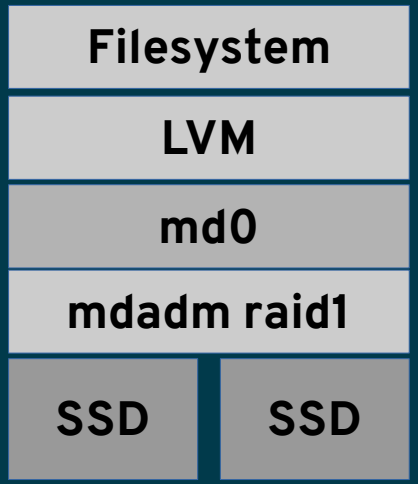




mdadm raid1

Linux software raid (mdadm)

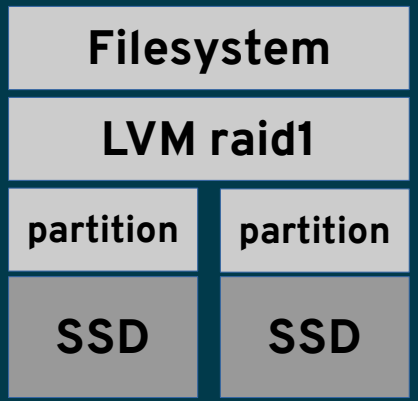
- Discard is passed-thru correctly
  - Subject to your kernel version / distro
- Initialisation may write to all blocks
  - Potentially impacting performance



LVM raid1

LVM based raid 1

- Discard is passed-thru correctly
  - Less initialisation overhead compared with mdadm



KVM Tips

1) Trim / discard with libvirt + KVM

2) Using virt-sparsify

3) Nesting with KVM

4) Enable KVM serial Console

KVM Tips

1) Trim / discard with libvirt + KVM

2) Using virt-sparsify

3) Nesting with KVM

4) Enable KVM serial Console (including Grub)

KVM serial console

- Need to modify /etc/default/grub
  - GRUB_TERMINAL="serial console”
  - GRUB_SERIAL_COMMAND="serial --speed=38400 --unit=0 --word=8 --parity=no -stop=1"
  - GRUB_CMDLINE_LINUX
    - remove “rhgb quiet”
    - Add “console=tty0 console=ttyS0,38400n8”
- Rebuild grub2 config
  - grub2-mkconfig -o /boot/grub2/grub.cfg

KVM serial console + Ansible

- Example playbook
  - https://github.com/steven-ellis/ansible-playpen
    - `grub_console.yaml`
  - Currently RHEL/Centos/Fedora centric
    - Pull requests welcomed
- Demo

redhat.

Questions?

http://people.redhat.com/sellis

Open?