Red Hat

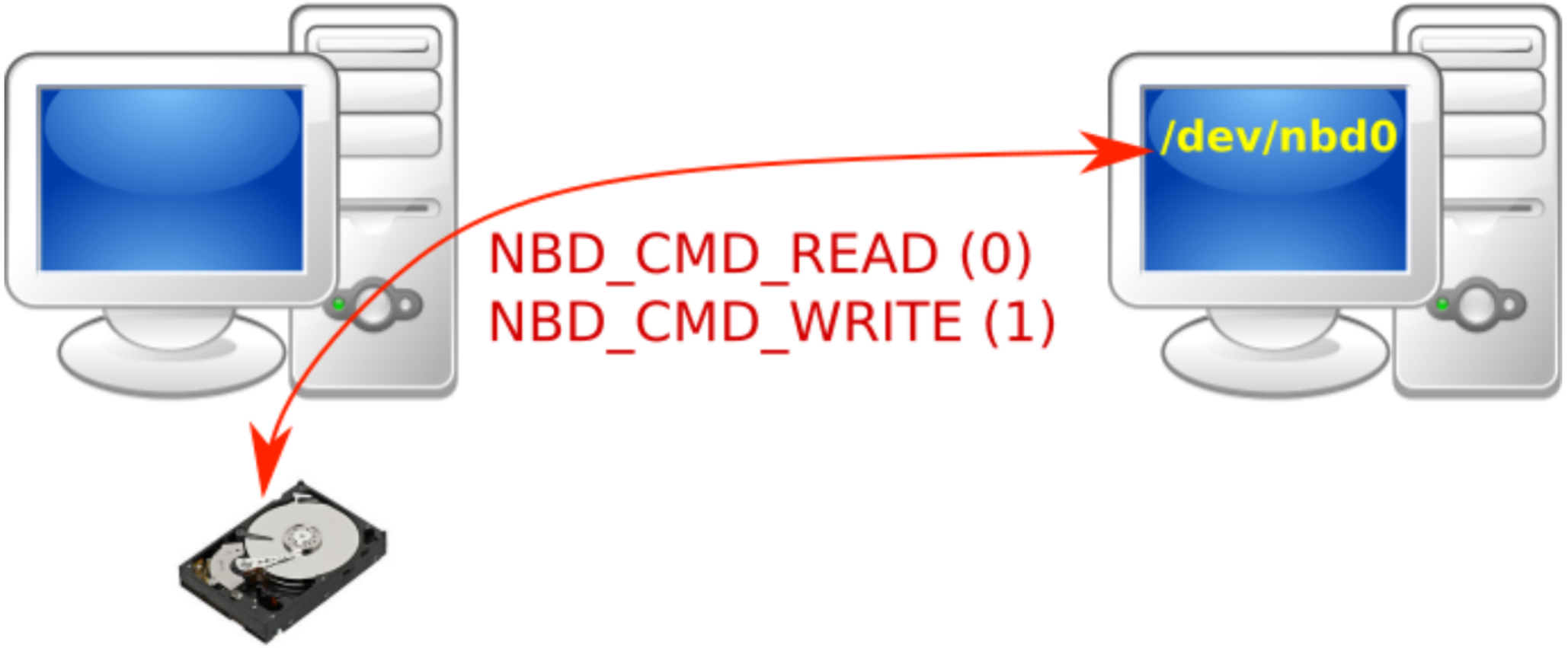
Making the Most of NBD

Eric Blake (eblake @ redhat.com)
Richard W.M. Jones (rjones @ redhat.com)
16:45 Thursday October 31st, 2019

The **Network Block Device (NBD) protocol** dates back to Linux 2.1.55 in April 1997, pre-dating iSCSI as a means for block device access of remote storage.
In recent years, the protocol has seen a revival as virtualization scenarios have used and extended its features for a variety of tasks.

Linux 2.1.55 (1997)

Red Hat



Hard drive photo by Evan-Amos https://commons.wikimedia.org/wiki/File:35-Desktop-Hard-Drive.jpg

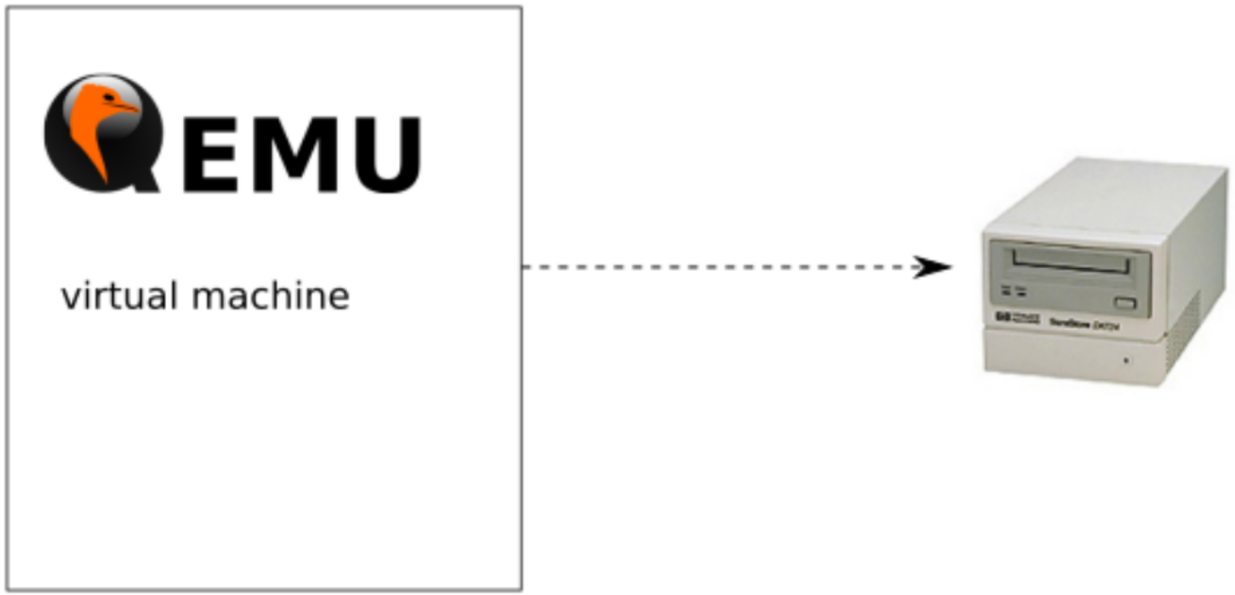
NBD in 2019 and beyond

Red Hat

NBD_CMD_READ (0)
NBD_CMD_WRITE (1)
NBD_CMD_DISC (2)
NBD_CMD_FLUSH (3)
NBD_CMD_TRIM (4)
NBD_CMD_CACHE (5)
NBD_CMD_WRITE_ZEROES (6)
NBD_CMD_BLOCK_STATUS (7)
NBD_CMD_RESIZE (8) [proposed]

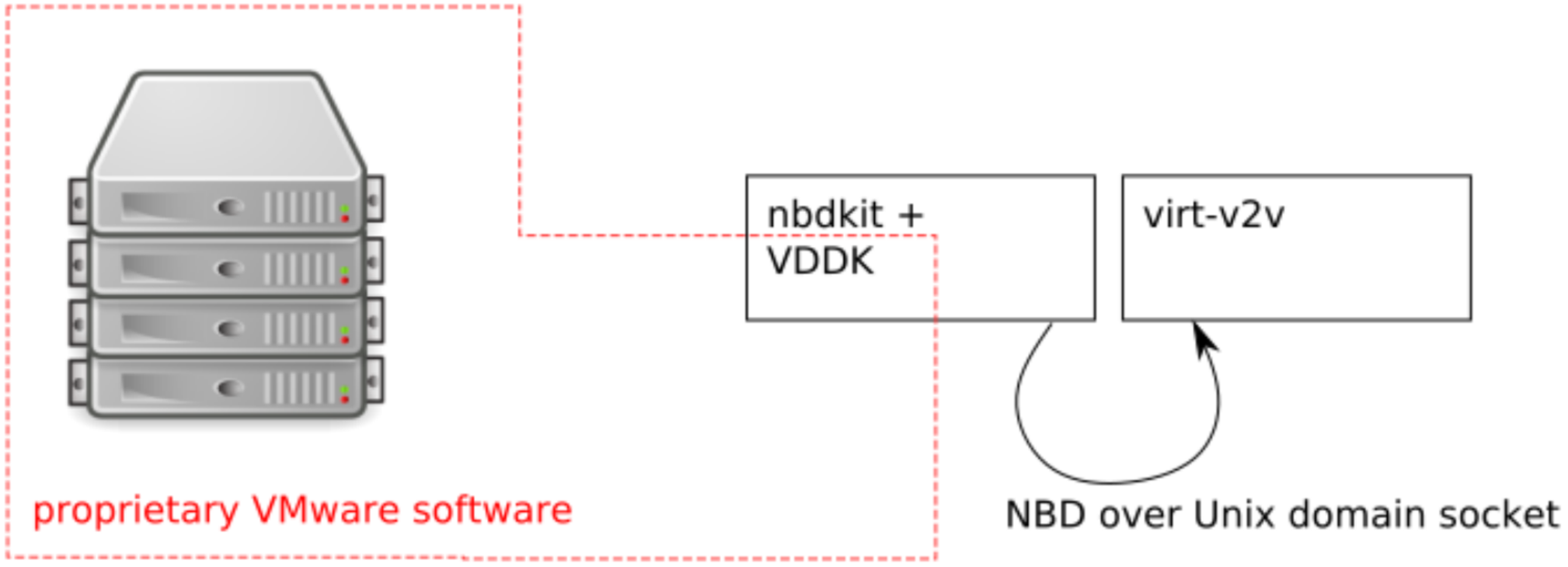
Hard drive photo by Evan-Amos https://commons.wikimedia.org/wiki/File:35-Desktop-Hard-Drive.jpg

Export a point in time snapshot from qemu




```
blockdev-backup device=A target=Target sync=none job-id=job1

nbd-server-start addr={"type":"unix","data":{"path":"./nbd-sock"}}

nbd-server-add device=Target
```

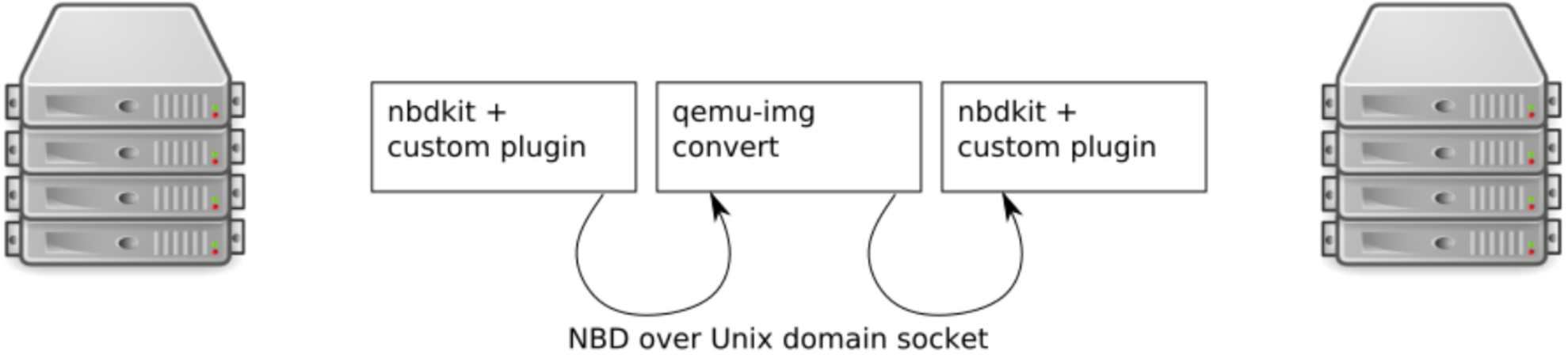
Virt-v2v reading a VMware disk




```
virt-v2v -ic vpx://... -it vddk "My Guest" ...
```

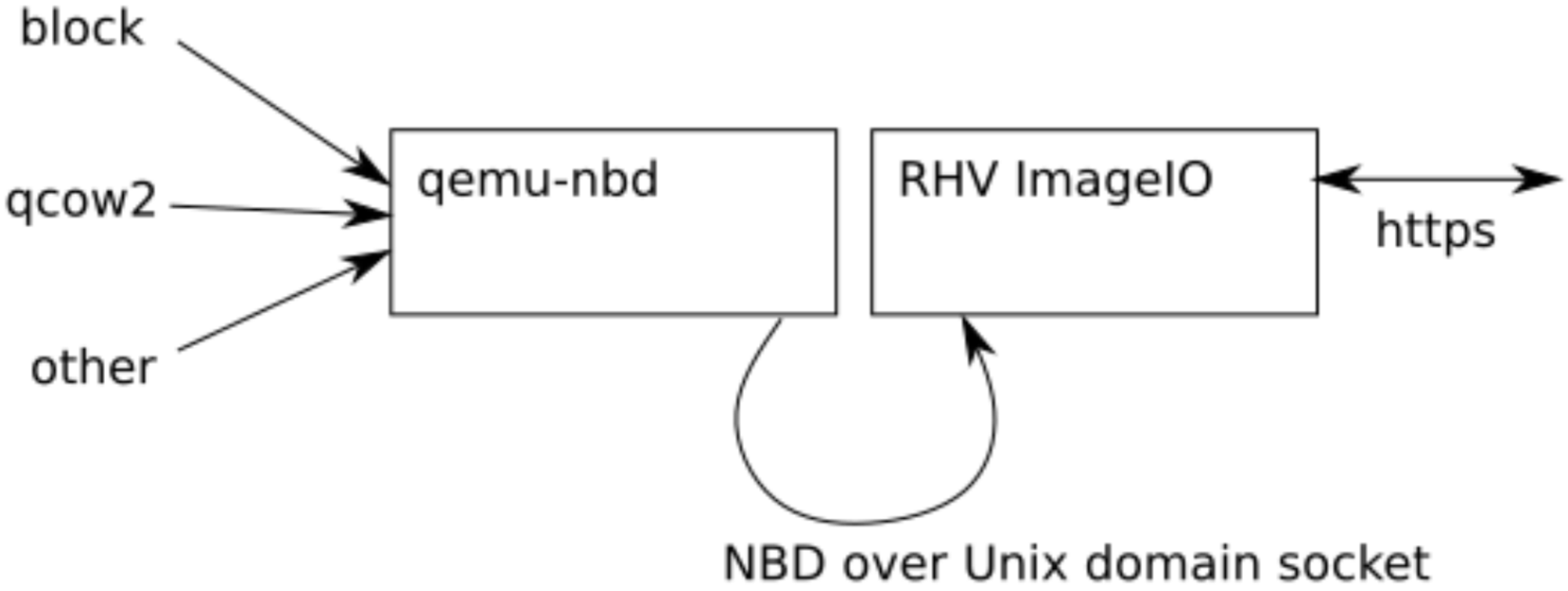
Server icon by RRZEicons https://commons.wikimedia.org/wiki/File:Server-multiple.svg

Convert between KVM-based management systems



Server icon by RRZEicons https://commons.wikimedia.org/wiki/File:Server-multiple.svg

qemu-nbd as a high-performance server for RHV ImageIO



Run RISC-V VMs from remote compressed images

Red Hat

```
nbdkit --filter=cow --filter=xz curl \
    https://dl.fedoraproject.org/pub/alt/risc-v/
    disk-images/fedora/rawhide/20190703.n.0/
    Developer/Fedora-Developer-Rawhide-20190703.n.0-sda.raw.xz

qemu-system-riscv64 \
    -nographic \
    -machine virt \
    -smp 8 \
    -m 2G \
    -drive file.driver=nbd,file.host=localhost,
           file.port=10809
```

nbdkit sh plugin for testing

```
#!/bin/bash

case "$1" in
    get_size) echo 64M ;;
    pread)
        if [ $4 -le 100000 ] &&
           [ $(( $4+$3 )) -gt 100000 ]; then
            echo EIO Bad block >&2
            exit 1
        else
            dd if=/dev/zero count=$3 iflag=count_bytes
        fi ;;
    *) exit 2 ;;
esac
```

On Youtube: *"Better loop mounts with NBD – Take your loop mounts to the next level with nbdkit"*

What is NBD today?

Red Hat

You can think of NBD as a universal protocol that you can use whenever you need to access a virtual machine disk, disk image, or block device between two processes, whether or not the processes are running on the same machine.

Based on https://www.redhat.com/archives/libguestfs/2019-August/msg00322.html

Case Study: copying sparse images

```
$ qemu-img create -f qcow2 src.qcow2 100m
Formatting 'src.qcow2', fmt=qcow2 size=104857600 cluster_size=65536 lazy_refcounts=off refcount_bits=16

$ for i in $(seq 0 2 99); do qemu-io -f qcow2 -c "w ${i}m 1m" src.qcow2; done >/dev/null

$ cat ./convert
#!/bin/bash
nbdkit -U - --filter=log --filter=nozero --filter=blocksize \
  --filter=delay --filter=stats --filter=noextents memory 100m \
  logfile=>(sed -n '/Zero.*\.\./ s/.*\(fast=.\).*/\1/p' |sort|uniq -c) \
  statsfile=/dev/stderr delay-write=20ms delay-zero=5ms maxdata=256k \
  --run 'qemu-img convert -n -f qcow2 -O raw src.qcow2 $nbd' "$@"
st=$?; sleep .2; exit $st

$ ./convert zeromode=none
elapsed time: 8.49035 s
write: 400 ops, 104857600 bytes, 9.88017e+07 bits/s

$
```

Writing zeroes: much ado about nothing

```
$ ./convert zeromode=plugin fastzeromode=none
elapsed time: 4.51148 s
write: 200 ops, 52428800 bytes, 9.29696e+07 bits/s
zero: 50 ops, 52428800 bytes, 9.29696e+07 bits/s
     50 fast=0

$ ./convert zeromode=emulate fastzeromode=none
elapsed time: 8.39201 s
write: 400 ops, 104857600 bytes, 9.99595e+07 bits/s
     50 fast=0

$
```

Block status: does it change the status quo?

Red Hat

Checking whether a block is already zero will eliminate the need to write zeroes to that block, but at what cost?

block status reports hole | block status reports data | write zeroes

	O(1) server	O(n) server
no block status		
block status sees zero		
block status sees data		
block status is slow		

If checking once per block is bad, can we instead check once per image?

Or why even check - what if we ensure the image starts with all zeroes?

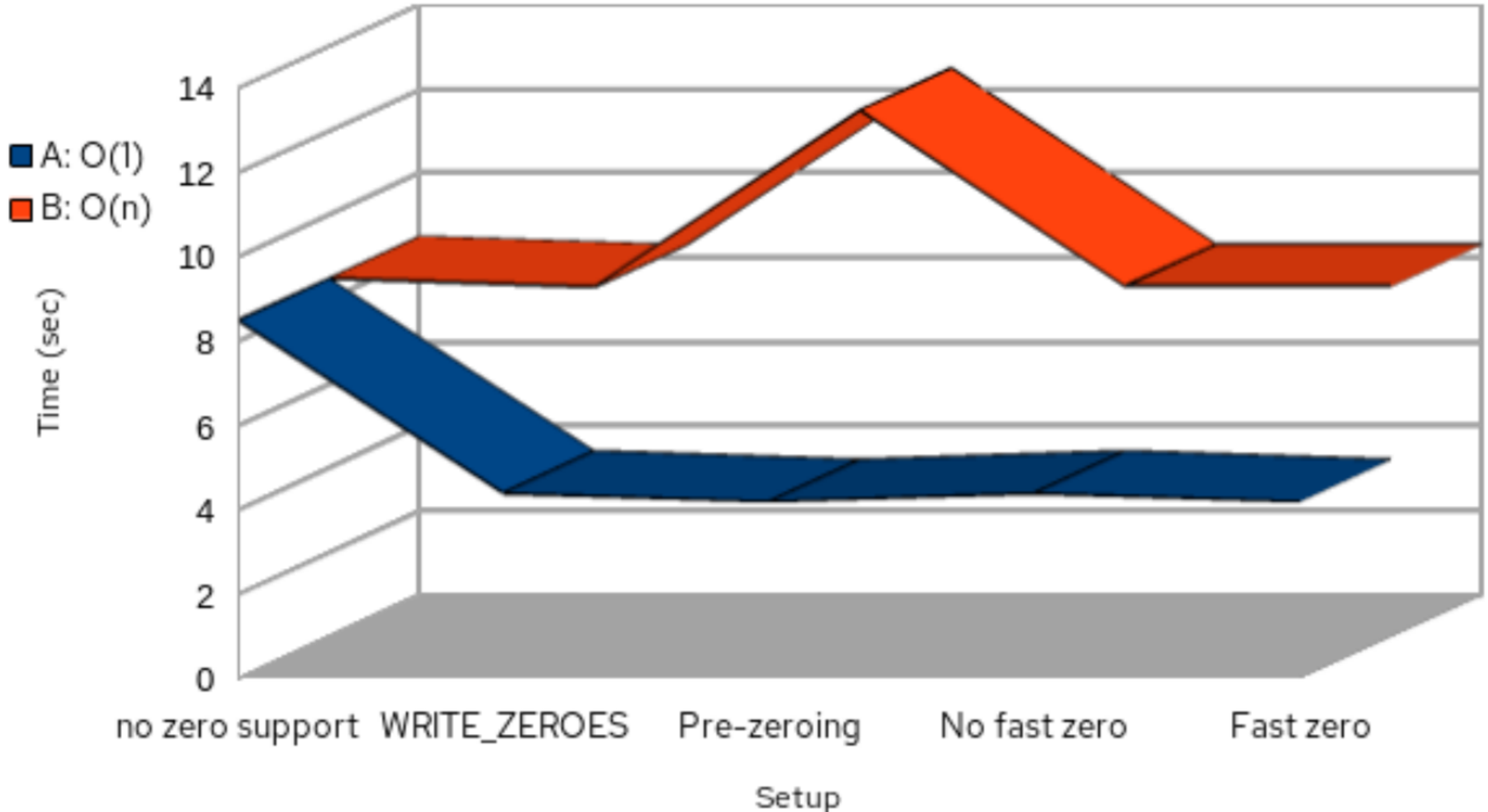
Pre-zeroing: a tale of two servers

```
$ ./convert zeromode=plugin fastzeromode=ignore
elapsed time: 4.25498 s
write: 200 ops, 52428800 bytes, 9.85741e+07 bits/s
zero: 4 ops, 104857600 bytes, 1.97148e+08 bits/s
      4 fast=1

$ ./convert zeromode=emulate fastzeromode=ignore
elapsed time: 12.4766 s
write: 600 ops, 157286400 bytes, 1.00852e+08 bits/s
      4 fast=1

$
```

The fix: raise the victory flag

QEMU 4.0 added `BDRV_REQ_NO_FALLBACK` to detect if zeroing is fast, and automatically utilizes it during 'qemu-img convert':

- No support for flag: avoid pre-zeroing
- Flag supported and succeeds (for O(1) implementation): pre-zeroing is a win
- Flag supported but fails (for O(n) implementation): no penalty from pre-zeroing



Protocol extension: time to pull a fast one

```
$ ./convert zeromode=plugin fastzeromode=default
elapsed time: 4.25743 s
write: 200 ops, 52428800 bytes, 9.85174e+07 bits/s
zero: 4 ops, 104857600 bytes, 1.97035e+08 bits/s
      4 fast=1

$ ./convert zeromode=emulate fastzeromode=default
elapsed time: 8.39064 s
write: 400 ops, 104857600 bytes, 9.99757e+07 bits/s
     50 fast=0
      1 fast=1

$
```

Can you provide me some references?

Red Hat

qemu: wire `BDRV_REQ_NO_FALLBACK` to `NBD_CMD_FLAG_FAST_ZERO` in both server and client

libnbd: add `ENOTSUP` errno, add `LIBNBD_CMD_FLAG_FAST_ZERO` to `nbd_zero()` and `nbd_aio_zero()`

nbdkit: add support in the following filters and plugins: always fails always fast conditional

curl	**info**	**null**	**rust**
data	iso	**ocaml**	**sh**
ext2	libvirt	partitioning	split
file	linuxdisk	pattern	streaming
floppy	lua	perl	tar
full	**memory**	python	tcl
guestfs	**nbd**	random	vddk
gzip		ruby	zero

plugins available in nbdkit 1.15.6

blocksize	error	noparallel	readahead
cache	fua	**nozero**	**retry**
cacheextents	**log**	offset	stats
cow	nocache	partition	**truncate**
delay	noextents	rate	xz

filters available in nbdkit 1.15.6

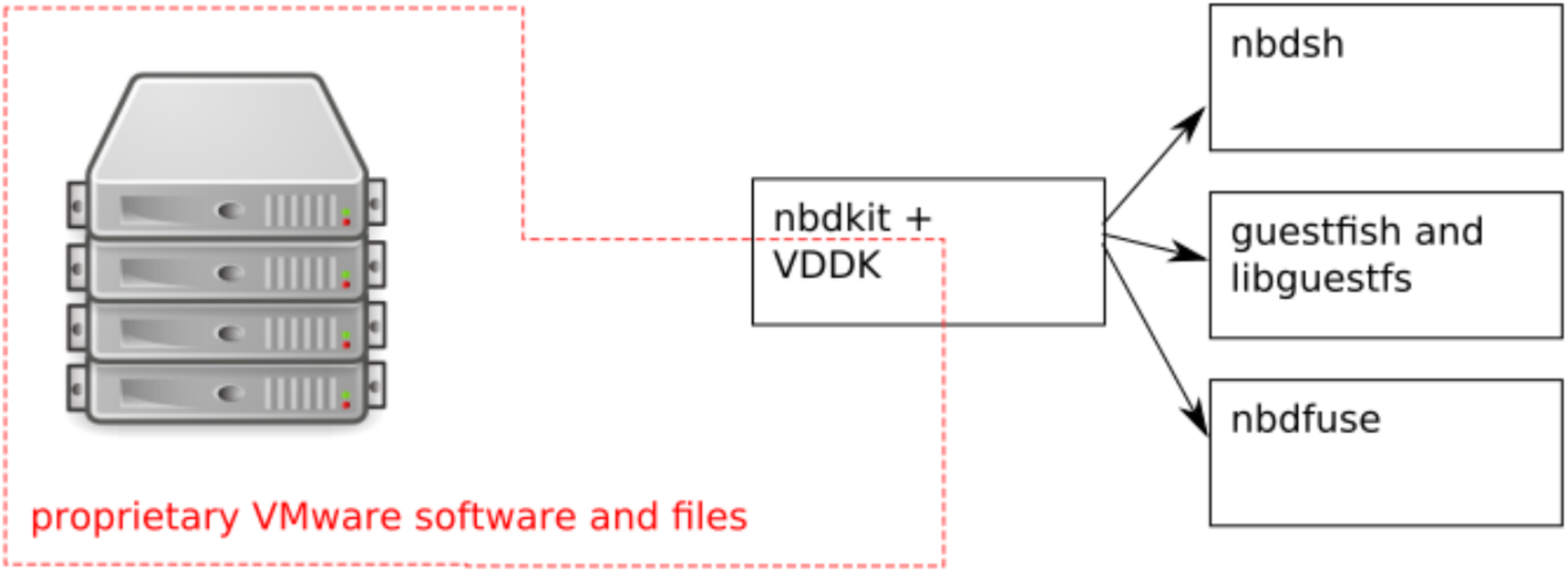
Some fun things we can do with nbdkit

Red Hat

nbdkit is an **NBD server with plugins**

https://github.com/libguestfs/nbdkit

VDDK



Server icon by RRZEicons https://commons.wikimedia.org/wiki/File:Server-multiple.svg

Next slide Back Kill & restart Options

VDDK

```
$ LD_LIBRARY_PATH=vddk/lib64 nbdkit vddk file=$PWD/fedora.vmdk

$ nbdsh -u nbd://localhost

Welcome to nbdsh, the shell for interacting with
Network Block Device (NBD) servers.

The ‘nbd’ module has already been imported and there
is an open NBD handle called ‘h’.

h.connect_tcp ("remote", "10809")   # Connect to a remote server.
h.get_size ()                       # Get size of the remote disk.
buf = h.pread (512, 0, 0)           # Read the first sector.
exit() or Ctrl-D                    # Quit the shell
help (nbd)                          # Display documentation

nbd> buf = h.pread (512, 0)
nbd> print (buf)
b"\xebH\x90\x10\x8e\xd0\xbc\x00\xb0\xb8\x00\x00\x8e\xd8\x8e\xc0\xfb\xbe\x00|\xbf\x00\x06\xb9\x00\x02\xf3\xa4\xea!\x06\x00\x00\xbe\xbe\x078\x04u\x0b\x83\xc6\x10\x81\xfe\xfe\x0
7u\xf3\xeb\x16\xb4\x02\xb0\x01\xbb\x00|\xb2\x80\x8at\x03\x02\xff\x00\x00\x80\x18t\xc4\x00\x00\x08\xfa\x90\x90\xf6\xc2\x80u\x02\xb2\x80\xeaY|\x00\x001\xc0\x8e\xd8\x8e\xd0\xbc\
x00 \xfb\xa0@|<\xfft\x02\x88\xc2R\xf6\xc2\x80tT\xb4A\xbb\xaaU\xcd\x13ZRrI\x81\xfbU\xaauC\xa0A|\x84\xc0u\x05\x83\xe1\x01t7f\x8bL\x10\xbe\x05|\xc6D\xff\x01f\x8b\x1eD|\xc7\x04\x
10\x00\xc7D\x02\x01\x00f\x89\\\x08\xc7D\x06\x00pf1\xc0\x89D\x04f\x89D\x0c\xb4B\xcd\x13r\x05\xbb\x00p\xeb}\xb4\x08\xcd\x13s\n\xf6\xc2\x80\x0f\x84\xf0\x00\xe9\x8d\x00\xbe\x05|\
xc6D\xff\x00f1\xc0\x88\xf0@f\x89D\x041\xd2\x88\xca\xc1\xe2\x02\x88\xe8\x88\xf4@\x89D\x081\xc0\x88\xd0\xc0\xe8\x02f\x89\x04f\xa1D|f1\xd2f\xf74\x88T\nf1\xd2f\xf7t\x04\x88T\x0b\
x89D\x0c;D\x08}<\x8aT\r\xc0\xe2\x06\x8aL\n\xfe\xc1\x08\xd1\x8al\x0cZ\x8at\x0b\xbb\x00p\x8e\xc31\xdb\xb8\x01\x02\xcd\x13r*\x8c\xc3\x8e\x06H|`\x1e\xb9\x00\x01\x8e\xdb1\xf61\xff
\xfc\xf3\xa5\x1fa\xff&B|\xbe\x7f}\xe8@\x00\xeb\x0e\xbe\x84}\xe88\x00\xeb\x06\xbe\x8e}\xe80\x00\xbe\x93}\xe8*\x00\xeb\xfeGRUB \x00Geom\x00Hard Disk\x00Read\x00 Error\x00\xbb\x
01\x00\xb4\x0e\xcd\x10\xac<\x00u\xf4\xc3\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xae'\x01\x00\x00\x00\x00\x01\x10\x83\x03\xe0\xff\x00\x08\x00\
x00\x00\xf8?\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00U\xaa"
nbd>

$ guestfish --ro --format=raw -a nbd://localhost -i
```

```
$ guestfish --ro --format=raw -a nbd://localhost -i

Welcome to guestfish, the guest filesystem shell for
editing virtual machine filesystems and disk images.

Type: 'help' for help on commands
      'man' to read the manual
      'quit' to quit the shell

Operating system: Fedora 17 (Beefy Miracle)
/dev/sda1 mounted on /

><fs> ls /
bin
boot
dev
etc
home
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
><fs> quit


$ nbdfuse mp nbd://localhost &
[1] 13700

$ ls -lh mp
total 0
-rw-rw-rw-. 1 eblake eblake 10G Oct 31 14:23 nbd

$
```

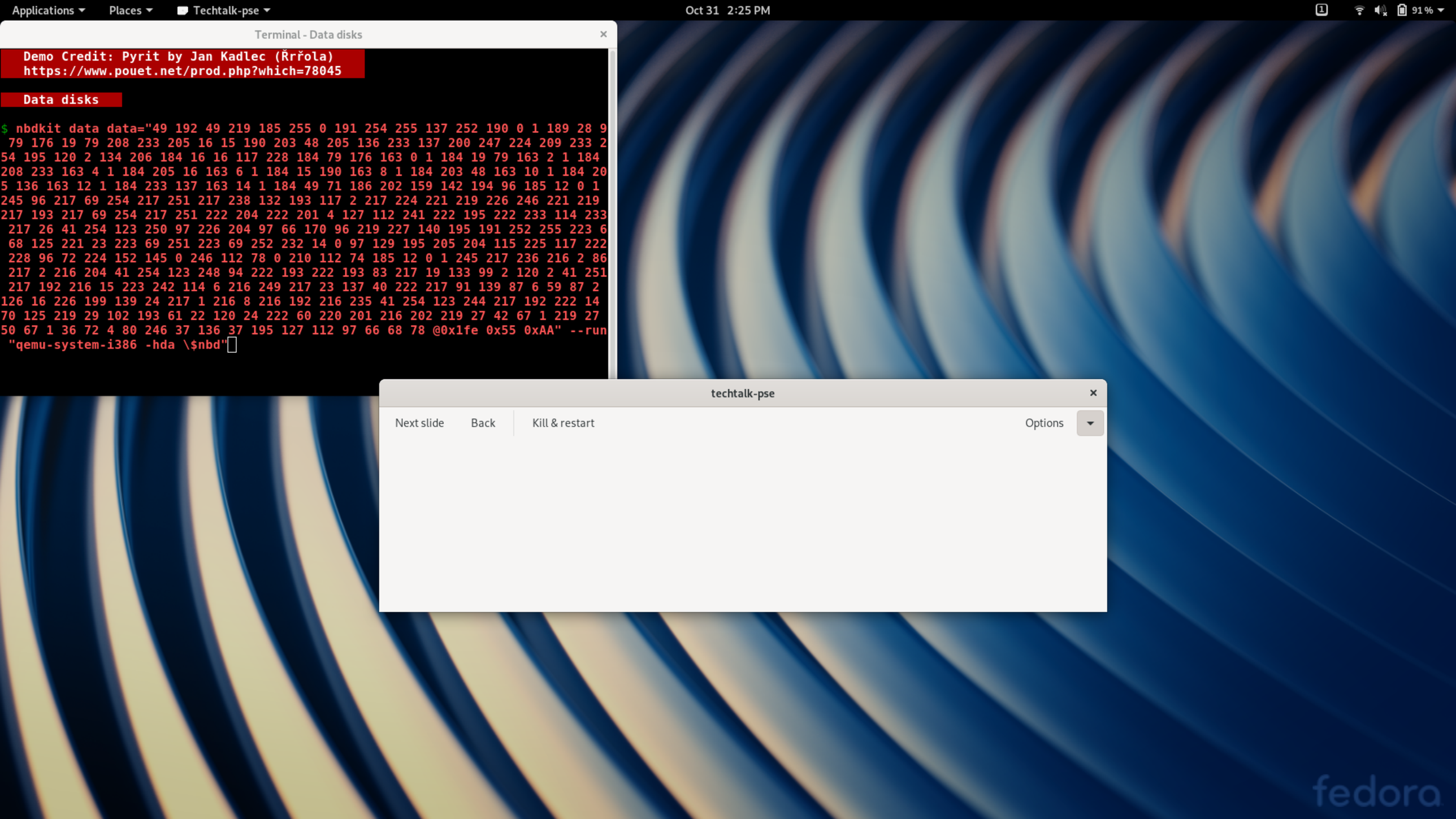
```
var
><fs> quit


$ nbdfuse mp nbd://localhost &
[1] 13700

$ ls -lh mp
total 0
-rw-rw-rw-. 1 eblake eblake 10G Oct 31 14:23 nbd

$ hexdump -C mp/nbd | head -30
00000000  eb 48 90 10 8e d0 bc 00  b0 b8 00 00 8e d8 8e c0  |.H..............|
00000010  fb be 00 7c bf 00 06 b9  00 02 f3 a4 ea 21 06 00  |...|.........!..|
00000020  00 be be 07 38 04 75 0b  83 c6 10 81 fe fe 07 75  |....8.u........u|
00000030  f3 eb 16 b4 02 b0 01 bb  00 7c b2 80 8a 74 03 02  |.........|...t..|
00000040  ff 00 00 80 18 74 c4 00  00 08 fa 90 90 f6 c2 80  |.....t..........|
00000050  75 02 b2 80 ea 59 7c 00  00 31 c0 8e d8 8e d0 bc  |u....Y|..1......|
00000060  00 20 fb a0 40 7c 3c ff  74 02 88 c2 52 f6 c2 80  |. ..@|<.t...R...|
00000070  74 54 b4 41 bb aa 55 cd  13 5a 52 72 49 81 fb 55  |tT.A..U..ZRrI..U|
00000080  aa 75 43 a0 41 7c 84 c0  75 05 83 e1 01 74 37 66  |.uC.A|..u....t7f|
00000090  8b 4c 10 be 05 7c c6 44  ff 01 66 8b 1e 44 7c c7  |.L...|.D..f..D|.|
000000a0  04 10 00 c7 44 02 01 00  66 89 5c 08 c7 44 06 00  |....D...f.\..D..|
000000b0  70 66 31 c0 89 44 04 66  89 44 0c b4 42 cd 13 72  |pf1..D.f.D..B..r|
000000c0  05 bb 00 70 eb 7d b4 08  cd 13 73 0a f6 c2 80 0f  |...p.}....s.....|
000000d0  84 f0 00 e9 8d 00 be 05  7c c6 44 ff 00 66 31 c0  |........|.D..f1.|
000000e0  88 f0 40 66 89 44 04 31  d2 88 ca c1 e2 02 88 e8  |..@f.D.1........|
000000f0  88 f4 40 89 44 08 31 c0  88 d0 c0 e8 02 66 89 04  |..@.D.1......f..|
00000100  66 a1 44 7c 66 31 d2 66  f7 34 88 54 0a 66 31 d2  |f.D|f1.f.4.T.f1.|
00000110  66 f7 74 04 88 54 0b 89  44 0c 3b 44 08 7d 3c 8a  |f.t..T..D.;D.}<.|
00000120  54 0d c0 e2 06 8a 4c 0a  fe c1 08 d1 8a 6c 0c 5a  |T.....L......l.Z|
00000130  8a 74 0b bb 00 70 8e c3  31 db b8 01 02 cd 13 72  |.t...p..1......r|
00000140  2a 8c c3 8e 06 48 7c 60  1e b9 00 01 8e db 31 f6  |*....H|`......1.|
00000150  31 ff fc f3 a5 1f 61 ff  26 42 7c be 7f 7d e8 40  |1.....a.&B|..}.@|
00000160  00 eb 0e be 84 7d e8 38  00 eb 06 be 8e 7d e8 30  |.....}.8.....}.0|
00000170  00 be 93 7d e8 2a 00 eb  fe 47 52 55 42 20 00 47  |...}.*...GRUB .G|
00000180  65 6f 6d 00 48 61 72 64  20 44 69 73 6b 00 52 65  |eom.Hard Disk.Re|
00000190  61 64 00 20 45 72 72 6f  72 00 bb 01 00 b4 0e cd  |ad. Error.......|
000001a0  10 ac 3c 00 75 f4 c3 00  00 00 00 00 00 00 00 00  |..<.u...........|
000001b0  00 00 00 00 00 00 00 00  ae 27 01 00 00 00 00 00  |.........'......|
000001c0  01 10 83 03 e0 ff 00 08  00 00 00 f8 3f 01 00 00  |............?...|
000001d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|

$
```

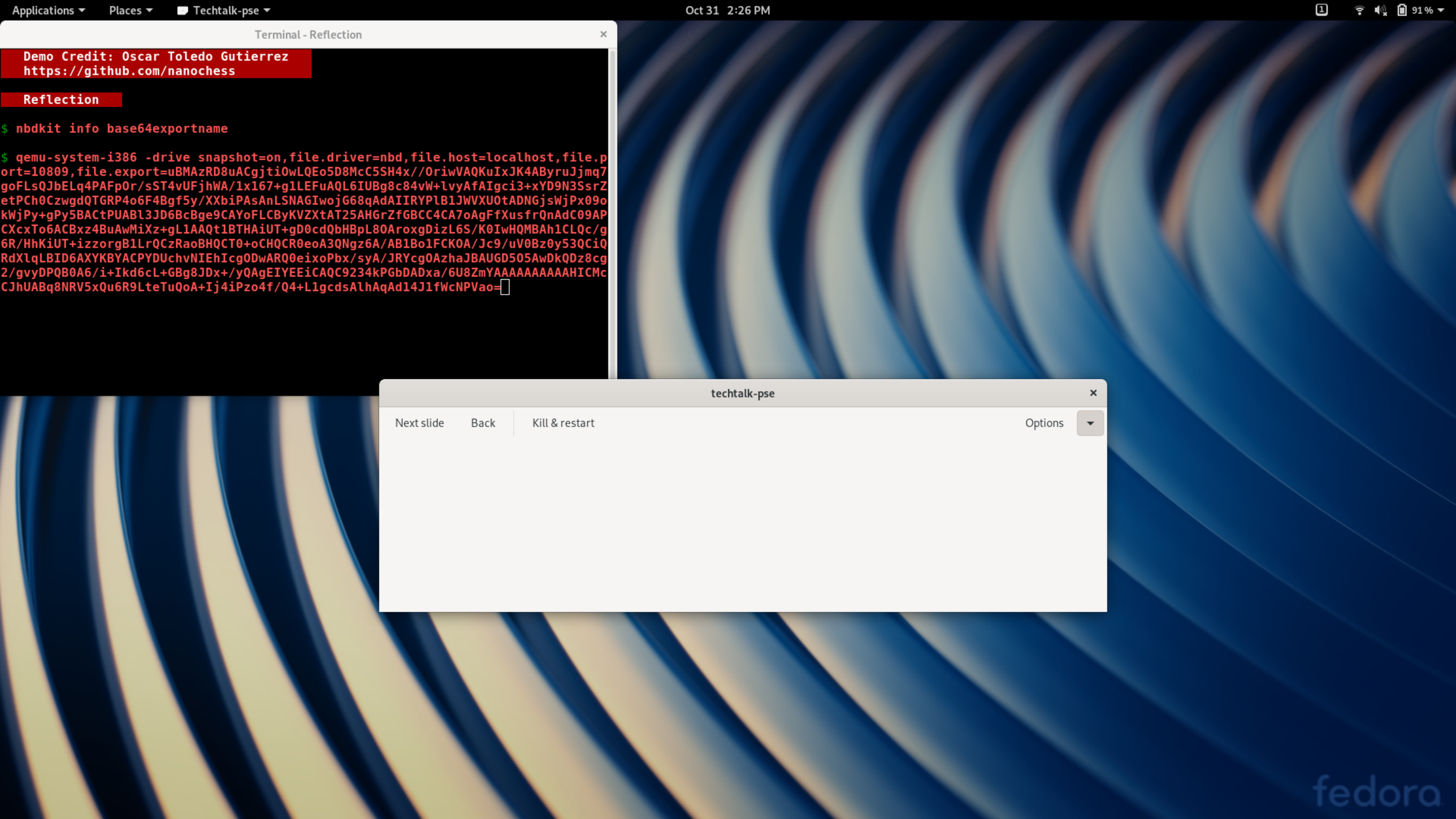
Tiny VMs

```
$ mkdir -p root/dev root/sbin root/bin root/usr/sbin root/usr/bin
$ sudo mknod root/dev/console c 5 1
$ cp /sbin/busybox root/sbin/
$ ln root/sbin/busybox root/bin/ls
$ ln root/sbin/busybox root/bin/sh
$ ln root/sbin/busybox root/bin/init
$ ls -lR root
```

```
$ sudo mknod root/dev/console c 5 1

$ cp /sbin/busybox root/sbin/

$ ln root/sbin/busybox root/bin/ls

$ ln root/sbin/busybox root/bin/sh

$ ln root/sbin/busybox root/bin/init

$ ls -lR root
root:
total 16
drwxrwxr-x. 2 eblake eblake 4096 Oct 31 14:24 bin
drwxrwxr-x. 2 eblake eblake 4096 Oct 31 14:24 dev
drwxrwxr-x. 2 eblake eblake 4096 Oct 31 14:24 sbin
drwxrwxr-x. 4 eblake eblake 4096 Oct 31 14:24 usr

root/bin:
total 3540
-rwxr-xr-x. 4 eblake eblake 1204976 Oct 31 14:24 init
-rwxr-xr-x. 4 eblake eblake 1204976 Oct 31 14:24 ls
-rwxr-xr-x. 4 eblake eblake 1204976 Oct 31 14:24 sh

root/dev:
total 0
crw-r--r--. 1 root root 5, 1 Oct 31 14:24 console

root/sbin:
total 1180
-rwxr-xr-x. 4 eblake eblake 1204976 Oct 31 14:24 busybox

root/usr:
total 8
drwxrwxr-x. 2 eblake eblake 4096 Oct 31 14:24 bin
drwxrwxr-x. 2 eblake eblake 4096 Oct 31 14:24 sbin

root/usr/bin:
total 0

root/usr/sbin:
total 0

$ nbdkit -U - linuxdisk root --run 'export unixsocket; ./qemu.sh'
```

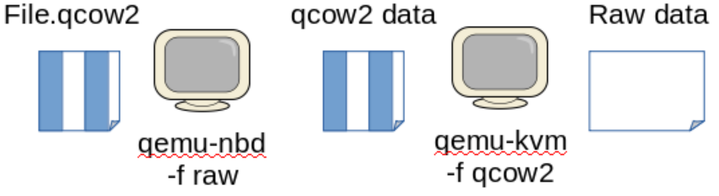
```
[    0.826974] EXT4-fs (sda1): mounting ext2 file system using the ext4 subsystem
[    0.830040] EXT4-fs (sda1): mounted filesystem without journal. Opts: (null)
[    0.831154] VFS: Mounted root (ext2 filesystem) on device 8:1.
[    0.832099] devtmpfs: mounted
[    0.833510] Freeing unused decrypted memory: 2040K
[    0.835173] Freeing unused kernel image memory: 2272K
[    0.836066] Write protecting the kernel read-only data: 20480k
[    0.837239] Freeing unused kernel image memory: 2016K
[    0.838132] Freeing unused kernel image memory: 1580K
[    0.846217] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[    0.847150] rodata_test: all tests were successful
[    0.847877] x86/mm: Checking user space page tables
[    0.855589] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[    0.856475] Run /sbin/init as init process
[    0.857810] Run /etc/init as init process
[    0.858383] Run /bin/init as init process
[    0.863978] random: init: uninitialized urandom read (8 bytes read)
can't run '/etc/init.d/rcS': No such file or directory

Please press Enter to activate this console.
[    2.810963] random: sh: uninitialized urandom read (8 bytes read)
# /sbin/busybox --install
[    8.734471] random: busybox: uninitialized urandom read (8 bytes read)
# ls /sbin
[   14.167193] random: ls: uninitialized urandom read (8 bytes read)
acpid            hdparm           losetup          route
adjtimex         hwclock          lsmod            run-init
arp              ifconfig         makedevs         runlevel
blkid            ifdown           mdev             setconsole
blockdev         ifenslave        mkdosfs          slattach
bootchartd       ifup             mke2fs           start-stop-daemon
busybox          init             mkfs.ext2        sulogin
depmod           insmod           mkfs.minix       swapoff
devmem           ip               mkfs.vfat        swapon
fbsplash         ipaddr           mkswap           switch_root
fdisk            iplink           modinfo          sysctl
findfs           ipneigh          modprobe         syslogd
freeramdisk      iproute          nameif           tunctl
fsck             iprule           pivot_root       udhcpc
fsck.minix       iptunnel         poweroff         uevent
fstrim           klogd            raidautorun      vconfig
getty            loadkmap         reboot           watchdog
halt             logread          rmmod            zcip
#
```


Applications
Places
Techtalk-pse
Oct 31 2:25 PM
91 %
Terminal - Data disks
Demo Credit: Pyrit by Jan Kadlec (Řrřola)
https://www.pouet.net/prod.php?which=78045
Data disks
$ nbdkit data data="49 192 49 219 185 255 0 191 254 255 137 252 190 0 1 189 28 9
79 176 19 79 208 233 205 16 15 190 203 48 205 136 233 137 200 247 224 209 233 2
54 195 120 2 134 206 184 16 16 117 228 184 79 176 163 0 1 184 19 79 163 2 1 184
208 233 163 4 1 184 205 16 163 6 1 184 15 190 163 8 1 184 203 48 163 10 1 184 20
5 136 163 12 1 184 233 137 163 14 1 184 49 71 186 202 159 142 194 96 185 12 0 1
245 96 217 69 254 217 251 217 238 132 193 117 2 217 224 221 219 226 246 221 219
217 193 217 69 254 217 251 222 204 222 201 4 127 112 241 222 195 222 233 114 233
217 26 41 254 123 250 97 226 204 97 66 170 96 219 227 140 195 191 252 255 223 6
68 125 221 23 223 69 251 223 69 252 232 14 0 97 129 195 205 204 115 225 117 222
228 96 72 224 152 145 0 246 112 78 0 210 112 74 185 12 0 1 245 217 236 216 2 86
217 2 216 204 41 254 123 248 94 222 193 222 193 83 217 19 133 99 2 120 2 41 251
217 192 216 15 223 242 114 6 216 249 217 23 137 40 222 217 91 139 87 6 59 87 2
126 16 226 199 139 24 217 1 216 8 216 192 216 235 41 254 123 244 217 192 222 14
70 125 219 29 102 193 61 22 120 24 222 60 220 201 216 202 219 27 42 67 1 219 27
50 67 1 36 72 4 80 246 37 136 37 195 127 112 97 66 68 78 @0x1fe 0x55 0xAA" --run
"qemu-system-i386 -hda \$nbd"
techtalk-pse
Next slide
Back
Kill & restart
Options
fedora


Applications
Places
Techtalk-pse
Oct 31 2:26 PM
91 %
Terminal - Reflection
Demo Credit: Oscar Toledo Gutierrez
https://github.com/nanochess
Reflection
$ nbdkit info base64exportname
$ qemu-system-i386 -drive snapshot=on,file.driver=nbd,file.host=localhost,file.port=10809,file.export=uBMAzRD8uACgjtiOwLQEo5D8McC5SH4x//OriwVAQKuIxJK4AByruJjmq7goFLsQJbELq4PAFpOr/sST4vUFjhWA/1x167+g1LEFuAQL6IUBg8c84vW+lvyAfAIgci3+xYD9N3SsrZetPCh0CzwgdQTGRP4o6F4Bgf5y/XXbiPAsAnLSNAGIwojG68qAdAIIRYPlB1JWVXUOtADNGjsWjPx09okWjPy+gPy5BACtPUABl3JD6BcBge9CAYoFLCByKVZXtAT25AHGrZfGBCC4CA7oAgFfXusfrQnAdC09APCXcxTo6ACBxz4BuAwMiXz+gL1AAQt1BTHAiUT+gD0cdQbHBpL8OAroxgDizL6S/K0IwHQMBAh1CLQc/g6R/HhKiUT+izzorgB1LrQCzRaoBHQCT0+oCHQCR0eoA3QNgz6A/AB1Bo1FCKOA/Jc9/uV0Bz0y53QCiQRdXlqLBID6AXYKBYACPYDUchvNIEhIcgODwARQ0eixoPbx/syA/JRYcgOAzhaJBAUGD5O5AwDkQDz8cg2/gvyDPQB0A6/i+Ikd6cL+GBg8JDx+/yQAgEIYEEiCAQC9234kPGbDADxa/6U8ZmYAAAAAAAAAAHICMcCJhUABq8NRV5xQu6R9LteTuQoA+Ij4iPzo4f/Q4+L1gcdsAlhAqAd14J1fWcNPVao=
techtalk-pse
Next slide
Back
Kill & restart
Options
fedora

Using qcow2 with NBD

Red Hat

		
Pro	• Matches existing use • Server can start with thin qcow2, and grow it as needed with guest activity	• Guest-visible size can be changed • Backing files • Dirty bitmap tracking • Any other qcow2 features...
Con	• Guest size is fixed • No access to qcow2 features from client	• Server file must be preallocated, or else guest hits ENOSPC • Internal snapshots are unlikely to work

Can we merge the best of both worlds, giving the client access to all qcow2 features, but permitting resize of the underlying file on the host?

Clipart of computer http://clipart-library.com/clipart/76irEkBcK.htm

Tradeoff: Automatic or explicit

Red Hat

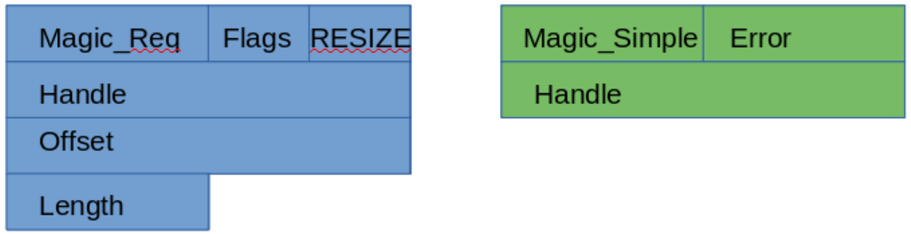
When should resize happen? Starting with a 1k image:

Automatic (like file system):
NBD_CMD_WRITE(offset=0,len=2k) -> success, size now 2k

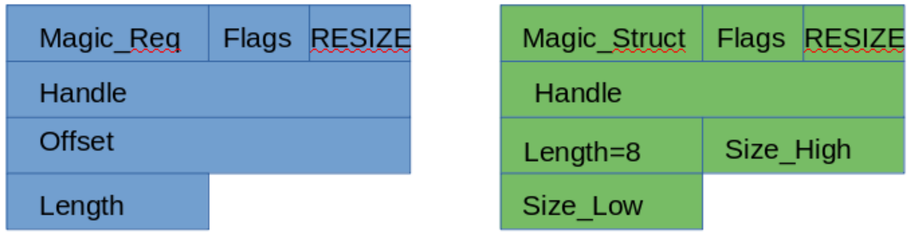
Explicit (like block device):
NBD_CMD_WRITE(offset=0,len=2k) -> fails
NBD_CMD_RESIZE(2k) -> success, size now 2k
NBD_CMD_WRITE(offset=0,len=2k) -> success

Tradeoff: Simple or structured

Simple: 16-byte response, boolean answer



Structured: one or more 20+-byte response (simplest is a single 28-byte response)
Requires adding `NBD_REPLY_TYPE_RESIZE`



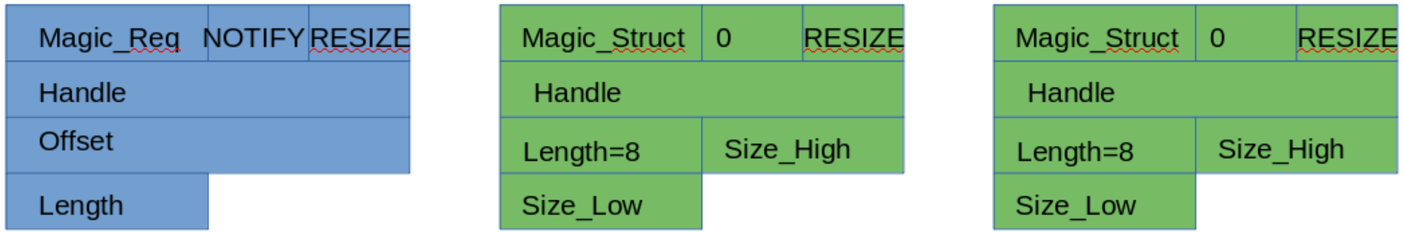
Tradeoff: poll or notify

Polling:

- Server cannot report size unless client asks
- Each server reply sets `NBD_REPLY_DONE` flag bit
- Client must ask in a loop when polling for a change from the server

Notify:

- Client uses `NBD_CMD_FLAG_RESIZE_NOTIFY` to request an open-ended command
- Server replies with `NBD_REPLY_DONE` flag clear whenever it has a new size to report



Tradeoff: how much complexity

Red Hat

Tradeoffs in implementation complexity

	Lots of knobs in the NBD spec	Mandate all-or-none server implementation
Pro	• Implement as much or as little as convenient for the server • Matches reality that no one resize solution fits all	• Easier interoperability testing • Client can rely on all servers giving same response
Con	• More combinations requires more testing • Odd or untested combinations could produce weird behaviors or even CVEs • Clients must be prepared for more fallbacks • NBD protocol tries to be as simple as possible	• Not all tradeoffs map easily to existing implementation • Open-ended structured replies requires more efforts for writing a compliant server

In summary

- NBD is a simple yet flexible protocol.
- Extensions are easy to prototype in nbdkit and libnbd.
- Interoperability testing is worthwhile.
- Expect more extensions as we find more ways to utilize NBD in virtualization.

Get it

- nbdkit ≥ 1.15.6 https://github.com/libguestfs/nbdkit
- libnbd ≥ 1.1.5 https://github.com/libguestfs/libnbd
- qemu ≥ 4.2 (not yet released). https://qemu.org
- This talk: http://git.annexia.org/?p=libguestfs-talks.git
- Presentation software: Tech Talk PSE 1.2

Any Questions?