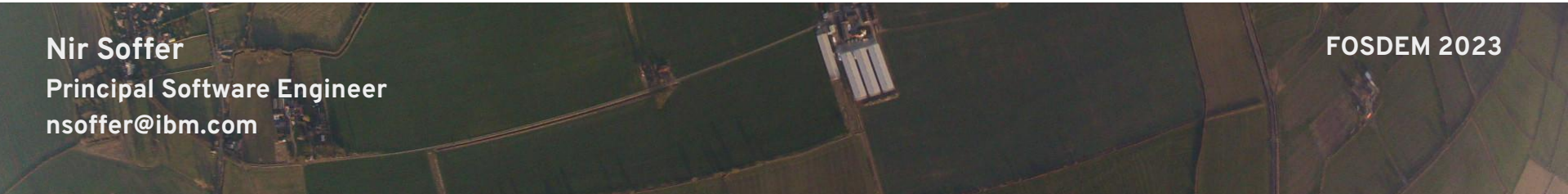
blkhash

Fast disk image checksums



Nir Soffer
Principal Software Engineer
nsoffer@ibm.com

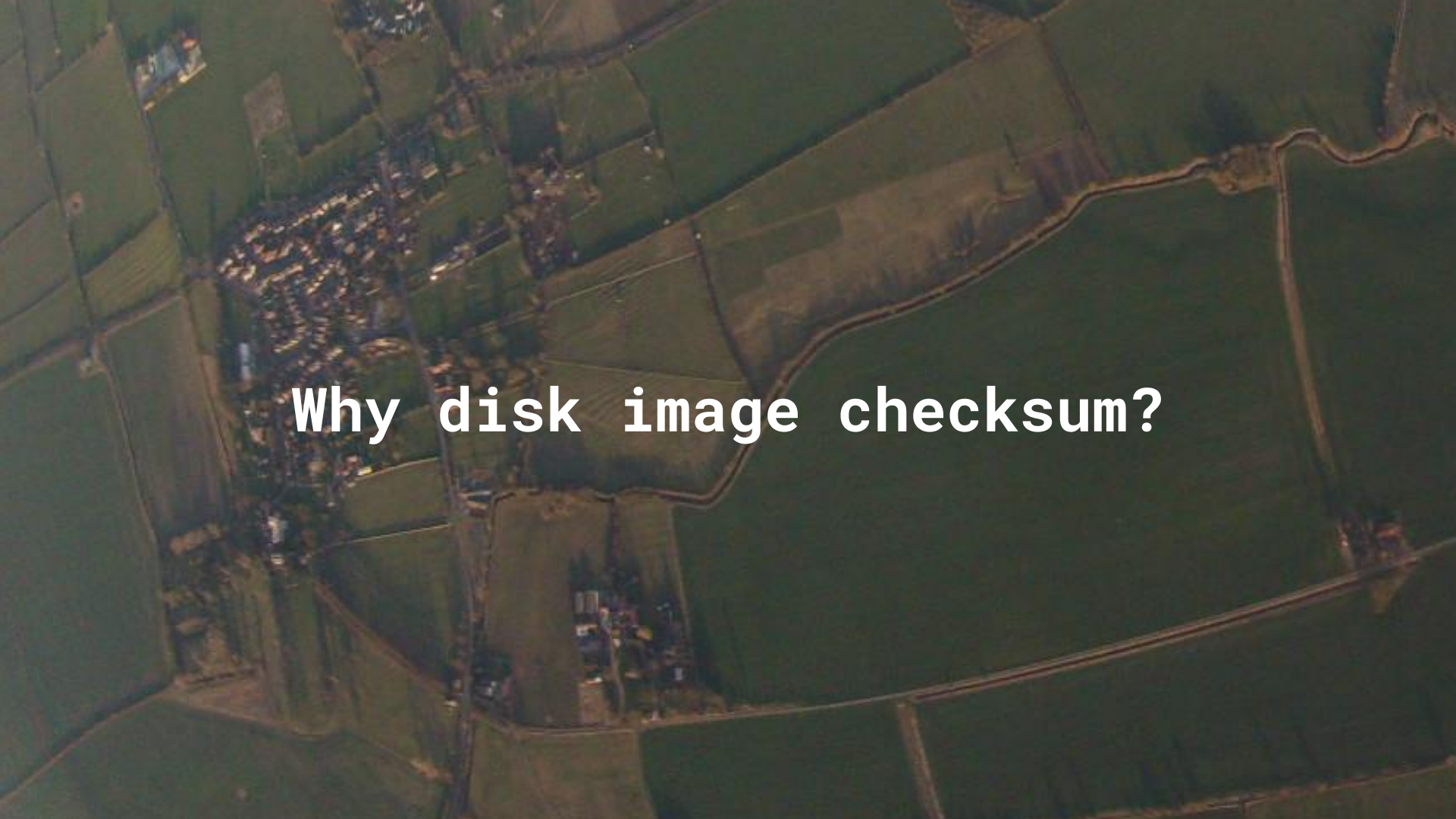
FOSDEM 2023

Who am I?

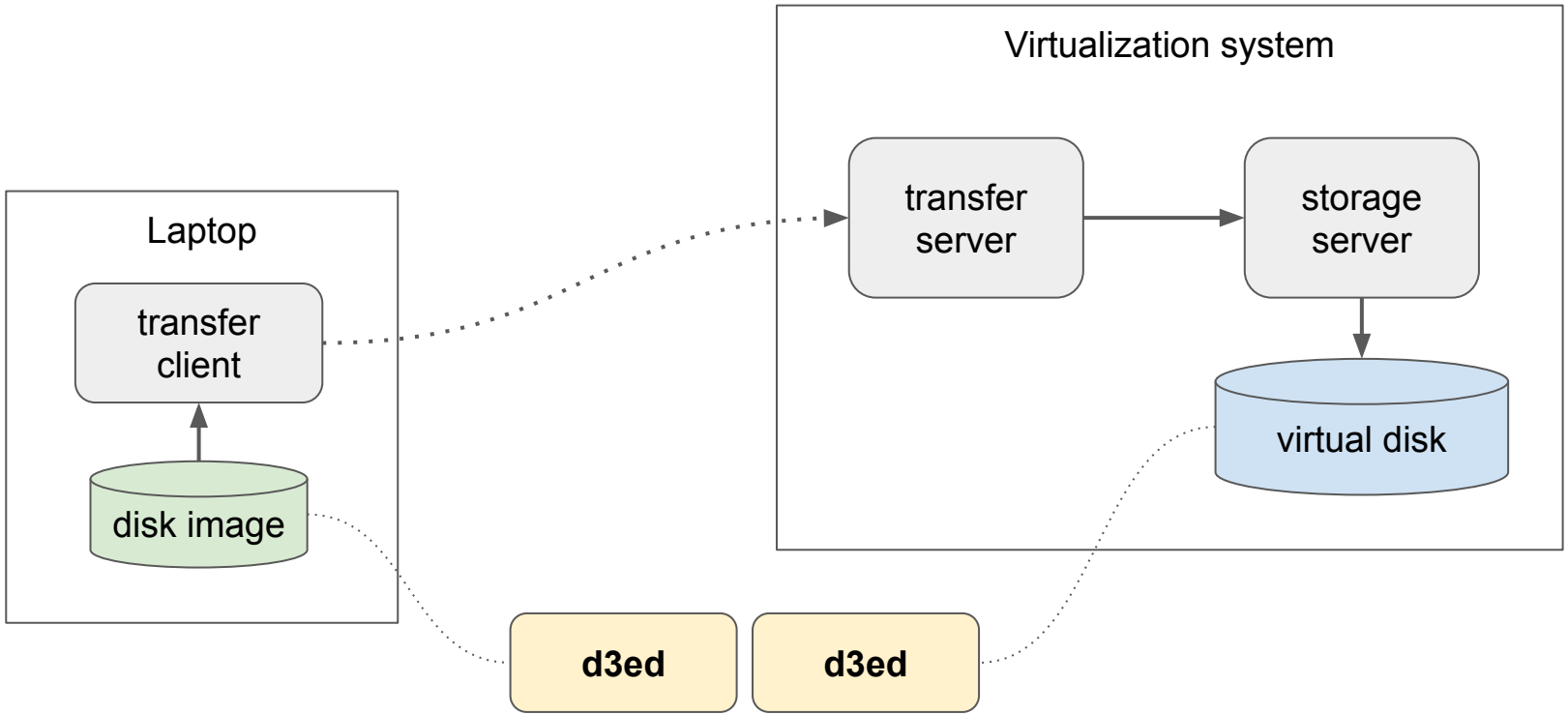
- Long-time contributor to free software projects
- Worked 9 years for Red Hat on oVirt storage
- Focused on incremental backup, image transfer, and NBD tools

Agenda

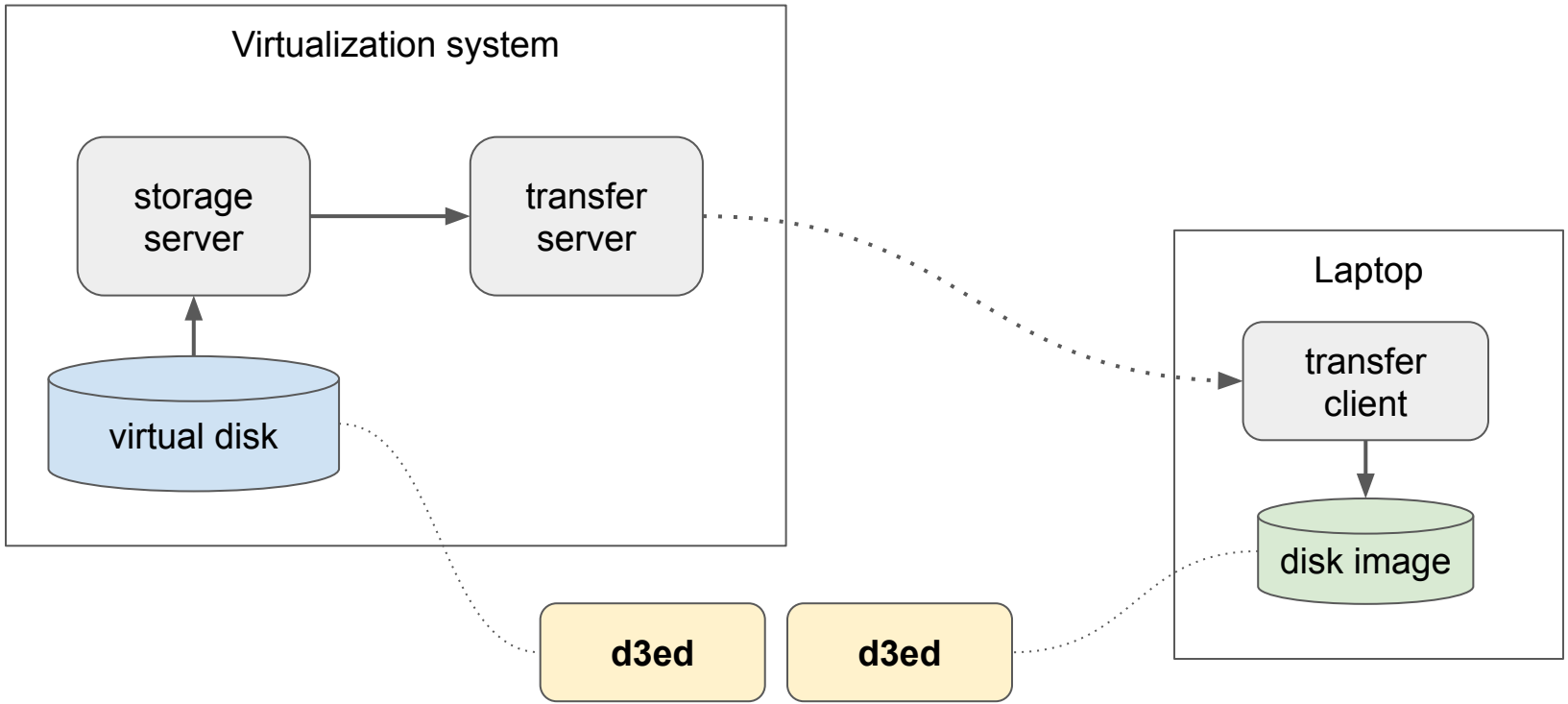
- Why disk image checksums?
- Issues with standard tools
- The blksum command
- The blkhash library
- Integration with qemu-img
- Live demo
- How to contribute
- Questions


Why disk image checksum?

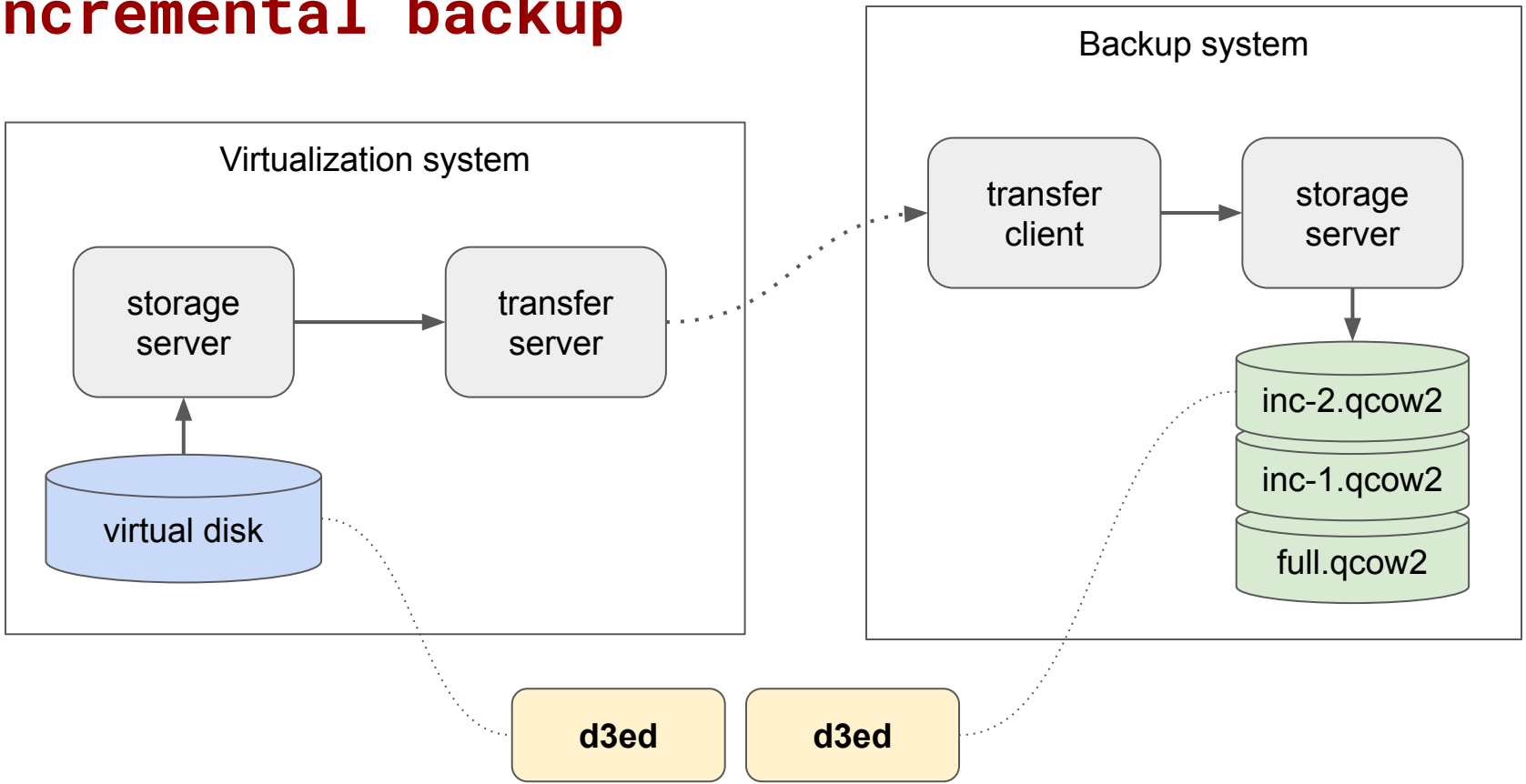
Upload disk image to virtual disk



Download virtual disk to disk image



Incremental backup




Issues with standard tools

Image format

Standard tools like sha*sum do not understand disk image format:

```
$ qemu-img compare fedora-35.raw fedora-35.qcow2
Images are identical.

$ sha256sum fedora-35.raw fedora-35.qcow2
88da042d3c4ad61091c25414513e74d2eaa7183f6c555476a9be55372ab284c6  fedora-35.raw
ae33f66851f5306bad5667ba7aabfea2d65cc2a13989e4fb0f73f4627861dbf2  fedora-35.qcow2
```

Image compression

Standard tools like sha*sum cannot cope with image compression:

```
$ qemu-img compare fedora-35.qcow2 fedora-35.comp.qcow2
Images are identical.

$ sha256sum fedora-35.qcow2 fedora-35.comp.qcow2
ae33f66851f5306bad5667ba7aabfea2d65cc2a13989e4fb0f73f4627861dbf2  fedora-35.qcow2
b4499a1170acd29657bf95eb3c97e54fb5f248234ddb75ca03a4d653a7a3c25f  fedora-35.comp.qcow2
```

Host data layout

Converting with unordered writes (-W) creates different host data:

```
$ qemu-img convert -f qcow2 -O qcow2 -W fedora-35.qcow2 fedora-35.copy.qcow2

$ qemu-img compare fedora-35.qcow2 fedora-35.copy.qcow2
Images are identical.

$ sha256sum fedora-35.qcow2 fedora-35.copy.qcow2
ae33f66851f5306bad5667ba7aabfea2d65cc2a13989e4fb0f73f4627861dbf2  fedora-35.qcow2
f35608ec810d6cc6b42231d630cb409056f0d4bb98748f16b095ff596fd37a6f  fedora-35.copy.qcow2
```

Image sparseness

Standard tools must read and process the entire image even if large portion of the image is unallocated.

```
$ qemu-img info fedora-35.raw
virtual size: 6 GiB (6442450944 bytes)
disk size: 1.13 GiB

$ time sha256sum fedora-35.raw
88da042d3c4ad61091c25414513e74d2eaa7183f6c555476a9be55372ab284c6  fedora-35.raw

real  0m11.421s
user  0m10.699s
sys   0m0.662s
```

Computing a checksum is slow

Computing checksum for a big image is not practical:

```
$ qemu-img create -f raw empty-100g.raw 100g
Formatting 'empty-100g.raw', fmt=raw size=107374182400

$ time sha256sum empty-100g.raw
f0b14a8da7f1c48a0846647a078b97956edd8df451a62fc4b466879aa24d4fd7  empty-100g.raw

real 3m17.608s
user 3m4.453s
sys  0m12.601s
```


The blksum command

Feels like sha*sum

If you know how to use sha*sum you know how to use blksum:

```
$ blksum fedora-35.raw
6f84badc7d20700d01487724f7d8f2dd602abc866d42fec301817242daef28bf  fedora-35.raw
```

Understands image formats

Identical images with different format have the same checksum:

```
$ blksum fedora-35.raw
6f84badc7d20700d01487724f7d8f2dd602abc866d42fec301817242daef28bf  fedora-35.raw

$ blksum fedora-35.qcow2
6f84badc7d20700d01487724f7d8f2dd602abc866d42fec301817242daef28bf  fedora-35.qcow2
```

Supports compressed qcow2

Compressed and uncompressed qcow2 have the same checksum:

```
$ blksum fedora-35.qcow2
6f84badc7d20700d01487724f7d8f2dd602abc866d42fec301817242daef28bf  fedora-35.qcow2

$ blksum fedora-35.comp.qcow2
6f84badc7d20700d01487724f7d8f2dd602abc866d42fec301817242daef28bf  fedora-35.comp.qcow2
```

Supports snapshots

Can compute a checksum of a qcow2 chain:

```
$ qemu-img create -f qcow2 -b fedora-35.raw -F raw snapshot.qcow2

$ blksum snapshot.qcow2
6f84badc7d20700d01487724f7d8f2dd602abc866d42fec301817242daef28bf  snapshot.qcow2
```

Supports NBD URL

You can access NBD server instead of a local image:

```
$ qemu-nbd -t -e0 -f qcow2 fedora-35.qcow2 &

$ blksum nbd://localhost
6f84badc7d20700d01487724f7d8f2dd602abc866d42fec301817242daef28bf  nbd://localhost
```

Supports reading from pipe

You can read raw image from a pipe:

```
$ blksum < fedora-35.raw
6f84badc7d20700d01487724f7d8f2dd602abc866d42fec301817242daef28bf  -
```

blksum is fast

```
$ hyperfine -r5 -w1 "blksum fedora-35.raw" "sha256sum fedora-35.raw"
Benchmark 1: blksum fedora-35.raw
  Time (mean ± σ):     933.9 ms ±   5.8 ms    [User: 2865.2 ms, System: 547.4 ms]
  Range (min … max):   927.3 ms … 939.9 ms    5 runs

Benchmark 2: sha256sum fedora-35.raw
  Time (mean ± σ):     14.726 s ±  0.040 s    [User: 13.973 s, System: 0.713 s]
  Range (min … max):   14.666 s … 14.762 s    5 runs

Summary
  'blksum fedora-35.raw' ran
    15.77 ± 0.11 times faster than 'sha256sum fedora-35.raw'
```

blksum is fast

blksum - 8 TiB empty image:

```
$ hyperfine -r5 -w1 "blksum empty-8t.raw"
Benchmark 1: blksum empty-8t.raw
  Time (mean ± σ):      2.590 s ±  0.074 s    [User: 10.242 s, System: 0.088 s]
  Range (min … max):    2.498 s …  2.670 s    5 runs
```

sha256sum:

```
measured time for 100 GiB: 197 s
estimated time for 8 TiB: 16138 s (4h:29m)
```

blksum is ~6000 times faster

blksum checksum is different

Using different algorithm you get different checksum:

```
$ sha1sum fedora-35.raw
784013d23c7ce1f60adb688e4d1d48003a5dac95  fedora-35.raw

$ sha256sum fedora-35.raw
88da042d3c4ad61091c25414513e74d2eaa7183f6c555476a9be55372ab284c6  fedora-35.raw

$ blksum fedora-35.raw
6f84badc7d20700d01487724f7d8f2dd602abc866d42fec301817242daef28bf  fedora-35.raw
```

How to install

Enable the copr repo:

```
dnf copr enable nsoffer/blkhash
```

And install the `blkhash` package:

```
dnf install blkhash
```


The libblkhash library

```
#include <blkhash.h>

h = blkhash_new(BLOCK_SIZE, "sha256");

/* Hash data, detecting zero blocks for faster hashing.  */
blkhash_update(h, buf, BUF_SIZE);

/* Hash 1g of zeroes extremely fast. */
blkhash_zero(h, 1024 * 1024 * 1024);

blkhash_final(h, md, &md_len);

blkhash_free(h);
```

blkhash performance (X86_64)

Results from Lenovo ThinkPad P1 Gen 3:

```
$ build/blkhash_bench | grep -v PASS | grep -v sha1
update-data (sha256): 2.00 GiB in 0.978 seconds (2.04 GiB/s)
update-zero (sha256): 50.00 GiB in 1.050 seconds (47.61 GiB/s)
zero (sha256): 2.44 TiB in 0.858 seconds (2.85 TiB/s)
```

blkhash performance (M1)

Results from MacBook Air M1:

```
$ build/blkhash_bench | grep -v PASS | grep -v sha1
update-data (sha256): 2.00 GiB in 0.286 seconds (6.99 GiB/s)
update-zero (sha256): 50.00 GiB in 1.389 seconds (36.01 GiB/s)
zero (sha256): 2.44 TiB in 0.182 seconds (13.41 TiB/s)
```

How to install

To use `libblkhash` in your app, install the `blkhash-devel` package:

```
dnf install blkhash-devel
```

Your app will depend on the `blkhash-libs` package.


Integration with qemu-img

qemu-img checksum

Patches in review for new checksum command:

```
$ ./qemu-img checksum --help | grep checksum
  checksum [--object objectdef] [--image-opts] [-f fmt] [-T src_cache] [-p] filename
```

Example:

```
$ qemu-img checksum disk.qcow2
5c92496ada47fb5f5c0d76e13373038cb8b7297f3cbd9f8294c2c7e26145e03c  disk.qcow2
```

For more info see https://gitlab.com/nirs/qemu/-/tree/checksum

`qemu-img checksum` vs `blksum`

```
$ hyperfine -w1 -r10 "./qemu-img checksum zero-6g.raw" \
                     "blksum --cache zero-6g.raw"
Benchmark 1: ./qemu-img checksum zero-6g.raw
  Time (mean ± σ):     690.5 ms ±  18.8 ms    [User: 633.8 ms, System: 2127.9 ms]
  Range (min … max):   664.2 ms … 722.3 ms    10 runs

Benchmark 2: blksum --cache zero-6g.raw
  Time (mean ± σ):     994.1 ms ±  42.5 ms    [User: 392.0 ms, System: 2821.4 ms]
  Range (min … max):   916.8 ms … 1042.9 ms    10 runs

Summary
  './qemu-img checksum zero-6g.raw' ran
    1.44 ± 0.07 times faster than 'blksum --cache zero-6g.raw'
```


Live demo

How to contribute

Testing

- Install and play with it
- Report issues
- Add benchmarks results from your machines

Packaging

- Fedora, CentOS - 90% done
- Other Linux distros
- macOS (via macport, brew) - needs libnbd
- FreeBSD - needs libnbd
- Other?

Missing features

- Support any image format supported by qemu-nbd
- Checksum multiple images
- Check image against a checksum file
- Supports extents without qemu-nbd
- Improve CI, only Fedora and CentOS Stream are tested

Integration

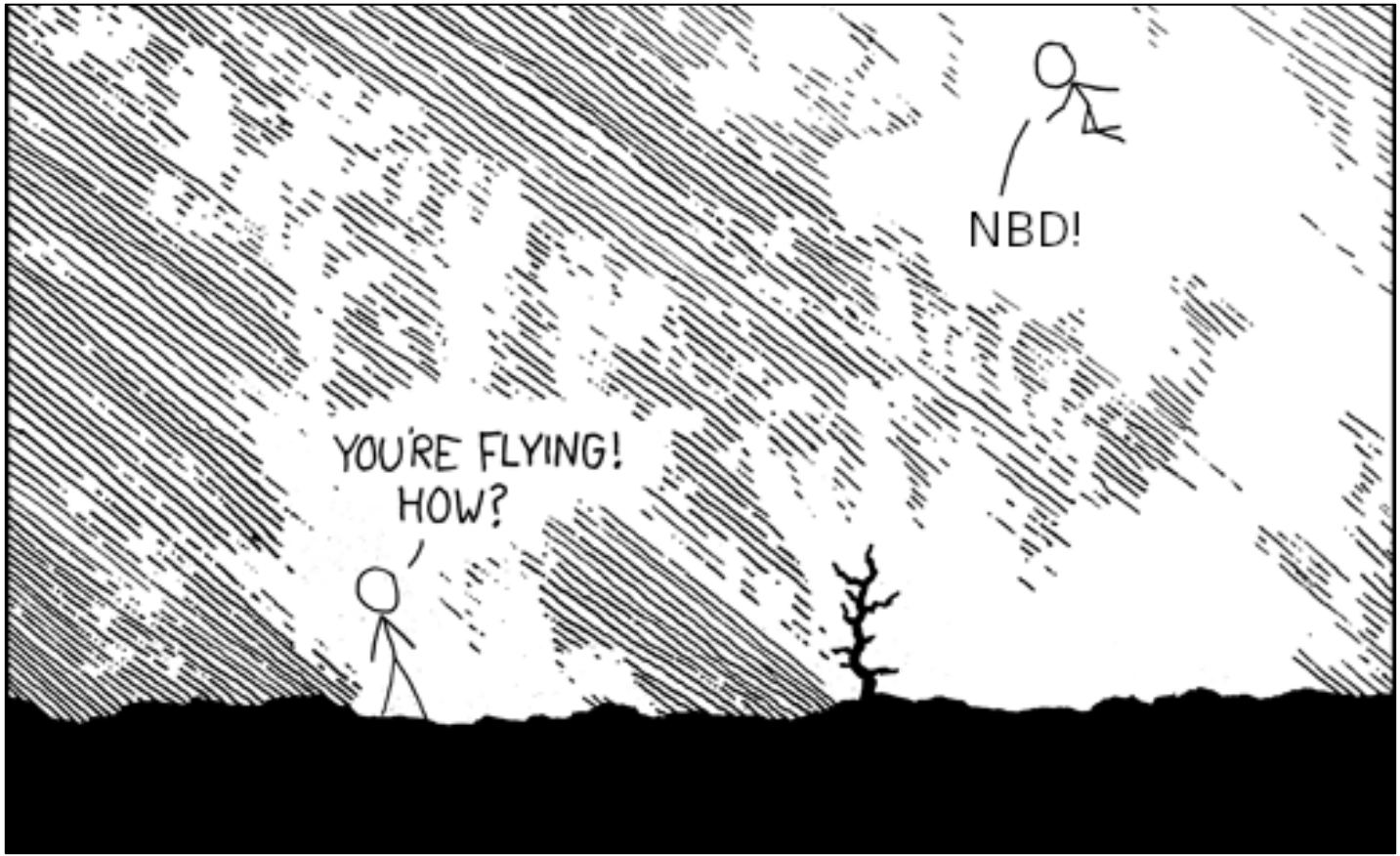
- oVirt - has checksum API using older implementation
- ovirt-stress - using the checksums API to verify incremental backup in oVirt
- KubeVirt?
- proxmox?
- other?

More info

- Project: https://gitlab.com/nirs/blkhash
- Issue tracker: https://gitlab.com/nirs/blkhash/-/issues
- Copr repo: https://copr.fedorainfracloud.org/coprs/nsoffer/blkhash/

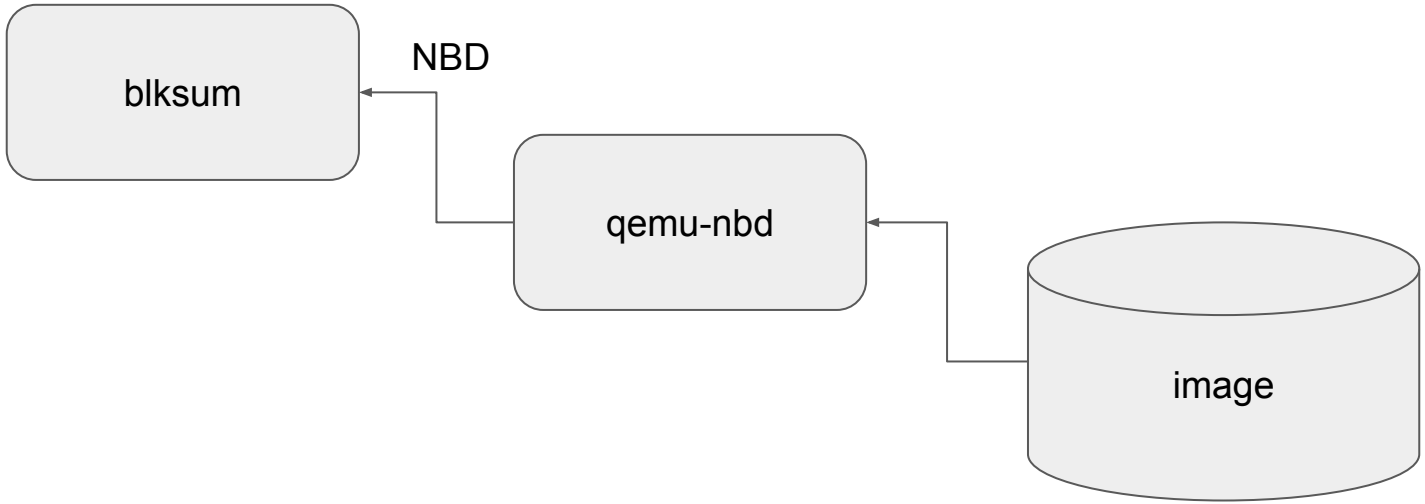

Questions?


Bonus: how it works



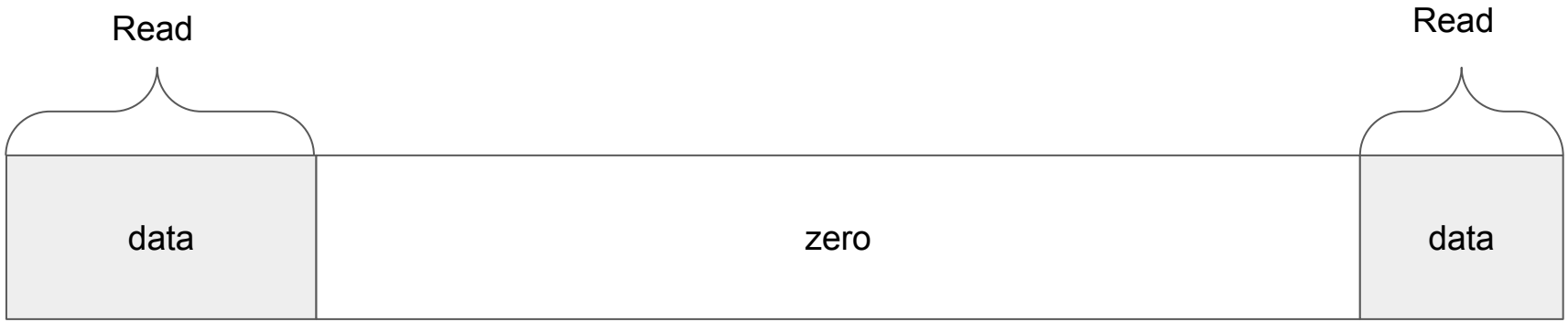
based on https://xkcd.com/353/

Accessing guest data



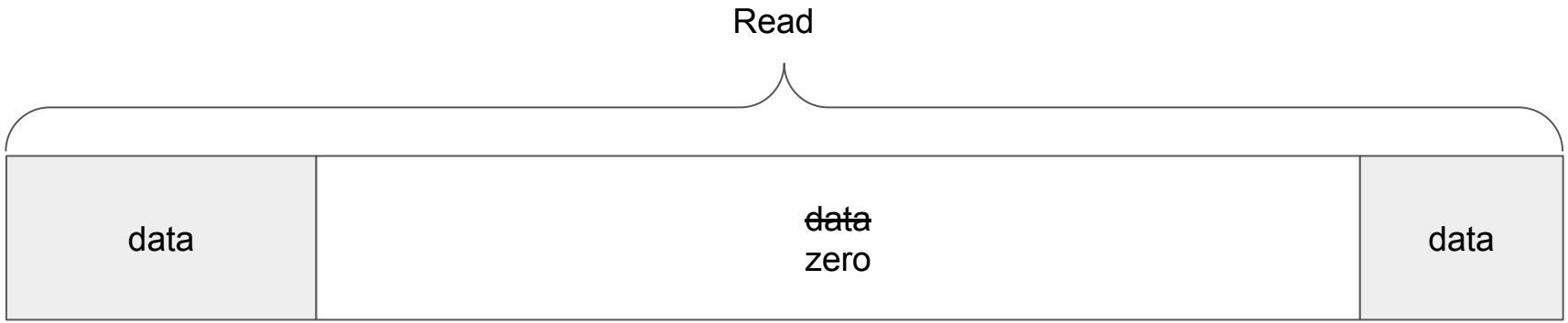
`blksum` runs `qemu-nbd` as child process and access it using `libnbd`.

Read only data extents



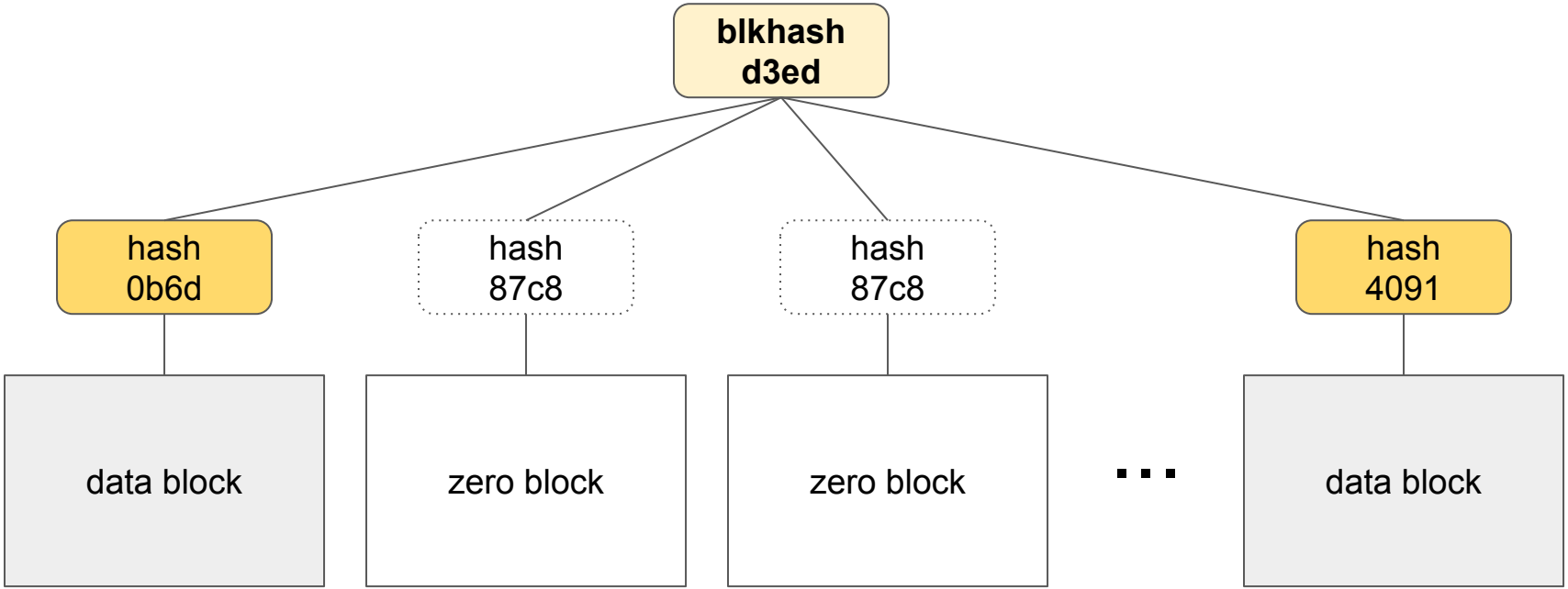
For raw file or qcow2 image we can usually read only the data extents.

Zero detection

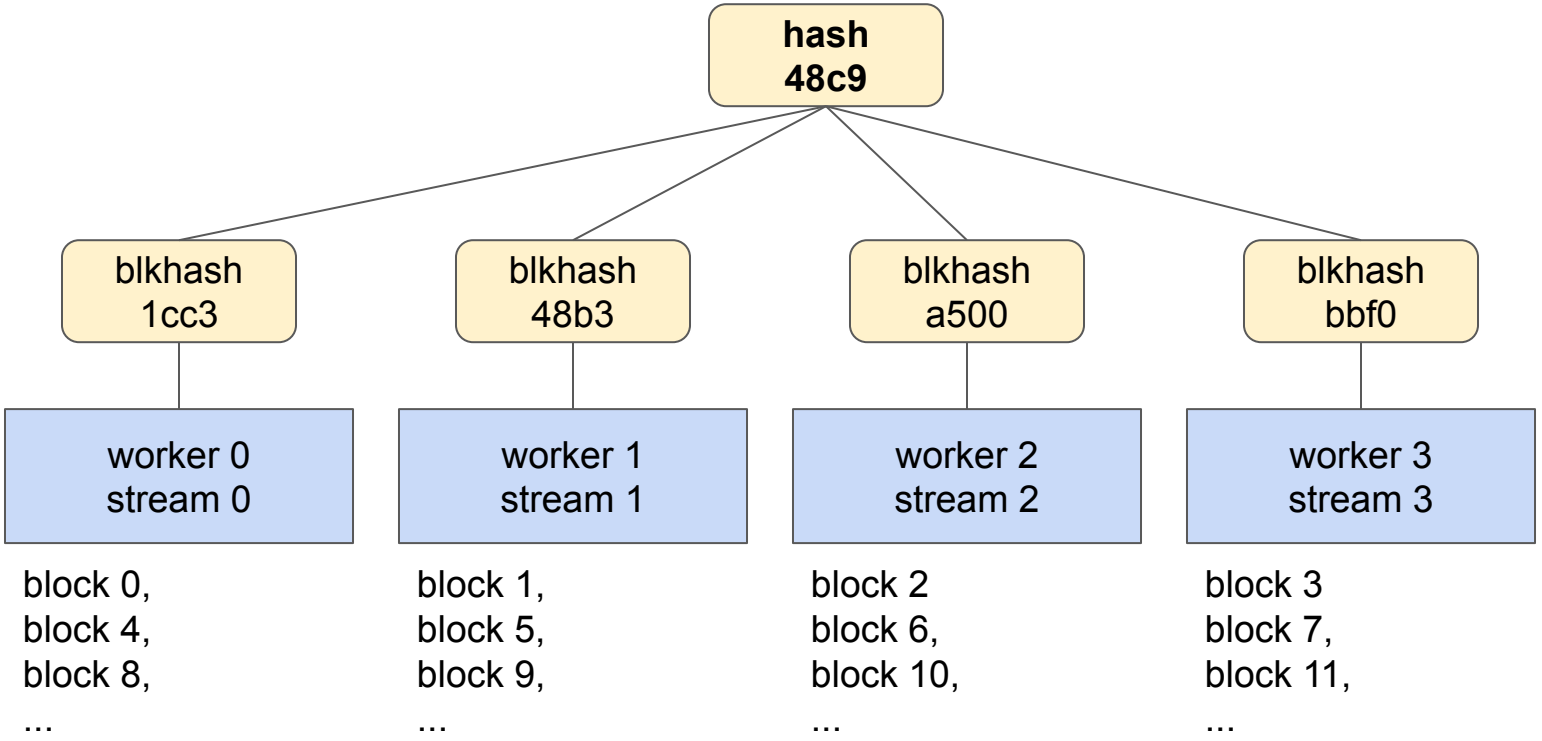


If extents information is not available we read the entire image and detect zeros.

Optimizing zero block hashing



Parallel hashing



The blkhash construction

Image hash:

```
    H( H(stream 0) || H(stream 1) || H(stream 2) || H(stream 3) )
```

Dispatching block to stream:

```
    stream_index == block_index % 4
```

Example with 16 blocks image:

```
    H(stream 0) = H( H(block 0) || H(block 4) || H(block 8)  || H(block 12) )
    H(stream 1) = H( H(block 1) || H(block 5) || H(block 9)  || H(block 13) )
    H(stream 2) = H( H(block 2) || H(block 6) || H(block 10) || H(block 14) )
    H(stream 3) = H( H(block 3) || H(block 7) || H(block 11) || H(block 15) )
```