Fault-injection Testing for Distributed Storage System on Linux

Kyle Zhang 张凯
Cofounder & CTO
SmartX

MAKE IT SIMPLE

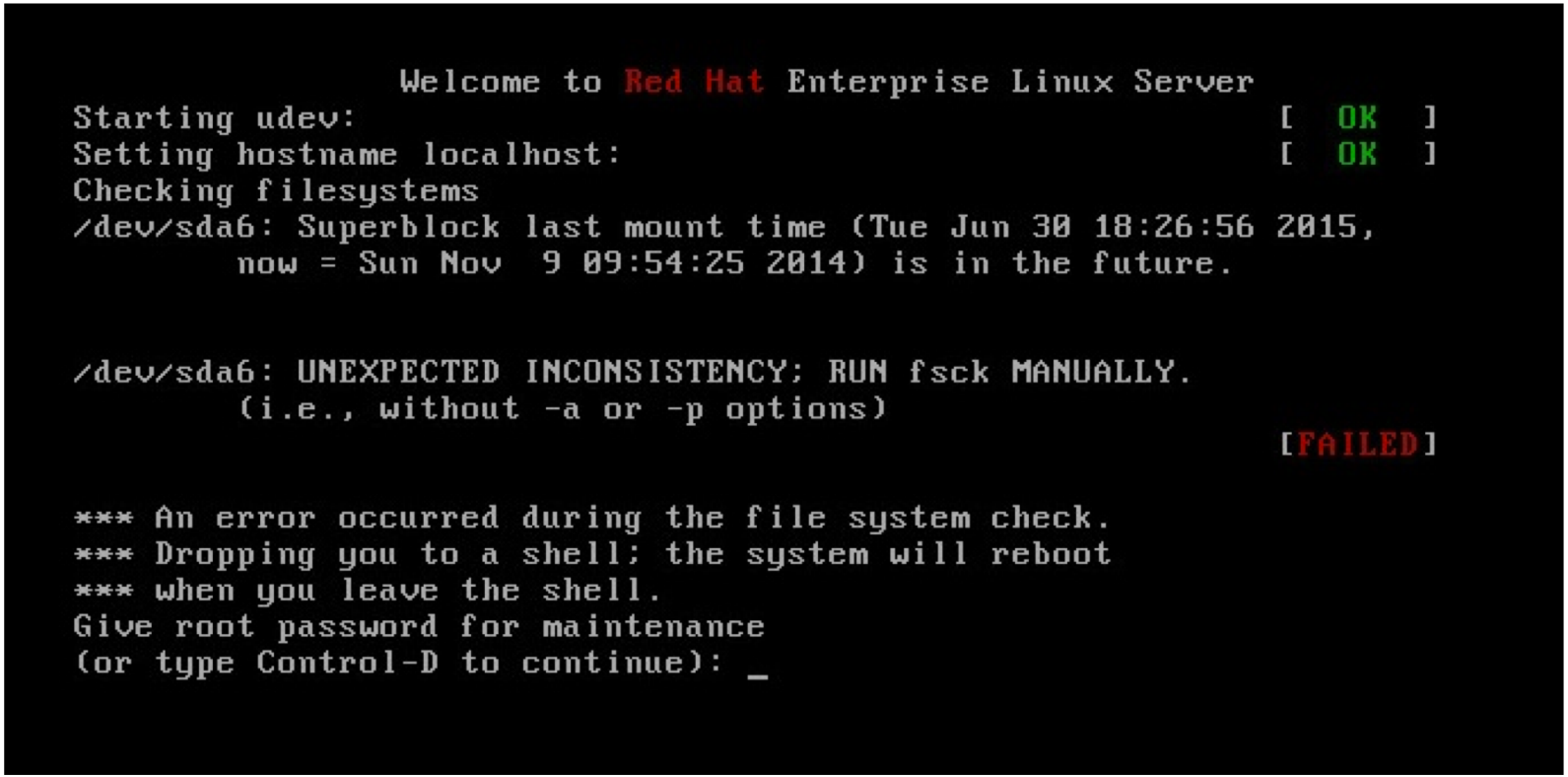
About Me

- Kyle Zhang 张凯
- Computer Science, Tsinghua University
- Software Engineer, Infrastructure Team, Baidu
- Cofounder and CTO, SmartX
- Focused on distributed block storage system

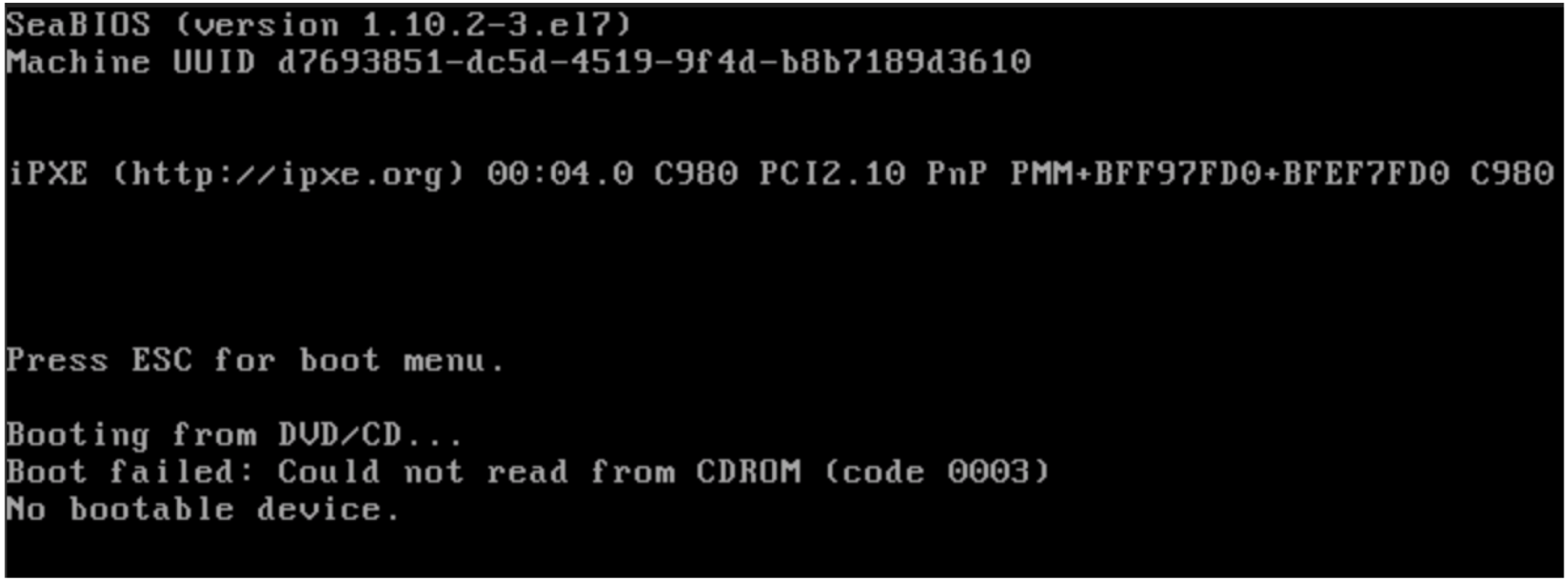
After a storage developer release a new version...

The Terrible Thing...

smartx
MAKE IT SIMPLE



The More Terrible Thing...



The Most Terrible Thing…

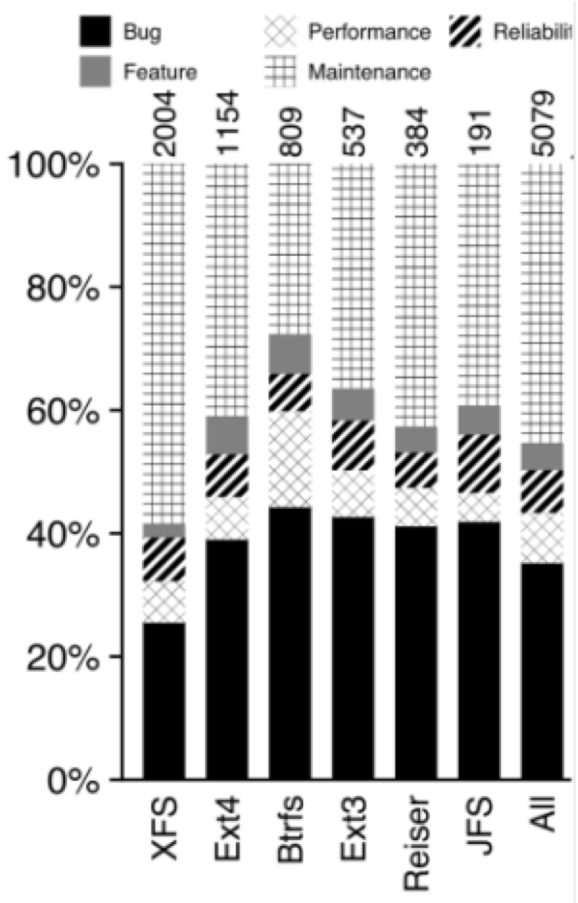
smartx
MAKE IT SIMPLE

“Anything that can go wrong,
will go wrong”

Murphy’s law

A Study of Linux File System Evolution

- 40% patches are for bug fix
- 40% patches are for maintenance



A Study of Linux File System Evolution

Embracing Bugs

smartx
MAKE IT SIMPLE

It is normal that software has bugs.

Our mission is to find out bugs.

Fault-injection Testing

Fault-injection in Physical Way

smartx
MAKE IT SIMPLE

Fault-injection in Physical Way

smartx
MAKE IT SIMPLE

- *Subject*: [PATCH] Btrfs: fix log replay failure after unlink and link combination
- *From*: fdmanana@xxxxxxxxxx
- *Date*: Wed, 28 Feb 2018 15:56:10 +0000

```
From: Filipe Manana <fdmanana@xxxxxxxx>

If we have a file with 2 (or more) hard links in the same directory,
remove one of the hard links, create a new file (or link an existing file)
in the same directory with the name of the removed hard link, and then
finally fsync the new file, we end up with a log that fails to replay,
causing a mount failure.

Example:

  $ mkfs.btrfs -f /dev/sdb
  $ mount /dev/sdb /mnt

  $ mkdir /mnt/testdir
  $ touch /mnt/testdir/foo
  $ ln /mnt/testdir/foo /mnt/testdir/bar

  $ sync

  $ unlink /mnt/testdir/bar
  $ touch /mnt/testdir/bar
  $ xfs_io -c "fsync" /mnt/testdir/bar

  <power failure>

  $ mount /dev/sdb /mnt
  mount: mount(2) failed: /mnt: No such file or directory
```

Fault-injection in Physical Way

smartx
MAKE IT SIMPLE

- **Nice to have**
  - real world failures!
- **Limitations**
  - costly
  - not efficiency
  - hard to automate and reproduce

Fault-injection at Application-level

smartx
MAKE IT SIMPLE

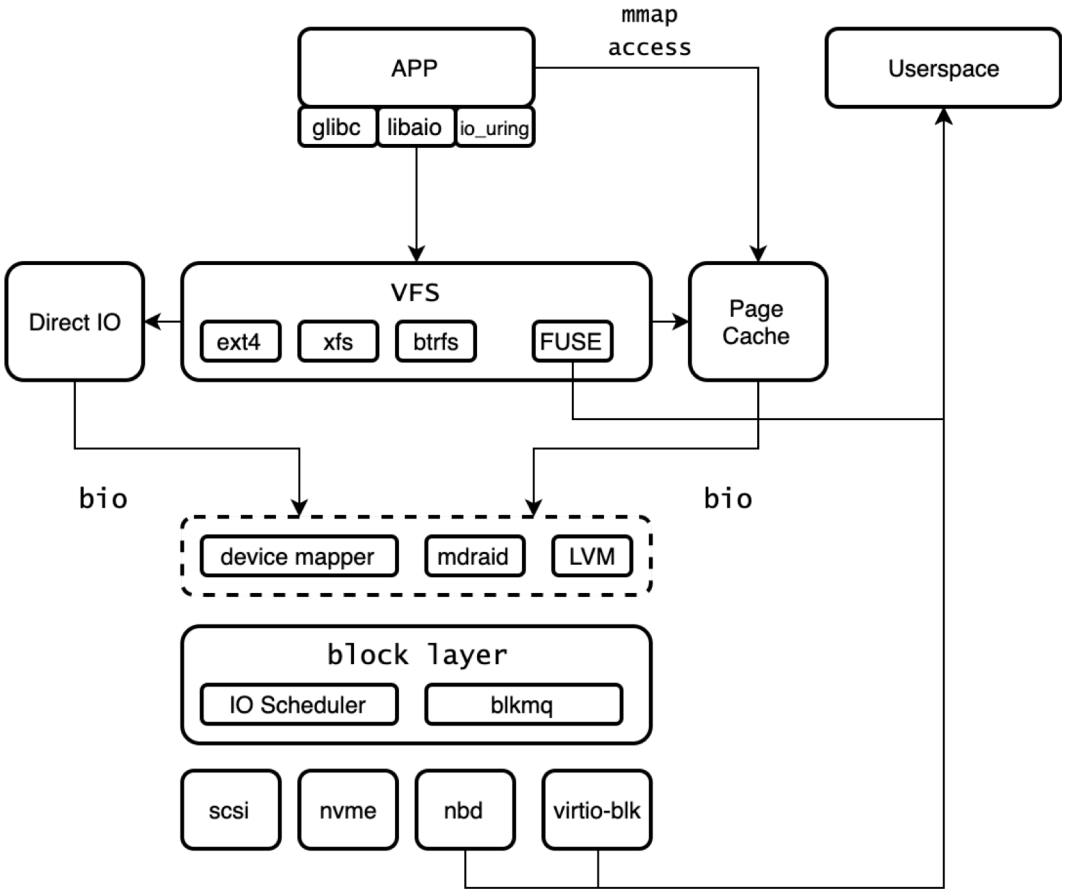
- **Nice to have**
  - easy to automate and reproduce
  - find-grained control
- **Limitations**
  - redundant source codes for different languages/applications
  - hard to cover all cases

Fault-injection in Kernel

smartx
MAKE IT SIMPLE

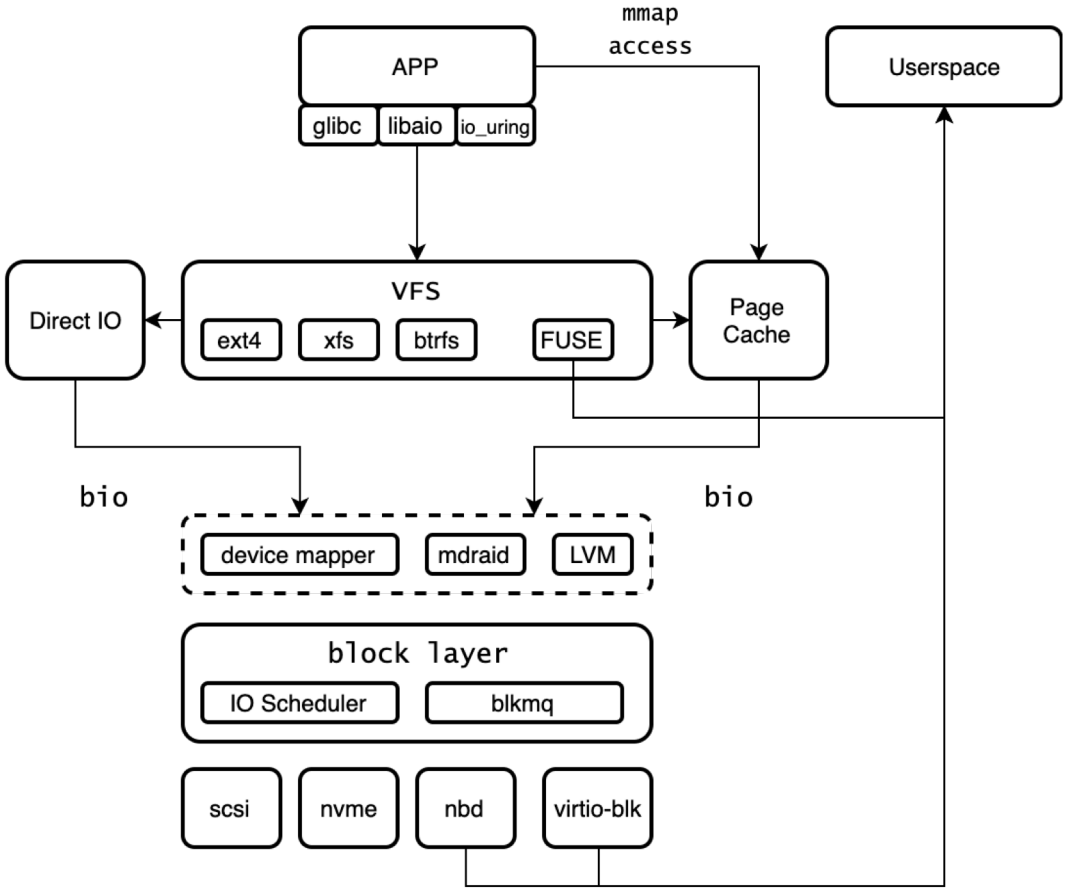
- **Nice to have**
  - easy to automate and reproduce
  - system-wide
  - find-grained control
- **Limitations**
  - depends on what you use

Linux Kernel IO Stack



Where to Inject Failures?

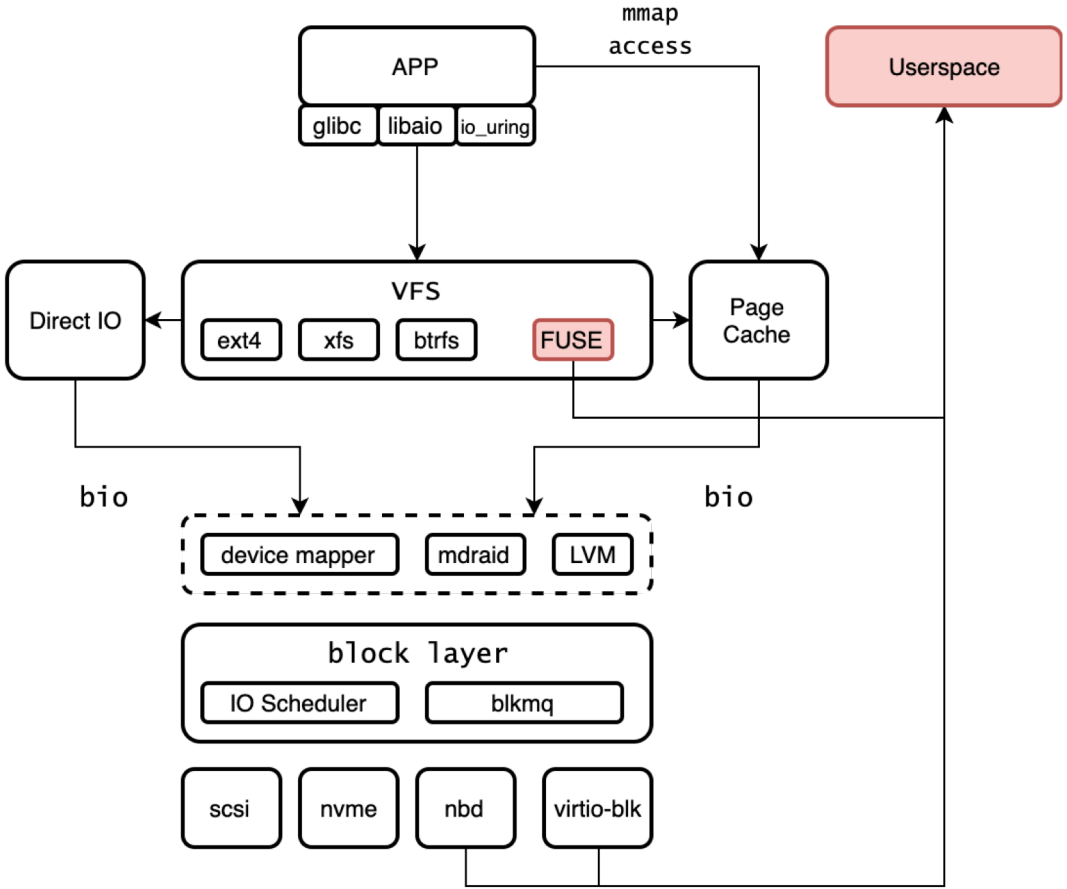
- Don’t modify application source code
- Portable with different kernel versions
- Easy to use
- Fine-grained control
  - fail/delay request
  - probability
  - dynamic configuration
  - mutate data



FUSE

- Hook IO at VFS layer
- Inject failures with userspace code
- Good at portability

- Namazu
  - static config
- CharybdeFS
  - dynamic config with thrift



CharybdeFS

smartx
MAKE IT SIMPLE

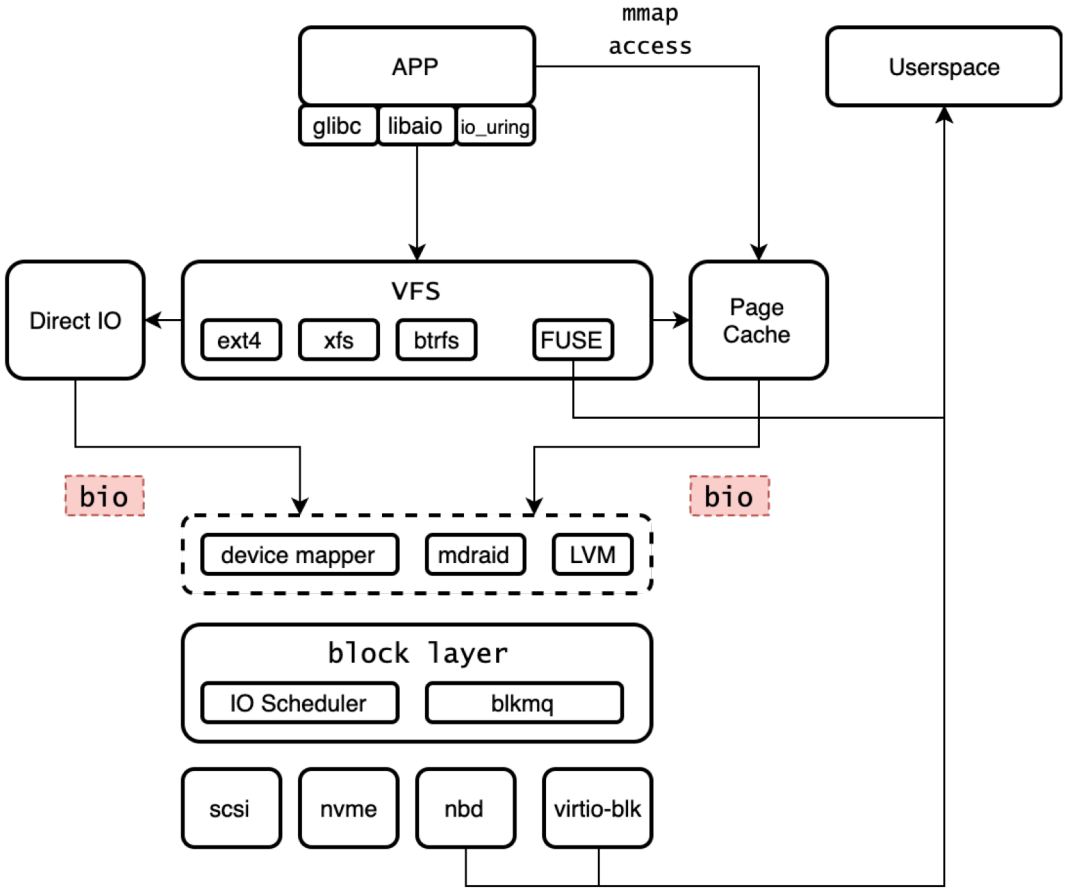
```
> sudo ./charybdefs /mnt/charybde -omodules=subdir,subdir=/xxx
> cd /mnt/charybde && ls

> python cookbook/recipes.py --io-error
Simulating IO error

> ls
ls: cannot open directory .: Input/output error
```

bio

- Fail bio request
  - return error at *generic_make_request()*
  - recompile kernel with *CONFIG_FAIL_MAKE_REQUEST*

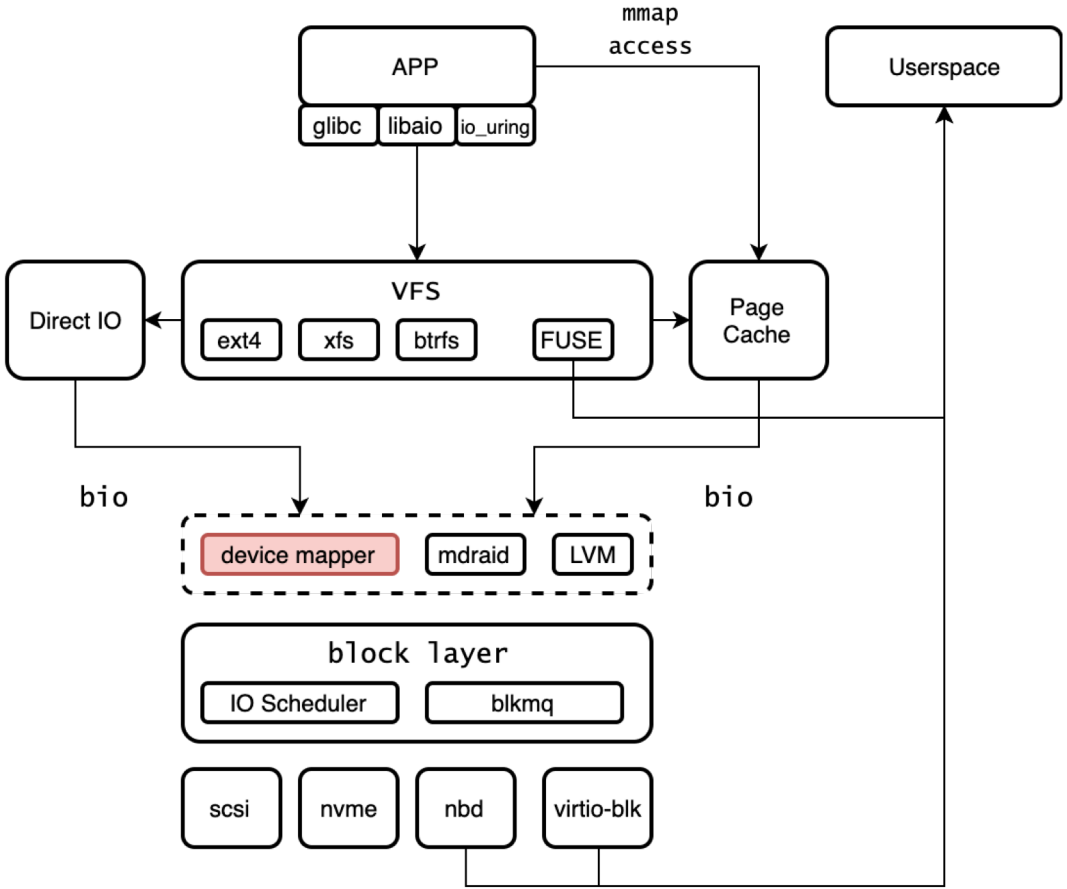
/sys/block/<device>/make-it-fail

- Not good at fine-grained control



Device Mapper

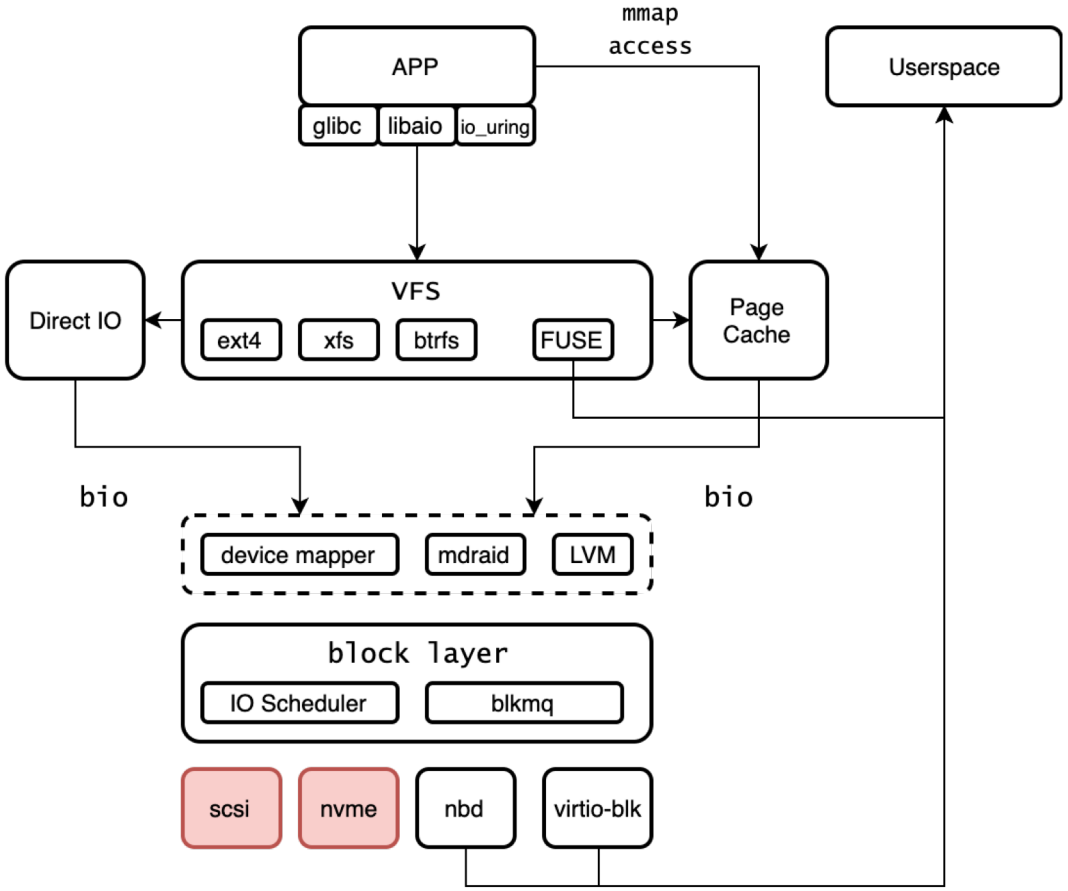
```
> dmsetup create bad_disk << EOF
0    1024 linear /dev/vdb 0
1024 8    error
1032 1016 linear /dev/vdb 1032
EOF

> badblocks -b 4096 /dev/dm-2
128
```



Driver Layer

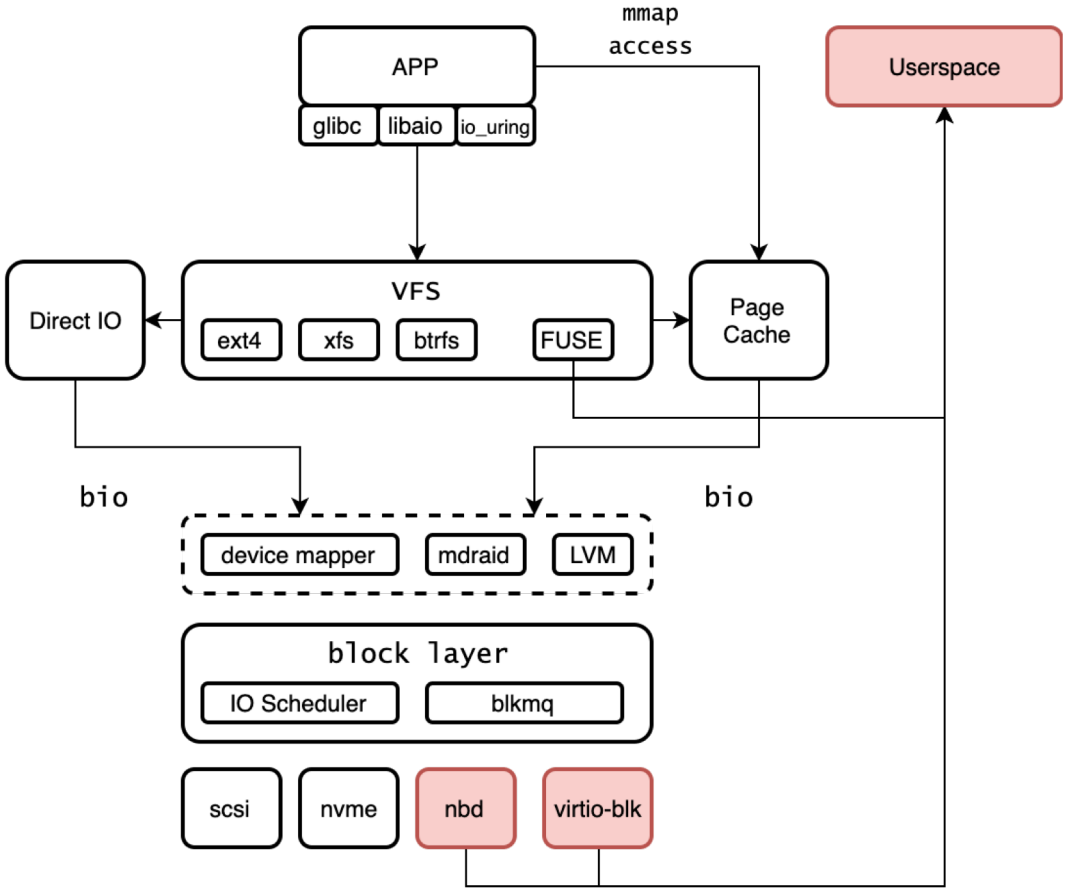
```
echo 1 >
/sys/block/<name>/device/delete
```

- No fine-grained control



NBD/virtio-blk

- Similar to FUSE



nbdkit Examples

smartx
MAKE IT SIMPLE

```
> fallocate -l 10m disk.img
> nbdkit -f --filter=error file disk.img error-pread-rate=10%
> nbd-client -b 512 localhost /dev/nbd0
> badblocks /dev/nbd0
5696
5697
5712
5716
5728
5740
```

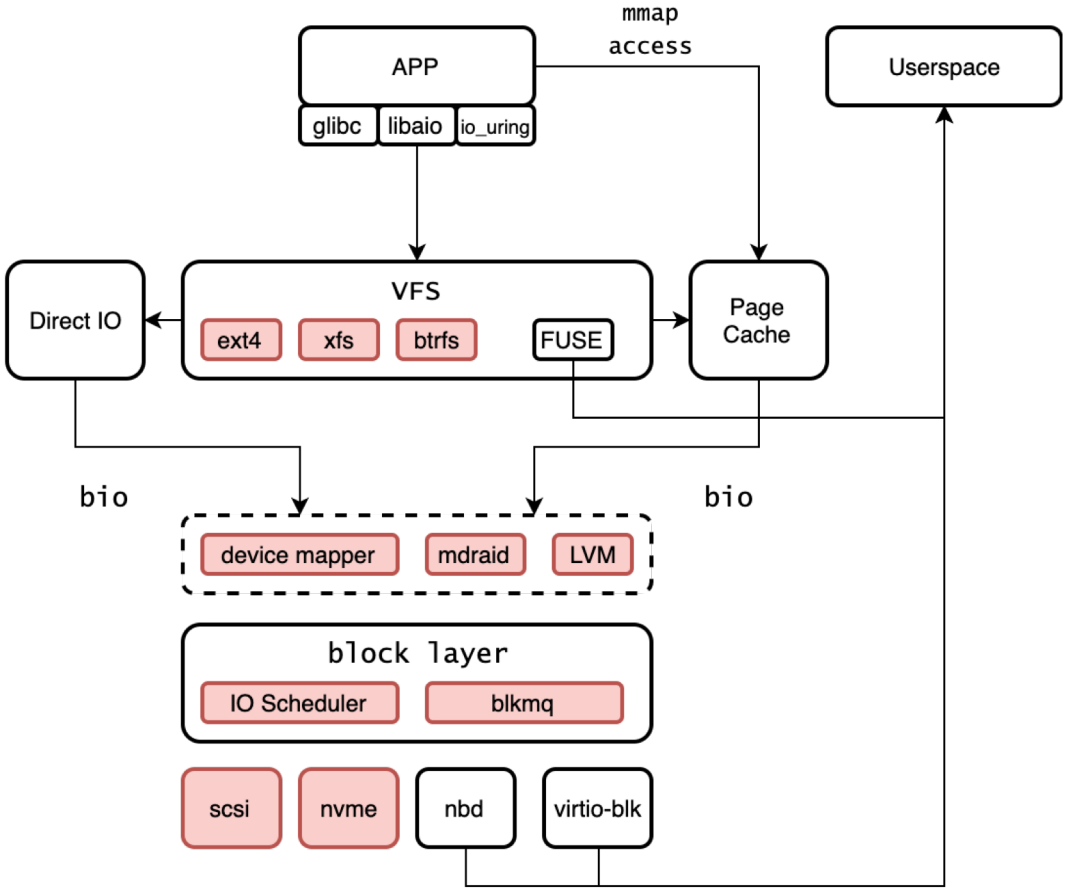
Demo?

```
$ cd /tmp
$ rm -f disk.img && fallocate -l 100m disk.img
$ nbdkit file disk.img

$ mkdir -p /mnt/nbd
$ mkfs.btrfs /dev/ndb0
$ mount /dev/nbd0 /mnt/nbd
$ cd /mnt/nbd
$ touch foo && ln foo bar
$ sync
$ unlink bar && touch bar
$ xfs_io -c "fsync" bar

# simulate power failure
$ killall nbdkit
$ cd .. && umount /mnt/nbd

$ nbdkit file disk.img
$ mount /dev/nbd0 /mnt/nbd
```

SystemTap

- Based on **kprobes**
- Fine-grained control
- Inject from remote

- Not portable with different versions of kernel



SystemTap Example

smartx
MAKE IT SIMPLE

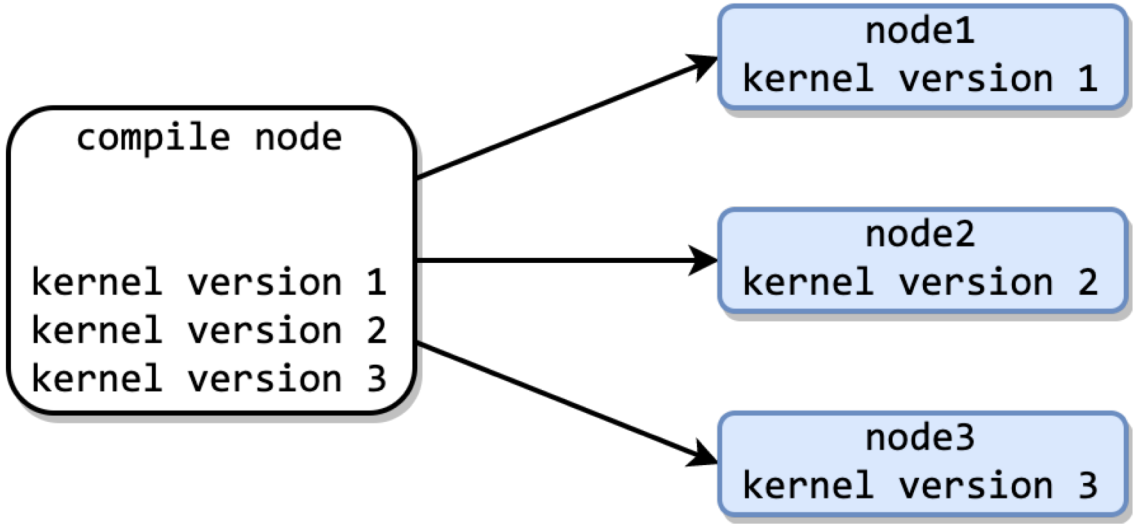
```
#! /usr/bin/stap -g

global target_bdev_name
global target_bdev_part
global iocbs%[10000]

probe begin
{
  target_bdev_name = @1
  target_bdev_part = $2
  printf("\nbegin\n")
}

probe end
{
  printf("\nend\n")
}
```

```
probe kernel.function("do_blockdev_direct_IO@fs/direct-io.c")
{
  if ($bdev->bd_part)
    partno = $bdev->bd_part->partno
  else
    partno = $bdev->bd_disk->first_minor

  bdevname = kernel_string($bdev->bd_disk->disk_name)

  if (target_bdev_name == bdevname && target_bdev_part == partno) {
    iocbs[$iocb] = $iocb
  }
}

probe kernel.function("aio_complete@fs/aio.c")
{
  i = iocbs[$iocb]
  if (i) {
      delete iocbs[$iocb]
      // -5 is EIO
      $res = -5
  }
}
```

SystemTap Example

```
> stap -g --suppress-handler-errors -r 3.10.0-957.21.3.el7.x86_64 \
     --remote=root@<ip> tests/disk_error.stp sda 5

> fio --name=test --filename=/dev/sda5 --ioengine=libaio --direct=1
test: (g=0): rw=read, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=1
fio-2.15
Starting 1 process
fio: io_u error on file /dev/sda5: Input/output error: read offset=0, buflen=4096
fio: pid=24508, err=5/file:io_u.c:1708, func=io_u error, error=Input/output error

> fio --name=test --filename=/dev/sda5 --direct=1
test: (g=0): rw=read, bs=4K-4K/4K-4K/4K-4K, ioengine=psync, iodepth=1
fio-2.15
Starting 1 process
Jobs: 1 (f=1): [R(1)] [0.3% done] [144.9MB/0KB/0KB /s] [37.8K/0/0 iops] [eta 29m:11s]
```

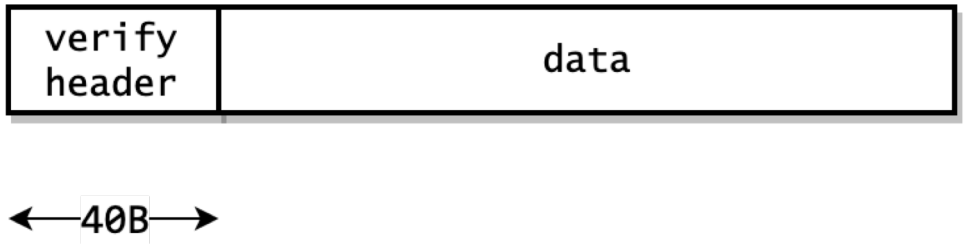
SystemTap Example



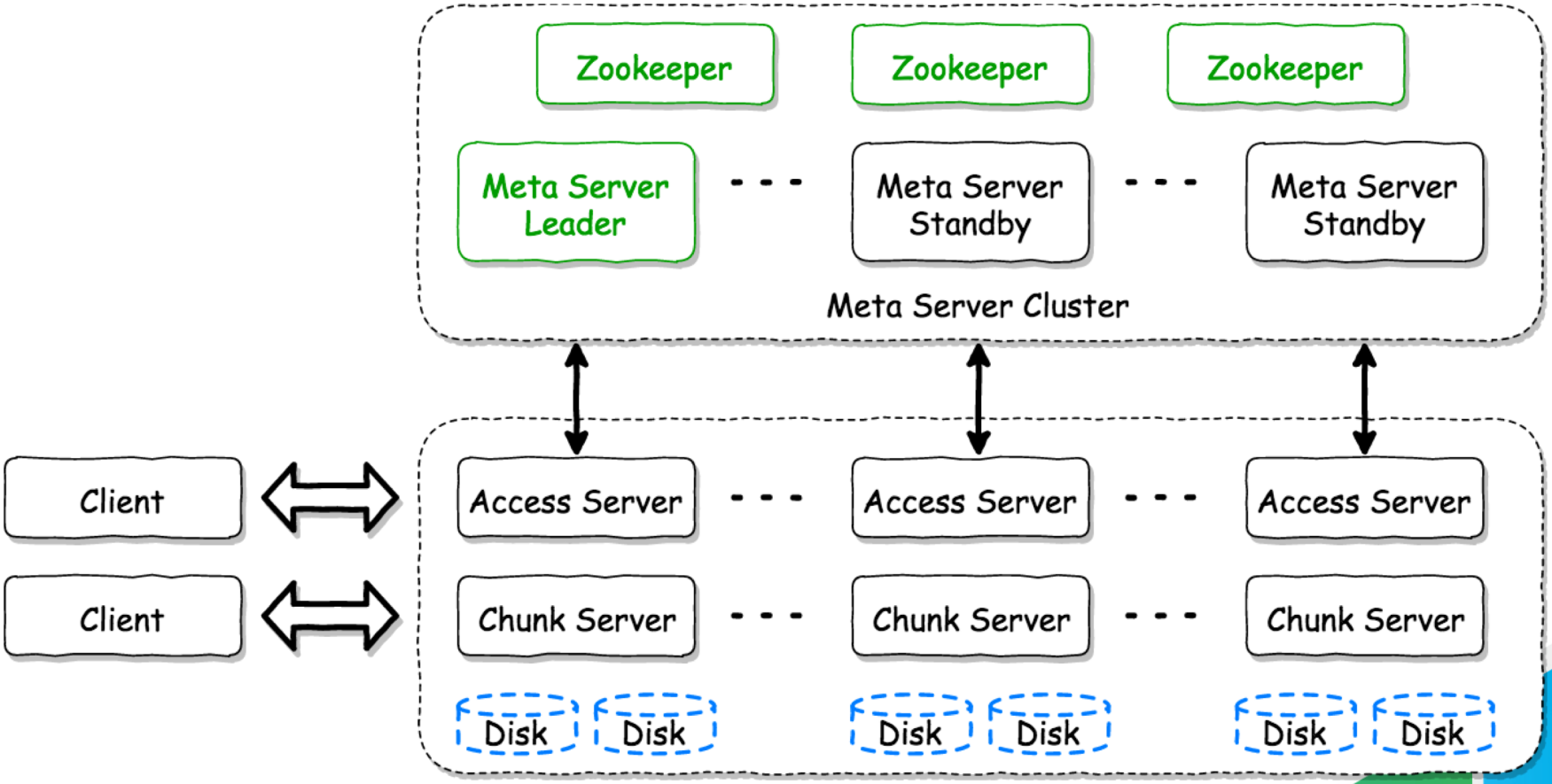
Use remote mode to handle multiple kernel versions

Data Verification

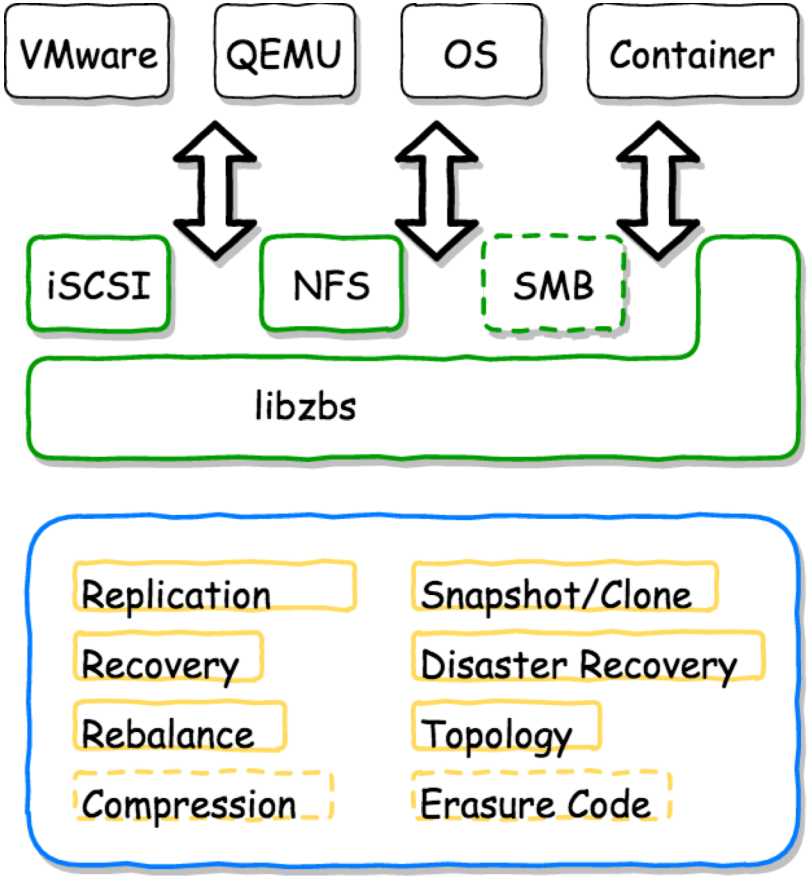
- **Filesystem**
  - xfstest
  - btrfs
- **FIO**
  - verify data
  - libiscsi engine



ZBS Architecture

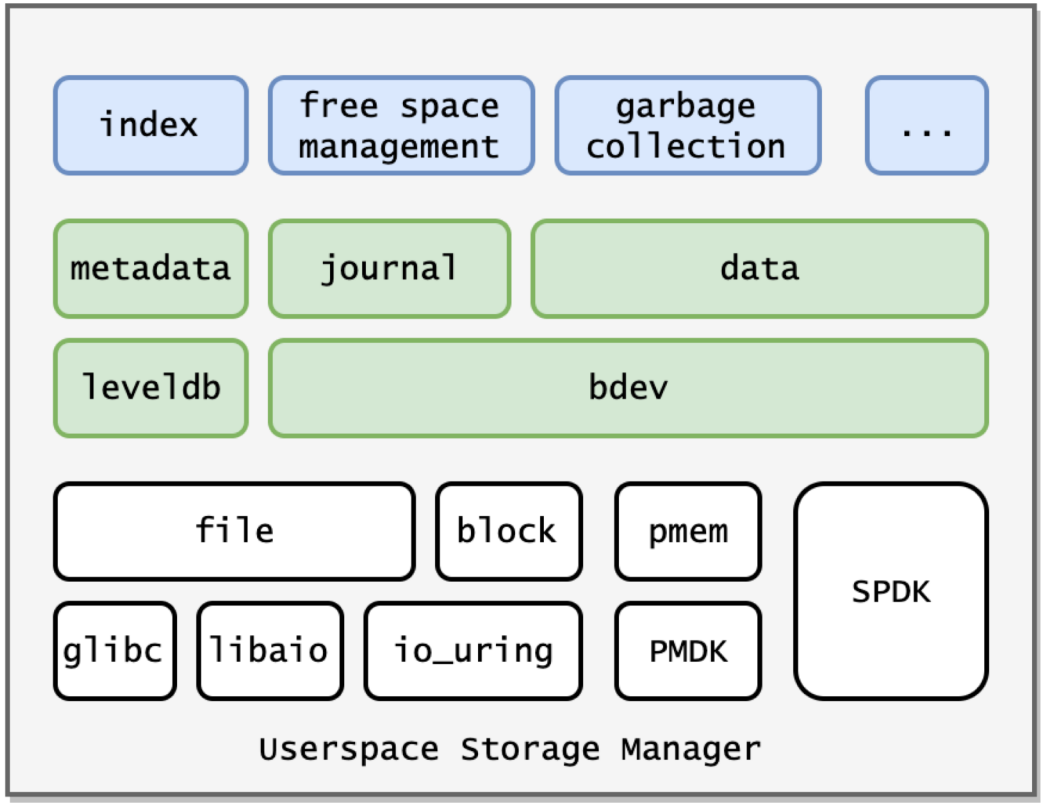


ZBS Interface

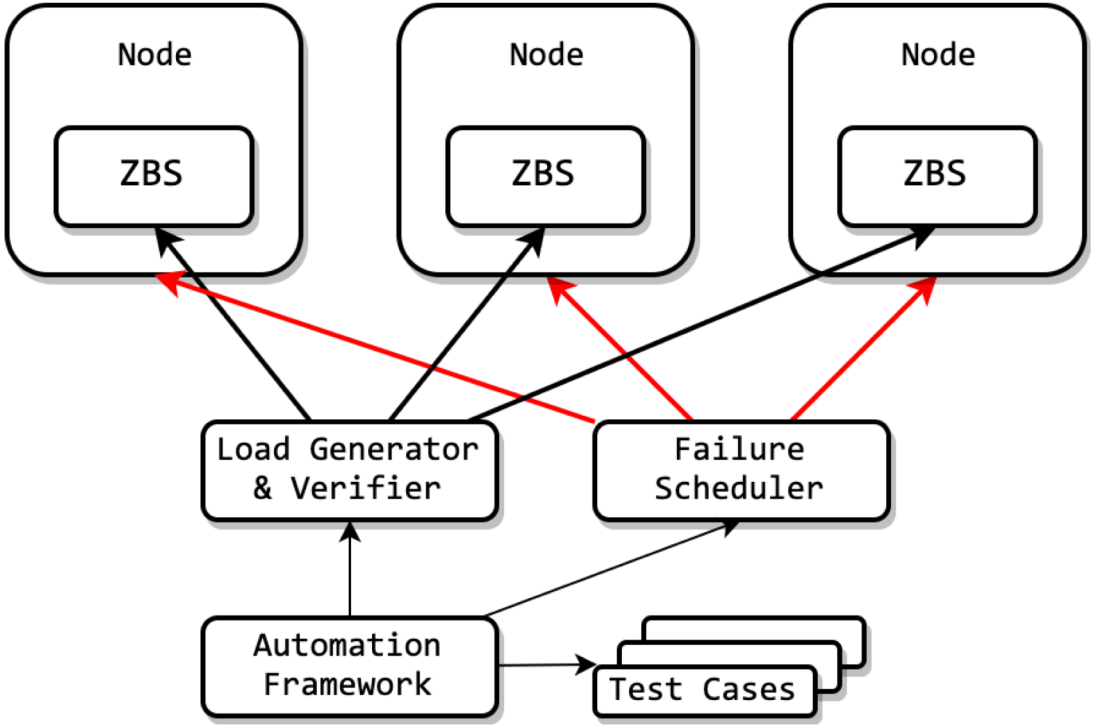


- **libzbs (c library)**
  - Optimized for performance
- **iSCSI/NFS**
  - Based on libzbs
  - Friendly for operating system, hypervisor...

ZBS Local Storage Manager



Fault Injection Framework



Testing ZBS

- Unit test
  - simulate failures with mocked classes
- Longevity test
  - simulate user operations
- Fault-injection test
  - inject failure at driver layer
  - SystemTap
- Dogfood

We are Walking on the Edge of the Cliff

smartx
MAKE IT SIMPLE

Thanks for listening!

smartx
MAKE IT SIMPLE

MAKE IT SIMPLE