DevOps

General technical system administration and devops documentation.

Linux

General Linux tidbits.

Sudo and root

You may see `sudo su -` used instead of `sudo -i` but there are some subtle differences between them. The `sudo su -` command sets up the root environment exactly like a normal login because the `su -` command ignores the settings made by sudo and sets up the environment from scratch. The default configuration of the `sudo -i` command actually sets up some details of the root user's environment differently than a normal login. For example, it sets the PATH environment variable slightly differently. This affects where the shell will look to find commands. You can make `sudo -i` behave more like `su -` by editing `/etc/sudoers` with `visudo`. Find the line

Text Only

```
Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin
```

and replace it with the following two lines:

Text Only

```
Defaults secure_path = /usr/local/bin:/usr/bin
Defaults>root secure_path = /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
```

For most purposes, this is not a major difference. However, for consistency of PATH settings on systems with the default `/etc/sudoers file`, it must be considered.

SSH

Fix SSH permissions

Bash

```
find .ssh/ -type f -exec chmod 600 {} \;; find .ssh/ -type d -exec chmod 700 {} \;; find .ssh/ -type f -name "*.pub" -exec chmod 644 {} \;
```

Virtualization

Check nested virtualization support

Intel:

- `cat /sys/module/kvm_intel/parameters/nested`
- `modinfo kvm_intel | grep -i nested`

AMD:

- `cat /sys/module/kvm_amd/parameters/nested`
- `modinfo kvm_amd | grep -i nested`

Disk

Check if disk is SSD or HDD

Text Only

```
lsblk -d -o name,rota
```

Disk performance testing with FIO

Flexible I/O tester docs fio output explained ArsTechnica fio recommended tests

SINGLE 4KIB RANDOM WRITE PROCESS

This is a single process doing random 4K writes. This is where the pain really, really lives; it's basically the worst possible thing you can ask a disk to do. Where this happens most frequently in real life: copying home directories and dotfiles, manipulating email stuff, some database operations, source code trees.

Bash

```

```

```
fio --filename=sdX --name=random-write --ioengine=posixaio --rw=randwrite --
bs=4k --size=4g --numjobs=1 --iodepth=1 --runtime=60 --time_based --end_fsync=1
```

16 PARALLEL 64KIB RANDOM WRITE PROCESSES

This time, we're creating 16 separate 256MB files (still totaling 4GB, when all put together) and we're issuing 64KB blocksized random write operations. We're doing it with sixteen separate processes running in parallel, and we're queuing up to 16 simultaneous asynchronous ops before we pause and wait for the OS to start acknowledging their receipt. This is a pretty decent approximation of a significantly busy system. It's not doing any one particularly nasty thing—like running a database engine or copying tons of dotfiles from a user's home directory—but it is coping with a bunch of applications doing moderately demanding stuff all at once.

This is also a pretty good, slightly pessimistic approximation of a busy, multi-user system like a NAS, which needs to handle multiple 1MB operations simultaneously for different users. If several people or processes are trying to read or write big files (photos, movies, whatever) at once, the OS tries to feed them all data simultaneously. This pretty quickly devolves down to a pattern of multiple random small block access. So in addition to "busy desktop with lots of apps," think "busy fileserver with several people actively using it."

Bash

```
1  fio --filename=sdX --name=random-write --ioengine=posixaio --rw=randwrite --
   bs=64k --size=256m --numjobs=16 --iodepth=16 --runtime=60 --time_based --
   end_fsync=1
```

SINGLE 1MIB RANDOM WRITE PROCESS

This is pretty close to the best-case scenario for a real-world system doing real-world things. No, it's not quite as fast as a single, truly contiguous write... but the 1MiB blocksize is large enough that it's quite close. Besides, if literally any other disk activity is requested simultaneously with a contiguous write, the "contiguous" write devolves to this level of performance pretty much instantly, so this is a much more realistic test of the upper end of storage performance on a typical system.

You'll see some kooky fluctuations on SSDs when doing this test. This is largely due to the SSD's firmware having better luck or worse luck at any given time, when it's trying to queue operations so that it can write across all physical media stripes cleanly at once. Rust disks will tend to provide a much more consistent, though typically lower, throughput across the run.

You can also see SSD performance fall off a cliff here if you exhaust an onboard write cache—TLC and QLC drives tend to have small write cache areas made of much faster MLC or SLC media. Once those get exhausted, the disk has to drop to writing directly to the much slower TLC/QLC media where the data eventually lands. This is the major difference between, for example, Samsung EVO and Pro SSDs—the EVOs have slow TLC media with a fast MLC cache, where the Pros use the higher-performance, higher-longevity MLC media throughout the entire SSD.

Bash

```
fio --filename=sdX --name=random-write --ioengine=posixaio --rw=randwrite --bs=1m --size=16g --numjobs=1 --iodepth=1 --runtime=60 --time_based --end_fsync=1
```