

How to maximise SSD performance with Linux

Article Number: 676 | Rating: 5/5 from 2 votes | Last Updated: Fri, May 15, 2020 11:47 AM

Now that SSDs have moved into the realm of affordability there's little reason not to use one for your next PC if you're not already using one. But Linux as with Windows has spent decades being tuned for spinning platter drives and while performance is noticeably increased when using an SSD under Linux there are a number of tweaks and filesystem changes you can do to make the most of your SSDs.

Filesystem layer

The first of these is easy to do and can both improve performance and more importantly the longevity of your SSD by reducing unnecessary writes (keeping in mind the memory used in SSDs has limited write-rewrite cycles).

By default many distributions including Ubuntu use the 'relatime' flag for updating file metadata when files are accessed but you're unlikely to care about last access times. Additionally Linux supports TRIM with Ext4. TRIM is important for maintaining the performance of an SSD over time as files are added deleted and changed and lets the SSD know which blocks can be safely cleared. No distributions currently enable it by default but it's simple to do by adding the 'discard' flag to any mounted SSDs.

To make all these changes open up a terminal and run:

```
sudo nano -w /etc/fstab
```

Then for all SSD devices in your system remove 'relatime' if present and add 'noatimenodiratimediscard' so it looks something like this:

```
/dev/sda / ext4 noatime,nodiratime,discard,errors=remount-ro 0 1
```

Scheduler

The scheduler helps organise reads and writes in the I/O queue to maximise performance. The default scheduler in the Linux kernel is CFQ (Completely Fair Queuing) which is designed with the rotational latencies of spinning platter drives in mind. So while it works well for standard hard drives it doesn't work so well when it comes to SSDs.

Fortunately the kernel comes with some other schedulers to play with and here the deadline and NOOP schedulers are ideal. Both are basic schedulers that guarantee fast turnaround of I/O requests. NOOP is basically no scheduler at all it's a basic FIFO (First In First Out) queue whereas deadline does some sorting to guarantee read requests take priority over write which is useful if you want to guarantee read responsiveness under heavy writes.

Changing scheduler is easy and even better — you can do it on a per-device basis if you have a mixed SSD and spinning platter hard drive system using deadline for SSDs and CFQ for traditional drives. As CFQ is the default change SSDs to use deadline by opening up a terminal and running:

```
sudo nano -w /etc/rc.local
```

Then add the following line for each SSD in your system:

```
echo deadline >/sys/block/sda/queue/scheduler
```

Changing 'sda' to 'sdb' and so on for each SSD device. If you only have SSDs in your system you can instead set the global scheduler policy to apply to all devices at boot time.

For Ubuntu and other distributions using GRUB2 edit the /etc/default/grub file and add 'deadline' to the GRUB_CMDLINE_LINUX_DEFAULT line like so:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash elevator=deadline"
```

Then run 'sudo update-grub2'.

Swap and tmp

Linux is pretty good at only using swap if it really needs to but even so if you're installing to an SSD and you have a mechanical hard drive in your system be sure to put the swap partition on it instead of the SSD. If you've already installed Linux and allocated a swap partition on the SSD you can simply set aside a partition on a spinning platter drive and edit your /etc/fstab swap entry to point to it instead. For example assuming /dev/sdb is a normal hard drive:

```
/dev/sdb2 none swap sw 0 0
```

And reboot or alternatively issue 'swapoff -a && swapon -a' to update on the fly. If you have a purely SSD system and lots of memory you can disable swap almost entirely. Keep a swap partition available but add the following to your /etc/rc.local file:

```
echo 0 > /proc/sys/vm/swappiness
```

Linux won't use swap at all unless physical memory is completely filled.

Next to reduce unnecessary writes to the SSD move the temp directories into a ram disk using the 'tmpfs' filesystem which dynamically expands and shrinks as needed.

In your `/etc/fstab` add the following:

```
tmpfs /tmp tmpfs defaults,noatime,mode=1777 0 0
```

```
tmpfs /var/spool tmpfs defaults,noatime,mode=1777 0 0
```

```
tmpfs /var/tmp tmpfs defaults,noatime,mode=1777 0 0
```

If you don't mind losing log files between boots and unless you're running a server you can probably live without them also add:

```
tmpfs /var/log tmpfs defaults,noatime,mode=0755 0 0
```

Considering that even everyday applications generate a lot of log files it's not a bad idea to do this.

Applications

Any applications that write excessively to a hard drive are also candidates for moving data. Browsers are a fine example of this — the browser cache is nice but it'd work just as well from a spinning platter drive and save your SSD from thousands of writes a day that don't make a huge difference to you.

To move the cache in Firefox in the browser type 'about:config' right-click anywhere and select New → String and add 'browser.cache.disk.parent_directory'. Edit the variable and point it to a directory on a non-SSD drive or if you don't mind losing the cache between boots and you're using the tweaks above point it to /tmp for a super-fast memory cache.

Moving the cache in Chrome is a little harder. The directory is hardcoded but you can use symbolic links to point it to a directory on another drive or to /tmp. You'll find the cache under ~/.cache/chromium. You could also redirect the entire .cache directory as many programs use this for caching data.

Have a look at other applications you use and see if you can redirect any unnecessary writes as well.

Partition alignment

Finally there's partition alignment but this can only be done with a clean system before you install either Linux or Windows. Partition alignment is critical for SSDs as being memory-based devices data is written and read in blocks known as pages. When partitions aren't aligned the block size of filesystem writes isn't aligned to the block size of the SSD causing extra overhead as data crosses page boundaries.

Aligning partitions is simply a matter of ensuring the first partition starts on a clean 1MB boundary from the start of the disk ensuring whatever block size the filesystem uses will align with the block size of the SSD (which can also vary). If you create partitions using Windows 7 on an empty drive it will start partitions at the 1MB boundary automatically.

In Linux simply run 'fdisk -cu (device)' on the drive you want to partition press 'n' for new partition 'p' for primary and enter a start sector of at least 2048. The general

rule is that the starting sector must be divisible by 512 but to cater for all variations of SSD page size and filesystem block size 2048 is a good idea (and equates to 1MB).

Posted - Fri, May 15, 2020 11:47 AM. This article has been viewed 9715 times.

Online URL: <http://kb.ictbanking.net/article.php?id=676>