# SSDOptimization - Debian Wiki

15-19 minutes

---

Translation(s): none

---

This page is about optimal set up of an SSD (Solid State Drive). This page should be kept clean enough for beginners to get the most basic idea.

⚠ Note that some of the configuration improvements listed below happen automatically today for new installations.

Contents

## WARNING

Some firmware versions on some SSD models have bugs that result in data corruption when used in certain ways. For this reason the Linux ata driver maintains a blacklist of certain things it shouldn't do on certain drive/firmware combinations. If you have a blacklisted controller/drive combination, you are at risk until a newer kernel avoids the problem.

In particular, many drives, including Samsung, Micron, Crucial have problems with discard/TRIM. Also see 790520

Make sure you review the latest version of that file for your model, and if present then make sure it's also in the version of the kernel you intend to run or find some other way to avoid the problems (like not using discard/TRIM, or a particular firmware version).

## Basics

- Use a reasonably recent Linux (kernel)
- Use the latest firmware for the SSD
    - Use a command like: "`sudo smartctl -a /dev/sda`" to check for issues.
    - Update firmware as needed.
- Use the ext4 filesystem (the most mature filesystem) unless you have reason not to.
- Have enough DRAM required to operate without swap space under normal workloads.
    - You need a swap partition that is larger than your DRAM to save all the DRAM content securely for hibernation.
    - If your SSD size is too small for your DRAM size, think about placing your swap on the larger classic HDD.
    - If you are planning on doing a huge amount of writes (more than 40-50 GB per day), it is advised to avoid SSDs that use TLC NAND.

These may also help:

- https://wiki.archlinux.org/index.php/Solid_State_Drives
- http://siduction.org/index.php?module=news&func=display&sid=78
- http://forums.debian.net/viewtopic.php?f=16&t=76921
- http://wiki.ubuntuusers.de/SSD (German)

## Partitions and Alignment

You should consider to use the Multi HDD/SSD Partition Scheme to keep variable and bulk data on the HDD(s) and establish a fallback redundancy, if the system also has a HDD available (internal or external spinning disk).

All tools should automatically align filesystems and partitions to the 4096 byte page size. This is one of the most important optimization aspects. Here are good links for this subject:

- Block Alignment issues and analysis on Debian (not specific to SSD)
- Advanced Format
- Aligning SSD Partitions (Linux magazine)
- Aligning filesystems to an SSD’s erase block size] by Ted T'so (kernel FS developer)
- http://blog.nuclex-games.com/2009/12/aligning-an-ssd-on-linux/
- http://lifehacker.com/5837769/make-sure-your-partitions-are-correctly-aligned-for-optimal-solid-state-drive-performance

## Mounting SSD filesystems

The performance of SSDs is also influenced by filesystem mounting options:

- Add the "noatime" (or the default "relatime") mount option in /etc/fstab, to disable (or significantly

reduce) disk writes whenever a file is read.
    - This improves filesystem read performance for both SSDs and HDDs.
- First read the WARNING at the top of this page.

  **If desirable**, enable the "discard" filesystem options for automatic online TRIM.

    - Set "issue_discards" option in /etc/lvm/lvm.conf for LVM if you want LVM to discard on lvremove.

  See lvm.conf(5).
    - Set "discard" option in /etc/crypttab for dm-crypt.
    - Enable weekly trim.

  http://blog.neutrino.es/2013/howto-properly-activate-trim-for-your-ssd-on-linux-fstrim-lvm-and-dmcrypt/

  ```
  sudo systemctl enable fstrim.timer
  sudo systemctl start fstrim.timer
  ```

    - Alternatively, and often not recommended: Set "discard" mount option in /etc/fstab for the ext4 filesystem, swap partition, Btrfs, etc.

  See mount(8).

- The "discard" options **is not needed** if your SSD has enough overprovisioning (spare space) or you leave (unpartitioned) free space on the SSD.

    - See http://www.spinics.net/lists/raid/msg40866.html

- The "discard" options with on-disk-cryptography (like dm-crypt) have **drawbacks with security/ cryptography**.

    - See crypttab(5).

Example for dm-crypt's /etc/crypttab:

```
#<target name>    <source device>            <key file>  <options>
var  UUID=01234567-89ab-cdef-0123-456789abcdef  none  luks,discard
```

After changing filesystem options, update settings in all initramfs images:

```
$ sudo update-initramfs -u -k all
```

Alternative to setting the "discard" options is to set up an offline-trim cronjob that runs `time fstrim -v` on the ssd mountpoints periodically (but the WARNING at the top of the page is relevant here too). For older versions software raid (md device layer) that lack trim support, you could use something like mdtrim (https://github.com/Cyberax/mdtrim/).

Alternative to LVM's "issue_discards" is to `blkdiscard` on the LV before lvremove, or afterwards on a temporarily created LV with `lvcreate -l100%FREE` to trim all unused LVM space.

## Reduction of SSD write frequency via RAMDISK

Use of RAMDISK can stop constantly changing files from hitting on the SSD (it may hit SSD via swap). RAMDISK configuration may be performed via

- enable system-managed ramdisks Of course, this will not improve writing to the SSD if your system (filesystem holding / or /tmp) is not located on it.
  - using systemd:

```
sudo cp /usr/share/systemd/tmp.mount /etc/systemd/system/
sudo systemctl enable tmp.mount
```

  - using systemV, set RAMTMP, RAMRUN and RAMLOCK to "yes" in /etc/default/tmpfs.
- /etc/fstab: line such as "tmpfs /tmp tmpfs noatime,nosuid 0 0"
- Optionally, make system only flush data to the disk every 10 minutes or more:
  - Kernel settings like the "dirty_buffer_ratio" etc.

  may only be available as non-path/mount specific global settings.
  - Mount option "commit=600" in /etc/fstab.

  See mount(8).

  - Or better, use pm-utils (Debian BTS #659260), tlp, or laptop-mode-tools (also optimizes read buffers) to configure the laptop-mode even under AC operation.

  ⚠ Attention: Increasing the flushing interval from the default 5 seconds (maybe even until proper shutdown) leaves your data much more vulnerable in case of lock-ups or power failures, and seems to be a global setting.

Enabling RAMTMP may cause some (broken) applications to run out of temporary write disk space. In this case, setting TMPDIR environment variable for those programs pointing to a writable disk space should fix their situation. It might as well cause other unwanted side effects if the disk space occupied in your /tmp is high (e.g. unexpected swapping).

Please note files in /tmp are wiped out upon reboot unless /etc/default/rcS is set to something other than TMPTIME=0 (if not set, 0 is the default value).

⚠ You might also want to be aware of Summary: Moving /tmp to tmpfs makes it useless

## Persistent RAMDISK

As an advanced option, you may consider to use persistent RAMDISK (dedicated read/write RAM buffer that gets synced periodically and on startup/shutdown) with anything-sync-daemon or goanysync set up:

- /home (synced to work-data-fs raid only once a day?), you only risk settings the true work in /home/*/work-data is on a dedicated raid
- /home/*/work-data/volatile (synced more frequently, once per hour?)
- /home/*/Downloads (synced to bulk-data-fs once a day?)
- /var completely if supported (syncing once a day? avoids spin-ups and allows to save /var also to SSD), at least set this up for
  - /var/log if supported

    - /var/cache/apt/archives
        - Configure apt to delete package files after installing, to minimize the data to sync.

Options to having logs copied into RAM:

- http://www.debian-administration.org/articles/661
- http://www.tremende.com/ramlog
- https://github.com/graysky2/anything-sync-daemon (if it supports this)
- https://github.com/wor/goanysync

If /home is not on a persistent ramdisk, use profile-sync-daemon to have the browser database and cache copied into RAM during uptime (http://ubuntuforums.org/showthread.php?t=1921800 https://github.com/graysky2/profile-sync-daemon)

- /home/*/<browser-cache-and-profiles> (synced to root-fs or home-fs)

Further improvement: Patch anything-sync-daemon or goanysync to use a (copy-on-write) union filesystem mount (e.g. http://aufs.sourceforge.net) to keep changes in RAM and only save to SSD on unmount/shutdown (aubrsync), instead of copying all data to RAM and having to sync it all back.

## Low-Latency IO-Scheduler

This step is not necessary for SSDs using the NVMe protocol instead of SATA, since NVMe uses the blk-mq module instead of the traditional I/O scheduler.

The default I/O scheduler queues data to minimize seeks on HDDs, which is not necessary for SSDs. Thus, use the "deadline" scheduler that just ensures bulk transactions won't slow down small transactions: Install sysfsutils and

```
  # echo "block/sdX/queue/scheduler = deadline" >> /etc/sysfs.conf
```

(adjust sdX to match your SSD) reboot or

```
  # echo deadline > /sys/block/sdX/queue/scheduler
```

In systems with different drive types you can adjust settings with a udev rule (create /etc/udev/rules.d/60-ssd-scheduler.rules):

```
# set deadline scheduler for non-rotating disks
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="0",
ATTR{queue/scheduler}="deadline"
```

To check if the kernel knows about SSDs try:

```
  # for f in /sys/block/sd?/queue/rotational; do printf "$f is "; cat $f;
done
```

```
/sys/block/sda/queue/rotational is 1
/sys/block/sdb/queue/rotational is 1
/sys/block/sdc/queue/rotational is 0   <=== Only this is SSD!
```

```
  $ grep . /sys/block/sd?/queue/rotational
/sys/block/sda/queue/rotational:1
/sys/block/sdb/queue/rotational:1
/sys/block/sdc/queue/rotational:0   <=== Only this is SSD!
```

To switch scheduler manually issue:

```
# echo deadline > /sys/block/$YOURDRIVE/queue/scheduler
```

To see results:

```
# for f in /sys/block/sd?/queue/scheduler; do printf "$f is "; cat $f; done
/sys/block/sda/queue/scheduler is noop deadline [cfq]
/sys/block/sdb/queue/scheduler is noop deadline [cfq]
/sys/block/sdc/queue/scheduler is noop [deadline] cfq  <== That is!
```

```
$ grep . /sys/block/sd?/queue/scheduler
/sys/block/sda/queue/scheduler:noop deadline [cfq]
/sys/block/sdb/queue/scheduler:noop deadline [cfq]
/sys/block/sdc/queue/scheduler:noop [deadline] cfq  <== That is!
```

The dumb "noop" scheduler may be a little faster in benchmarks that max out the throughput, but this scheduler causes noticeable delays for other tasks while large file transfers are in progress.

## /etc/fstab example

Here is an example of /etc/fstab (read WARNING at top about discard before copying from this example)

```
# /etc/fstab: static file system information.
#
# Use 'vol_id --uuid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
### SSD: discard,noatime
### match battery operation default for commit JOURNAL_COMMIT_TIME_AC in
Add files in /etc/pm/config.d/*
/dev/mapper/goofy-root /               ext4
discard,noatime,commit=600,errors=remount-ro 0       1
# /boot was on /dev/sda1 during installation
```

```
UUID=709cbe4a-80c1-46cb-8bb1-dbce3059d1f7 /boot           ext4
discard,noatime,commit=600,defaults        0       2
### SSD: discard
/dev/mapper/goofy-swap none            swap    sw,discard             0
0
/dev/mapper/goofy-chroot /srv/chroot        btrfs    ssd,discard,noatime 0
2
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0       0
```

## /etc/lvm/lvm.conf example

(read WARNING at top about discard before copying from this example)

You do not need this setting for filesystem discard; it is for LVM to discard on lvremove etc. If you prefer to do this manually, use blkdiscard on to-be-removed LV.

```
...
# This section allows you to configure which block devices should
# be used by the LVM system.
devices {
...
    # Issue discards to a logical volumes's underlying physical volume(s)
when
    # the logical volume is no longer using the physical volumes' space
(e.g.
    # lvremove, lvreduce, etc).  Discards inform the storage that a region
is
    # no longer in use.  Storage that supports discards advertise the
protocol
    # specific way discards should be issued by the kernel (TRIM, UNMAP, or
    # WRITE SAME with UNMAP bit set).  Not all storage will support or
benefit
    # from discards but SSDs and thinly provisioned LUNs generally do.  If
set
    # to 1, discards will only be issued if both the storage and kernel
provide
    # support.
    # 1 enables; 0 disables.
    #issue_discards = 0
    issue_discards = 1
}
...
```

## Smaller system with SSD

See

- Acer Aspire One: "Reducing Disk Access for laptops with SSDs"

## Adding vm.swappiness in sysctl for the kernel

As per http://evolvisforge.blog.tarent.de/archives/68 which has come on p.d.o. as well as on 29/07/2013 the following needs to be added in /etc/sysctl.conf.d/ for better longevity of the disc:

```
# /etc/sysctl.d/local.conf

#vm.swappiness=0
vm.swappiness=1
```

Setting vm.swappiness=0 is more aggressive but may cause out-of-memory events. http://java.dzone.com/articles/OOM-relation-to-swappiness

**Note:** together with other settings vm.swappiness _may_ be a considerable enhancement to I/O performance; worth reading: http://rudd-o.com/linux-and-free-software/tales-from-responsivenessland-why-linux-feels-slow-and-how-to-fix-that

#### vm.swappiness=0 considered harmful?

I just had a multi-minutes long system freeze (ping still worked, nc to port 22 still opened a connection but sshd was silent) due to lack of "avail Mem" in top(8) output; with vm.swappiness=0 the kernel used a mere ⅓ MiB of swap out of 3 GiB I have. This is the second time this week this happened; Firefox and IntelliJ together ate up all RAM.

From more reading about SSDs, I’m now considering vm.swappiness=10 instead; some swapping is, probably, still, overall, better. --ThorstenGlaser

## reduce swapping activities with ZRam

Using ZRam (aka compcache) it is possible to use data compression on the contents of the system memory (RAM). This effectively trades in some CPU cycles for the ability to stuff a lot more into the available system memory and thereby reduces the need for swapping out memory pages to the SSD. It uses specialised high speed data compression algorithms that are especially light on the CPU and yet are said to give typically about 1:3 compression on this type of content. Because compressing is typically faster than writing to swap devices it also brings performance improvements to systems without excessive amounts of physical memory. See the respective article on how to activate it.

## Upgrading the Firmware

(read WARNING at top before making firmware changes)

Some problems might occur due to firmware bugs. Therefore, use tools like smartctl (terminal) or GSmartControl (GUI) to check for warnings to update the firmware. Normally the manufacturer provides these proprietary firmware images packed within update tools, available as bootable CD ISO images.

In case they used FreeDOS started by syslinux, instead of booting from CD, you can extract the (floppy) image inside the ISO and create a new entry in your GRUB2 boot menu.

For the Crucial m4 firmware update, the *syslinux.cfg* looks like this:

```
LABEL default
        KERNEL memdisk
        append initrd=boot2880.img floppy raw
```

Copying the files *memdisk* and *boot2880.img* to the boot partition allows them to be started from GRUB2, by adding an entry in */etc/grub.d/40_custom*:

```
# assuming an ext2 boot partition on (hd0,5) -- compare to your other GRUB2
entries
menuentry "m4 firmware update" {
  insmod ext2
  linux16 (hd0,5)/isos/m4memdisk floppy raw
  initrd16 (hd0,5)/isos/boot2880.img
}
```

Afterwards run `update-grub` to apply the changes. Reboot and select the created boot entry, then follow the firmware update menu. After completion, restart the computer.

There are some articles for specific SSD models:

- Micron 1100 Series

Updating firmware for other models is documented on the Arch Linux SSD wiki page.

This is a list of SSD related bug reports. It is probably better to come up with a user tag to use instead of listing them all here.

- 600523
- 603735
- 648868
- 683698
- 717313
- 790520
- 889668

---