SSD Optimization

Kernel SSD Awareness

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!
```

Partitions and Alignment

Since Debian Wheezy, all tools should automatically align filesystems and partitions to the 4096 byte page size. This is one of the most important optimization aspects. Alignment can be checked with the Parted command, which has an align-check build in.

```
# parted /dev/sda
align-check opt n (n is the partition you want to check)
```

Mounting SSD Filesystems

relatime is a default mount option and is much better than atime. The former requires a write for the first read after a write, the latter requires a write for every read. But with noatime each read is free of a write. Therefore add **noatime, nodiratime** to the mount options.

```
# nano /etc/fstab

UUID=b2fa2xxx-670f-4d44-becc-d9xxxxxa41 /               ext4
noatime,nodiratime,errors=remount-ro 0       1
```

TRIM

Continues TRIM

The real reason Ubuntu doesn’t TRIM SSDs by default is because the Linux kernel’s implementation of TRIM is slow and results in poor performance in normal use. Windows sends a TRIM command each time it deletes a file, telling the drive to immediately delete the bits of the file. Linux supports this when file systems are mounted with the “discard” option. However, Debian, Ubuntu — and other distributions — don’t do this by default for performance reasons. Continuous TRIM is not the most preferred way to issue TRIM commands in Linux. Continuous TRIM is enabled by the discard option for

a mount in /etc/fstab: enables continuous TRIM in device operations:

```
/dev/sda1  /           ext4  defaults,discard   0  1
```

Periodic trim

Check if the fstrim service is already existing in /usr/lib/systemd/service. If not, create the service and timer files with the content below:

```
# nano /usr/lib/systemd/system/fstrim.service
-------------------------------------------

[Unit]
Description=Discard unused blocks on filesystems from /etc/fstab
Documentation=man:fstrim(8)

[Service]
Type=oneshot
ExecStart=/sbin/fstrim -Av
```

```
# nano /usr/lib/systemd/system/fstrim.timer
-----------------------------------------

[Unit]
Description=Discard unused blocks once a week
Documentation=man:fstrim

[Timer]
OnCalendar=weekly
AccuracySec=1h
Persistent=true

[Install]
WantedBy=timers.target
```

Start the timer and check if timer is activated properly:

```
# systemctl enable fstrim.timer
# systemctl start fstrim.timer
# systemctl list-timers
```

Ramdisks

Add ramdisks for temporary data files in /etc/fstab.

```
#nano /etc/fstab
```

```
tmpfs  /tmp         tmpfs    defaults,noatime,mode=1777   0   0
tmpfs  /var/log     tmpfs    defaults,noatime,mode=0755   0   0
tmpfs  /var/spool   tmpfs    defaults,noatime,mode=1777   0   0
tmpfs  /var/tmp     tmpfs    defaults,noatime,mode=1777   0   0
tmpfs  /var/log/apt tmpfs    defaults,noatime   0   0
```

Swappiness

The default value of vm.swappiness is 60 and represents the percentage of the free memory before activating swap. The lower the value, the less swapping is used and the more memory pages are kept in physical memory.These values are defined:

- 0: swap is disable
- 1: minimum amount of swapping without disabling it entirely
- 10: recommended value to improve performance when sufficient memory exists in a system
- 100: aggressive swapping

Get current value

With the following 2 commands the current value of vm.swappiness can be retrieved:

```
# sysctl vm.swappiness
or
# cat /proc/sys/vm/swappiness
```

Set new value

With the following 2 commands the current value of vm.swappiness can be set:

```
# echo 10 > /proc/sys/vm/swappiness
or
# sysctl -w vm.swappiness=10
```

To set the value permanently, open the file **/etc/sysctl.conf** as an administrative user and add the following line:

```
# nano /etc/sysctl.conf
---------------------------------
Add line to the bottom of the file:

vm.swappiness = 10
```

Turn off Journaling in EXT4

Check current status

First thing's first, we can confirm that our ext4 partition is running a journal with:

```
sudo dumpe2fs /dev/sdaX | grep has_journal
```

If nothing is returned journaling is off. If something is returned, like: “Filesystem features: has_journal ext_attr.…” then journaling is on.

Disable journaling

Disabling journaling is rather easy, the only drag is that to make structural changes to a filesystem, the filesystem cannot be mounted with read/write privileges. So, run a live cd and hit

```
sudo tune2fs -O ^has_journal /dev/sdaX
```

And it's done. Now when you boot, the change will be noted and the disk will be checked for errors. When the system is finally up we can run this again to confirm that in fact ext4 is running without a journal

```
sudo dumpe2fs /dev/sdaX | grep has_journal
```

should now return nothing. With this quick fix, no more constant IO peaks. BTW, one good way to format ext4 from scratch, W/O journalling, onto a device natively-named /dev/sda1? is to specifically issue the command

```
mke2fs -t ext4 -O ^has_journal -cv /dev/sda1
```

IO-Scheduler

This step is not necessary for SSDs using the NVMe protocol instead of SATA, which bypass the traditional I/O scheduler and use the blk-mq module instead. An I/O scheduler decides the order in which applications are given access to the SSD for writes and reads. In Linux, the default I/O scheduler in Linux is called CFQ, Completely Fair Queuing. CFQ generally works well, but it is not perfect for SSDs. Its default behavior is to give drive access on a first come - first serve basis. Sometimes overall system performance can be enhanced by using a different scheduling order. Also, there are schedulers which cause less wear on solid state drives. Two of the better I/O schedulers for SSDs are noop and **deadline**, and they are faster than CFQ when an application is writing large files.

Query current scheduler mode

To see the current scheduler in use for a disk on your system (and alternative I/O schedulers), use the “cat” command to view the scheduler file. The active scheduler will be in brackets:

```
# cat /sys/block/sda/queue/scheduler
```

```
noop [deadline] cfq
```

Set proper scheduler mode

If the scheduler mode is not deadline, we need to change it. To change the I/O scheduler of a disk to deadline (temporarily - until shutdown or reboot), use the “echo” command to write “deadline” into the scheduler file.

```
echo deadline > /sys/block/sdX/queue/scheduler (with X is the proper drive letter).
```

Firefox

Save Interval

The Firefox browser writes between 300 kb and 2 MB every 15 seconds to a recovery file to recover unexpectedly terminated sessions. The solution is to increase that interval. Recommends setting it to 30 minutes (1800000). You can adjust the value in **about:config** after the **browser.sessionstore.interval** parameter. Same phenomenon with other browsers like Chrome: it even writes 24 gigabytes.

Move Firefox cache to ram

Moving the cache into RAM enhances the browser's speed and reduces writes to the SSD. Be aware that each user on the computer has their own Firefox configuration. Repeat the following steps for all user accounts on the system. Open Firefox, type **about:config** in the address bar. In an empty space on the screen, right-click and create a new String: **browser.cache.disk.parent_directory** Set the new string value to: **/tmp/firefox**