5 TWEAKS TO MAKE AFTER INSTALLING AN SSD DRIVE

First, you may be wondering how I came to add a solid-state drive (SSD) to my existing notebook computer. If not, click here to skip to the recommendations.
This past holiday season, I decided to spend some money on myself and update my computer hardware. I wanted the speed of an solid-state drive (SSD) and lots of RAM, the weight of an ultrabook and the long battery life afforded by the latest mobile processor technology. After toying with replacing my two year old notebook with a thin, light, sexy ultrabook with Linux pre-installed, I decided to upgrade my existing computer instead. What I realized is that, with a couple of upgrades, my existing notebook computer could match most of the specs of what pass for ultrabooks these days. I would have to give up a 6 hour battery of course, and my i5 processor wouldn't have the capabilities of the newest mobile processors, but those were almost the only tradoffs.

During my research for a wholesale replacement, boasting a 6 hour battery life and weighing-in at a mere 3.25 pounds, ZaReason's UltraLap 430 easily beat the competition. Maxed-out, I could get a 14.1" ultrabook with an i5-3317U processor, 16GB RAM, 512GB SSD and a one year warranty for about $1,500. The runner-up was the 14.1" System76 Lemur Ultra with an i5-310M processor, 16GB RAM, a 512GB SSD drive and a 1 year warranty for under $1,200. What I realized was that the System76, and most of the other "ultrabooks" with Linux pre-installed were about the same weight as my current computer, 4.5 pounds. If I was going to spend over $1,000 on an ultrabook, I wanted it to be an ultra-lightweight ultrabook with ultra-long battery life. It would be the UltraLap 430 or nothing. So now it was time to compare replacing, with upgrading, my existing hardware.

My 14.1" HP Pavilion dm4-1063cl is just out of warranty after a little over 2 years of use. With a meager 4GB of RAM and with the battery already upgraded a year ago to a 9-cell version, I am getting a respectable 2 hours of battery life. It's i5-M430 processor clocks in at an also respectable maximum speed of 2.53GHZ. The cost of maxing-out the RAM to 8GB and replacing the 500GB spinning drive with a 512GB SSD came out to 1/3 of that sexy UltraLap 430! When I realized that I could upgrade my trusty notebook to almost the same specs as the the Lemur Ultra for 1/2 of its price, my decision was made. I would give up the longer battery life and settle for half the RAM, but I would end up with my high-performance dream machine. I'm willing to wait another couple of years for the technology to advance even further before replacing the HP outright.

After I ordered the 8GB of RAM and the 512GB M4 Series SSD drive recommended for my HP Pavilion dm4-1063cl notebook on the Crucial website, I began to look into what special considerations I would need when using an SSD drive on a Linux-powered notebook. I had read that SSD drives can read data 2-3 times faster than spinning SATA drives, that they boot and shut down faster, and that they are more reliable, quieter and cooler because they have no moving parts. I also read that SSD drives have a limited number of writes - 3,000-10,000 depending on design. So I did some research and found a handful of things you need to do to optimize a Linux notebook for an SSD drive.

Five tweaks to make to optimize your Linux computer for a solid-state drive

1. Enable the TRIM command to clean up garbage on the drive. You can do this as long as the Linux kernel is 2.6.33 or later, and you are use ext4 or other TRIM-friendly file systems.
2. Stop the system from recording every time files are accessed. By default Linux writes this information onto the drive for each file your system touches - every time it touches a file!
3. Use RAM instead of the SSD drive for storing temp and log files.
4. Swap more to RAM instead of the SSD.
5. Prioritize "reads" over "writes".

Enable TRIM and reduce writes to the hard drive

The first three tweaks are made by editing /etc/fstab. You use these flags to reduce SSD writes:

```
noatime,nodiratime,discard
```

1. Edit the fstab file as an administrator. On Mint, I open a terminal window and type the command:

   ```
   sudo gedit /etc/fstab
   ```

2. Edit the file to add the flags to the line that represents your SSD drive. It may be a line that begins with 'UUID' or with '/dev/sda1' or something similar, depending on your Linux distribution. Here is how my file looks with the old line commented out and a new one added with the additional flags:

   ```
   ###################################################################
   # / was on /dev/sda1 during installation
   # use these flags to reduce SSD writes: noatime,nodiratime,discard
   # noatime and nodiratime flags turns off writing "last access time"
   # discard enables TRIM as long as kernel >= 2.6.33
   # UUID=59d0f1dd-30b1-43c4-8d67-977a7130e353 / ext4 discard,errors=remount-ro
   0 1
   UUID=59d0f1dd-30b1-43c4-8d67-977a7130e353 / ext4
   noatime,nodiratime,discard,errors=remount-ro 0 1
   ```

3. While you have the fstab file open for editing, add the following 4 lines so that you use RAM instead of your SSD for temp and log files:

```
###################################################################
# Added 4 lines to use RAM instead of SSD for temp and log files
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
```

4. Now save the file and restart the computer to mount the SSD drive with your changes.

Reduce swappiness

Add these two lines to /etc/sysctl.conf to make the system swap more to RAM.

```
###################################################################
# Manual settings - these settings are to optimize for SSD drive
# Ref: https://wiki.archlinux.org/index.php/Solid_State_Drives
#
vm.swappiness=1
vm.vfs_cache_pressure=50
#
```

Prioritize "reads" over "writes"

Edit your GRUB boot loader's configuration file to organize the I/O scheduler to maximize performance:

1. Run this command to check the currently active scheduler:

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

   After installing my SSD, mine reads:

   ```
   noop [deadline] cfq
   ```

   The active scheduler appears in brackets.
   '**noop**' is essentially no scheduler at all. It uses a first-in, first-out (FIFO) prioritization.
   '**deadline**' prioritizes reads over writes. This is what you want for SSD drives.
   '**cfq**' completely fair queuing is the default and is designed for traditional rotating drives.

   Before you make the change to grub, ensure that you use the command above to determine which scheduler is active. On my Mint 14 64-bit system, the active scheduler was already set to "deadline" even though the grub file does not reflect the edits I'll detail below. Mint 14 appears to have detected the presence of the new SSD on its own and automatically switched itself to use "deadline".
2. For Ubuntu and other distributions using GRUB2:

   ```
   sudo gedit /etc/default/grub
   ```

3. Add 'deadline' to the GRUB_CMDLINE_LINUX_DEFAULT line like this:

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

4. Then to finalize the changes, run '`sudo update-grub2`' or '`sudo update-grub`'. You'll find the correct command to use referenced in the comments at the top of the grub file itself.

With increased RAM, a solid-state drive and these 5 tweaks, my two year old HP dm4 has become a fast booting, high-performance dream machine.

Source : http://goinglinux.com/articles/Optimize_Linux_For_SSD_Drive_en.htm