Installing on USB Thumbdrive

The main thing with using a USB thumbdrive is to decrease the number of writes. Compared to SSD and platter based block storage devices (disk drives, called “spinner” in this document), USB has a very limited number of writes available.

As soon as possible after doing the initial install, begin decreasing writes by modifying the atime parameter on anything stored on the USB, then moving active partitions (/tmp, /var/log, /var/run, etc…) off of the USB, either into a Memory Device (aka md, RAMDisk), and SSD or, preferably, a “spinner”.

Quick and Dirty

- Install your base system using UFS. I do a manual creation and simply use the entire space for /, then began tweaking
  - When given the opportunity to create the partition, do manual partitioning and set the following options in the Configure section:
    - SUJ off (turn off journaling, since this generates a lot of writes)
    - TRIM on (let TRIM work to balance where the writes go, SSD only)
    - SU on
  - The most important thing to decrease writes is to set that *noatime* parameter. This greatly reduces writes (we'll handle the other constant writes later).
- Put /tmp in md and point /var/tmp to it. If you have plenty of memory, this is very simple and really, really fast.
- Build your disk set of spinners. In my setup, I had a ZFS Z-RAID on some 10k drives, and an SSD card for the ZFS cache (I needed speed since it was planned to be a iSCSI server).
- put the following onto separate partitions on the spinners:
  - /var/log
  - /var/run
  - swap

More Detail

When I am done, my fstab looks like this.

```
# Device    Mountpoint    FStype    Options    Dump            Pass#
/dev/da4s1a /        ufs    rw,noatime            1    1
md11            /tmp            mfs     rw,-s256M,-M,nosuid,noatime,noexec 0
0
md              /var/run        mfs     rw,-s4M,nosuid,noatime  0   0
/dev/ada0p2     none            swap    sw                      0   0
/dev/ada1p2     none            swap    sw                      0       0
```

Note If you use *pkg* for your installation, it will use /tmp for most of its work. In that case, you could get a complaint that there is not enough room. You will either need to increase the size of your

ramdisk, or point pkg's work directory someplace else. I ran across that at one time, but now I can't find it. I'll update this when I do.

The above is a little different, since I already had an SSD in the system as the L2ARC for ZFS. FreeBSD does not require L2ARC have an entire device, so I was able to partition this out to give myself a block for swap. This particular card is actually two small devices which can be RAID'd, but I broke the RAID set and simply set up each device on its own, thus creating the two swap devices above (ada0p2 and ada1p2). ada0p1 and ada1p1 were for the L2ARC.

I also created a small mfs for /tmp and /var/run. /tmp really needs 256M (128 will work) if you are using pkg to install. If you use ports, most of the action takes place in the /usr/ports and /usr/src directories, and you should create partitions for that on the spinners (I would use ZFS since I'm already set up for it).

One thing is to point /var/tmp to /tmp. That is as simple as making sure nothing is using /var/tmp (do an ls, some files are ok), then run the following:

```
rm -fRv /var/tmp
ln -s /tmp /var/tmp
```

I set up /var/log on the zfs platter I created after the fact, so it doesn't show up in fstab. My pool is called Storage in this example:

```
zfs create storage/varlog
mv /var/log/* /storage/varlog
zfs set checksum=off atime=off exec=off  mountpoint=/var/log storage/varlog
service syslogd restart
mount
```

If you are going to use the Ports collection, it will do a lot of work in /usr/ports, so we want to fix that also. **Note** I think it does all its work in the /usr/port/* directory, but I have not been able to verify that.

```
zfs create storage/ports
mv /usr/ports/* /storage/ports
zfs set checksum=off atime=off exec=off mountpoint=/usr/ports storage/ports
```

Now, you can see the results of your mounted directories

```
mount
```

The output of the above command will show you the mounted file system

```
/dev/da4s1a on / (ufs, local, noatime, journaled soft-updates)
devfs on /dev (devfs, local, multilabel)
/dev/md0 on /tmp (ufs, local, noatime, nosuid, soft-updates)
/dev/md1 on /var/run (ufs, local, noatime, nosuid, soft-updates)
storage on /storage (zfs, local, nfsv4acls)
storage/varlog on /var/log (zfs, local, nfsv4acls)
```

Other Options

Note I have found that booting from an image on some enterprise servers requires you to reconfigure the BIOS, telling it to boot off the new “drive.” It appears the BIOS tracks which *device* it is to boot off of, not the particular slot. If you image a flash, be prepared to go into BIOS and reconfigure your boot device.

Duplicate your boot device

There are some really great options available when booting from USB that I ran into. Manual installs, installations with two copies of the boot image on the flash (http://www.cabstand.com/usbflash.html). The nice thing is, you can quickly make a copy of the low cost device and swap them out when/if needed.

The best one is, however, the ability to store an image very nicely and recreate the boot device if you ever need to. To do this.

1. Remove USB from drive
2. Plug into a second system
3. Use the *dd* command to make a copy.
    1. Determine the USB drive name
    2. Determine an output file on a device which has sufficient space
    3. `dd if=/dev/ad0 of=/my/path/to/image/ad0.img && sync`
4. You can now create a second device by simply using *dd* to copy from the file back to it.
    1. Determine the USB drive name (assume ad0 for this)
    2. dd if=/my/path/to/image/ad0.img of=/dev/ad0 && sync

The *&& sync* in the above command ensures all data is written before the cursor returns. Note that the target device (the second USB) must be as large or larger than the original one. I tend to make it the exact make and model.

Manually tune file system

If you forgot to set journaling off, trim on, etc…, you can do it now.

You can not modify a slice (partition) which is mounted rw, so reboot into single user mode. It will show you the volume mounted as read only.

Look at what is currently set on your volume

```
tunefs -p /dev/da0p3
```

Now, you can fix anything which is set incorrectly. We are going to disable

- -j soft updates journaling
- -J gjournal flag

- -n soft updates

and enable

- -t trim

```
tune2fs -j disable -J disable -n disable -t enable /dev/da0p3
```

If you had to add trim, there are likely a bunch of blocks which are not flagged. To fix this, issue:

```
fsck_ufs -Ey /dev/da0p3
```

Set label on root file system

```
tunefs -L root /dev/da0p3
```

References

- http://www.a1poweruser.com/30.00-USB_installing_article.php (Basic USB Install)
- http://www.cabstand.com/usbflash.html (Duplicate Image on one USB)
- https://box.matto.nl/ramdiskhome.html (MFS)
- http://daemon-notes.com/articles/system/install-ufs/gpart-mbr (Install FreeBSD 10 on gmirror/gstripe/simple UFS partitions (GPT or MBR))
- http://www.freebsdwiki.net/index.php/Create_RAMdisks_under_FreeBSD_5.x (RAMDisk creation. Set up for FreeBSD 5 but still works on 10)
- https://superuser.com/questions/1183946/how-to-trim-the-whole-partition-in-freebsd-to-save-space-in-virtualbox
- https://forums.freebsd.org/threads/everything-you-want-to-know-about-installing-freebsd-on-a-usb-stick.11715/