Physical storage device (HDD, SDD) partitions

Table of Contents

Hard disk drive (HDD) technology

Block device partitioning has a long history, related to the physical construction of hard disk drives – to their cylinders, heads and sectors. Current hard disk drives present themselves to the operating system as a simple sequence of blocks. For practical partitioning purposes the physical construction no longer matters except that Zone Bit Recording means that a partition at the beginning of the drive (physically the outer edge) transfers data faster than one at the end of the drive. With the introduction of 4k sector sizes, aligning partitions on sector boundaries has become important.

Wikipedia hard disk drive

Wikipedia disk sector

Wikipedia zone bit recording (ZBR a.k.a Zone Constant Angular Velocity (Zone CAV, Z-CAV or ZCAV))

Wikipedia disk partitioning

Wikipedia advanced format (> 512 B sectors)

Solid state drive (SSD) technology

TBC

Partition tables

Reference: http://en.wikipedia.org/wiki/Partition_table

⚠ SSD partitioning considerations

SSDs need some free space for over-provisioning, so it's advised to leave about 5% unpartitioned (e.g. 8 GB from a 128 GB) - however, many new SSDs have this space reserved on firmware level so there's no need to do that. Usually that's the case if capacity is just a little lower than the closest power of 2: e.g. 120 instead of 128 GB, 60 instead of 64 GB, etc.

DOS partition table (a.k.a MBR)

Also known as the MBR partition table.

⚠ Cannot be used with UEFI boot; must use a GUID partition table (GPT)

Reference: http://en.wikipedia.org/wiki/Master_boot_record

The EBR/EPBR

Each logical partition is preceded by an Extended Boot Record (EBR) a.k.a. Extended Partition Boot Record (EPBR) of one sector followed by a number of unused sectors. There are commonly 62 unused sectors which, with the EBR, fill a historical 63 sector track.

Which partition management utility to use?

⚠ If grub is installed in the MBR, using gdisk or cgdisk will make the system unbootable by overwriting the MBR.

⚠️ For 4096 kB sector HDDs, fdisk is better than cfdisk because it defaults to creating correctly aligned partitions whereas cfdisk offers only the beginning or end of the free space.

From the fdisk man page:

> *There are several *fdisk programs around. Each has its problems and strengths. Try them in the order **cfdisk**, **fdisk**, **sfdisk**. (Indeed, cfdisk is a beautiful program that has strict requirements on the partition tables it accepts, and produces high quality partition tables. Use it if you can. fdisk is a buggy program that does fuzzy things - usually it happens to produce reasonable results. Its single advantage is that it has some support for BSD disk labels and other non-DOS partition tables. Avoid it if you can. sfdisk is for hackers only -- the user interface is terrible, but it is more correct than fdisk and more powerful than both fdisk and cfdisk. Moreover, it can be used noninteractively.)*
>
> *These days there also is **parted**. The cfdisk interface is nicer, but parted does much more: it not only resizes partitions, but also the filesystems that live in them.*

GUID partition table (GPT)

GPT is required when the HDD presents 512 kB sectors and is greater than 2 TiB and for UEFI boot.

References:

- https://wiki.archlinux.org/index.php/GPT

GRUB's GPT requirements

TODO: duplicated text!!! Merge this with similar information at UEFI and GPT and cross-reference as required.

BIOS systems: when GRUB is to be installed on an HDD with GPT partitions, a bios_grub partition is required. We normally use the free space left before block 2048 when gdisk's default partition alignment is used.

EFI systems: when GRUB is to be installed on a UEFI system, an "EFI System Partition (ESP)" is required (called "EFI boot partition" in Debian and Ubuntu documentation), partition type ef00. It has to be big enough to contain all the boot loaders that will be installed in it, say 100 MB for a single OS system and 200 MB for multi-boot.

Which partition management utility to use?

For GPT-partitioning, there is **gdisk**, its full-screen version **cgdisk** and **sgdisk**. They automatically do 4096 kB physical sector alignment. TODO: confirm that sgdisk does. sgdisk can be used to replicate a partition table from one block device to another.

Appendix – exact DOS partition sizes

Find an exact partition size

The easiest way to find an exact partition size is to read the /sys/block/sd*/sd*/size file, for example `cat /sys/block/sda/sda2/size`. This gives the size of the sda2 partition in sectors.

⚠️ TODO: check what happens with a 4 kB sector size HDD. If /sys/block/sd*/sd*/size then lists the size in 4k sectors, how best to find the sector size? fdisk?

The alternatives are:

- **fdisk**

  The partition size shown by fdisk (in 1024 bytes Blocks) is rounded; if an exact partition size is required it can be calculated by End - Start + 1 (fd isk shows End and Start in sectors). For example:

  ```
  root@CW8:~# fdisk -l /dev/sdb
  [snip]
  Device Boot Start End Blocks Id System
  [snip]
  /dev/sdb7 1224268508 1250261614 12996553+ 83 Linux
  ```

  End - Start + 1 is 1250261614 - 1224268508 + 1 = 25993107.

  Exact size in sectors, converted to blocks: 25993107 / 2 = 12996553.5

- **cfdisk**

  The most informative listing is produced by the print option -P with the display sectors option -s (other units are rounded for display).

In the output, illustrated below:

- **Length** includes the EPBR, any following unused sectors and the logical partition itself.
- **Offset** is the EPBR plus any following unused sectors.
- **First Sector** is the first (and only) sector of the EPBR
- The partition size is not displayed. It can be calculated from Length less Offset. For example:

```
root@CW8:~# cfdisk -Ps /dev/sdb
Partition Table for /dev/sdb
              First       Last
 # Type       Sector      Sector   Offset    Length   Filesystem Type (ID) Flag
-- ------- ----------- ----------- ------ ----------- -------------------- ----
[snip]
 7 Logical  1224268445* 1250261614*    63    25993170 Linux (83)           None
```

Length less Offset: 25993170 - 63 = 25993107.

- **sfdisk** TBC
- **parted** TBC
- **/proc/partitions** The blocks listed are 1024 byte blocks, rounded down to the nearest integer.

Create a partition with an exact size

- **fdisk**

  The u option ("change display/entry units") toggles between cylinders and sectors. It changes the Start and End addresses but not the Blocks which are always shown as 1024 byte blocks. (Tested using util-linux 2.20.1).

  TODO: test the text below

  When creating a new partition, the default starting block leaves 62 unused sectors after the EPBR. This results in a historically conventional 63 sector track before the partition itself.

- **cfdisk** The size entered is the EPBR+unused+partition. If an exact partition size is required, the EPBR+unused size must be added to the required size. This can most easily be done by specifying the size in sectors (append an S) to avoid rounding effects.
- **sfdisk** TBC
- **parted** TBC