

Hardware

# Europe's first exascale supercomputer is now up and running, using 24,000 Nvidia GH200 Superchips to perform more than one quintillion operations per second with nearly 1,000,000 terabytes of storage

News

By [Andy Edser](#) published September 12, 2025

Yeah, but can it play... y'know what, I'm not even gonna go there.



Comments (4)

Europe's first exabyte system



## World's most energy-efficient AI supercomputer comes online

JUPITER, the European Union's new exascale supercomputer, is 100% powered by renewable energy. Can it compete in the global AI race?

By [Jonathan O'Callaghan](#)



German Chancellor Friedrich Merz inaugurated the new European supercomputer called JUPITER on 5 September. Credit: Ina Fassbender/AFP via Getty

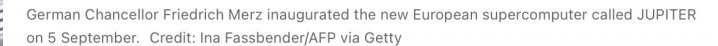


# Europes first exabyte system

- |   |                                                                                                                                                                 |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 | <b>El Capitan</b> - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE                                                  |
| 2 | <b>Frontier</b> - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE                              |
| 3 | <b>Aurora</b> - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel                        |
| 4 | <b>JUPITER Booster</b> - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN |
| 5 | <b>Eagle</b> - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure                                                |







# Lecture 2: Disks and Filesystems

High Performance Computers



# Outline

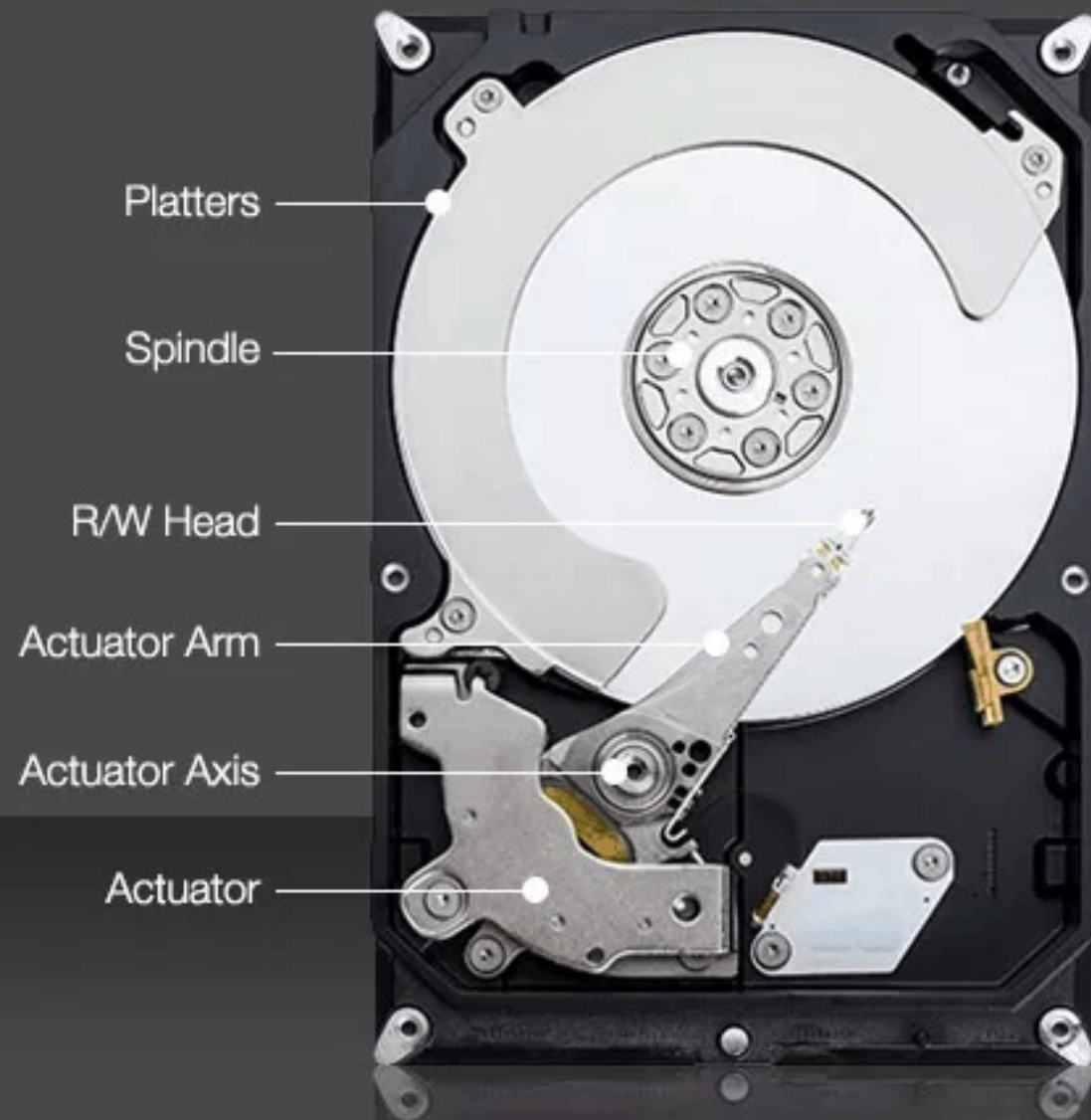
- Disk Hardware
  - Spinning Disks, Solid State Drives
- Partitioning
  - Viewing, Modifying, Creating, Geometry, SSDs
- Filesystems
  - Filesystem Types, Creating, Mounting, UUIDs, Caching, fstab, repairing
- Swap Space
- LVMs
- Internals
  - Inodes, blocks

# Relevance to HPC

- Performance bottlenecks often come from filesystem IO not compute
  - In HPC, CPUs and GPUs are extremely fast, but if the filesystem can't feed them data quickly enough, performance collapses.
- Managing large clusters involves creating virtual filesystems
  - You have to have some idea of how filesystems work to understand this
- HPC centers deal with massive amounts of data. Understanding limits introduced by filesystems is critical.
- Filesystems underpin everything a computer does.

# HDD

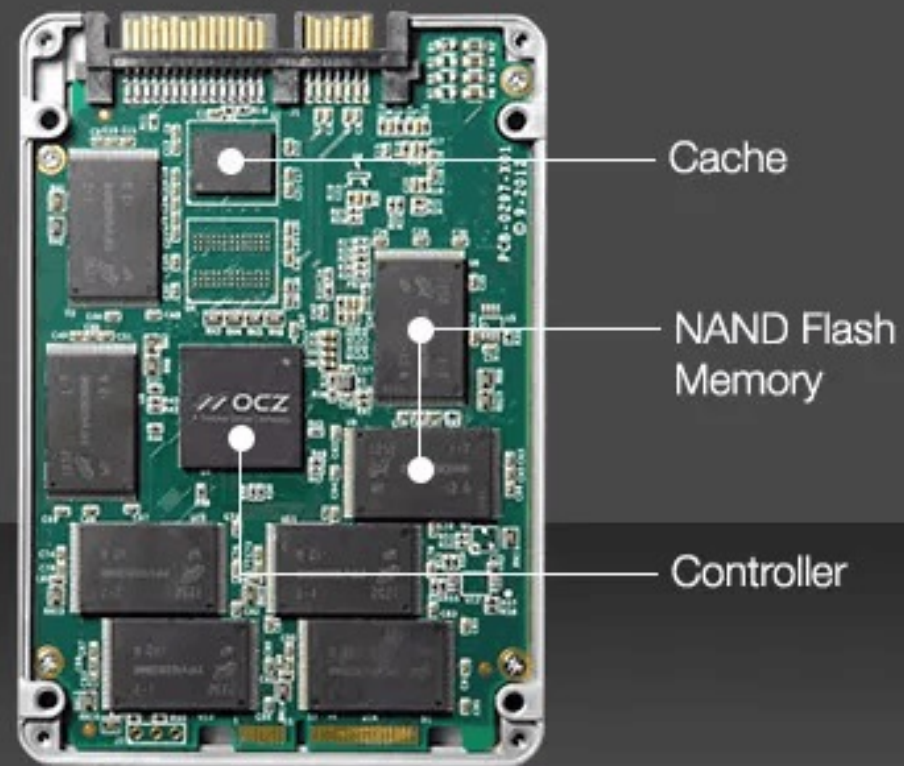
3.5"



Shock resistant up to 350g/2ms

# SSD

2.5"



Shock resistant up to 1500g/0.5ms

CHS Addressing (Cylinder, Head, Sector)

Maps to Logical Block Addresses (LBA)

5400, 7200, 10K, 15K RPM

Head is 2-3 nm above the spinning disk  
(1/30k the width of a human skin cell)

The relative motion of the head over  
Platter is 115 km/h at the outside and  
30 km/h towards the center.

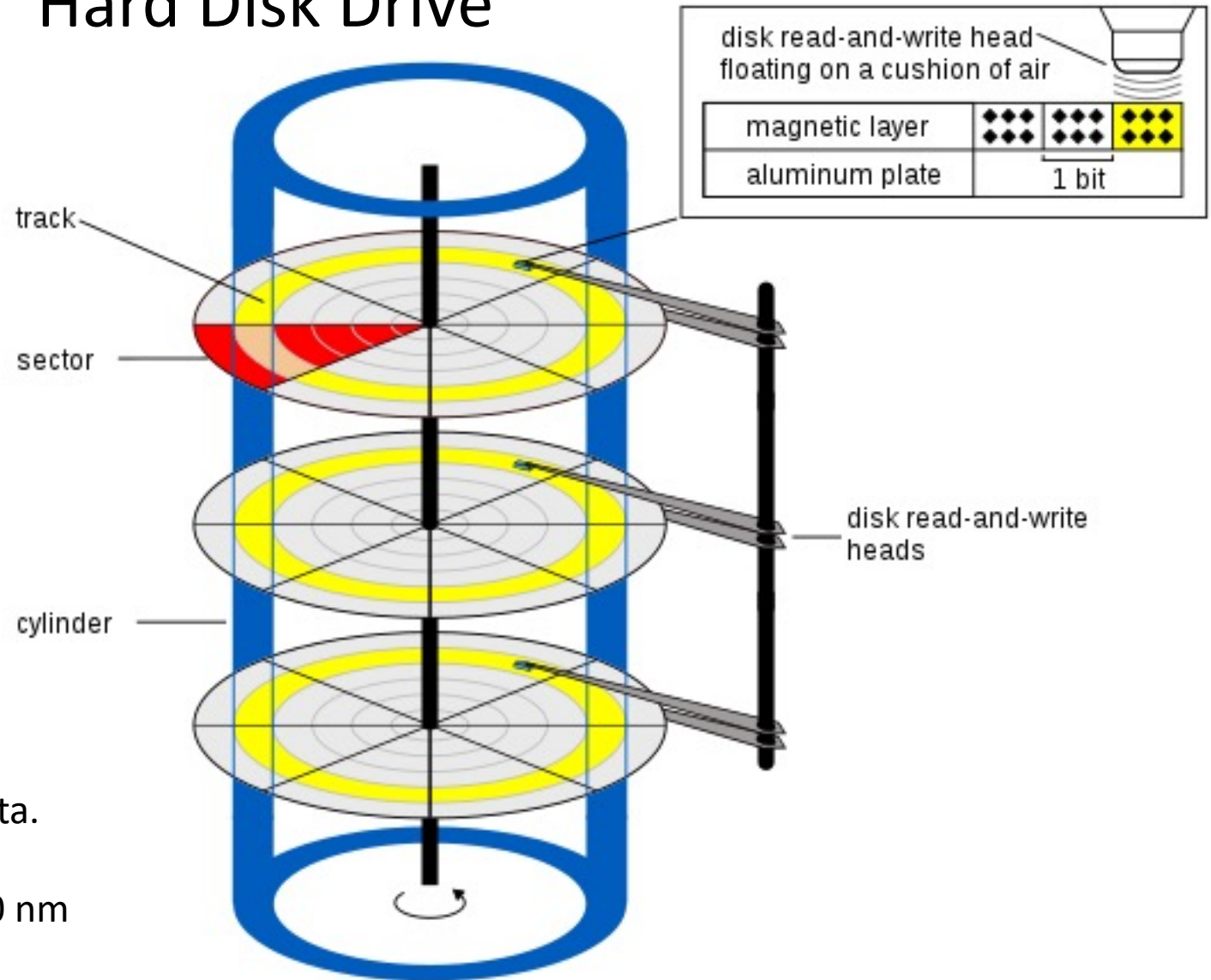
The HDD head can polarize the iron atoms  
covering the HDD platter to store 0s and 1s.

It can also detect the polarization to read data.

The area that stores the bit value is about 50 nm  
wide.

A disk crash can literally mean that the head  
crashed into the platter causing damage.

# Hard Disk Drive



**These regions can be rewritten an unlimited number of times.**

# Hard Disk Drive

CHS Addressing (Cylinder, Head, Sector)

Maps to Logical Block Addresses (LBA)

5400, 7200, 10K

Head is 2-3 nm a  
(1/30k the width

The relative mot  
Platter is 115 km  
30 km/h toward

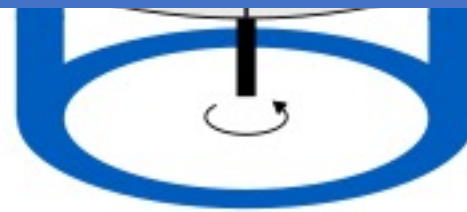
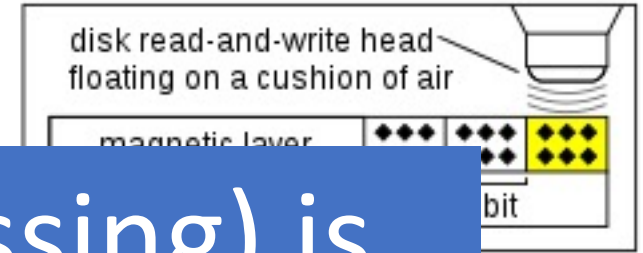
The HDD head c  
covering the HD

It can also detect the polarization to read data.

The area that stores the bit value is about 50 nm wide.

A disk crash can literally mean that the head crashed into the platter causing damage.

LBA (Logical Block Addressing) is used now even for spinning disks not CHS. But you will still see CHS mentioned for backwards compatibility.



**These regions can be rewritten an unlimited number of times.**

# Hard Disk Drive

CHS Addressing (Cylinder, Head, Sector)

Maps to Logical Block Addresses (LBA)

5400, 7200

Head is 2-3  
(1/30k the

The relative  
Platter is 11  
30 km/h to

The HDD he  
covering th

It can also c

The area th  
wide.

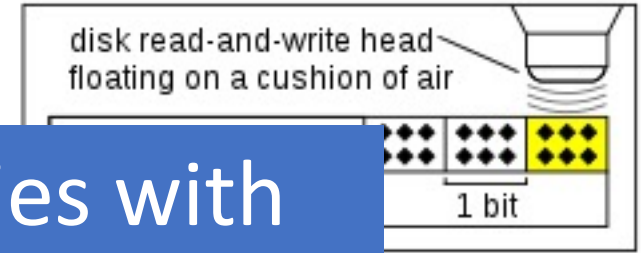
A disk crash can literally mean that the head crashed into the platter causing damage.

Explains those “weird” 16-byte entries with CHS and LBA fields side by side.

Explains why the MBR had such small limits (2 TB).

BIOS, bootloaders, and partition tables weren't designed for today's disks.

**These regions can be rewritten an unlimited number of times.**



Faster  
(Recall NVME (25 GB/s) vs SCSI (6 GB/s))

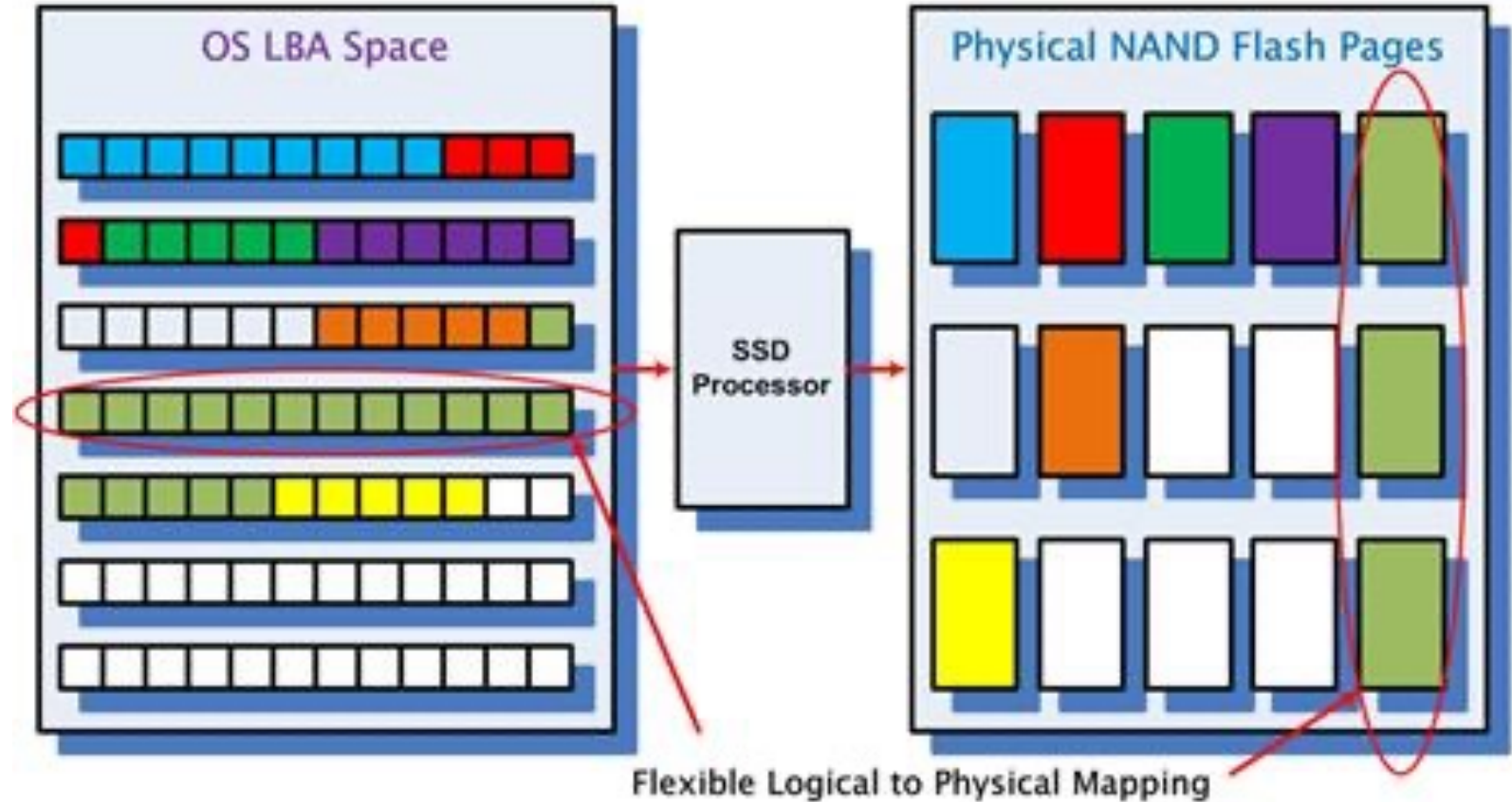
Data sectors can only be rewritten a limited number of times (a few 10Ks).

**Write Amplification Factor (WAF)** makes the rewrite limit worse. **The problem is that data is written at a more granular level (pages) than erasure happens (blocks).**

Basically, an SSD has to erase more storage than it writes.

As the drive fills up data gets more fragmented (is spread across more areas of the drive) and erasing data in all those different areas gets more and more expensive relative to the data written and the rewrite limits are reached fast.

# Logical Block Addressing\*



\* Spinning Disks have supported LBA addressing since the 1990s  
**NT 13h Extensions** ("Enhanced Disk Drive Services").  
Natural for SSDs

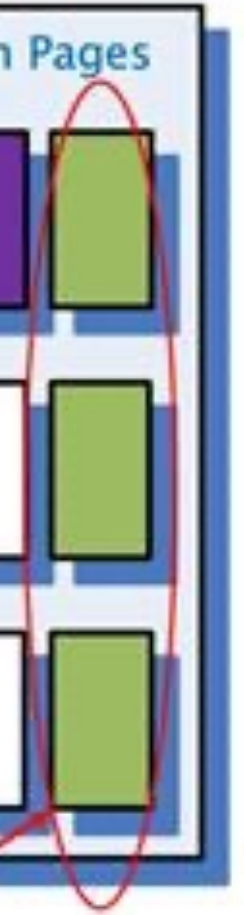
**Current Samsung drives are good for about 300 TB of writing. That's fine for a desktop but not for many HPC applications.**

HOST accesses **SSD through LBA (Logical Block Address)**. Each LBA represents a Sector (generally 512B in size) and the operating system generally accesses [SSD](#) in 4K

## Solid State Drive

Modern Enterprise SSDs have tradeoffs between write longevity, read longevity, or both with reduced performance.

Centers still by spinning disks for cost control and SSD cache (tiering).



applications.

512B in

Faster  
(Recall NVME (25GB/s) vs SATA (6GB/s))

Data sectors can only be rewritten a limited number of times

**Write Amplification**  
makes the rewrite limit a problem is that data is written at a more granular level than erasure happens

Basically, an SSD has less usable storage than it appears to

As the drive fills up data becomes fragmented (is spread across different areas of the drive) and accessing data in all those different locations gets more and more expensive relative to the data written when rewrite limits are reached

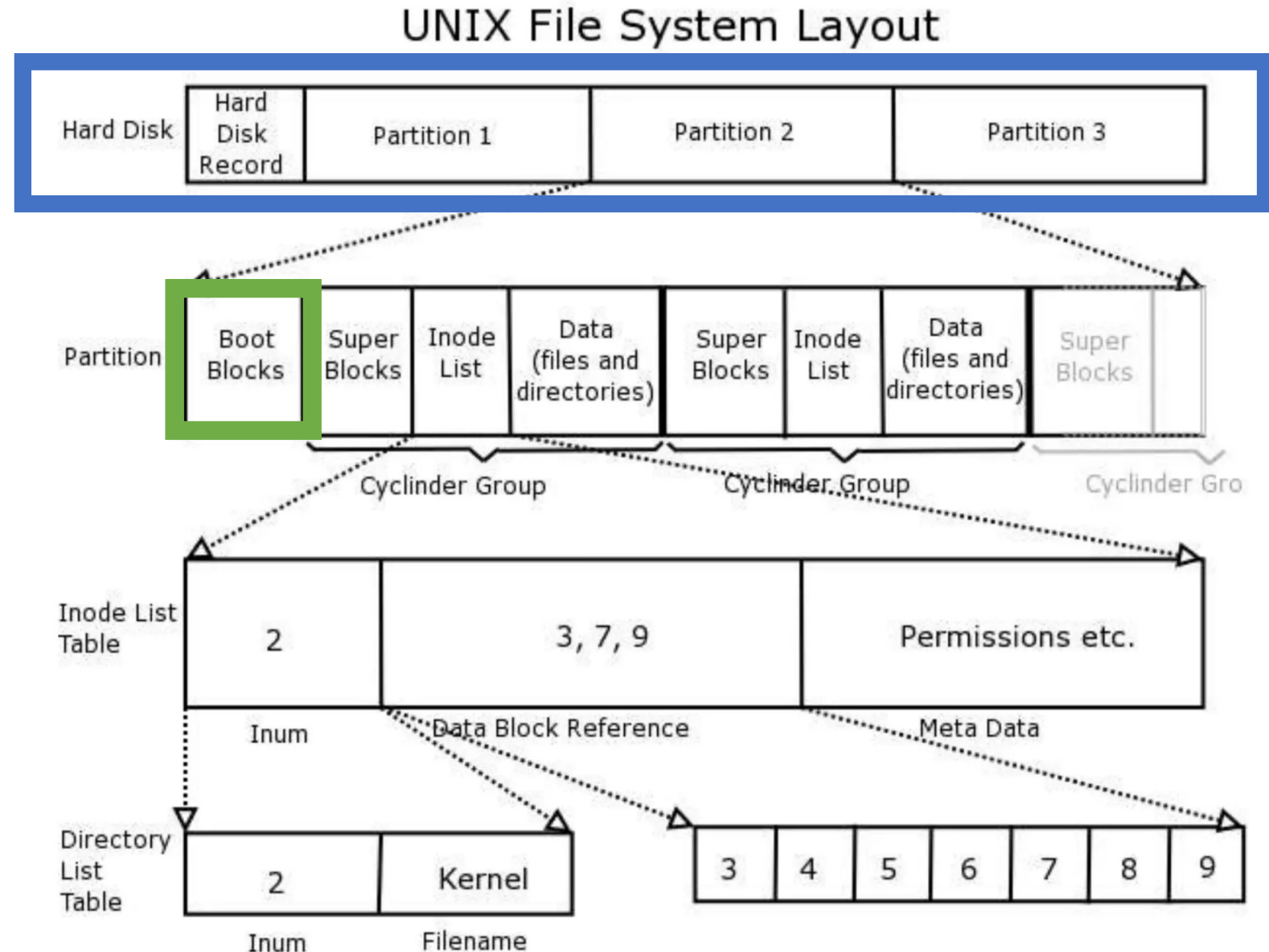
**Current Samsung**

HOST accesses

(size) and the operating system generally accesses [SSD](#) in 4K

# Harddrive Filesystem Layout

- We can do raw reads and writes to disks through `/dev/`
- But are usually much more organized so we can manage the data

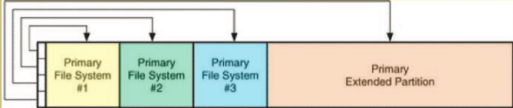


# Partitions: Master Boot Record

- An older standard for organizing disk storage (sometimes called the MSDOS partition scheme).
- The MBR tells boot loaders where to find things like the operating system.
- You can have multiple operating systems on different partitions.
- Partitions might also be used to separate data.
- In homework 1 you created separate partitions for /boot and /home
- This is often done to make sure that even if users fill up their partition that doesn't affect vital areas the OS want's to use. Like /boot.







| Offset | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  |                   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|
| 0      | 33 | 0C | 8E | D0 | BC | 00 | 7C | 8E | C0 | 8E | D8 | BE | 00 | 7C | BF | 00 | 3A1D%  1A10%  c   |
| 16     | 06 | B9 | 00 | 02 | FC | F3 | A4 | 50 | 68 | 1C | 06 | CB | FB | B9 | 04 | 00 | ' uoPh Eù¹        |
| 32     | BD | BE | 07 | 80 | 7E | 00 | 00 | 7C | 0B | 0F | 85 | 0E | 01 | 83 | C5 | 10 | % I~   I 1A       |
| 48     | E2 | F1 | CD | 18 | 88 | 56 | 00 | 55 | C6 | 46 | 11 | 05 | C6 | 46 | 10 | 00 | añI IV UeF æF     |
| 64     | B4 | 41 | BB | AA | 55 | CD | 13 | 5D | 72 | 0F | 81 | FB | 55 | AA | 75 | 09 | 'A»UI jr uU#u     |
| 80     | F7 | C1 | 01 | 00 | 74 | 03 | FE | 46 | 10 | 66 | 60 | 80 | 7E | 10 | 00 | 74 | +Á t pF f'lv t    |
| 96     | 26 | 66 | 68 | 00 | 00 | 00 | 00 | 66 | FF | 76 | 08 | 68 | 00 | 00 | 68 | 00 | &fh fyv h h       |
| 112    | 7C | 68 | 01 | 00 | 68 | 10 | 00 | B4 | 42 | 8A | 56 | 00 | 8B | F4 | CD | 13 | h h 'BIV iol      |
| 128    | 9F | 83 | C4 | 10 | 9E | EB | 14 | B8 | 01 | 02 | BB | 00 | 7C | 8A | 56 | 00 | 11A ie , »  IV    |
| 144    | 8A | 76 | 01 | 8A | 4E | 02 | 8A | 6E | 03 | CD | 13 | 66 | 61 | 73 | 1C | FE | lv IN ln i fas b  |
| 160    | 4E | 11 | 75 | 0C | 80 | 7E | 00 | 80 | 0F | 84 | 8A | 00 | B2 | 80 | EB | 84 | N u l~ l l l 'lel |
| 176    | 55 | 32 | E4 | 8A | 56 | 00 | CD | 13 | 5D | EB | 9E | 81 | 3E | FE | 7D | 55 | U2aIV i jol >U    |
| 192    | AA | 75 | 6E | FE | 76 | 00 | E8 | 8D | 00 | 75 | 17 | FA | B0 | D1 | E6 | 64 | #unyv è u ú'Ned   |
| 208    | E8 | 83 | 00 | B0 | DF | E6 | 60 | E8 | 7C | 00 | B0 | FF | E6 | 64 | E8 | 75 | èl 'Bæ'èl 'yadèu  |
| 224    | 00 | FB | B8 | 00 | BB | CD | 1A | 66 | 23 | C0 | 75 | 3B | 66 | 81 | FB | 54 | ù, »I f#Au:f ùT   |
| 240    | 43 | 50 | 41 | 75 | 32 | 81 | F9 | 02 | 01 | 72 | 2C | 66 | 68 | 07 | BB | 00 | CPAu2 ù r.fh »    |
| 256    | 00 | 66 | 68 | 00 | 02 | 00 | 00 | 66 | 68 | 08 | 00 | 00 | 00 | 66 | 53 | 66 | fh fh fSf         |
| 272    | 53 | 66 | 55 | 66 | 68 | 00 | 00 | 00 | 00 | 66 | 68 | 00 | 7C | 00 | 00 | 66 | SfUfh fh   f      |
| 288    | 61 | 68 | 00 | 00 | 07 | CD | 1A | 5A | 32 | F6 | EA | 00 | 7C | 00 | 00 | CD | ah I Z2èè   f     |
| 304    | 18 | A0 | B7 | 07 | EB | 08 | A0 | B6 | 07 | EB | 03 | A0 | B5 | 07 | 32 | E4 | . è ¶ è µ 2ä      |
| 320    | 05 | 00 | 07 | 8B | F0 | AC | 3C | 00 | 74 | 09 | BB | 07 | 00 | B4 | 0E | CD | Iè< t » ' I       |
| 336    | 10 | EB | F2 | F4 | EB | FD | 2B | C9 | E4 | 64 | EB | 00 | 24 | 02 | E0 | F8 | ëööÿ+Eadè \$ æ    |
| 352    | 24 | 02 | C3 | 49 | 6E | 76 | 61 | 6C | 69 | 64 | 20 | 70 | 61 | 72 | 74 | 69 | \$ ÅInvalid parti |
| 368    | 74 | 69 | 6F | 6E | 20 | 74 | 61 | 62 | 6C | 65 | 00 | 45 | 72 | 72 | 6F | 72 | tion table Error  |
| 384    | 20 | 6C | 6F | 61 | 64 | 69 | 6E | 67 | 20 | 6F | 70 | 65 | 72 | 61 | 74 | 69 | loading operati   |
| 400    | 6E | 67 | 20 | 73 | 79 | 73 | 74 | 65 | 6D | 00 | 4D | 69 | 73 | 73 | 69 | 6E | ng system Missin  |
| 416    | 67 | 20 | 6F | 70 | 65 | 72 | 61 | 74 | 69 | 6E | 67 | 20 | 73 | 79 | 73 | 74 | g operating syst  |
| 432    | 65 | 6D | 00 | 00 | 00 | 63 | 7B | 9A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | em c(l            |
| 448    | 03 | 00 | 06 | FE | 7F | E1 | 80 | 00 | 00 | 00 | 80 | 37 | 76 | 00 | 00 | 00 | p ál I7v          |
| 464    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |                   |
| 480    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |                   |
| 496    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |                   |

| Bytes   | Purpose                  |
|---------|--------------------------|
| 0-445   | Boot code                |
| 446-461 | Partition Table Entry #1 |
| 462-477 | Partition Table Entry #2 |
| 478-493 | Partition Table Entry #3 |
| 494-509 | Partition Table Entry #4 |
| 510-511 | Signature value (0xAA55) |

| Bytes | Purpose                                                                                                             |
|-------|---------------------------------------------------------------------------------------------------------------------|
| 0     | Bootable flag (0x80=active; else 0x00)                                                                              |
| 1-3   | Starting CHS address                                                                                                |
| 4     | Partition type (e.g., 0x00=empty, 0x01=FAT12, 0x07=NTFS, 0x0b=FAT32 (CHS), 0x83=Linux, 0xa5=FreeBSD, 0xa8=MacOS X*) |
| 5-7   | Ending CHS address                                                                                                  |
| 8-11  | Starting LBA address                                                                                                |
| 12-15 | Size (in sectors)                                                                                                   |

| Flag | Starting CHS | Partition Type | Ending CHS | Starting LBA | Size     |
|------|--------------|----------------|------------|--------------|----------|
| 00   | 000302       | 06             | E17FFE     | 00000080     | 00763780 |

\*Data is in an IA32-based system, Little endian (least significant byte is first)

# Partitions: Master Boot Record

- This is a hexadecimal representation of the binary data at the start of an MBR partitioned disk.
- Useful because characters become readable.
- You can display this data with

## Log into your Roadrunner cluster

```
sudo dd if=/dev/sda bs=1  
count=1024 | hexdump -C
```

| <b>Bytes</b> | <b>Purpose</b>           |
|--------------|--------------------------|
| 0-445        | Boot code                |
| 446-461      | Partition Table Entry #1 |
| 462-477      | Partition Table Entry #2 |
| 478-493      | Partition Table Entry #3 |
| 494-509      | Partition Table Entry #4 |
| 510-511      | Signature value (0xAA55) |

Also known as the magic number

| Offset | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  |                   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|
| 0      | 33 | C0 | 8E | D0 | BC | 00 | 7C | 8E | C0 | 8E | D8 | BE | 00 | 7C | BF | 00 | 3Ä!D%   Ä!0%  c   |
| 16     | 06 | B9 | 00 | 02 | FC | F3 | A4 | 50 | 68 | 1C | 06 | CB | FB | B9 | 04 | 00 | ' üóPPh Èü¹       |
| 32     | BD | BE | 07 | 80 | 7E | 00 | 00 | 7C | 0B | 0F | 85 | 0E | 01 | 83 | C5 | 10 | ¼%  ~     Ä       |
| 48     | E2 | F1 | CD | 18 | 88 | 56 | 00 | 55 | C6 | 46 | 11 | 05 | C6 | 46 | 10 | 00 | äñí  V UÆF æF     |
| 64     | B4 | 41 | BB | AA | 55 | CD | 13 | 5D | 72 | 0F | 81 | FB | 55 | AA | 75 | 09 | 'A»²Uí ]r ûU²u    |
| 80     | F7 | C1 | 01 | 00 | 74 | 03 | FE | 46 | 10 | 66 | 60 | 80 | 7E | 10 | 00 | 74 | ÷Á t þF f'  ~ t   |
| 96     | 26 | 66 | 68 | 00 | 00 | 00 | 00 | 66 | FF | 76 | 08 | 68 | 00 | 00 | 68 | 00 | &fh fýv h h       |
| 112    | 7C | 68 | 01 | 00 | 68 | 10 | 00 | B4 | 42 | 8A | 56 | 00 | 8B | F4 | CD | 13 | h h 'B V  óí      |
| 128    | 9F | 83 | C4 | 10 | 9E | EB | 14 | B8 | 01 | 02 | BB | 00 | 7C | 8A | 56 | 00 | Ä  è , »  íV      |
| 144    | 8A | 76 | 01 | 8A | 4E | 02 | 8A | 6E | 03 | CD | 13 | 66 | 61 | 73 | 1C | FE | v  N  n í fas þ   |
| 160    | 4E | 11 | 75 | 0C | 80 | 7E | 00 | 80 | 0F | 84 | 8A | 00 | B2 | 80 | EB | 84 | N u  ~      ² è   |
| 176    | 55 | 32 | E4 | 8A | 56 | 00 | CD | 13 | 5D | EB | 9E | 81 | 3E | FE | 7D | 55 | U2ä V í  è  >þ}U  |
| 192    | AA | 75 | 6E | FF | 76 | 00 | E8 | 8D | 00 | 75 | 17 | FA | B0 | D1 | E6 | 64 | ²unýv è u ú²Ñæd   |
| 208    | E8 | 83 | 00 | B0 | DF | E6 | 60 | E8 | 7C | 00 | B0 | FF | E6 | 64 | E8 | 75 | è  ²Bæ`è  ²yæðèu  |
| 224    | 00 | FB | B8 | 00 | BB | CD | 1A | 66 | 23 | C0 | 75 | 3B | 66 | 81 | FB | 54 | û, »í f#Äu;f ûT   |
| 240    | 43 | 50 | 41 | 75 | 32 | 81 | F9 | 02 | 01 | 72 | 2C | 66 | 68 | 07 | BB | 00 | CPÄu2 û r,fh »    |
| 256    | 00 | 66 | 68 | 00 | 02 | 00 | 00 | 66 | 68 | 08 | 00 | 00 | 00 | 66 | 53 | 66 | fh fh fSf         |
| 272    | 53 | 66 | 55 | 66 | 68 | 00 | 00 | 00 | 00 | 66 | 68 | 00 | 7C | 00 | 00 | 66 | SfUfh fh   f      |
| 288    | 61 | 68 | 00 | 00 | 07 | CD | 1A | 5A | 32 | F6 | EA | 00 | 7C | 00 | 00 | CD | ah í Z2öè   í     |
| 304    | 18 | A0 | B7 | 07 | EB | 08 | A0 | B6 | 07 | EB | 03 | A0 | B5 | 07 | 32 | E4 | · è ¶ è µ 2ä      |
| 320    | 05 | 00 | 07 | 8B | F0 | AC | 3C | 00 | 74 | 09 | BB | 07 | 00 | B4 | 0E | CD | ð~< t » ' í       |
| 336    | 10 | EB | F2 | F4 | EB | FD | 2B | C9 | E4 | 64 | EB | 00 | 24 | 02 | E0 | F8 | èòöèý+Èädè \$ ào  |
| 352    | 24 | 02 | C3 | 49 | 6E | 76 | 61 | 6C | 69 | 64 | 20 | 70 | 61 | 72 | 74 | 69 | \$ ÄInvalid parti |
| 368    | 74 | 69 | 6F | 6E | 20 | 74 | 61 | 62 | 6C | 65 | 00 | 45 | 72 | 72 | 6F | 72 | tion table Error  |
| 384    | 20 | 6C | 6F | 61 | 64 | 69 | 6E | 67 | 20 | 6F | 70 | 65 | 72 | 61 | 74 | 69 | loading operati   |
| 400    | 6E | 67 | 20 | 73 | 79 | 73 | 74 | 65 | 6D | 00 | 4D | 69 | 73 | 73 | 69 | 6E | ng system Missin  |
| 416    | 67 | 20 | 6F | 70 | 65 | 72 | 61 | 74 | 69 | 6E | 67 | 20 | 73 | 79 | 73 | 74 | g operating syst  |
| 432    | 65 | 6D | 00 | 00 | 00 | 63 | 7B | 9A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 02 | em c{             |
| 448    | 03 | 00 | 06 | FE | 7F | E1 | 80 | 00 | 00 | 00 | 80 | 37 | 76 | 00 | 00 | 00 | þ á  17v          |
| 464    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |                   |
| 480    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |                   |
| 496    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | U²                |

Only 445  
bytes  
available  
to contain  
the boot  
program (!)

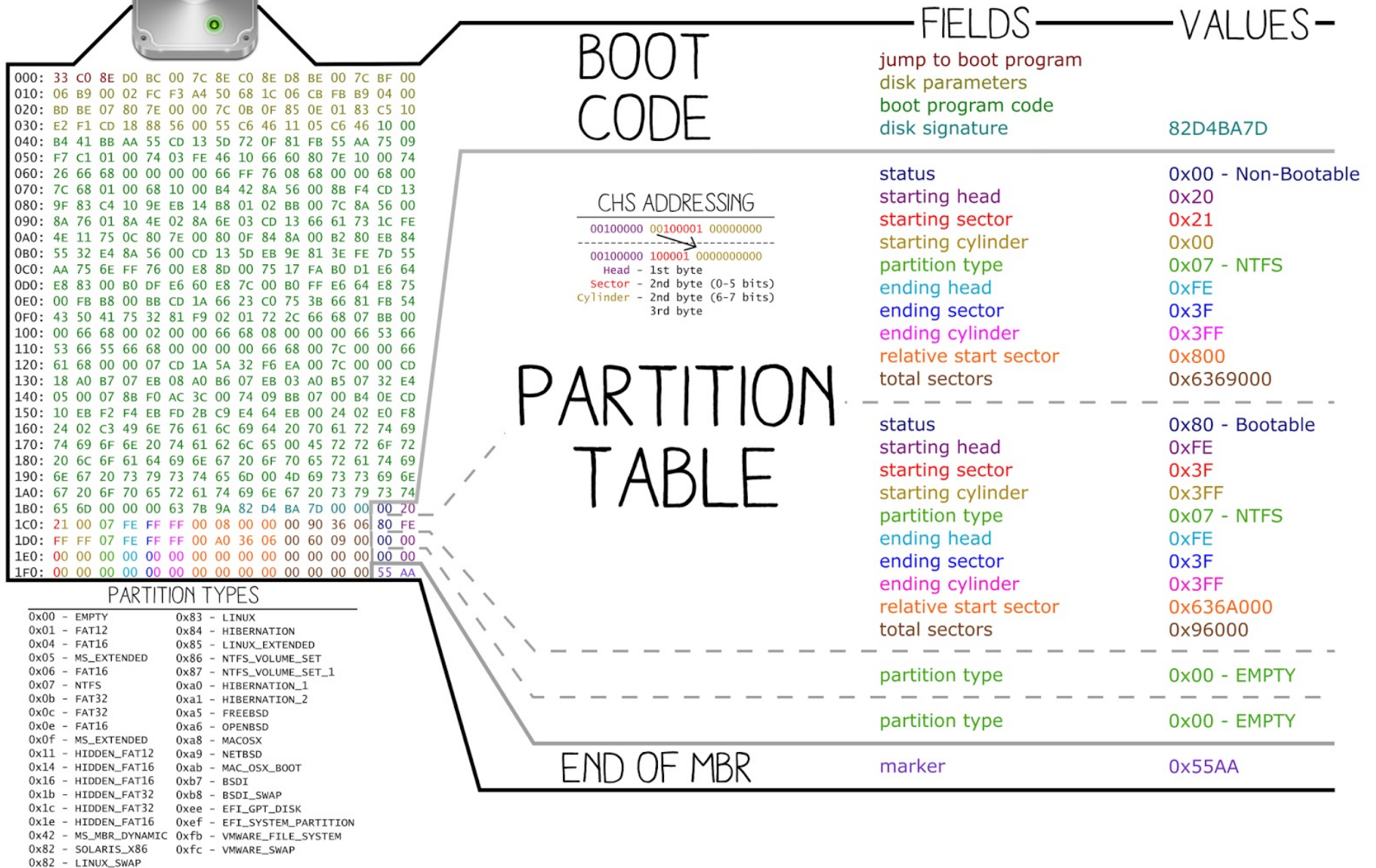


# MASTER BOOT RECORD

≥ INVOKE-IR

BY: JARED ATKINSON

TEMPLATE BY: ANGE ALBERTINI



# How the Master Boot Code Works

1. System startup self-check - BIOS checks the system hardware and CMOS Settings.
2. Read the master boot record - detect bootable devices, BIOS reads the MBR sector into memory.
3. Check whether the end flag of the MBR is 0000:7C00H equals 55AAH. When the boot device meets the requirements, the BIOS transfers control to the MBR to start the operating system.

The master boot code uses what's called **CHS** fields (Starting and Ending Cylinder, Head, and Sector fields) from the partition table to locate the boot sector portion of the partition.

The MBR can refer to partitions on other drives.

You can create a single partition on a drive and use it without an MBR.

FreeBSD: Bootstrap Source Code:

<https://svnweb.freebsd.org/base/stable/8/sys/boot/i386/boot0/boot0.S?revision=196045&view=markup>

```
/*
 * BOOT BLOCK STRUCTURE
 *
 * This code implements a Master Boot Record (MBR) for an Intel/PC disk.
 * It is 512 bytes long and it is normally loaded by the BIOS (or another
 * bootloader) at 0:0x7c00. This code depends on %cs:%ip being 0:0x7c00
 *
 * The initial chunk of instructions is used as a signature by external
 * tools (e.g. boot0cfg) which can manipulate the block itself.
 *
 * The area at offset 0x1b2 contains a magic string ('Drive '), also
 * used as a signature to detect the block, and some variables that can
 * be updated by boot0cfg (and optionally written back to the disk).
 * These variables control the operation of the bootloader itself,
 * e.g. which partitions to enable, the timeout, the use of LBA
 * (called 'packet') or CHS mode, whether to force a drive number,
 * and whether to write back the user's selection back to disk.
 *
 * As in every Master Boot Record, the partition table is at 0x1be,
 * made of four 16-byte entries each containing:
 *
 *   OFF SIZE DESCRIPTION
 *   0    1 status (0x80: bootable, 0: non bootable)
 *   1    3 start sector CHS
 *   8:head, 6:sector, 2:cyl bit 9..8, 8:cyl bit 7..0
 *   4    1 partition type
 *   5    3 end sector CHS
 *   8    4 LBA of first sector
 *  12    4 partition size in sectors
 *
 * and followed by the two bytes 0x55, 0xAA (MBR signature).
 */
```

# Bootstrap Loader Snippet

```
*
 * CONSTANTS
 *
 * NHRDRV is the address in segment 0 where the BIOS writes the
 * total number of hard disks in the system.
 * LOAD is the original load address and cannot be changed.
 * ORIGIN is the relocation address. If you change it, you also need
 * to change the value passed to the linker in the Makefile
 * PRT_OFF is the location of the partition table (from the MBR
standard).
 * B0_OFF is the location of the data area, known to boot0cfg so
 * it cannot be changed. Computed as a negative offset from 0x200
 * MAGIC is the signature of a boot block.
 */

.set NHRDRV,0x475 # Number of hard drives
.set ORIGIN,0x600 # Execution address
.set LOAD,0x7c00 # Load address

.set PRT_OFF,0x1be # Partition table
.set B0_OFF,(B0_BASE-0x200) # Offset of boot0 data

.set MAGIC,0xaa55 # Magic: bootable

.set KEY_ENTER,0x1c # Enter key scan code
.set KEY_F1,0x3b # F1 key scan code
.set KEY_1,0x02 # #1 key scan code

.set ASCII_BEL,'#' # ASCII code for <BEL>
.set ASCII_CR,0x0D # ASCII code for <CR>
```

```

/*
 * BOOT BLOCK STRUCTURE
 *
 * This code implements a Master Boot Record (MBR) for an Intel/PC disk.
 * It is 512 bytes long and it is normally loaded by the BIOS (or another
 * bootloader) at 0:0x7c00. This code depends on %cs:%ip being 0:0x7c00
 *
 * The initial chunk of instructions is used as a signature by external
 * tools (e.g. boot0cfg) which can manipulate the block itself.
 *
 * The area at offset 0x1b2 contains a magic string ('Drive '), also
 * used as a signature to detect the block, and some variables that can
 * be updated by boot0cfg (and optionally written back to the disk).
 * These variables control the operation of the bootloader itself,
 * e.g. which partitions to enable, the timeout, the use of LBA
 * (called 'packet') or CHS mode, whether to force a drive number,
 * and whether to write back the user's selection back to disk.
 *
 * As in every Master Boot Record, the partition table is at 0x1be,
 * made of four 16-byte entries each containing:
 *
 *   OFF SIZE DESCRIPTION
 *   0    1 status (0x80: bootable, 0: non bootable)
 *   1    3 start sector CHS
 *   8:head, 6:sector, 2:cyl bit 9..8, 8:cyl bit 7..0
 *   4    1 partition type
 *   5    3 end sector CHS
 *   8    4 LBA of first sector
 *  12    4 partition size in sectors

```

The MBR was designed in the CHS era, so it carries both CHS and LBA fields. BIOSes now rely on LBA, but the leftover CHS fields remain for backward compatibility. On SSDs they're meaningless filler.

LBA was introduced in the 90s as a layer on top of CHS to increase capacity. This happens all the time, to maintain Backwards compatibility a new abstraction layer is added on top of something to increase limits.

# Bootstrap Loader Snippet

```

*
 * CONSTANTS
 *
 * NHRDRV is the address in segment 0 where the BIOS writes the
 * total number of hard disks in the system.
 * LOAD is the original load address and cannot be changed.
 * ORIGIN is the relocation address. If you change it, you also need
 * to change the value passed to the linker in the Makefile
 * PRT_OFF is the location of the partition table (from the MBR
standard).
 * B0_OFF is the location of the data area, known to boot0cfg so
 * it cannot be changed. Computed as a negative offset from 0x200
 * MAGIC is the signature of a boot block.
 */

.set NHRDRV,0x475 # Number of hard drives
.set ORIGIN,0x600 # Execution address
.set LOAD,0x7c00 # Load address

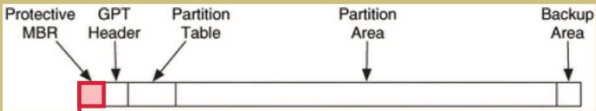
.set PRT_OFF,0x1be # Partition table
.set B0_OFF,(B0_BASE-0x200) # Offset of boot0 data

```

# GUID Partition Table (GPT)

## Global Unique ID (GUID)

- Newer
- MBR Supports 4 primary partitions (+ extended partition area)
- GPT supports 128 partitions.
- Requires Unified Extensible Firmware Interface (UEFI).



### Protective MBR (LBA 0)

| Offset | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0      | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 16     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 32     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 48     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 64     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 80     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 96     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 112    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 128    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 144    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 160    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 176    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 192    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 208    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 224    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 240    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 256    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 272    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 288    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 304    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 320    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 336    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 352    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 368    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 384    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 400    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 416    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 432    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 448    | 02 | 00 | EE | FF | FF | FF | 01 | 00 | 00 | 00 | FF | FF | FF | FF | 00 | 00 |
| 464    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 480    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 496    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 55 | AA |

| GUID Protective MBR |                                                                              |
|---------------------|------------------------------------------------------------------------------|
| Bytes               | Description                                                                  |
| 0-440               | Unused by UEFI systems                                                       |
| 440-443             | Unused and set to Zero                                                       |
| 444-445             | Unused and set to Zero                                                       |
| 446-509             | MBR partition records that only have one entry pointing to the EFI Partition |
| 510-511             | Set to AA55                                                                  |
| 512                 | The rest of the logical block, if any, is reserved. Set to Zero              |

|     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 432 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 448 | 02 | 00 | EE | FF | FF | FF | 01 | 00 | 00 | 00 | FF | FF | FF | FF | 00 | 00 |
| 464 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 480 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 496 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 55 | AA |

| GUID protected MBR entry format |                                                                                                                                                                                          |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Bytes                           | Purpose                                                                                                                                                                                  |
| 0                               | Set to 0x00 to indicate a non-bootable partition. If set to any value other than 0x00 the behavior of this flag on non-UEFI systems is undefined. Must be ignored by UFI implementations |
| 1-3                             | Set to 0x0002000, corresponding to the Starting LBA field                                                                                                                                |
| 4                               | Partition type set to "EE" with indicates the EFI partition                                                                                                                              |
| 5-7                             | Set to the CHS address of the last logical block on the disk. Set to 0xFFFFFFFF if it is not possible to represent the value in this field                                               |
| 8-11                            | Set to 0x00000001 (i.e., the LBA of the GPT Partition Header).                                                                                                                           |
| 12-15                           | Set to the size of the disk minus one. Set to 0xFFFFFFFF if the size of the disk is too large to be represented in this field.                                                           |



## GUID Protective MBR

| Bytes   | Description                                                                  |
|---------|------------------------------------------------------------------------------|
| 0-440   | Unused by UEFI systems                                                       |
| 440-443 | Unused and set to Zero                                                       |
| 444-445 | Unused and set to Zero                                                       |
| 446-509 | MBR partition records that only have one entry pointing to the EFI Partition |
| 510-511 | Set to AA55                                                                  |
| 512     | The rest of the logical block, if any, is reserved. Set to Zero              |

# GUID Protective MBR

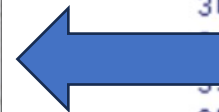
| Bytes   | Description                                                                  |
|---------|------------------------------------------------------------------------------|
| 0-440   | Unused by UEFI systems                                                       |
| 440-443 | Unused and set to Zero                                                       |
| 444-445 | Unused and set to Zero                                                       |
| 446-509 | MBR partition records that only have one entry pointing to the EFI Partition |
| 510-511 | Set to AA55                                                                  |
| 512     | The rest of the logical block, if any, is reserved. Set to Zero              |

| GUID protected MBR entry format |                                                                                                                                                                                          |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Bytes                           | Purpose                                                                                                                                                                                  |
| 0                               | Set to 0x00 to indicate a non-bootable partition. If set to any value other than 0x00 the behavior of this flag on non-UEFI systems is undefined. Must be ignored by UFI implementations |
| 1-3                             | Set to 0x0002000, corresponding to the Starting LBA field                                                                                                                                |
| 4                               | Partition type set to "EE" with indicates the EFI partition                                                                                                                              |
| 5-7                             | Set to the CHS address of the last logical block on the disk. Set to 0xFFFFFFFF if it is not possible to represent the value in this field                                               |
| 8-11                            | Set to 0x00000001 (i.e., the LBA of the GPT Partition Header).                                                                                                                           |
| 12-15                           | Set to the size of the disk minus one. Set to 0xFFFFFFFF if the size of the disk is too large to be represented in this field.                                                           |

Here the bootstrap loader gets a whole partition (no more 512 bytes)

EFI partition is typically 100-500 MB

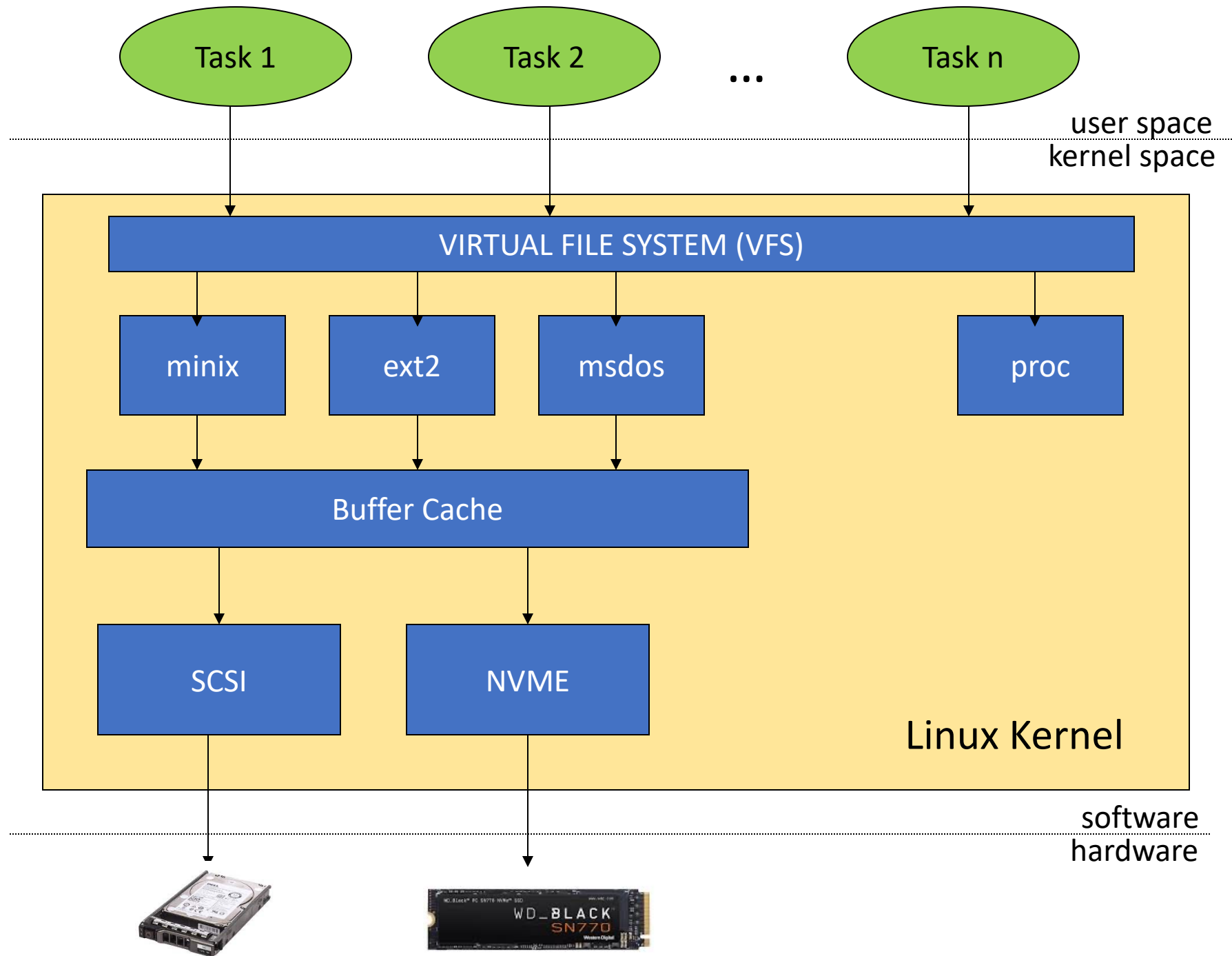
| Offset | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0      | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 16     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 32     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 48     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 64     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 80     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 96     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 112    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 128    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 144    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 160    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 176    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 192    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 208    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 224    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 240    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 256    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 272    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 288    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 304    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 320    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 336    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 352    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 368    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 384    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 400    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 416    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 432    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 448    | 02 | 00 | EE | FF | FF | FF | 01 | 00 | 00 | 00 | FF | FF | FF | FF | 00 | 00 |
| 464    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 480    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 496    | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 55 | AA |



```
sudo dd if=/dev/sda bs=1 count=1024 | hexdump -C
```

```
00000000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
000001c0  02 00 ee ff ff ff 01 00 00 00 ff 87 df e8 00 00 |.....|
000001d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
000001f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 aa |.....U.|
00000200  45 46 49 20 50 41 52 54 00 00 01 00 5c 00 00 00 |EFI PART....\...|
00000210  3b 51 e7 56 00 00 00 00 01 00 00 00 00 00 00 00 |;Q.V.....|
00000220  ff 87 df e8 00 00 00 00 22 00 00 00 00 00 00 00 |.....".....|
00000230  de 87 df e8 00 00 00 00 d0 cc 71 f7 27 f4 7c 44 |.....q.'.|D|
00000240  91 6c 14 53 7c 7c 43 2b 02 00 00 00 00 00 00 00 |.l.S||C+.....|
00000250  80 00 00 00 80 00 00 00 a2 63 be d1 00 00 00 00 |.....c.....|
00000260  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
1024+0 records in
1024+0 records out
1024 bytes (1.0 kB, 1.0 KiB) copied, 0.00254417 s, 402 kB/s
00000400
```

Is this a GPT or MBR partition layout?

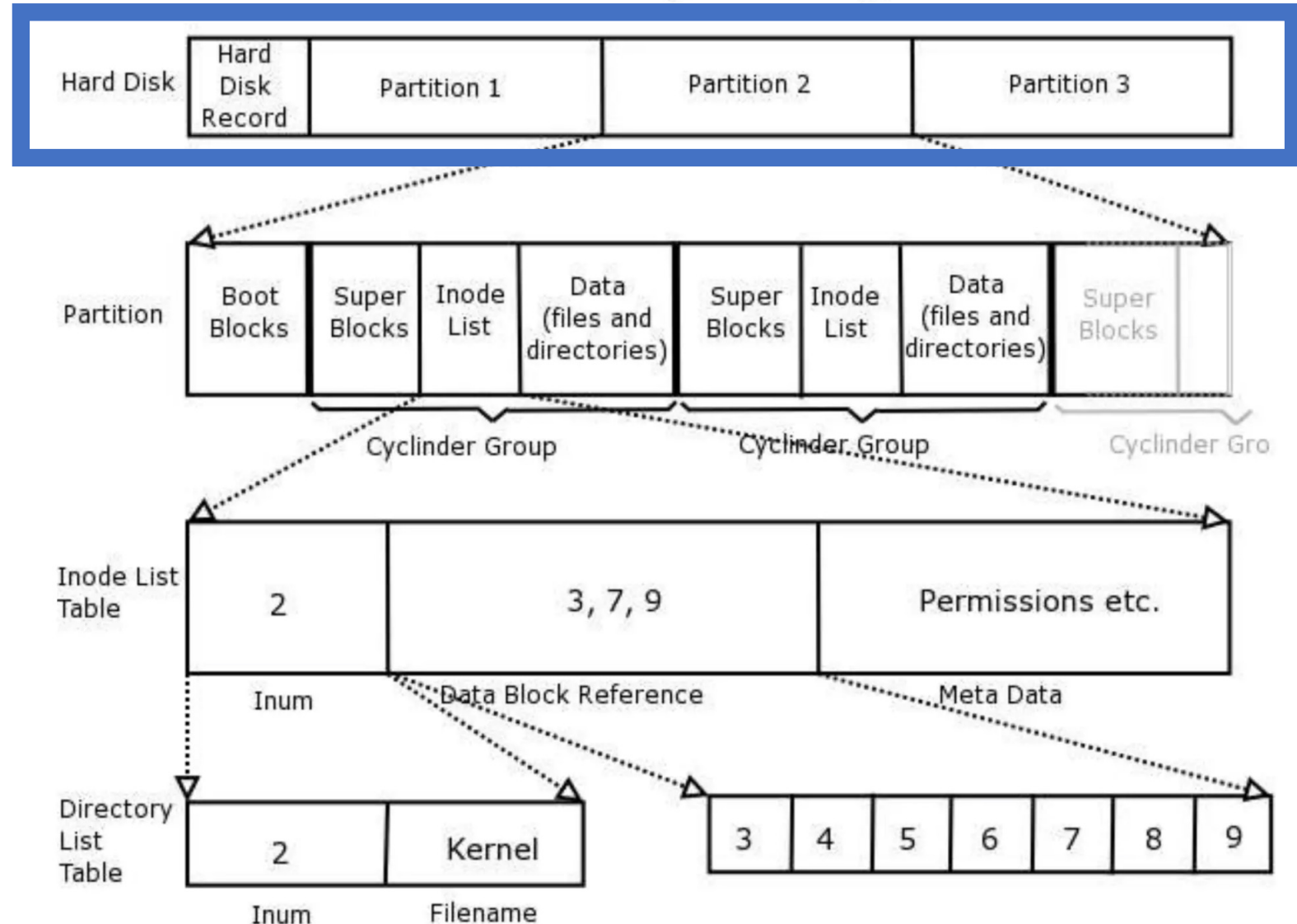


# Partitions

## UNIX File System Layout



- We can do raw reads and writes to disks through `/dev/`
- But are usually much more organized so we can manage the data through partitions



# Generalised Linux Filesystem

- The **superblock** contains all of the information about how the file system is configured, such as block size, block address range, and mount status.
- The **i-nodes** contain the file attributes and a map indicating where the blocks of the file are located on the disk. They are of 128 bytes.
- The **data blocks** are where file contents are stored.
- i-node in position 2 of the table usually points to the entry for the root directory file in the file system.

# File Systems

- In general file systems are simple
  - Abstraction for secondary storage
    - Files
  - Logical organization of files
    - Directories
  - Sharing of data between users/processes
    - Permissions/ACLs

# UNIX File System

- Implemented as part of original UNIX system
  - Ritchie and Thompson, Bell Labs, 1969
- Designed for workgroup scenario
  - Multiple users sharing a single system
- Still forms the basis of all UNIX based file systems

# 5 parts of a UNIX Disk

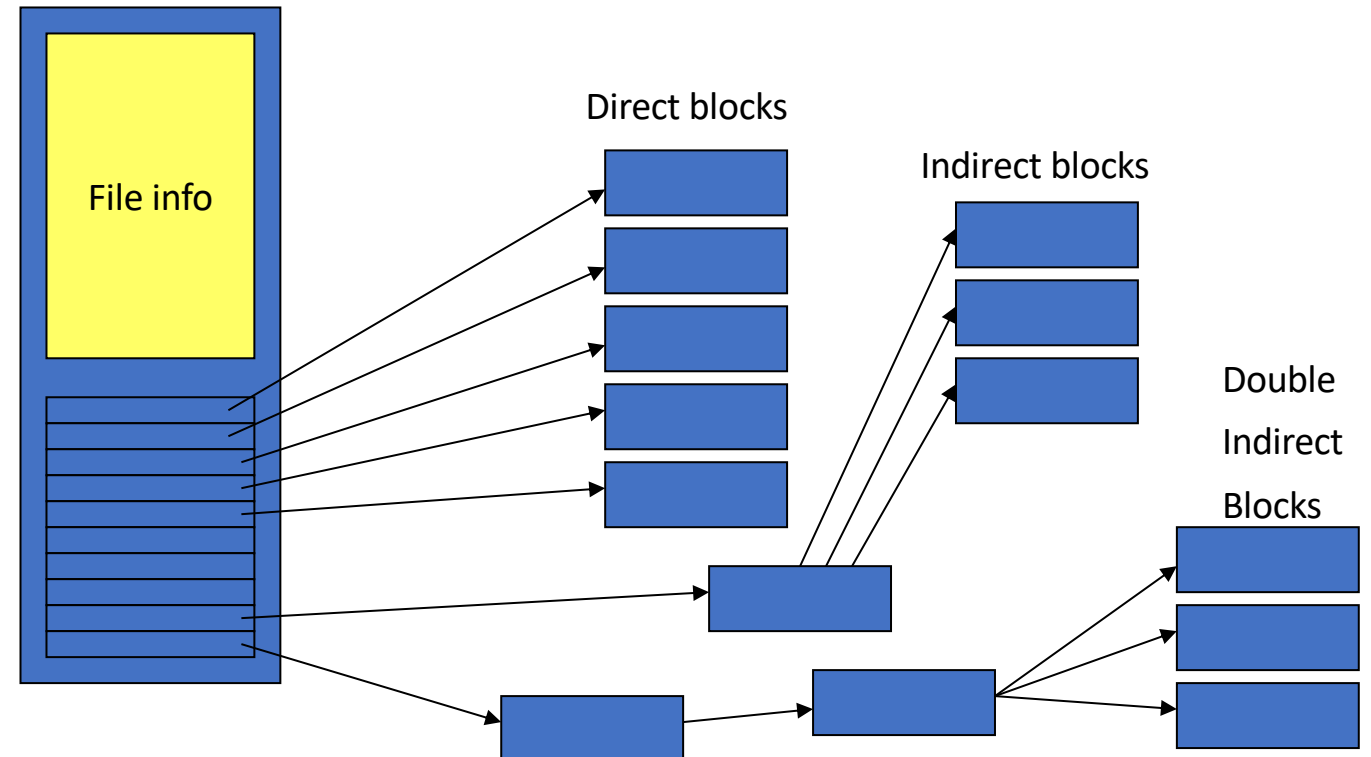
- Boot Block
  - Contains boot loader
- Superblock
  - The file systems “header”
  - Specifies location of file system data structures
- inode area
  - Contains descriptors (inodes) for each file on the disk
  - All inodes are the same size
  - Head of the inode free list is stored in superblock
- File contents area
  - Fixed size blocks containing data
  - Head of freelist stored in superblock
- Swap area
  - Part of disk given to virtual memory system

# Unix directory files

- A directory is a flat file of fixed size entries
- Each entry consists of an inode # and a file name

| i-node number | File name    |
|---------------|--------------|
| 152           | .            |
| 18            | ..           |
| 216           | my_file      |
| 4             | another_file |
| 93            | oh_my_god    |
| 144           | a_directory  |
|               |              |

inode



# Contiguous Allocation

- Allocate each file to contiguous blocks on disk
  - Meta-data includes first block and size of file (can also include creation/access/modification times)
  - OS allocates single chunk of free space
- Advantages
  - Low overhead for meta-data
  - Excellent sequential performance
  - Simple to calculate random addresses
- Disadvantages
  - Horrible external fragmentation (requires compaction)
  - Usually must move entire file to resize it

# Extent Based Allocation

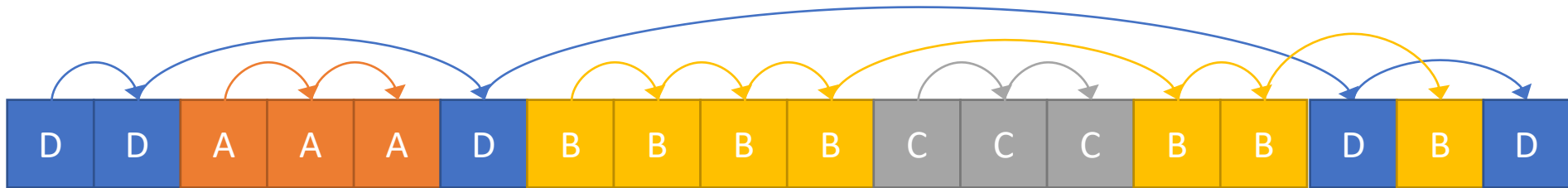
- Allocate multiple contiguous regions (extents)
  - Meta-data: Small array of extents (first block + size)



- Improves contiguous allocation
  - File can grow over time
  - External fragmentation reduced
- Advantages
  - Limited overhead for meta-data
  - Good performance with sequential accesses
  - Simple to calculate random addresses
- Disadvantages
  - External fragmentation can still be a problem
  - Extents can be exhausted (fixed size array in meta-data)

# Linked Allocation

- Allocate linked-list of fixed size blocks
  - Meta-data: location of file's first block
  - Each block stores pointer to next block

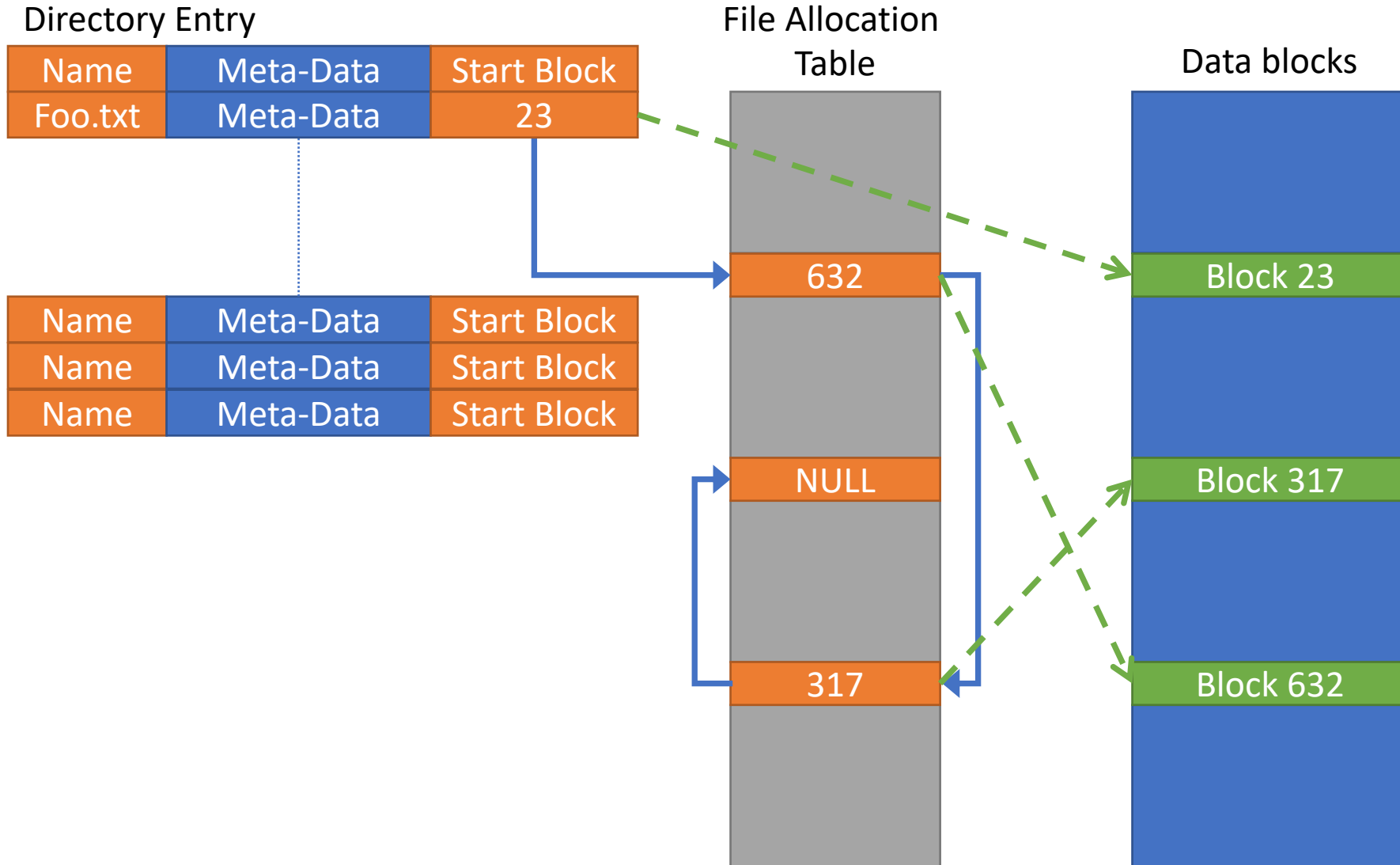


- Advantages
  - No External fragmentation
  - File size can be very dynamic
- Disadvantages
  - Random access takes a long time
  - Sequential accesses can be slow
    - Can try to allocate contiguously to avoid this
  - Very sensitive to corruption

# File Allocation Table (FAT)

- Variation of Linked Allocation
  - Linked list information stored in FAT table (on disk)
  - Meta-data: Location of first block of file
- Comparison to Linked Allocation
  - Same basic advantages and disadvantages
  - Additional disadvantage:
    - Two disk reads for 1 data block
  - Optimization: Cache FAT table in memory

# File-Allocation Table

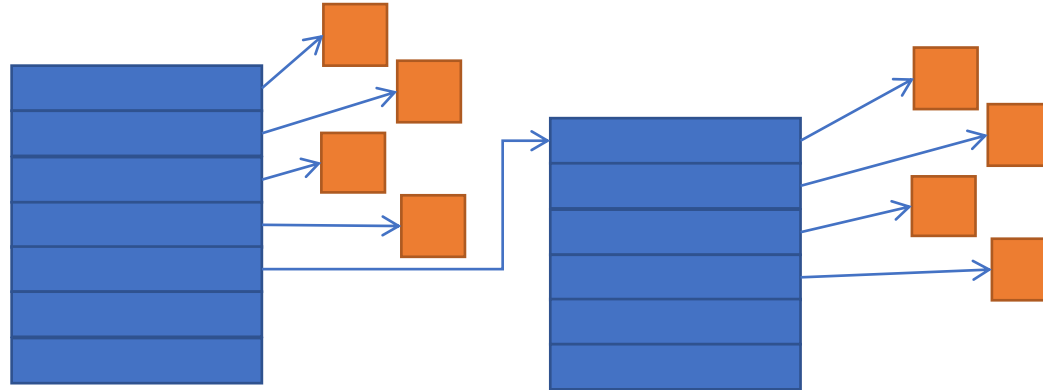


# Indexed Allocation

- Allocate fixed-size blocks for each file
  - Meta-data: Fixed size array of block pointers
    - Array allocated at file creation time
- Advantages
  - No external fragmentation
  - Files can be easily grown, with no limit
  - Supports random access
- Disadvantages
  - Large overhead for meta-data
    - Unneeded pointers are still allocated

# Multi-level Index Files

- Variation of Indexed Allocation
  - Dynamically allocate hierarchy of pointers to blocks as needed
  - Meta-data: Small number of pointers allocated statically
    - Allocate blocks of pointers as needed



- Comparison to Indexed Allocation
  - Advantage: Less wasted space
  - Disadvantage: Random reads require multiple disk reads

# Free Space Management

- How do you remember which blocks are free
  - Operations: Free block, allocate block
- Free List: Linked list of free blocks
  - Advantages: Simple, constant time operations
  - Disadvantage: Quickly loses locality
- Bitmap: Bitmap of all blocks indicating which are free
  - Advantages: Can find sequence of consecutive blocks
  - Disadvantage: Space overhead

# So...

- With a boot block you can boot a machine
  - Stores code for boot loader
- With a superblock you can access a file system
  - Superblock always kept at a fixed location
  - Specifies where you can find FS state information
    - By convention root directory ('/') is stored in second inode
  - Most current boot loaders read superblock to find kernel image

# Inode format

- User and group IDs
- Protection bits
- Access times
- File Type
  - Directory, normal file, symbolic link, etc
- Size
  - Length in bytes
- Block list
  - Location of data blocks in file contents area
- Link Count
  - Number of directories (hard links) referencing this inode

# Unix Inodes and Path Search

- Unix Inodes are not directories
  - Inodes describe where on disk a file's blocks are stored
  - Directories are files
    - Inodes describe where a directory's blocks are stored
- Directory entries map file names to inodes
  - To open `"/foo"`, use Master Block to find inode for `"/"`
    - Open `"/"`, search for entry `"foo"`
    - This entry specifies block number for inode of `"foo"`
  - Read `"foo"`'s inode into memory
    - Get first data block location from inode
    - Read block into memory

```
[matthew@moonshine ~]$ df -i
```

```
matthew@moonshine dev]$ df -i
```

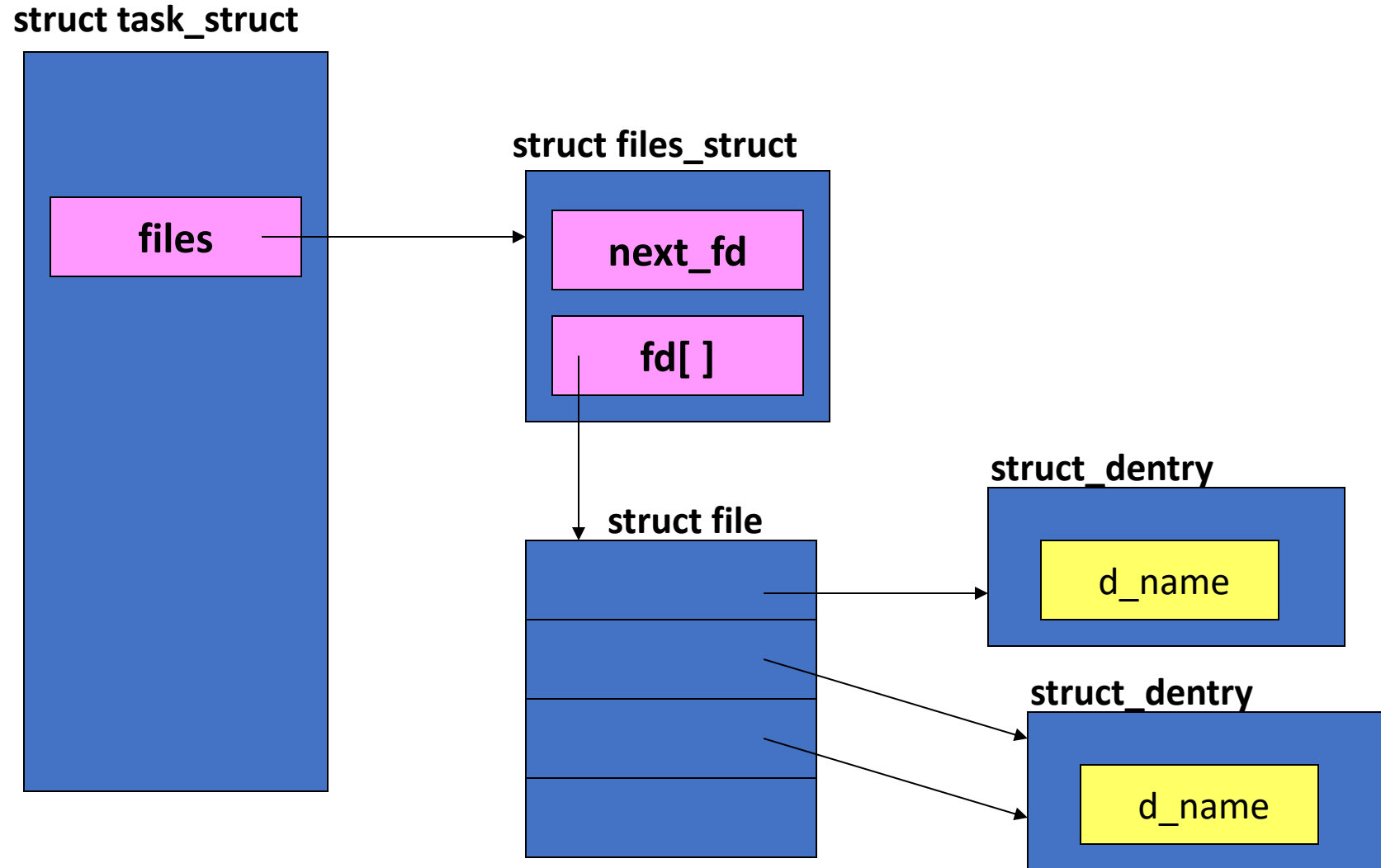
| Filesystem                 | Inodes    | IUsed  | IFree     | IUse% | Mounted on     |
|----------------------------|-----------|--------|-----------|-------|----------------|
| devtmpfs                   | 8157993   | 909    | 8157084   | 1%    | /dev           |
| tmpfs                      | 8163114   | 1      | 8163113   | 1%    | /dev/shm       |
| tmpfs                      | 819200    | 1138   | 818062    | 1%    | /run           |
| /dev/mapper/rl_dhcp52-root | 36700160  | 153255 | 36546905  | 1%    | /              |
| /dev/mapper/rl_dhcp52-home | 434087936 | 774    | 434087162 | 1%    | /home          |
| /dev/sda2                  | 524288    | 30     | 524258    | 1%    | /boot          |
| /dev/sda1                  | 0         | 0      | 0         | -     | /boot/efi      |
| tmpfs                      | 1632622   | 21     | 1632601   | 1%    | /run/user/1000 |

Inodes are a limited resource. End users usually know that the amount of storage space is limited, but the number of inodes available is just as important.

Inodes in the table above gives the total number of available inodes. Iused is how many have been used so far. If we run out no more files or directories can be created.

# Each task opens its own files

- Each task has its own **task\_struct**
- **task\_struct** points to a **files\_struct** (file descriptor table)
- **files\_struct** contains:
  - next\_fd (next available file descriptor)
  - fd[ ] (array of open files)
- Each entry points to a **struct file** (open file instance)
- **struct file** links to a **struct dentry** (directory entry)
- **dentry** holds the **d\_name** (filename in the filesystem)
- **file descriptors are per-process, even when files are shared**



# Why File Descriptors Matter in HPC

- Each process has its own set of file descriptors
- Task (file) **Per-process tables** – each rank has its own open files
  - \* **Limits** – too many opens → “*Too many files*” errors
- File **Performance** – metadata servers choke on massive parallel opens

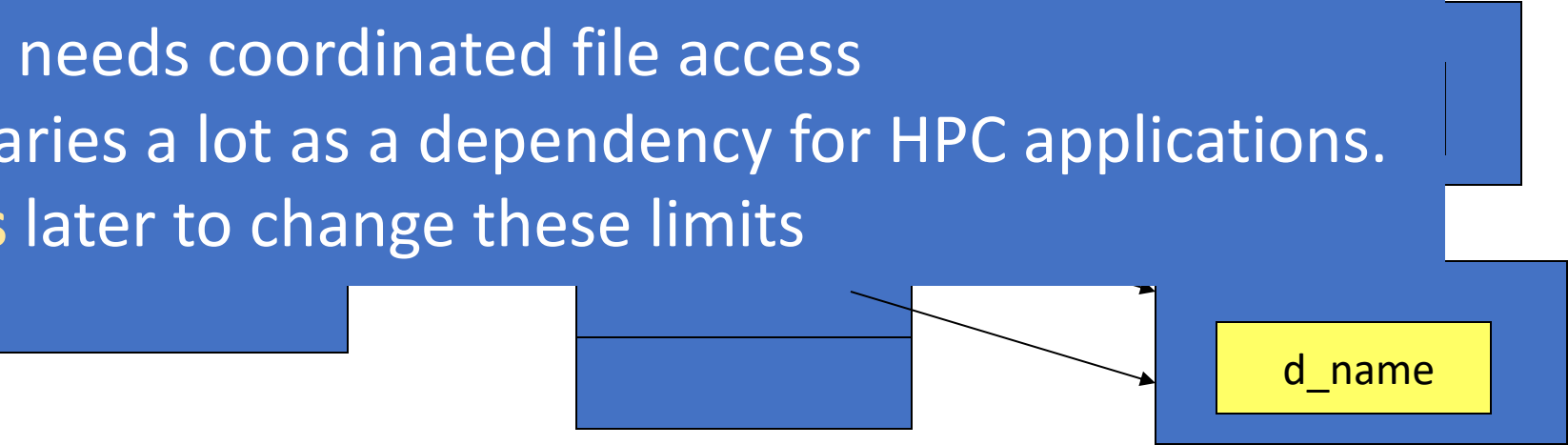
## HPC Coordinated output:

- Each process uses MPI-IO, HDF5, NetCDF; aggregate I/O

- Structure (directory) **Takeaway:** scalable HPC needs coordinated file access

- Dependencies You will see NetCDF libraries a lot as a dependency for HPC applications.
  - the \* You will change **ulimits** later to change these limits

- file descriptors are per-process, even when files are shared



d\_name

## Common Local Filesystems in HPC

### •ext4

-  Stable, default on many Linux distros
-  Good all-around performance
-  Limited scalability for very large filesystems (>100 TB)

### •XFS

-  Excellent scalability (large files, high concurrency)
-  Common in HPC clusters for scratch and compute nodes
-  Metadata-heavy workloads can be slower than ext4

### •ReiserFS (*historical*)

-  Once good at handling many small files
-  Deprecated/obsolete — not used in modern HPC

### •ZFS

-  Advanced features: checksumming, snapshots, compression
-  Very resilient to corruption
-  Higher memory use, licensing issues in Linux kernel

### •Btrfs

-  Copy-on-write, snapshots, checksums
-  Less mature under heavy HPC I/O;
- performance not as predictable

Developed as the successor to ext3 in the Linux kernel (2006–2008).

**Lead developer:** Theodore Ts'o

Originally created by **Silicon Graphics (SGI)** in 1993 for IRIX. Ported to Linux in 2001; Linux kernel by IBM/Redhat

Created by **Hans Reiser** and his company **Namesys**  
Introduced with Linux kernel 2.4 (2001).  
Now deprecated; Namesys is defunct.

Developed by **Sun Microsystems** (2001–2005) for Solaris.  
Now maintained by the **OpenZFS** project after Oracle closed the original code.

Initiated by **Oracle** (2007).

Main early developer: Chris Mason contributions by Meta.

## Common Local Filesystems in HPC

### •ext4



Ted Ts'o – Ext3/4

First North American Linux kernel developer (early 1990s).

Wrote the **Linux random number generator** (/dev/random) in his spare time

Ext4 wasn't meant to be flashy

**“boring, reliable plumbing,”** - Ted

•  very resilient to corruption

•  Higher memory use, licensing issues in Linux kernel

the original code.

## Common Local Filesystems in HPC

### •ext4

-  Stable, default on many Linux distros
-  Developed as the successor to ext2 in the Linux kernel

## • Why XFS dominates in HPC

### •

-  **Linux-native:** XFS is in the mainline Linux kernel, with long-term enterprise support (Red Hat, CentOS, RHEL, SUSE). Most HPC clusters are Linux-based.
-  or IRIX.

### •

### •

### •

**Performance:** XFS scales very well for large files, parallel writes, and high-concurrency workloads — a good match for scratch and job output.

### •ZFS

-  Advanced features: checksumming, snapshots, compression
  -  Very resilient to corruption
  -  Higher memory use, licensing issues in Linux kernel
- Developed by **Sun Microsystems** (2001–2005) for Solaris. Now maintained by the **OpenZFS** project after Oracle closed the original code.

```
Last login: Fri Feb  2 10:31:15 on ttys000
matthew@dhcp178 ~ % ssh matthew@129.24.245.16
matthew@129.24.245.16's password:
Last login: Mon Jan 29 10:11:40 2024 from
129.24.246.178
[matthew@moonshine ~]$
```

```
[matthew@moonshine ~]$ sudo parted -l
```

```
Model: DELL PERC H310 (scsi)
```

```
Disk /dev/sda: 1000GB
```

```
Sector size (logical/physical): 512B/512B
```

```
Partition Table: gpt
```

```
Disk Flags:
```

| Number | Start  | End    | Size   | File system | Name                 | Flags     |
|--------|--------|--------|--------|-------------|----------------------|-----------|
| 1      | 1049kB | 630MB  | 629MB  | fat32       | EFI System Partition | boot, esp |
| 2      | 630MB  | 1704MB | 1074MB | xfs         |                      |           |
| 3      | 1704MB | 1000GB | 998GB  |             |                      | lvm       |

```
Model: USB DISK 3.0 (scsi)
```

```
Disk /dev/sdb: 15.5GB
```

```
Sector size (logical/physical): 512B/512B
```

```
Partition Table: msdos
```

```
Disk Flags:
```

| Number | Start | End    | Size   | Type    | File system | Flags |
|--------|-------|--------|--------|---------|-------------|-------|
| 2      | 340kB | 7604kB | 7264kB | primary |             | esp   |

```
Last login: Fri Feb  2 10:31:21 on ttys001
matthew@dhcp178 ~ % ssh matthew@129.24.245.16
matthew@129.24.245.16's password:
Last login: Fri Feb  2 11:31:49 2024 from 129.24.246.178
[matthew@moonshine ~]$
```

Open another terminal to your server

```
[matthew@moonshine ~]$ sudo udevadm monitor --kernel  
[sudo] password for matthew:  
monitor will print the received events for:  
KERNEL – the kernel uevent
```

```
[matthew@moonshine ~]$ lsblk
```

```
[matthew@moonshine ~]$ sudo umount /dev/sdb
```

```
[matthew@moonshine ~]$ cat /etc/fstab
```

```
/dev/mapper/rl_dhcp52-root / xfs defaults 0 0
UUID=a447244c-dab4-41a2-b908-8a931134c113
/boot xfs defaults 0 0
UUID=A57E-CA77 /boot/efi vfat umask=0077,shortname=winnt 0 2
/dev/mapper/rl_dhcp52-home /home xfs defaults 0 0
/dev/mapper/rl_dhcp52-swap none swap defaults 0 0
```

If your system runs out of memory (RAM) then the least recently used data is moved to the **swap space** on the HDD.

This keeps you from crashing – but reading and writing data from disk is about 1/100<sup>th</sup> the speed of RAM.

So, if lots of data gets moved to disk the system becomes so slow it is unusable. This is called **disk thrashing**.

| Amount of RAM in the system | Recommended swap space     | Recommended swap space if allowing for hibernation |
|-----------------------------|----------------------------|----------------------------------------------------|
| ≤ 2 GB                      | 2 times the amount of RAM  | 3 times the amount of RAM                          |
| > 2 GB – 8 GB               | Equal to the amount of RAM | 2 times the amount of RAM                          |
| > 8 GB – 64 GB              | At least 4 GB              | 1.5 times the amount of RAM                        |
| > 64 GB                     | At least 4 GB              | Hibernation not recommended                        |

```
[matthew@moonshine ~]$ sudo fdisk /dev/sdb  
[sudo] password for matthew:
```

```
Welcome to fdisk (util-linux 2.37.4).  
Changes will remain in memory only, until you decide to write  
them.  
Be careful before using the write command.
```

```
The device contains 'iso9660' signature and it will be  
removed by a write command. See fdisk(8) man page and --wipe  
option for more details.
```

Command (m for help): p

Why do you think it says DOS and EFI?

Disk /dev/sdb: 14.46 GiB, 15525216256 bytes, 30322688 sectors

Disk model: USB DISK 3.0

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: dos

Disk identifier: 0x73b44ec8

| Device    | Boot | Start | End     | Sectors | Size | Id | Type                   |
|-----------|------|-------|---------|---------|------|----|------------------------|
| /dev/sdb1 | *    | 0     | 3293407 | 3293408 | 1.6G | 0  | Empty                  |
| /dev/sdb2 |      | 664   | 14851   | 14188   | 6.9M | ef | EFI (FAT-<br>12/16/32) |

Command (m for help): q

Why do you think it says DOS and EFI?

```
[matthew@moonshine ~]$ sudo dd if=/dev/sdb  
of=Rocky9.4_minimal.iso bs=4M status=progress
```

```
15509602816 bytes (16 GB, 14 GiB) copied, 415 s, 37.4 MB/s  
30322688+0 records in  
30322688+0 records out  
15525216256 bytes (16 GB, 14 GiB) copied, 416.363 s, 37.3 MB/s
```

```
[matthew@moonshine ~]$ lsblk
```

| NAME            | MAJ:MIN | RM | SIZE   | RO | TYPE |           |
|-----------------|---------|----|--------|----|------|-----------|
| MOUNTPOINTS     |         |    |        |    |      |           |
| sda             | 8:0     | 0  | 931G   | 0  | disk |           |
| —sda1           | 8:1     | 0  | 600M   | 0  | part | /boot/efi |
| —sda2           | 8:2     | 0  | 1G     | 0  | part | /boot     |
| —sda3           | 8:3     | 0  | 929.4G | 0  | part |           |
| —rl_dhcp52-root | 253:0   | 0  | 70G    | 0  | lvm  | /         |
| —rl_dhcp52-swap | 253:1   | 0  | 31.5G  | 0  | lvm  | [SWAP]    |
| —rl_dhcp52-home | 253:2   | 0  | 828G   | 0  | lvm  | /home     |
| sdb             | 8:16    | 1  | 14.5G  | 0  | disk |           |
| —sdb1           | 8:17    | 1  | 1.6G   | 0  | part |           |
| —sdb2           | 8:18    | 1  | 6.9M   | 0  | part |           |
| sr0             | 11:0    | 1  | 1024M  | 0  | rom  |           |

```
[matthew@moonshine ~]$ sudo fdisk /dev/sdb  
[sudo] password for matthew:
```

```
Welcome to fdisk (util-linux 2.37.4).  
Changes will remain in memory only, until you decide to write  
them.  
Be careful before using the write command.
```

```
The device contains 'iso9660' signature and it will be removed  
by a write command. See fdisk(8) man page and --wipe option for  
more details.
```

```
Command (m for help):
```

```
[matthew@moonshine ~]$ sudo fdisk /dev/sdb  
[sudo] password for matthew:
```

Welcome to fdisk (util-linux 2.37.4).  
Changes will remain in memory only, until you decide to write  
them.

Be careful before using the write command.

The device contains 'iso9660' signature. The signature will be removed  
by a write command. See fdisk(8) for the --signature option for  
more details.

Command (m for help):



```
Command (m for help): p
Disk /dev/sdb: 14.46 GiB, 15525216256 bytes, 30322688 sectors
Disk model: USB DISK 3.0
Geometry: 255 heads, 63 sectors/track, 14806 cylinders
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x73b44ec8
```

| Device    | Boot | Start | End     | Sectors | Size | Id | Type                   |
|-----------|------|-------|---------|---------|------|----|------------------------|
| /dev/sdb1 | *    | 0     | 3293407 | 3293408 | 1.6G | 0  | Empty                  |
| /dev/sdb2 |      | 664   | 14851   | 14188   | 6.9M | ef | EFI (FAT-<br>12/16/32) |

```
Command (m for help):
```

That it is listed as empty is a quirk of iso9660.  
That's where your installation data lives.

The device contains 'iso9660' signature and it will be removed by a write command. See fdisk(8) man page and --wipe option for more details.

Command (m for help): d

Partition number (1,2, default 2): 1

Partition 1 has been deleted.

Command (m for help): p

Disk /dev/sdb: 14.46 GiB, 15525216256 bytes, 30322688 sectors

Disk model: USB DISK 3.0

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: dos

Disk identifier: 0x73b44ec8

The partition table is gone so no partitions are displayed.

But Fdisk doesn't actually ask the kernel to make changes until after you hit **w** to write the changes.

Command (m for help): w

The partition table has been altered.  
Calling ioctl() to re-read partition table.  
Syncing disks.

The partition table is gone so no partitions are displayed.

But Fdisk doesn't actually ask the kernel to make changes until after you hit **w** to write the changes.

When you use the “w” command fdisk makes function calls to the kernel and the kernel makes the changes to the USB drive

```
KERNEL[1122527.084021] remove    /devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.2/2-1.2:1.0/host0/target0:0:0/0:0:0:0/block/sdb/sdb1 (block)
```

```
KERNEL[1122527.084060] remove    /devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.2/2-1.2:1.0/host0/target0:0:0/0:0:0:0/block/sdb/sdb2 (block)
```

```
KERNEL[1122527.086394] change    /devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.2/2-1.2:1.0/host0/target0:0:0/0:0:0:0/block/sdb (block)
```

```
KERNEL[1122527.093141] change    /devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.2/2-1.2:1.0/host0/target0:0:0/0:0:0:0/block/sdb (block)
```

```
KERNEL[1122527.094259] change    /devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.2/2-1.2:1.0/host0/target0:0:0/0:0:0:0/block/sdb (block)
```

Command (m for help): w

The partition table has been altered.  
Calling ioctl() to re-read partition table.  
Syncing disks.

The partition table is gone so no partitions are displayed.

But Fdisk doesn't actually ask the kernel to make changes until after you hit **w** to write the changes.

```
[matthew@moonshine ~]$ sudo fdisk /dev/sdb
```

```
Command (m for help): p
```

```
Disk /dev/sdb: 14.46 GiB, 15525216256 bytes, 30322688 sectors
```

```
Disk model: USB DISK 3.0
```

```
Units: sectors of 1 * 512 = 512 bytes
```

```
Sector size (logical/physical): 512 bytes / 512 bytes
```

```
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disklabel type: dos
```

```
Disk identifier: 0x73b44ec8
```

| Device    | Boot | Start | End   | Sectors | Size | Id | Type               |
|-----------|------|-------|-------|---------|------|----|--------------------|
| /dev/sdb2 |      | 664   | 14851 | 14188   | 6.9M | ef | EFI (FAT-12/16/32) |

```
Command (m for help): d
```

```
Selected partition 2
```

```
Partition 2 has been deleted.
```

```
Command (m for help): w
```

```
[matthew@moonshine ~]$ lsblk
```

| NAME             | MAJ:MIN | RM | SIZE   | RO | TYPE | MOUNTPOINTS |
|------------------|---------|----|--------|----|------|-------------|
| sda              | 8:0     | 0  | 931G   | 0  | disk |             |
| └─sda1           | 8:1     | 0  | 600M   | 0  | part | /boot/efi   |
| └─sda2           | 8:2     | 0  | 1G     | 0  | part | /boot       |
| └─sda3           | 8:3     | 0  | 929.4G | 0  | part |             |
| └─rl_dhcp52-root | 253:0   | 0  | 70G    | 0  | lvm  | /           |
| └─rl_dhcp52-swap | 253:1   | 0  | 31.5G  | 0  | lvm  | [SWAP]      |
| └─rl_dhcp52-home | 253:2   | 0  | 828G   | 0  | lvm  | /home       |
| sdb              | 8:16    | 1  | 14.5G  | 0  | disk |             |
| sr0              | 11:0    | 1  | 1024M  | 0  | rom  |             |

```
[matthew@localhost ~]$ sudo fdisk /dev/sdb
```

```
Welcome to fdisk (util-linux 2.37.4).
```

```
Changes will remain in memory only, until you decide to write  
them.
```

```
Be careful before using the write command.
```

```
Command (m for help): n
```

```
Partition type
```

```
    p    primary (0 primary, 0 extended, 4 free)
```

```
    e    extended (container for logical partitions)
```

```
Select (default p): p
```

```
Partition number (1-4, default 1): 1
```

```
First sector (2048-30322687, default 2048): {press enter}
```

```
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-30322687,  
default 30322687): {press enter}
```

```
Created a new partition 1 of type 'Linux' and of size 14.5 GiB.
```

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table.

Syncing disks.

```
[matthew@localhost ~]$ lsblk
```

| NAME             | MAJ:MIN | RM | SIZE   | RO | TYPE | MOUNTPOINTS |
|------------------|---------|----|--------|----|------|-------------|
| sda              | 8:0     | 0  | 931G   | 0  | disk |             |
| └─sda1           | 8:1     | 0  | 600M   | 0  | part | /boot/efi   |
| └─sda2           | 8:2     | 0  | 1G     | 0  | part | /boot       |
| └─sda3           | 8:3     | 0  | 929.4G | 0  | part |             |
| └─rl_dhcp52-root | 253:0   | 0  | 70G    | 0  | lvm  | /           |
| └─rl_dhcp52-swap | 253:1   | 0  | 31.5G  | 0  | lvm  | [SWAP]      |
| └─rl_dhcp52-home | 253:2   | 0  | 828G   | 0  | lvm  | /home       |
| sdb              | 8:16    | 1  | 14.5G  | 0  | disk |             |
| └─sdb1           | 8:17    | 1  | 14.5G  | 0  | part |             |
| sr0              | 11:0    | 1  | 1024M  | 0  | rom  |             |

```
[matthew@moonshine ~]$ sudo mkfs.ext4 /dev/sdb1
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 3790080 4k blocks and 948416 inodes
Filesystem UUID: 6e7aaea1-7e15-41dd-8fd3-b9b5248e6636
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632,
2654208

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks):
done
Writing superblocks and filesystem accounting information: done
```

```
[matthew@moonshine ~]$ sudo mkfs.xfs -f /dev/sdb1
meta-data=/dev/sdb1          isize=512    agcount=4, agsize=947520 blks
                =                       sectsz=512   attr=2, projid32bit=1
                =                       crc=1        finobt=1, sparse=1, rmapbt=0
                =                       reflink=1     bigtime=1 inobtcount=1
nnext64=0
data                =                       bsize=4096   blocks=3790080, imaxpct=25
                =                       sunit=0      swidth=0 blks
naming              =version 2             bsize=4096   ascii-ci=0, ftype=1
log                 =internal log          bsize=4096   blocks=16384, version=2
                =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime            =none                  extsz=4096   blocks=0, rtextents=0
```

```
[matthew@moonshine ~]$ sudo mount /dev/sdb1 /mnt  
[matthew@moonshine ~]$
```

Now we can mount the filesystem.

The mount command takes a block device (such as a partition) and an existing directory file.

/mnt exists for this purpose but you can put any directory as the argument.

```
matthew@moonshine mnt]$ sudo chown matthew: /mnt  
[matthew@moonshine mnt]$ cd /mnt  
[matthew@moonshine mnt]$ touch test.txt  
[matthew@moonshine mnt]$ ls  
test.txt  
[matthew@moonshine mnt]$ cd ~
```

By default only the root user can write to this external usb mount.

We use `chmod` to make matthew the owner and `touch` to create a new file.

# Input/Output Monitoring is important for HPC



EPEL: Extra Packages  
for Enterprise Linux

```
#First enable the EPEL repository
```

```
[matthew@moonshine ~]$ sudo dnf install epel-release -y
```

```
#Update repo cache
```

```
[matthew@moonshine ~]$ sudo dnf update -y
```

```
# Install iotop
```

```
[matthew@moonshine ~]$ sudo dnf install iotop -y
```

Only one of the users on your machine should run sudo commands.  
Coordinate with your partner if you have one.

# Input/Output Monitoring is important for HPC

```
[matthew@moonshine ~]$ sudo yum install iotop
```

```
[sudo] password for matthew:
```

```
Last metadata expiration check: 0:25:53 ago on Mon 05 Feb 2024 10:37:13 AM CST.
```

```
Dependencies resolved.
```

| =====       |              |            |            |      |
|-------------|--------------|------------|------------|------|
| Package     | Architecture | Version    | Repository | Size |
| =====       |              |            |            |      |
| Installing: |              |            |            |      |
| iotop       | noarch       | 0.63-0.el9 | baseos     | 62 k |

```
Transaction Summary
```

```
=====
```

|         |           |
|---------|-----------|
| Install | 1 Package |
|---------|-----------|

Now let's restore our ISO image to the USB drive:

```
[matthew@moonshine ~]$ sudo dd if=Rocky9.4_minimal.iso  
of=/dev/sdb bs=4M status=progress  
[sudo] password for matthew:  
12494831616 bytes (12 GB, 12 GiB) copied, 126 s, 99.1 MB/
```

# Input/Output Monitoring is important for HPC

```
[matthew@moonshine ~]$ sudo iotop
```

```
Total DISK READ : 0.00 B/s | Total DISK WRITE :      8.82 M/s  
Actual DISK READ: 0.00 B/s | Actual DISK WRITE:    12.97 M/s
```

| TID | PRI0 | USER | DISK READ | DISK WR | TID | PRI0 | USER | DISK READ | DISK WRITE | COMMAND |
|-----|------|------|-----------|---------|-----|------|------|-----------|------------|---------|
|-----|------|------|-----------|---------|-----|------|------|-----------|------------|---------|

|        |      |      |          |          |                                                              |  |  |  |  |  |
|--------|------|------|----------|----------|--------------------------------------------------------------|--|--|--|--|--|
| 171200 | be/4 | root | 0.00 B/s | 8.82 M/s | dd if=Rocky9.1_minimal.iso of=/dev/sdb bs=4M status=progress |  |  |  |  |  |
|--------|------|------|----------|----------|--------------------------------------------------------------|--|--|--|--|--|

|    |      |      |          |          |                                                   |  |  |  |  |  |
|----|------|------|----------|----------|---------------------------------------------------|--|--|--|--|--|
| 1  | be/4 | root | 0.00 B/s | 0.00 B/s | systemd --switched-root --system --deserialize 31 |  |  |  |  |  |
| 2  | be/4 | root | 0.00 B/s | 0.00 B/s | [kthreadd]                                        |  |  |  |  |  |
| 3  | be/0 | root | 0.00 B/s | 0.00 B/s | [rcu_gp]                                          |  |  |  |  |  |
| 4  | be/0 | root | 0.00 B/s | 0.00 B/s | [rcu_par_gp]                                      |  |  |  |  |  |
| 5  | be/0 | root | 0.00 B/s | 0.00 B/s | [slub_flushwq]                                    |  |  |  |  |  |
| 6  | be/0 | root | 0.00 B/s | 0.00 B/s | [netns]                                           |  |  |  |  |  |
| 8  | be/0 | root | 0.00 B/s | 0.00 B/s | [kworker/0:0H-events_highpri]                     |  |  |  |  |  |
| 11 | be/0 | root | 0.00 B/s | 0.00 B/s | [mm_percpu_wq]                                    |  |  |  |  |  |
| 12 | be/4 | root | 0.00 B/s | 0.00 B/s | [kworker/u64:1-mlx4]                              |  |  |  |  |  |
| 13 | be/4 | root | 0.00 B/s | 0.00 B/s | [rcu_tasks_kthre]                                 |  |  |  |  |  |
| 14 | be/4 | root | 0.00 B/s | 0.00 B/s | [rcu_tasks_rude_]                                 |  |  |  |  |  |
| 15 | be/4 | root | 0.00 B/s | 0.00 B/s | [rcu_tasks_trace]                                 |  |  |  |  |  |
| 16 | be/4 | root | 0.00 B/s | 0.00 B/s | [ksoftirqd/0]                                     |  |  |  |  |  |
| 17 | be/4 | root | 0.00 B/s | 0.00 B/s | [pr/tty0]                                         |  |  |  |  |  |
| 18 | be/4 | root | 0.00 B/s | 0.00 B/s | [rcu_preempt]                                     |  |  |  |  |  |
| 19 | rt/4 | root | 0.00 B/s | 0.00 B/s | [migration/0]                                     |  |  |  |  |  |
| 20 | rt/4 | root | 0.00 B/s | 0.00 B/s | [idle_inject/0]                                   |  |  |  |  |  |

Now let's restore our ISO image to the USB drive:

```
[matthew@moonshine ~]$ sudo dd if=Rocky9.1_minimal.iso  
of=/dev/sdb bs=4M status=progress  
[sudo] password for matthew:  
12494831616 bytes (12 GB, 12 GiB) copied, 126 s, 315 MB/s
```



Notice that the write speed is faster than the read speed! Compare it to the write speed reported by iotop.

Does that make sense?

Now let's restore our ISO image to the USB drive:

```
[matthew@moonshine ~]$ sudo dd if=Rocky9.1_minimal.iso  
of=/dev/sdb bs=4M status=progress  
[sudo] password for matthew:  
12494831616 bytes (12 GB, 12 GiB) copied, 126 s, 315 MB/s
```



The write is actually going to RAM. The kernel will write to the USB disk when it gets around to it.

Dangerous if you unplug the USB now.

Now let's restore our ISO image to the USB drive:

```
[matthew@moonshine ~]$ sudo dd if=Rocky9.1_minimal.iso  
of=/dev/sdb bs=4M status=progress oflag=direct  
[sudo] password for matthew:  
12494831616 bytes (12 GB, 12 GiB) copied, 126 s, 11 MB/s
```



Oflag=direct in dd will tell the kernel to bypass the RAM cache.