CSE 350: Advanced Data Structures and Indexes (Spring 2026)

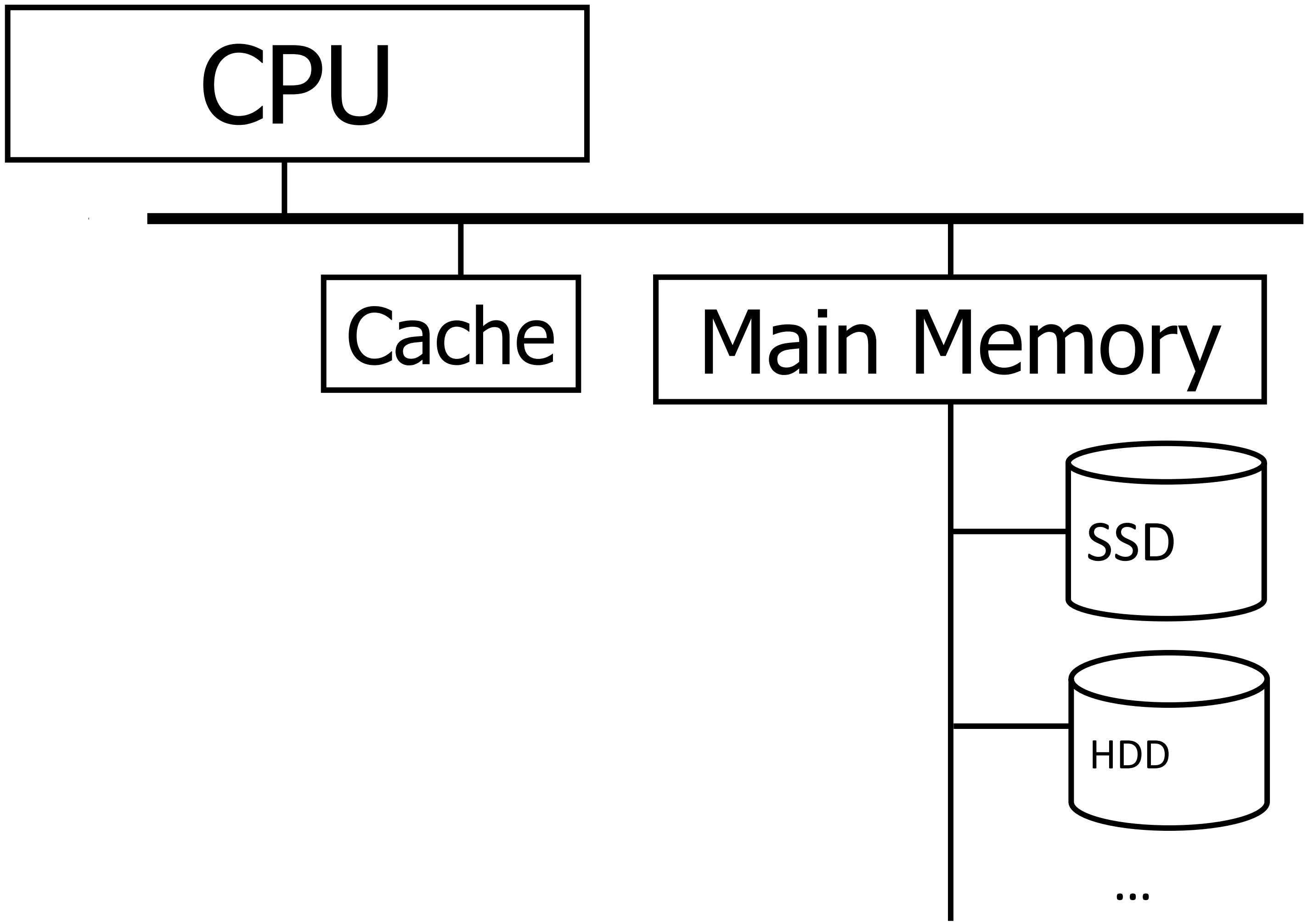
Lecture 2: Physical Storage Devices;

1/27/2026

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Typical (& oversimplified) computer architecture

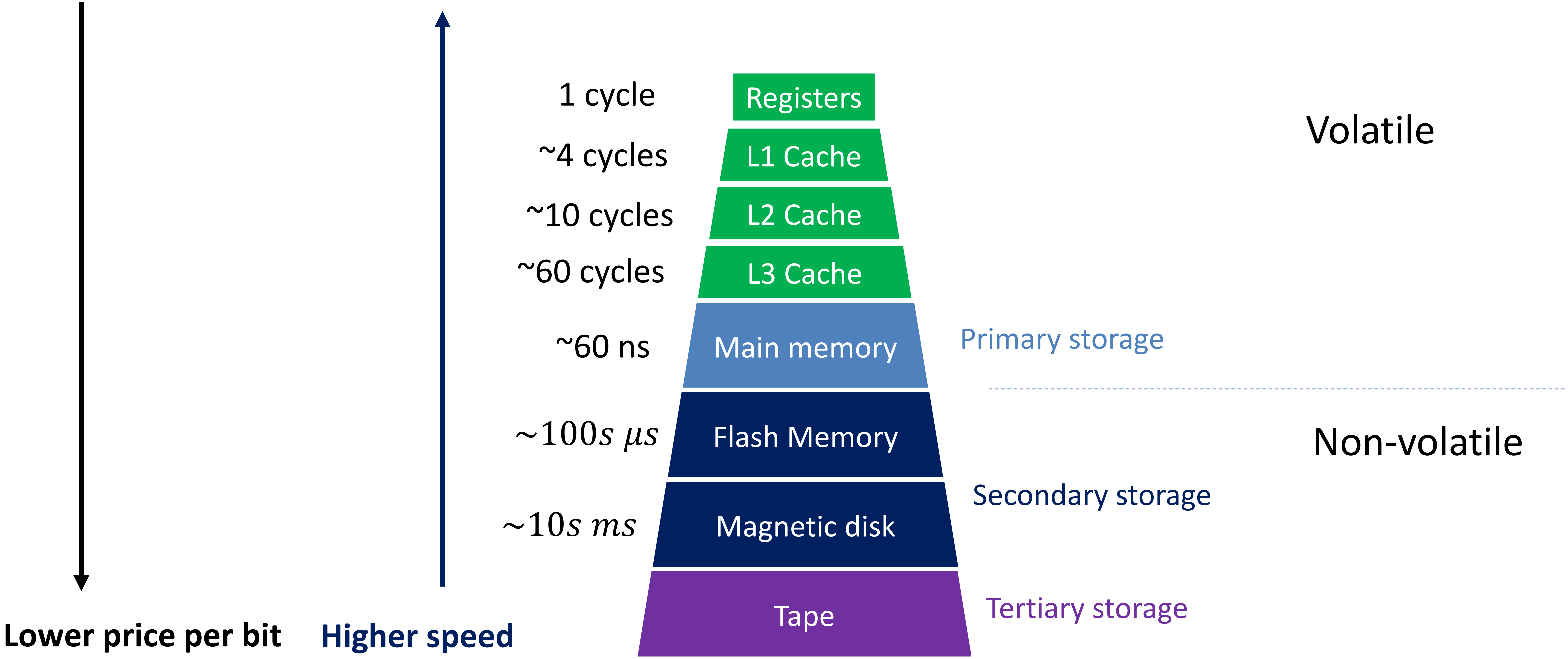
- A simplistic view of a computer



Typical Computer

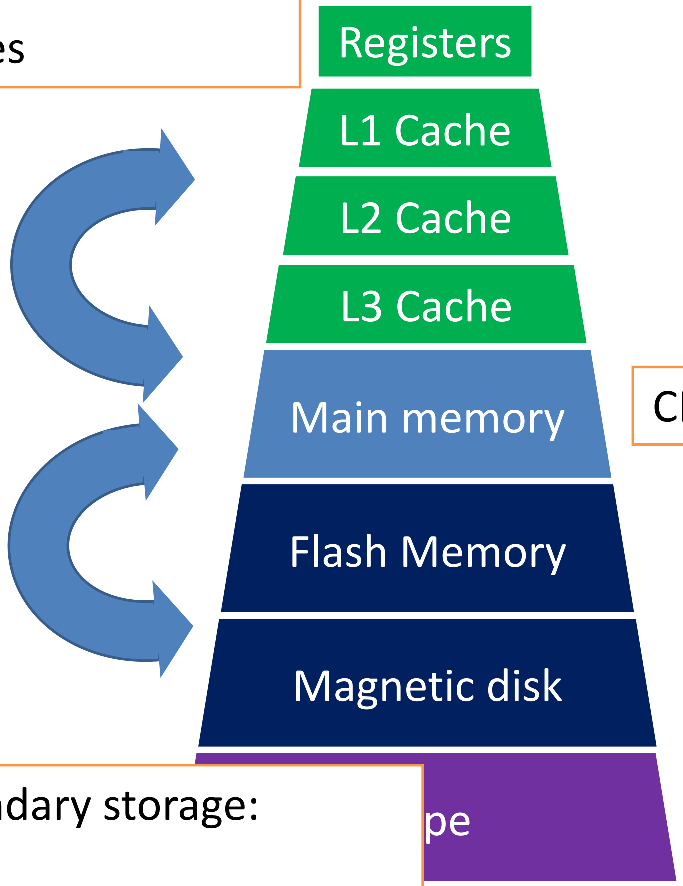
Secondary Storage

Storage Hierarchy



Data Transfers

Between cache and main memory:
hardware/OS controlled
usually in small units of cache lines



Volatile

CPU operates on main memory (byte addressable)

Non-volatile

Between main memory and secondary storage:
DBMS controlled (read/write)
usually with large block I/O

Non-volatile storage

- Common non-volatile (secondary) storage
  - Flash memory (e.g., SSD)
  - Magnetic disk

- Advantages
  - Cheaper -- can store much more data than memory with the same cost
  - Non-volatile – data are saved in server shutdown/power failure

- Disadvantages
  - Block device: read/write in the units of sectors (usually 512B/4096B)
  - Higher latency: usually >= 1 – 2 orders of magnitude slower than main memory

- Tertiary storage: tape (sequential I/O only)
  - Very slow but inexpensive; good for archiving data

Closer look at non-volatile storage

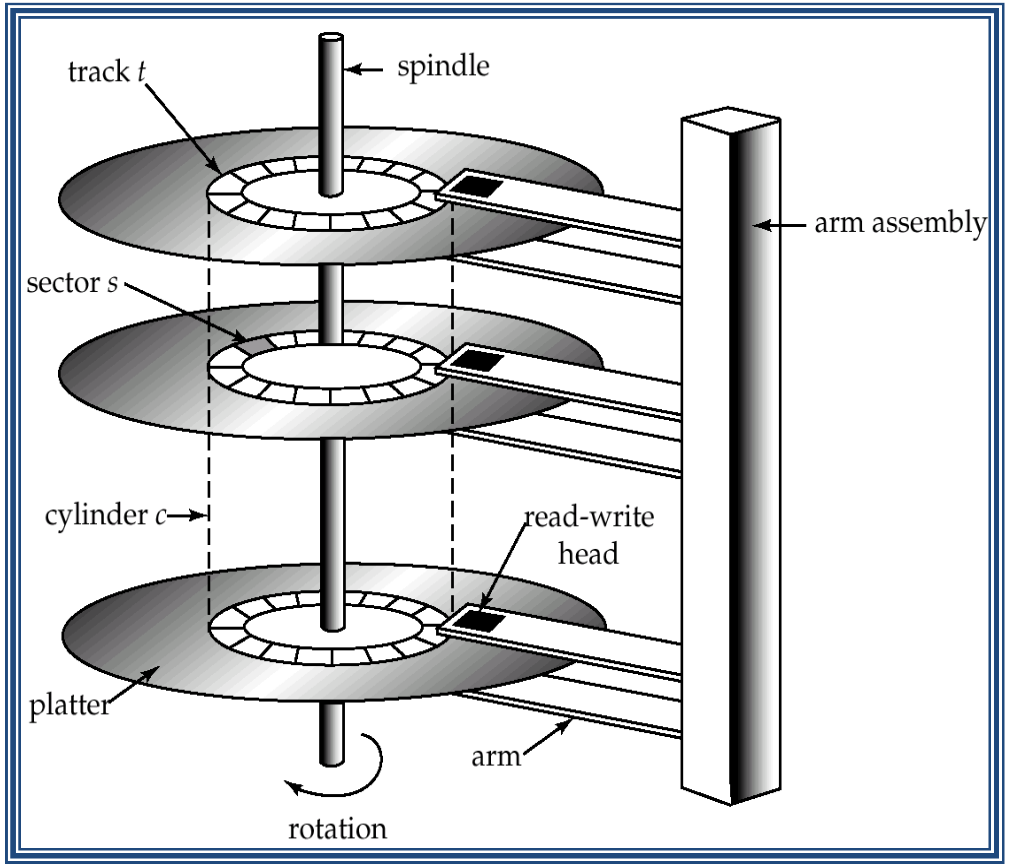
- We need to know the performance characteristics of non-volatile storage
  - to optimize database storage design

Magnetic disk (HDD)

Solid State Drive (SSD)

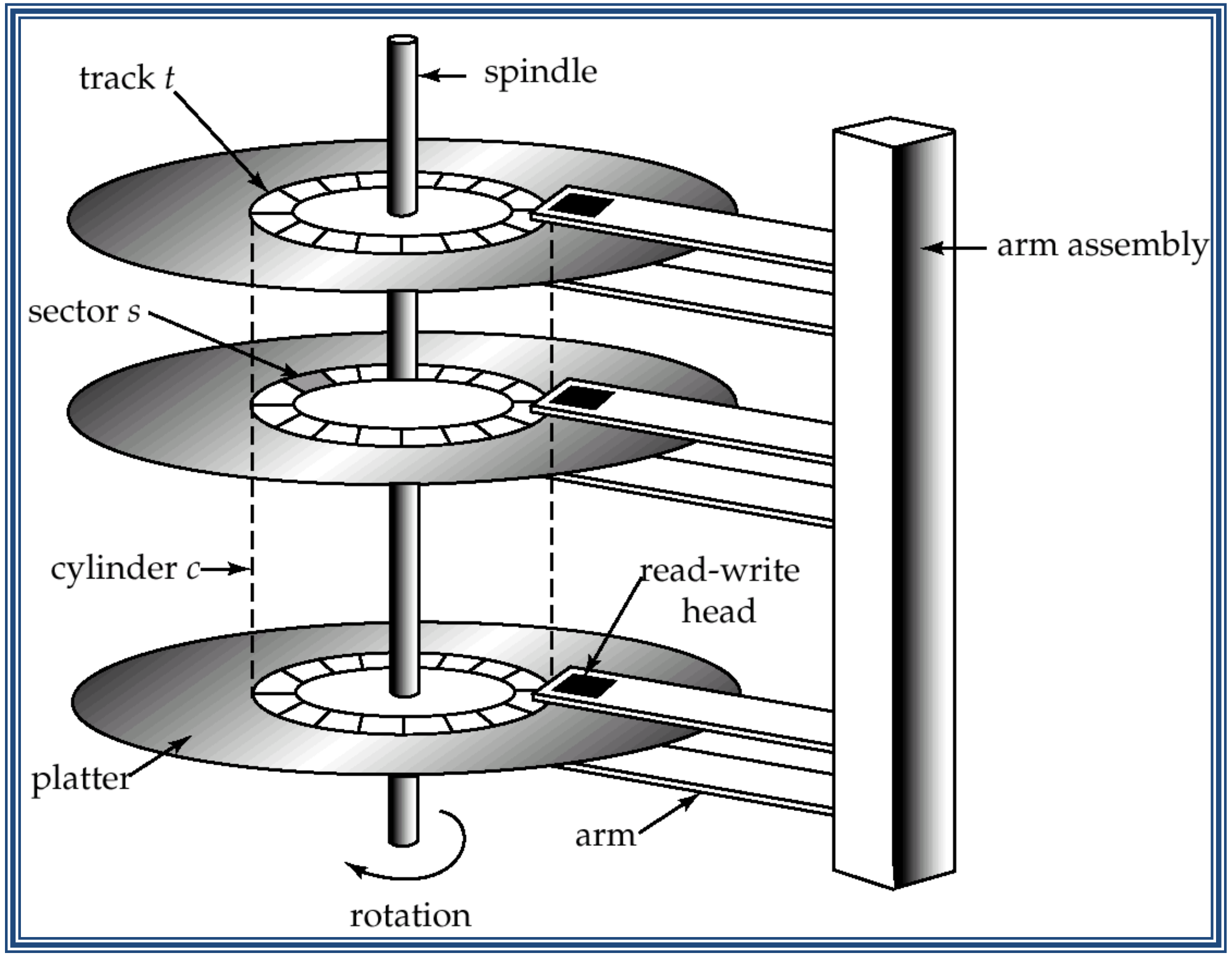
Magnetic disk organization

- Multiple platters
  - Each platter has *two* surfaces for data storage
  - Platters spin at the *same* rate (e.g., 7200 rpm)
  - A ring on a surface is called a track
    - A track is divided into many sectors of fixed size (512 B)
    - A sector is the *smallest* unit of I/O
- A single arm assembly with multiple disk heads
  - Can only move inward/outward *together*
  - The vertical stack of tracks is called a cylinder
    - Disk heads can be over the tracks of the *same* cylinder at the *same* time
  - Usually one read/writes at the same time
- Address of a sector: cylinder - head - sector
  - (0, 0, 0) : first sector; (0, 0, 1): second sector, …
    (0, 1, 0) : the S^{th} sector, (1, 0, 0) the $(SH)^{th}$
    where S is the max # of sectors/track and H is the # of heads
  - Reality: today's disks use logical block addressing (linear block #)
    - Translated to the actual geometry by disk controller
    - Nevertheless, this is still a good model for understanding HDD performance.



Magnetic disk I/O latency

- File systems perform I/O in units of multiple sector (page)
  - 4KB~16KB are most common
- Break-down of I/O latency of a page
  - Seek time: moving arms to the cylinder
    - 2 ~ 20 ms per seek
    - 4 ~ 10 ms on average
  - Rotation delay:
    wait for the sector to be under a head
    - Depending on rotation speed (5400 rpm - 15000 rpm)
    - E.g, 7200 rpm = 120 rotations/second
      => 1/120 = 8.33 ms / rotation
      on average it needs a half rotation
      => 8.33 / 2 = 4.17 ms on average
  - Transfer time: time for reading/writing data
    - Data transfer rate: 50 - 200 MB/s
    - ⇔ 0.02 ~ 0.08 ms for 4KB pages
- Average access time
  - 4KB page, 7200 rpm: roughly 8 ~ 15 ms

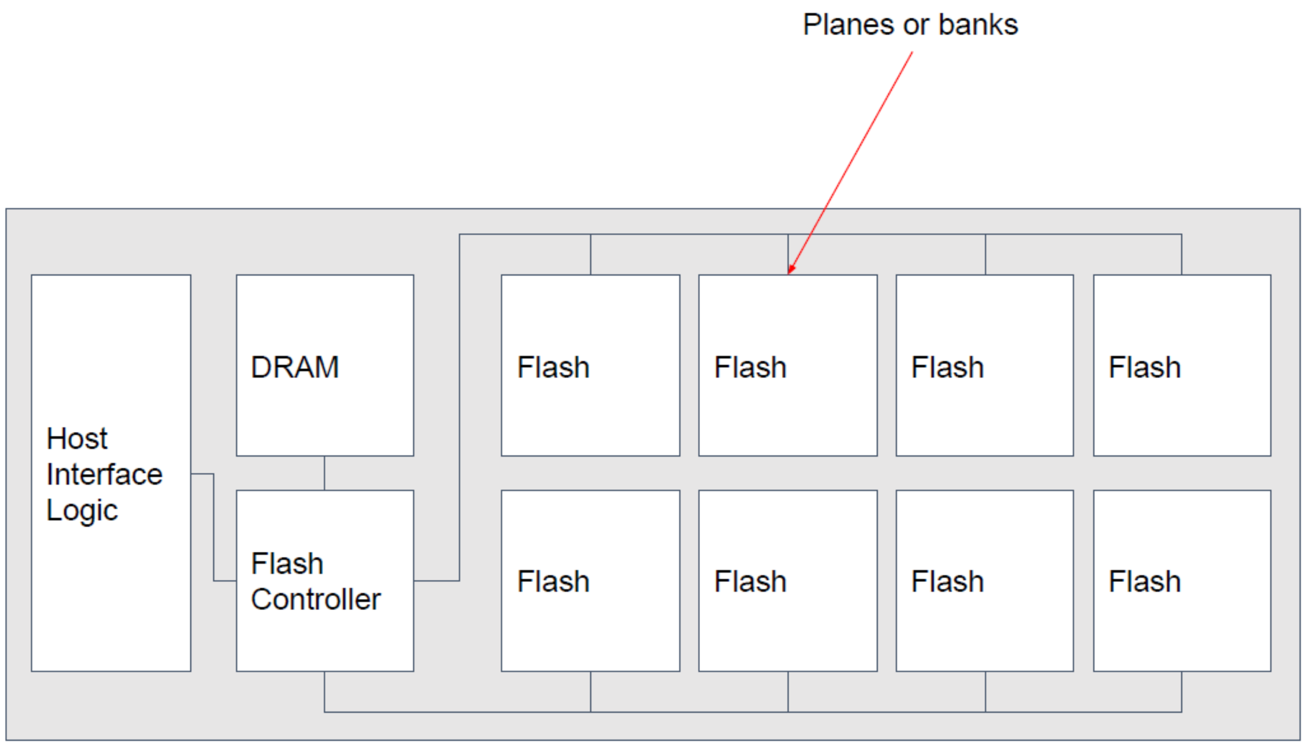


Impact of I/O pattern on magnetic disk

- I/O pattern has a huge impact on I/O performance
  - E.g., 4KB page size
    - Sequential read/write: usually 100 ~ 200+ MB/s
    - Random read/write: 50 ~ 200 IOPS ⬄ 200 KB ~ 800 KB /s
    - > 2 orders of magnitude difference in terms of data transfer rate
  - Rule of thumb:
    - Random I/O: very slow; avoid reading a lot of data from random location
    - Sequential I/O: better for accessing a lot of data

Flash memory / solid state drive

- NAND Flash is the most common storage media for solid state drives
- No mechanical parts (magnetic disk can have head crash => data corruption/loss)
  - More reliable; less likely to fail due to physical shocks
- Faster than magnetic disk



Flash memory / solid state drive

- NAND SSD has asymmetric read/write performance
  - 4KB page, typical SSD internal performance numbers
    - Read latency: 20 to 100 μs ; throughput: > 500 MB/s
    - Write latency: 200 μs; throughput: > 500 MB/s
    - Erase latency: ~2 ms

  - Three ops: read/write/erase
    - Read/write works on pages (usually 4KB)
      - Write can only change some bits from 1 to 0 (not the other way around!)
      - Muse erase before write a page.
    - Erase works on blocks (e.g., 256 KB)
      - Resets all bits in a block to 1
      - Flash translation layer: indirection of page numbers to physical pages
        - Solves two problems: slow erase and flash wear
  - Actual performance also often bound by peripheral bus's bandwidth and IOPS

File System Interface

- POSIX I/O interface
  - A standard synchronous I/O interface
  - Agnostic to the underlying storage device/file system

A *file descriptor* is a reference to an *open file description*, an entry in the system-wide table of open files that records file offsets and file status flags.

open(2): open and possibly create a file -> *file descriptor* (int)

```
int fd = open("/data/a.dat", O_RDONLY | O_CREAT, 0644);
```

opens the file at path `/data/a.dat`

1. read-only access
2. create the file if it does not exist

The permission bits if the file is created. 0644 = rw allowed for user (file owner); read only for group & others.

Case 1: fd >= 0 on success.
Case 2: fd == -1 if an error occurred -- check `errno` for reasons; also see `strerror(3)`

File System Interface

- POSIX I/O interface
  - A standard synchronous I/O interface
  - Agnostic to the underlying storage device/file system

A *file descriptor* is a reference to an *open file description*, an entry in the system-wide table of open files that records file offsets and file status flags.

open(2): open and possibly create a file -> *file descriptor* (int)

```
int fd = open("/data/a.dat", O_RDONLY | O_CREAT, 0644);
```

pread(2), pwrite(2): read from or write to a file descriptor at a given offset

```
char buf[4096];
ssize_t sz = pread(fd, buf, 4096, 1048576);
if (sz == 4096) /* success */; else /* error */;
```

reading 4096 bytes at file offset 1048576 = 4096 * 256 (i.e., reading page 255 from a file assuming 4KB pages)

File System Interface

- POSIX I/O interface
  - A standard synchronous I/O interface
  - Agnostic to the underlying storage device/file system

A *file descriptor* is a reference to an *open file description*, an entry in the system-wide table of open files that records file offsets and file status flags.

open(2): open and possibly create a file -> *file descriptor* (int)

```
int fd = open("/data/a.dat", O_RDONLY | O_CREAT, 0644);
```

pread(2), pwrite(2): read from or write to a file descriptor at a given offset

posix_fallocate(3), fallocate(2)

fsync(2), fdatasync(2),

close(2)

Check man pages for details (e.g., Linux man pages online (man7.org), or Linux man pages (die.net))