CS370 Operating Systems

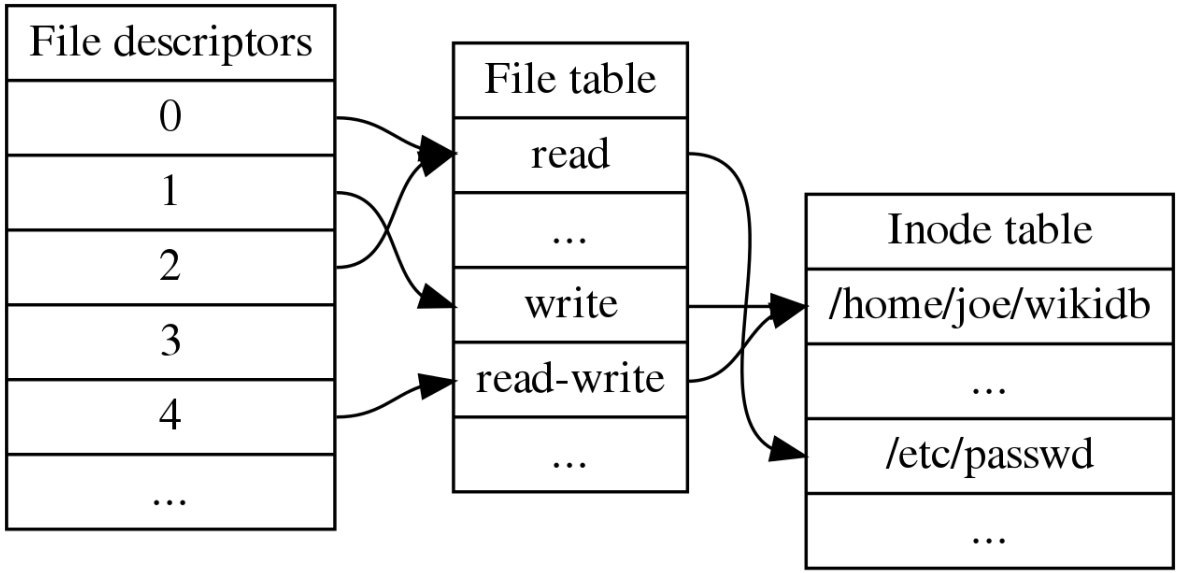
Colorado State University
Yashwant K Malaiya

Fall 2025 L21
File-system Implementation

Slides based on

- **Text by Silberschatz, Galvin, Gagne**
- **Various sources**

Process, System, Files



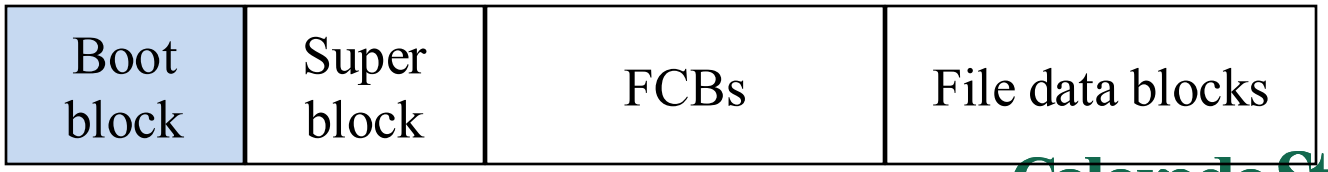
- **File descriptor table for a process:** File descriptor (FD int), pointer
- **System wide open File Table** in memory**:** r/w status, offset, open count, inode number
- **Inode table for all files/dirs:** indexed by inode numbers on the disk **(unix: ls – ia)**
  - **Inode for a file:** file/dir metadata (attributes etc), pointers to blocks

On-disk File-System Structures

1. **Boot control block** contains info needed by system to boot OS from that volume
   - Needed if volume contains OS, usually first block of volume

Volume: logical disk drive, perhaps a partition

2. **Volume control block (superblock** $_{ext}$ **or master file table**$_{NTFS}$**)** contains volume details
   - Total # of blocks, # of free blocks, block size, free block pointers or array
3. Directory structure organizes the files
   - File Names and inode numbers $_{UFS}$, master file table $_{NTFS}$



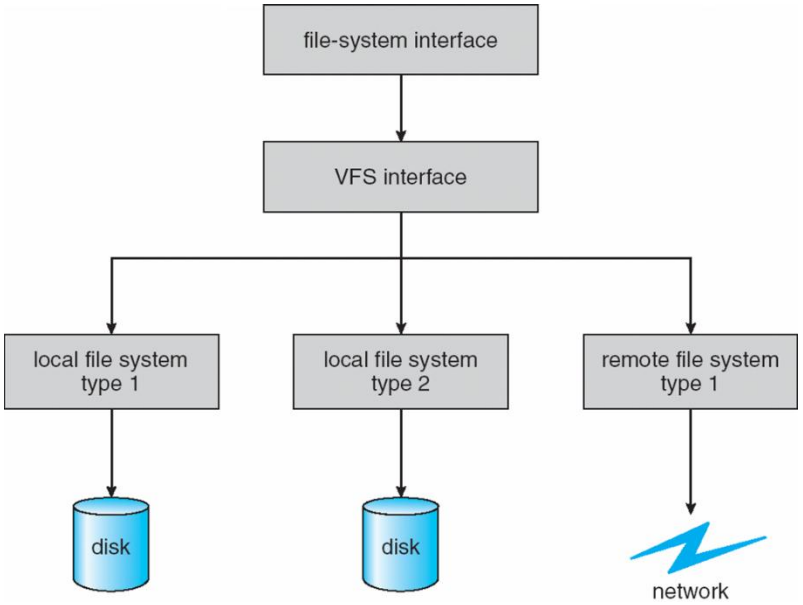
When a file is created

When a new file is created n a directory, the OS

- Allocates a new FCB.
- Update directory
  - Reads the appropriate directory into memory, in unix a directory is a file with special type field
  - updates it with the new file name and FCB,
  - writes it back to the disk.

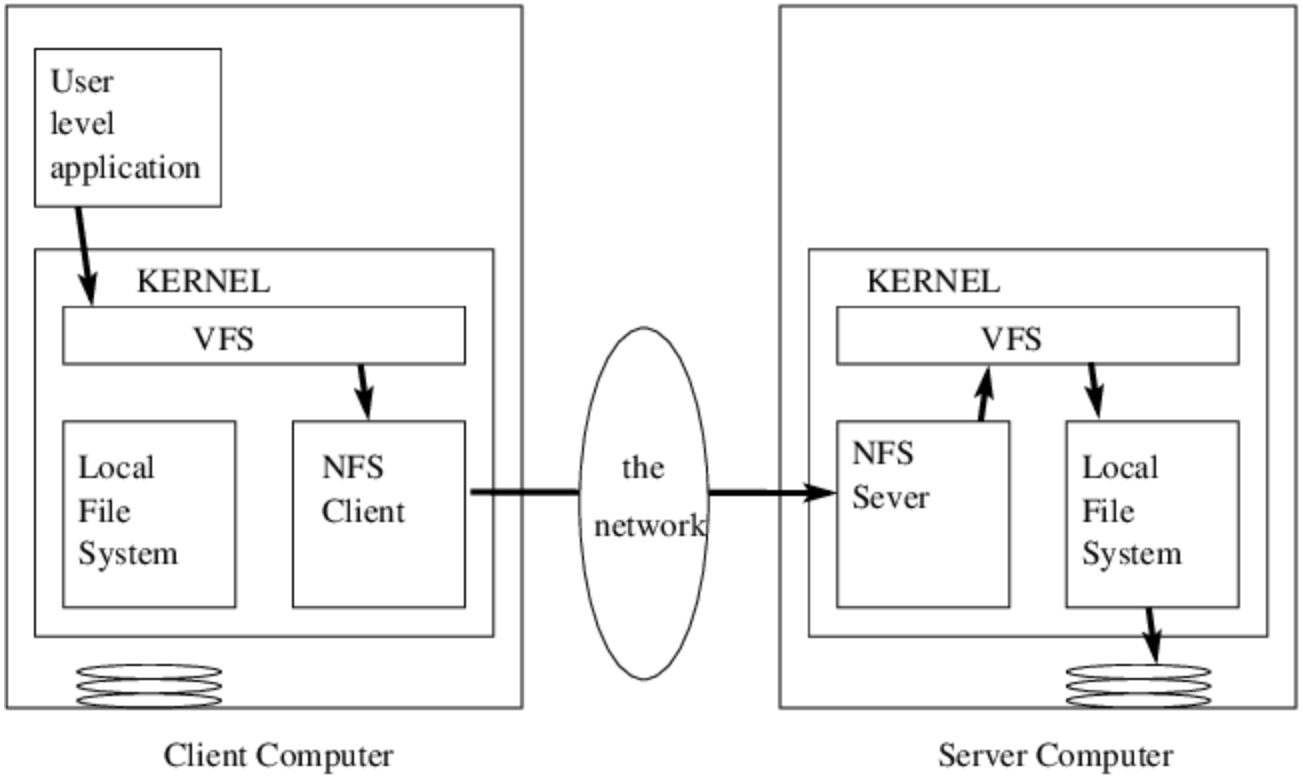
Virtual File Systems

- **Virtual File Systems** (**VFS**) in Unix kernel is an *abstraction layer* on top of specific file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems
- The API (POSIX system calls) is to the VFS interface, rather than any specific type of file system



Virtual to specific FS interface

NFS (Network File System)



Source

NFS is distributed file system *protocol* uses the Open Network Computing Remote Procedure Call (ONC RPC) system (1984). It uses the VFS layer to perform remote operations, translating file system requests into a network protocol to access files on a remote server.

File Sharing – Remote File Systems

- Uses networking to allow file system access between systems
  - Manually via programs like FTP/SFTP
  - Automatically, seamlessly using **distributed file systems**
  - Semi automatically via the **world wide web**
- **Client-server** model allows clients to mount remote file systems from servers
  - Server can serve multiple clients
  - Client and user-on-client identification is insecure or complicated
  - **NFS** is standard UNIX client-server file sharing protocol
  - **CIFS** is standard Windows protocol
  - Standard operating system file calls are translated into remote calls

Block Allocation Methods

An allocation method refers to how disk blocks are allocated for files:

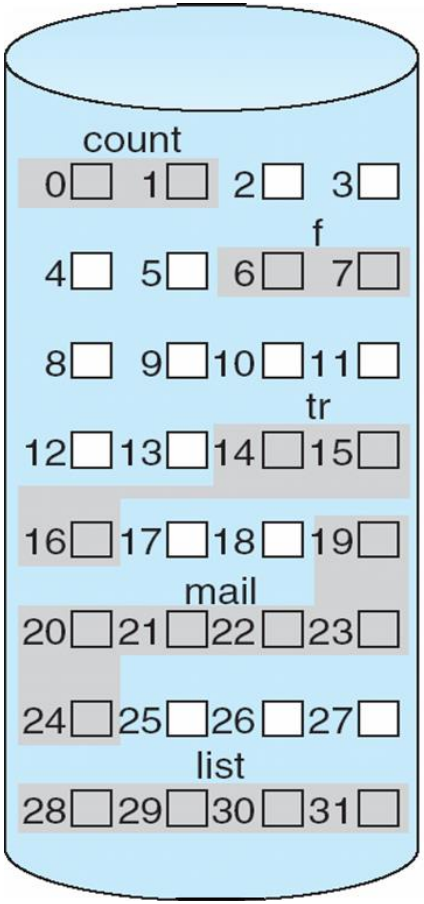
- Contiguous (not common, except for DVDs etc.)
- Linked blocks, Linked guide (e.g., FAT32)
- Indexed (e.g., ex4)

A disk block can be a physical sector. They are numbered using a linear sequence.

Actual implementations are more complex than the simple ones examined here. Contrast these with allocation for processes in memory.

You can use *stat -f* to get the details any filesystem. Ex:
% stat -f /home

Contiguous Allocation

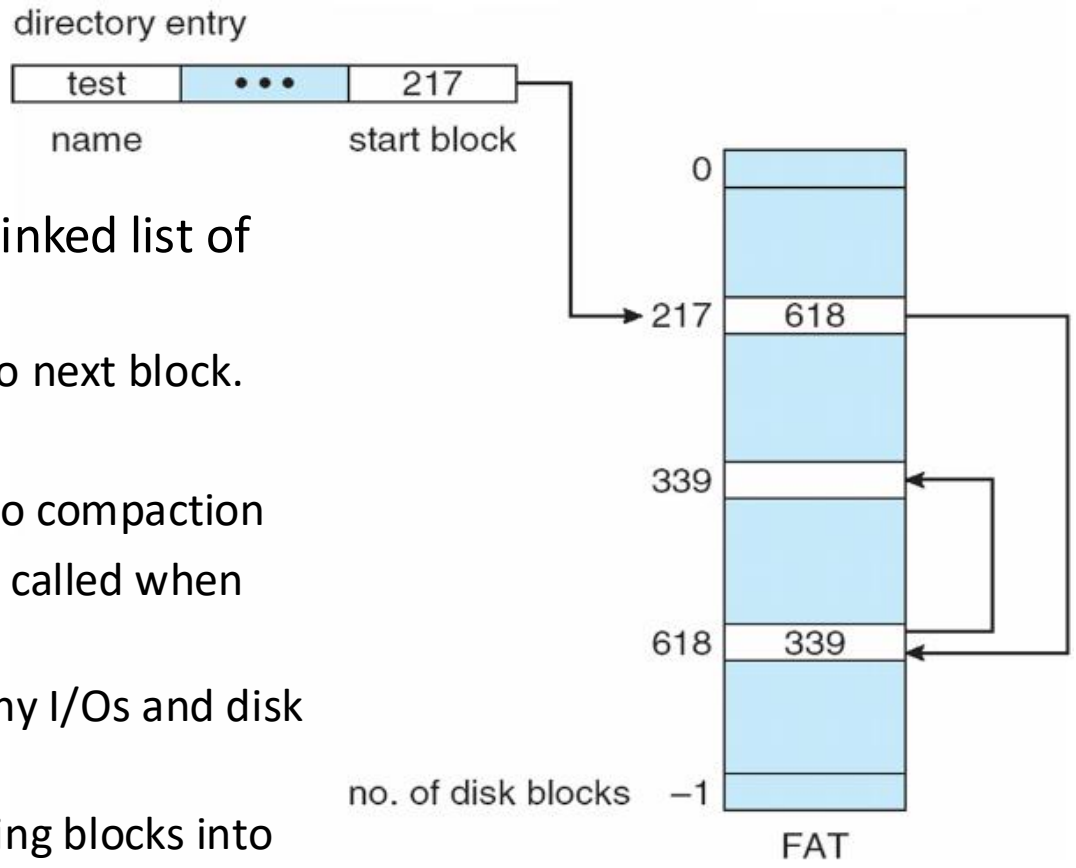


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

File **tr**: 3 blocks
Starting at block 14

Allocation Methods - Linked



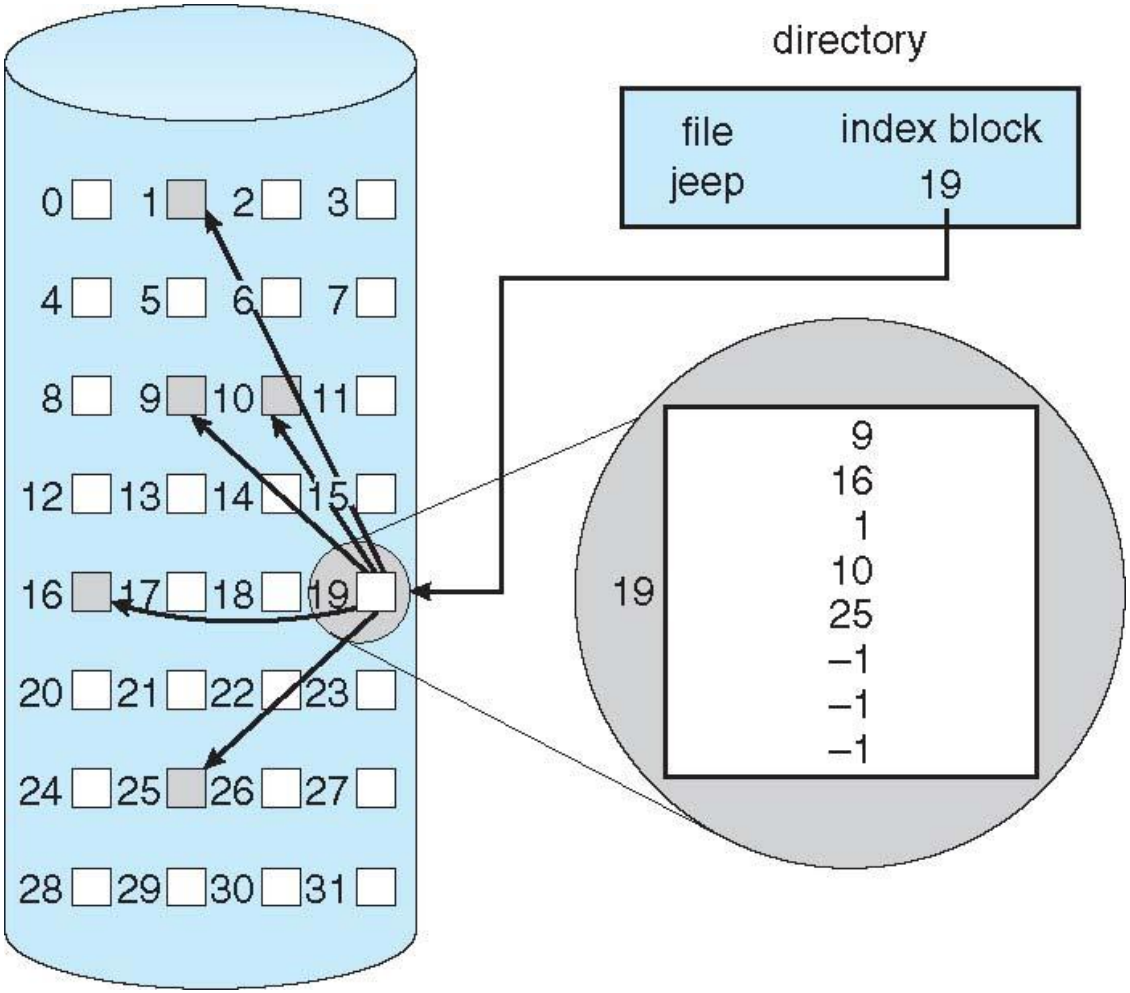
ii. Linked allocation – each file a linked list of blocks

- Each block contains pointer to next block.
- File ends at null pointer
- No external fragmentation, no compaction

Free space management system called when new block needed

- Locating a block can take many I/Os and disk seeks.
- Improve efficiency by clustering blocks into groups but increases internal fragmentation
- Reliability can be a problem, since every block in a file is linked.

Example of Indexed Allocation

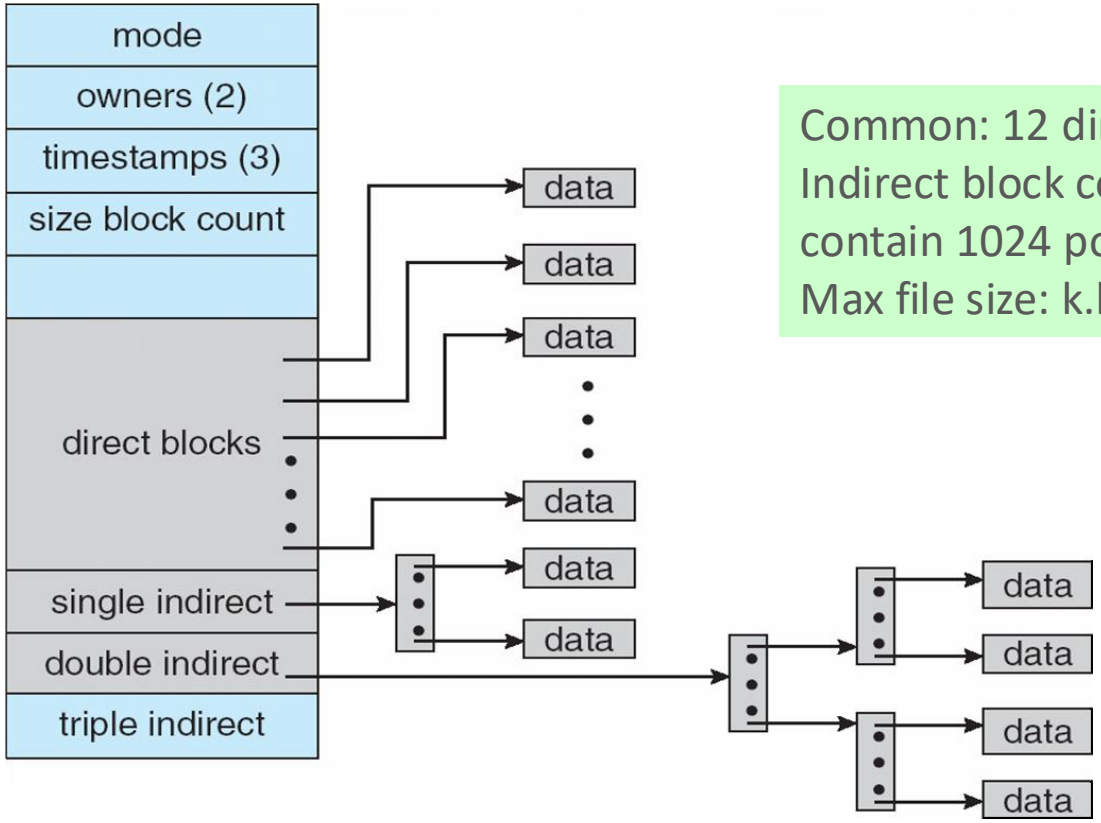


Uses Index blocks.
Index block has pointers to data blocks for a file.

Indexed Scheme: UNIX inodes

Assume 4K bytes per block, 32-bit addresses. Ext3 example.

Volume block:
Table with file names
Points to this inode
(file control block)



Common: 12 direct+3.
Indirect block could
contain 1024 pointers.
Max file size: k.k.k.4k (triple)+

More index blocks than can be addressed with 32-bit file pointer

Ext4: uses *extents* (pointer+ length of blocks)

Colorado State University

Performance

- Best method depends on file access type
  - Contiguous: OK when files don't change much
  - Linked: used for smaller file systems of the past: FAT, FAT32
  - Indexed more complex, modern
    - Single block access could require 0-3 index block reads then data block read
    - Clustering or disk caching can help improve throughput, reduce CPU overhead
    - Ex: Ext3, Ext4

Cluster: set of contiguous sectors

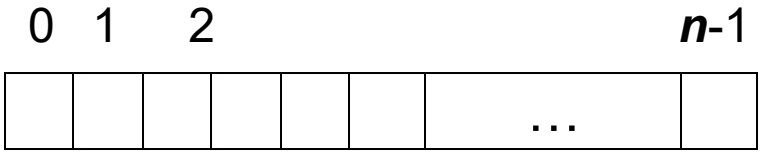
Performance (Cont.)

- Is adding instructions to the execution path to save one disk I/O is reasonable?
  - AMD Ryzen 7 9950X (2024)
    - 196,546 MIPS
    - http://en.wikipedia.org/wiki/Instructions_per_second
  - Typical disk drive at 250 I/Os per second
    - 196,546 MIPS / 250 = 786 million instructions during one disk I/O
  - Fast SSD drives provide 60,000 IOPS
    - 196,546 MIPS / 60,000 = 3.3 millions instructions during one disk I/O

Free-Space Management

- File system maintains **free-space list** to track available blocks/clusters
  - (Using term "block" for simplicity)
  - **Approaches: i. Bit vector ii. Linked list iii. Grouping iv. Counting**

i. Bit vector or **bit map** (***n*** blocks)



$$\text{bit}[i] = \begin{cases} 1 \Rightarrow \text{block}[i] \text{ free} \\ 0 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$

Block number calculation for first free block

(number of bits per word) *(number of 0-value words) + offset of first 1 bit

```
00000000
00000000
00111110
..
```

CPUs may have instructions to return offset within word of first "1" bit

Free-Space Management (Cont.)

Bit map requires extra space

- Example:

  block size = 4KB = 2^{12} bytes

  disk size = 2^{40} bytes (1 terabyte)

  blocks: $n = 2^{40}/2^{12} = 2^{28}$

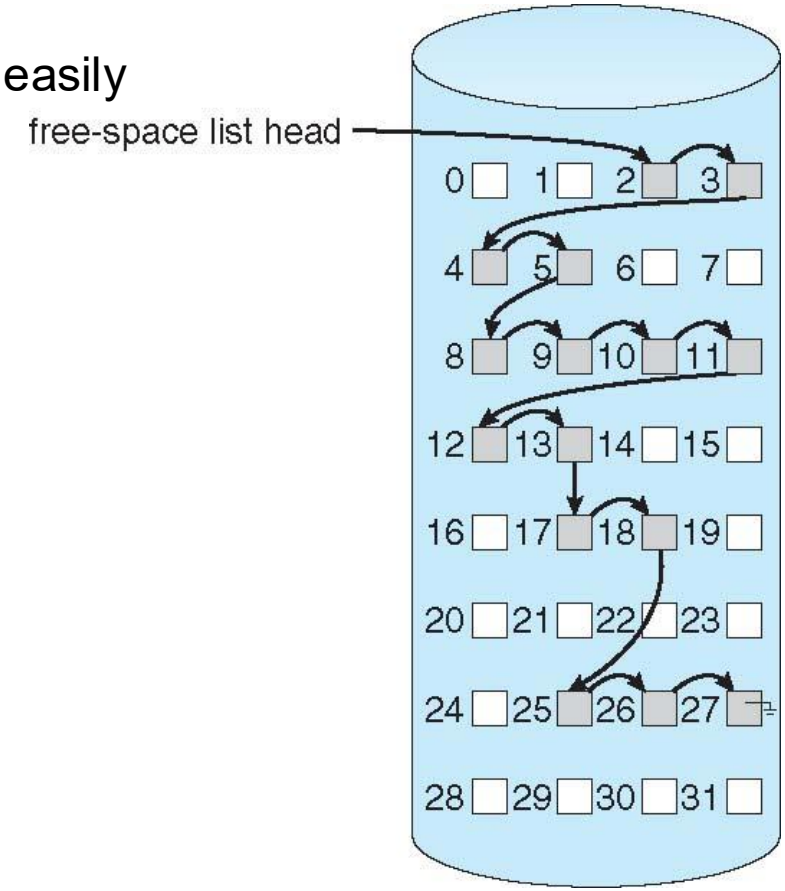
  Need 2^{28} bits or 32MB for map

  if clusters of 4 blocks -> 8MB of memory

Bit map makes it easy to get contiguous files if desired

Linked Free Space List on Disk

- ii. Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space



Superblock Can hold pointer to head of linked list

Free-Space Management (Cont.)

- iii. Grouping
  - Modify **linked list** to store address of next *n-1* free blocks in first free block, plus a pointer to next block that contains free-block-pointers free block pointer blocks in a linked list.
- iv. Counting
  - Because space is frequently contiguously used and freed, with contiguous-allocation allocation, extents, or clustering
    - Keep address of first free block and count of following free contiguous blocks
    - Free space list then has entries containing **addresses and counts**

UNIX directory structure

- Contains only file names and the corresponding inode numbers an inode uniquely identifies a file
- Use **ls – i** to retrieve inode numbers of the files in the directory
- Looking up path names in UNIX

  Example: /usr/tom/mbox

  – Lookup inode for /, then for usr, then for tom, then for mbox

Advantages of directory entries that have name and inode information

- Changing filename only requires changing the directory entry
- Only 1 physical copy of file needs to be on disk
  - File may have several names (or the same name) in different directories
- Directory entries are small
  - Most file info is kept in the inode

Hard and symbolic links

Hard Links:

- Both file names refer to the same inode (and hence same file)
  - Directory entry in /dirA

    ..[12345 filename1]..
  - Directory entry in /dirB

    ..[12345 filename2]..
- To create a hard link

  ln /dirA/filename1 /dirB/filename2
- Symbolic link *shortcut* in windows
  - To create a symbolic link

    ln -s /dirA/filenmame1 /dirB/filename3

    File filename3 just contains a pointer

File system based on inodes

Limitations

- File **must fit** in a single disk partition
- Partition size and number of files are **fixed** when system is set up

inode preallocation and distribution

- inodes are **preallocated** on a volume
  - Even on empty disks % of space lost to inodes
- Preallocating inodes
  - Improves performance
- Keep file's data block **close** to its inode
  - Reduce seek times

Checking up on the inodes

Command: df -i (df is for disk filesystem)

Gives inode statistics for the file systems: total, free and used nodes

```
Filesystem              Inodes   IUsed    IFree IUse% Mounted on
devtmpfs               2045460     484  2044976    1% /dev
tmpfs                  2053722       1  2053721    1% /dev/shm
tmpfs                  2053722     695  2053027    1% /run
tmpfs                  2053722      16  2053706    1% /sys/fs/cgroup
```

Command: ls –i (lists inodes of the files in current directory)

```
13320302                diskusage.txt
2408538                 Documents/
680003                  downloads/
```

CS370 Operating Systems

Colorado State University
Yashwant K Malaiya
Fall 2024. Ch 11

Mass Storage

Slides based on

- **Text by Silberschatz, Galvin, Gagne**
- **Various sources**

Chapter 11: Mass-Storage Systems

- Overview of Mass Storage
- Technologies, performance
- Disk Scheduling
- Disk Management
- RAID Structure

Colorado State University

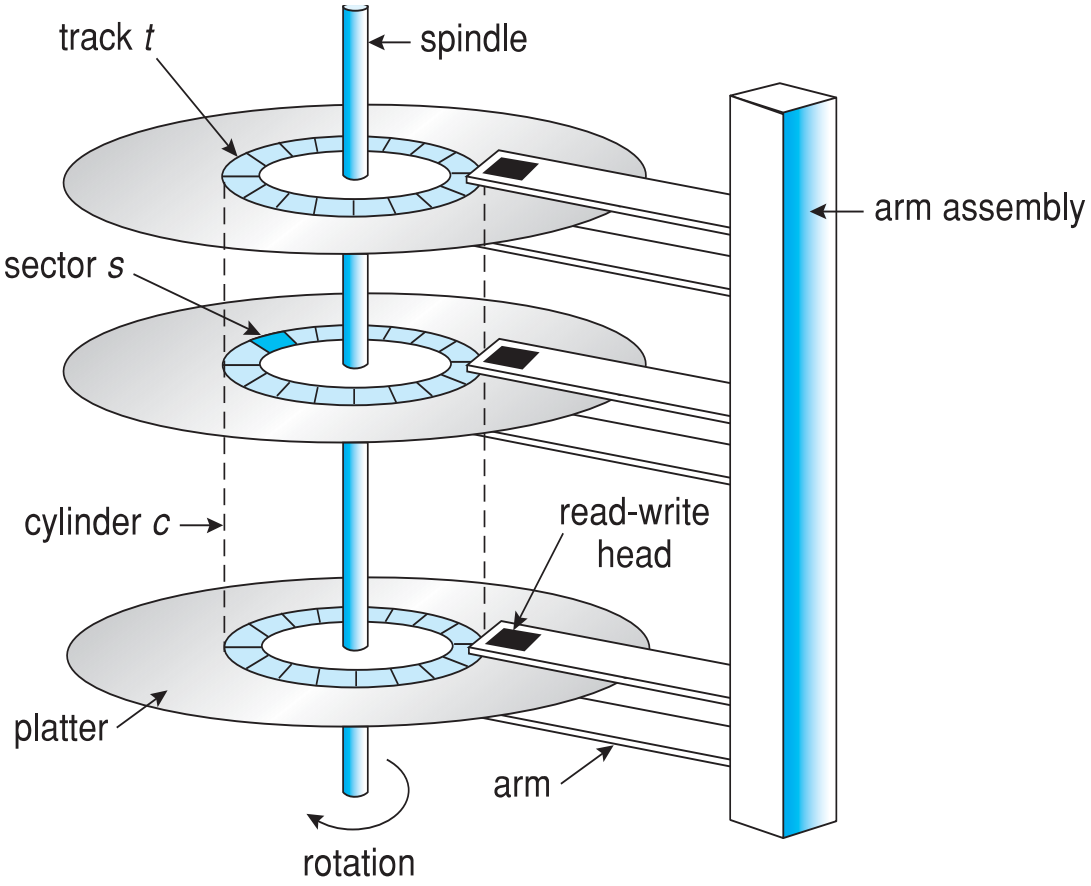
Objectives

- The physical structure of secondary storage devices and its effects on the uses of the devices
- To explain the performance characteristics of mass-storage devices
- To evaluate disk scheduling algorithms
- To discuss operating-system services provided for mass storage, including RAID

Overview of Mass Storage Structure

- **Magnetic disks** provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 250 times per second
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
  - **Head crash** results from disk head making contact with the disk surface -- That's bad
- Disks can be removable
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE**, **ATA**, **SATA**, **USB**, **Fibre Channel**, **SCSI, SAS, Firewire**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

Moving-head Disk Mechanism



Hard Disks

- Platters range from 0.85” to 14” (historically)
  - Commonly 3.5”, 2.5”, and 1.8”
- Range from 16GB to **12TB** per drive
- Performance
  - Transfer Rate – theoretical – 6 G**b**/sec
  - Effective Transfer Rate – real – 1G**b**/sec (about 150 M**B**/s)
  - Seek time from 2ms to 12ms – 9ms common for desktop drives
  - Average seek time measured or calculated based on 1/3 of tracks
  - Latency based on spindle speed
    - 1 / (RPM / 60) = 60 / RPM
  - Average latency = ½ latency

Spindle [rpm]	Average latency [ms]
4200	7.14
5400	5.56
7200	4.17
10000	3
15000	2

(From Wikipedia)

Hard Disk Performance

- **Average access time** = average seek time + average latency
  - For fastest disk 3ms + 2ms = 5ms
  - For slow disk 9ms + 5.56ms = 14.56ms
- Average I/O time = average access time + (amount to transfer / transfer rate) + controller overhead
- Example: Find expected I/O time to transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a 0.1ms controller overhead.

  Av latency =60/(7200*2)

  = (5ms + 4.17ms) + 0.1ms + transfer time
  - Transfer time = 4KB / 1Gb/s = 4x8K/G = 0.031 ms
  - Average I/O time for 4KB block = 9.27ms + .031ms = 9.301ms

Strategy: memorize formula or understand how it works?

Interfaces for HDD/SSD

Actual data rates (not raw)

Serial ATA (SATA): Serial: 4 Pin + grounds

- SATA-I: 125 MB/s
- SATA-II: 250 MB/s
- SATA-III: 500 MB/s

PCI Express (PCIe) v5.0

- 32 GB/s per lane, Up to 16 lanes
- Very low power and broad hardware support
- Very expensive, extremely high-performance applications

USB 3.2

- 610MB/s

Thunderbolt 3

- 4.88 GB/s

The First Commercial Disk Drive

1956
IBM RAMDAC computer included the IBM Model 350 disk storage system

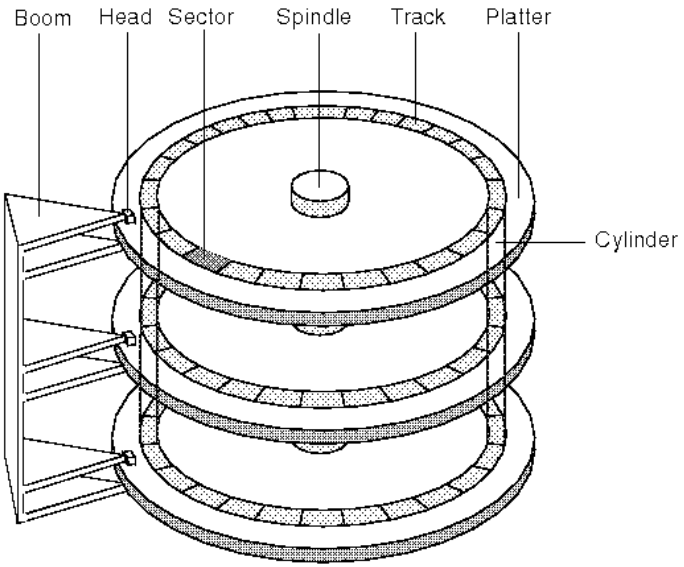
5M (7 bit) characters
50 x 24” platters
Access time = < 1 second

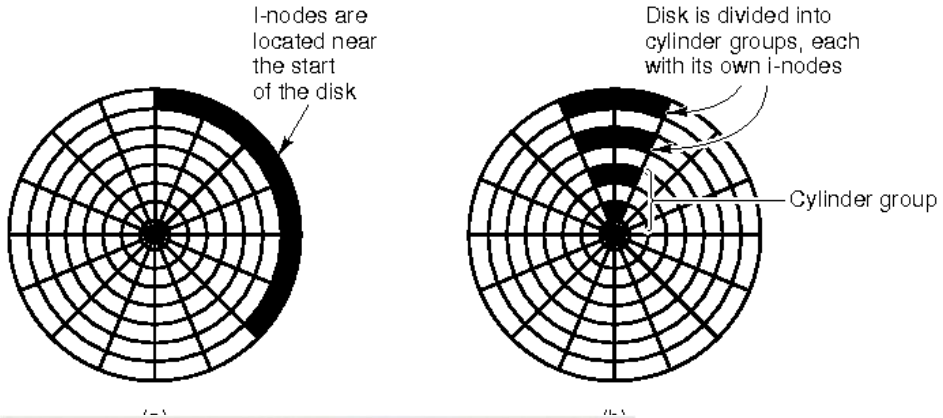
Disk Structure

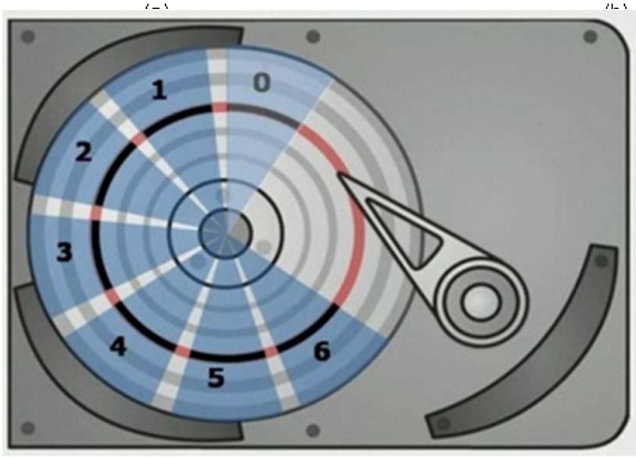
- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
  - Low-level formatting creates **sectors** on physical media (typically 512 bytes)
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
  - Sector 0 is the first sector of the first track on the outermost cylinder
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost
  - Logical to physical address should be easy
    - Except for bad sectors
    - Non-constant # of sectors per track via constant angular velocity

FAQ

- Physical: Drive, Cylinder, Head, sector
- Logical Block Addressing (LBA): blocks addressed by numbers.
- Inodes: where?







Colorado State University

Disk Formatting

- Low-level formatting marks the surfaces of the disks with markers indicating the start of a recording block (sector markers) and other information by the disk controller to read or write data.
- Partitioning divides a disk into one or more regions, writing data structures to the disk to indicate the beginning and end of the regions. Often includes checking for defective tracks/sectors.
- High-level formatting creates the file system format within a disk partition or a logical volume. This formatting includes the data structures used by the OS to identify the logical drive or partition's contents.

Solid-State Disks

- Nonvolatile memory used like a hard drive
  - Many technology variations
  - Same physical sizes, same interfaces (SATA, PCIe, SCSI)
- Can be more reliable than HDDs
- More expensive per MB ($0.30/GB vs $0.05 for HD)
- Life span (1-5 million write cycles) shorter/longer?
- Capacity ? (up to 16 TB vs 8 TB for HD)
- faster (access time <0.1 millisec, transfer rate 100MB-GB/s)
  - No moving parts, so no seek time or rotational latency
- Lower power consumption
- 3D Xpoint: 10x faster, 3x endurance, 4x denser than NAND flash

Colorado State University

SSD Architecture

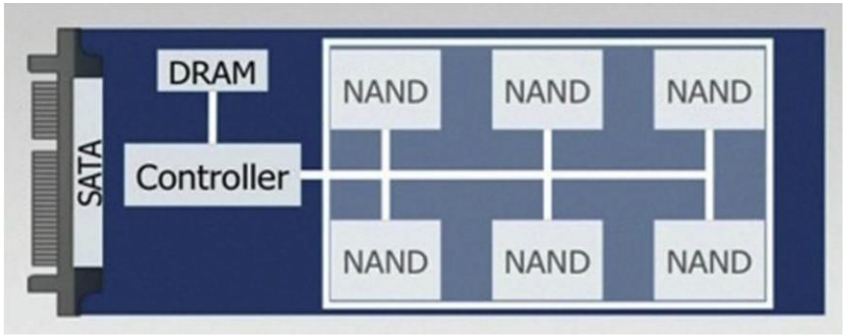
Controller

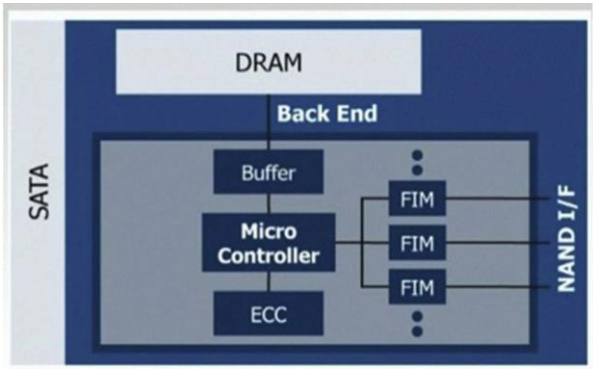
- Takes the raw data storage in the NAND flash and makes it look and act like hard disk drive
- Contains the micro controller, buffer, error correction, and flash interface modules

Micro Controller – a processor inside the controller that takes the incoming data and manipulates it

- Correcting errors
- Manages mapping
- Putting data into the flash or retrieving it from the flash

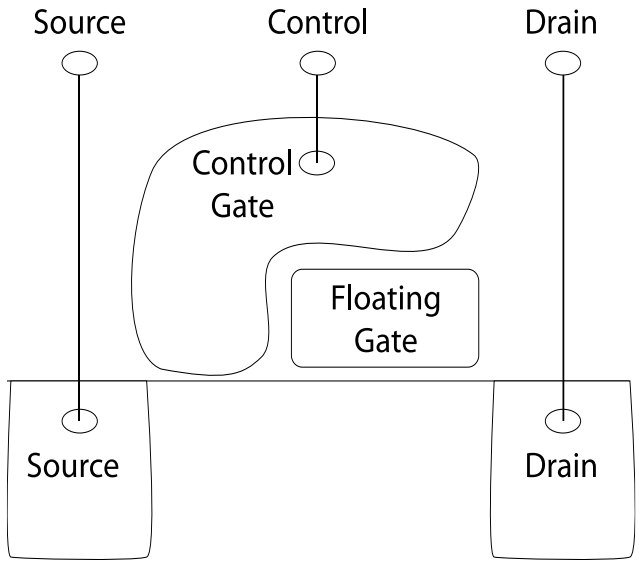
DRAM Cache – Reasonable amount of very low latency



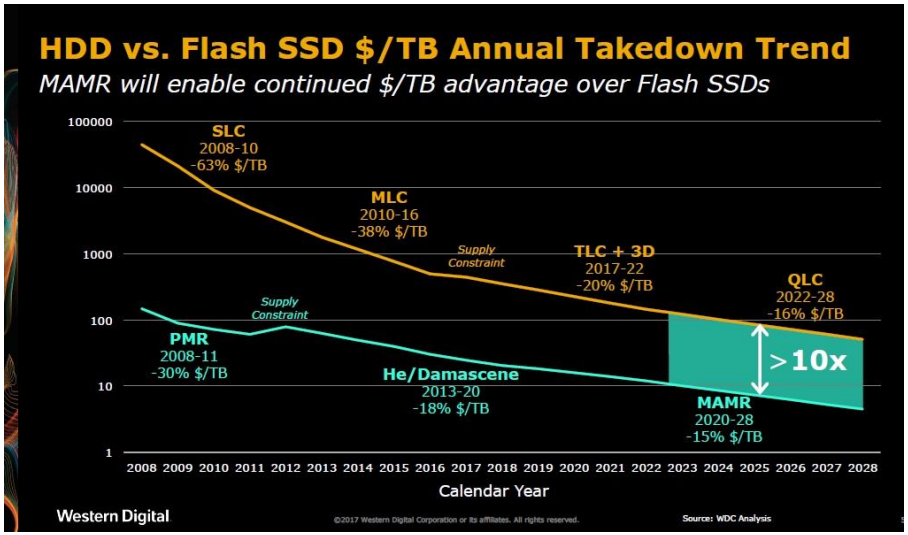


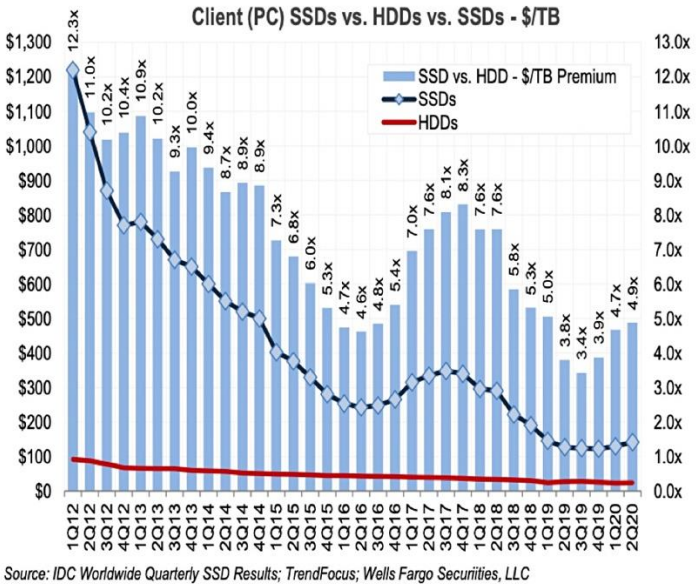
Flash Memory

- Writes must be to “clean” cells; no update in place
  - Large block erasure required before write
  - Erasure block: 128 – 512 KB
  - Erasure time: Several milliseconds
- Write/read page (2-4KB)
  - 50-100 usec

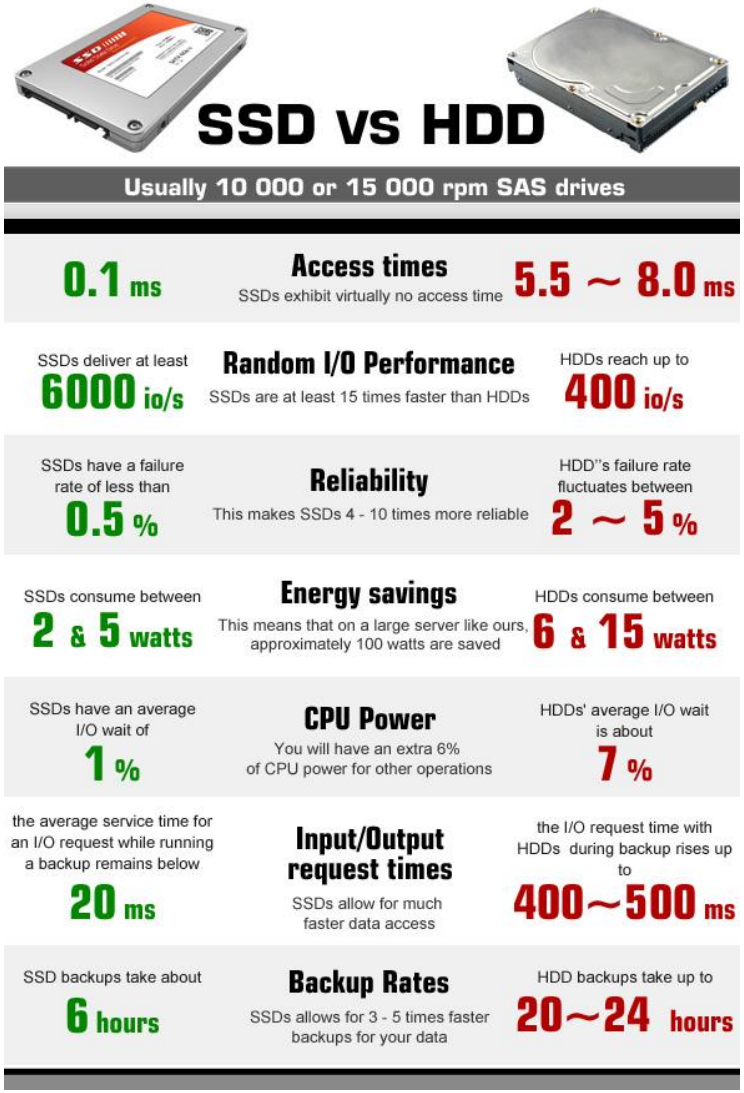


SSD vs HDD





Source: IDC Worldwide Quarterly SSD Results; TrendFocus; Wells Fargo Securities, LLC



Colorado State University

HDD vs SSD

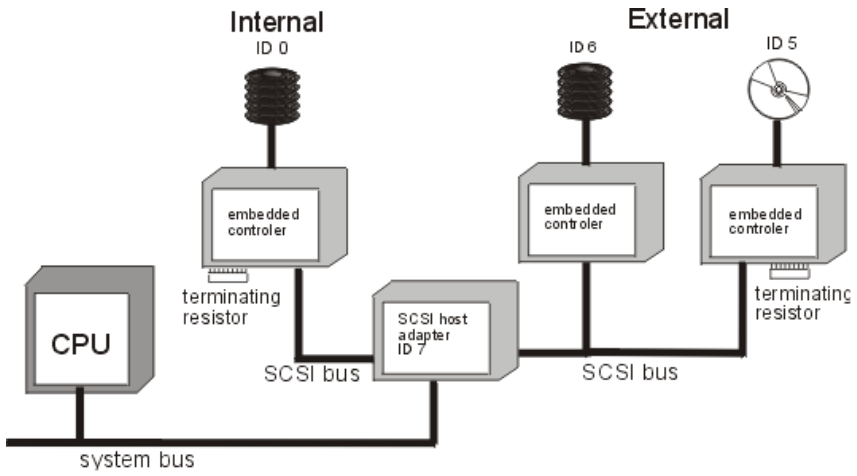
	HDD	SSD
	WD VelociRaptor	OCZ Vertex 3
Storage Capacity	600GB	120GB-360GB
Price for storage	48¢/ GB	2.08$/GB x4
Seek Time/Rotational Speed	7ms/157 MB/s	
MTBF	1.4 million hours?	2 million hours?
Sequential Read/Write	1 MB/s	413.5/371.4 MB/s
Random Read	1 MB/s	68.8 MB/s
Random Write	1 MB/s	332.5 MB/s
IOPS	905	60,000 x60

Magnetic Tape

- Was early secondary-storage medium (now tertiary)
  - Evolved from open spools to cartridges
- Relatively permanent and holds large quantities of data
- Access time slow
- Random access ~1000 times slower than disk
- Mainly used for backup, storage of infrequently-used data, transfer medium between systems
- Kept in spool and wound or rewound past read-write head
- Once data under head, transfer rates comparable to disk
  - 140MB/sec and greater
- 200GB to 1.5TB typical storage Sony: New 185 TB

Disk Attachment: I/O busses

- Host-attached storage accessed through I/O ports talking to I/O busses
- SCSI itself is a bus, up to 16 devices on one cable, **SCSI initiator** (adapter) requests operation and **SCSI targets** (controller) perform tasks
  - Each target can have up to 8 **logical units** (disks attached to device controller)
- FC (fibre channel) is high-speed serial architecture
  - Can be switched fabric with 24-bit address space – the basis of **storage area networks (SAN**s) in which many hosts attach to many storage units

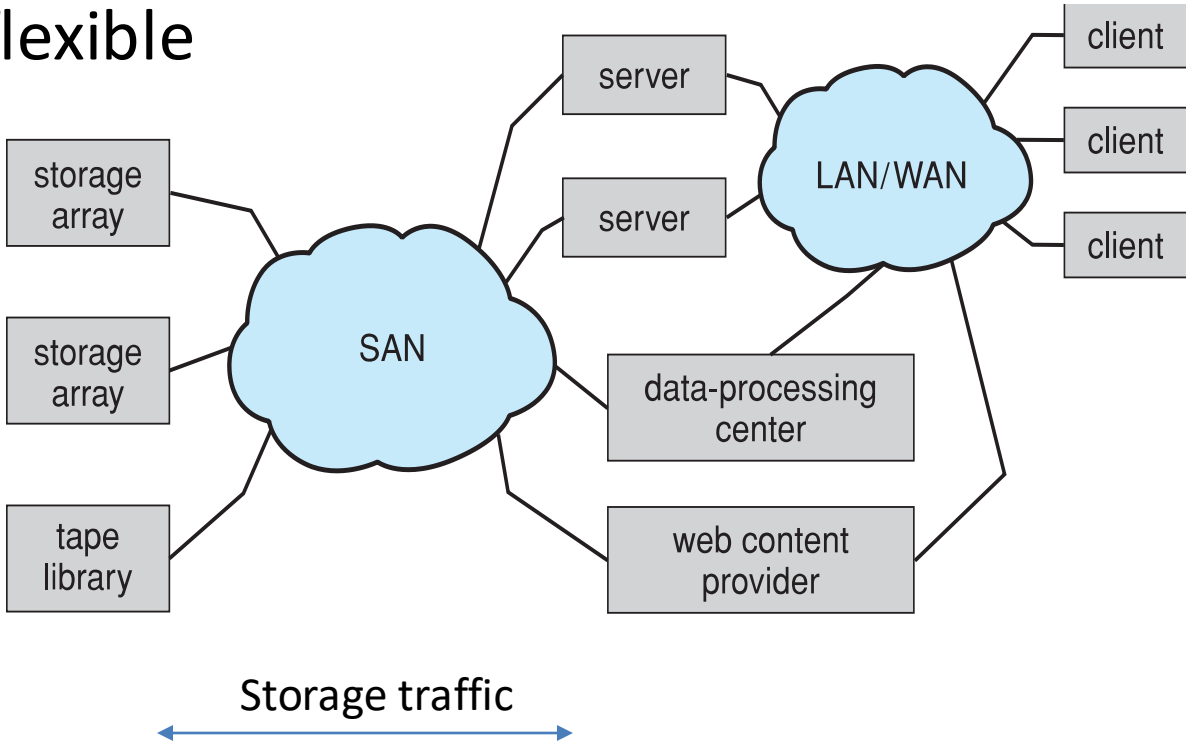


Storage Array

- Can just attach disks, or arrays of disks to an I/O port
- Storage Array has controller(s), provides features to attached host(s)
  - Ports to connect hosts to array
  - Memory, controlling software
  - A few to thousands of disks
  - RAID, hot spares, hot swap
  - Shared storage -> more efficiency

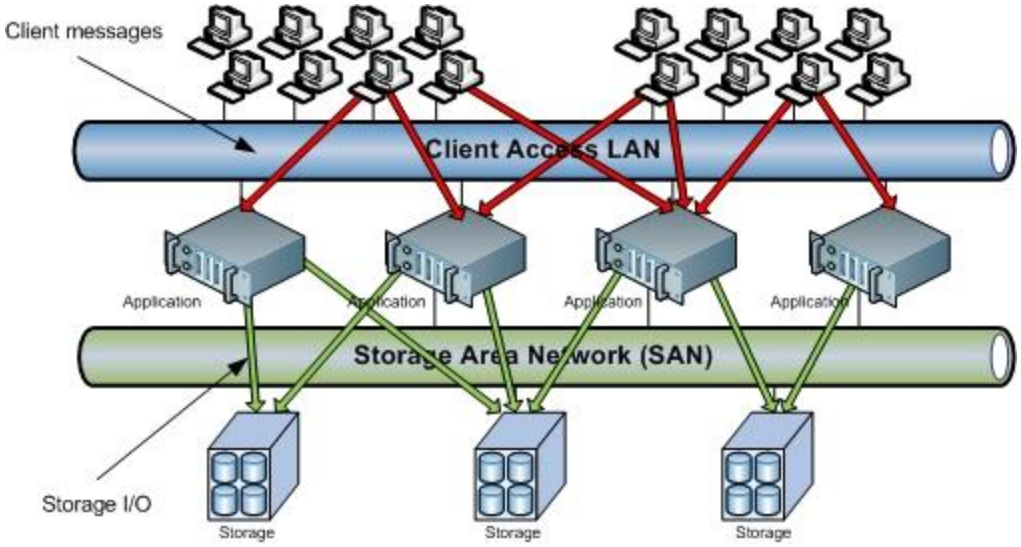
Storage Area Network

- Common in large storage environments
- Multiple hosts attached to multiple storage arrays - flexible



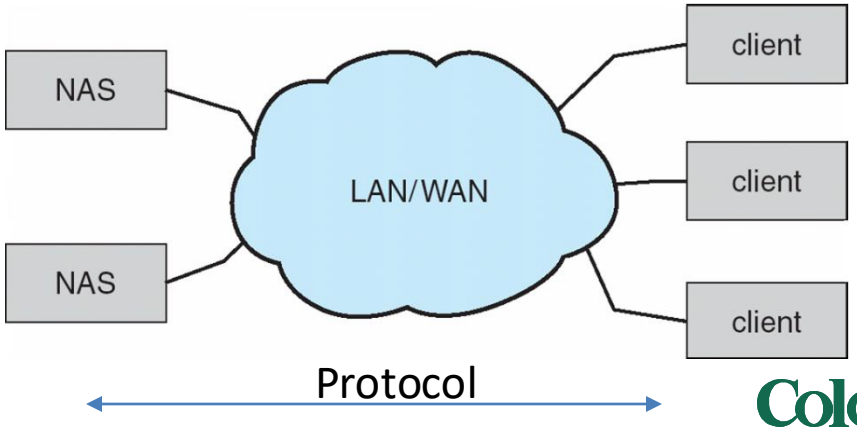
Storage Area Network (Cont.)

- SAN is one or more storage arrays
- Hosts also attach to the switches
- Storage made available from specific arrays to specific servers
- Easy to add or remove storage, add new host and allocate it storage
  - Over low-latency Fibre Channel fabric



Network-Attached Storage

- Network-attached storage (**NAS**) is storage made available over a network rather than over a local connection (such as a bus)
  - Remotely attaching to file systems
- NFS and CIFS (windows) are common protocols
- Implemented via remote procedure calls (RPCs) between host and storage over typically TCP or UDP on IP network
- **iSCSI** protocol uses IP network to carry the SCSI protocol
  - Remotely attaching to devices (blocks)



Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Minimize seek time
- Seek time $\approx \propto$ seek distance (between cylinders)
- Disk **bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

Disk Scheduling (Cont.)

- There are many sources of disk I/O request
  - OS
  - System processes
  - Users processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
  - Optimization algorithms only make sense when a queue exists

Disk Scheduling (Cont.)

- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying "depth")
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- We illustrate scheduling algorithms with a request queue (cylinders 0-199)

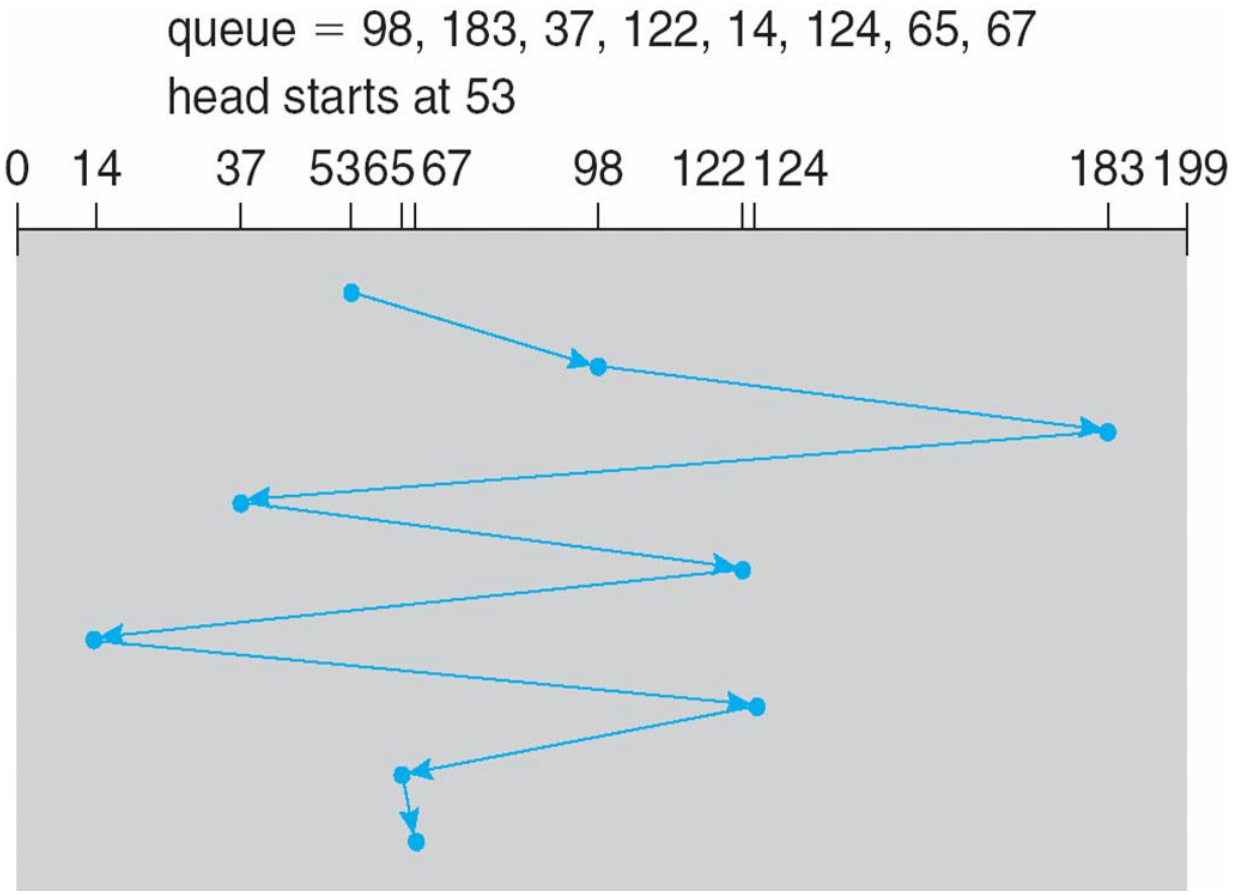
  98, 183, 37, 122, 14, 124, 65, 67

  Head pointer 53 (head is at cylinder 53)

Similar problems: limousine pickup/dropoff, elevator etc.

FCFS (First come first served)

Illustration shows total head movement. Cylinder 0 is outermost

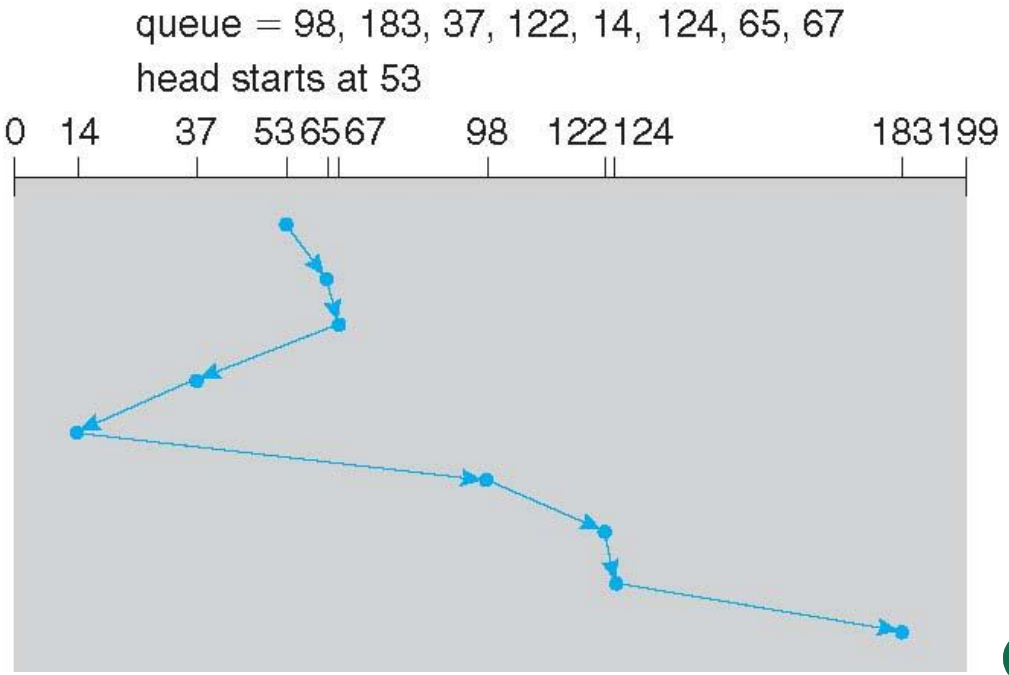


Total seek time = (98-53) + …..= **640** cylinders

Colorado State University

SSTF Shortest Seek Time First

- **Shortest Seek Time First** selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- total head movement of **236** cylinders

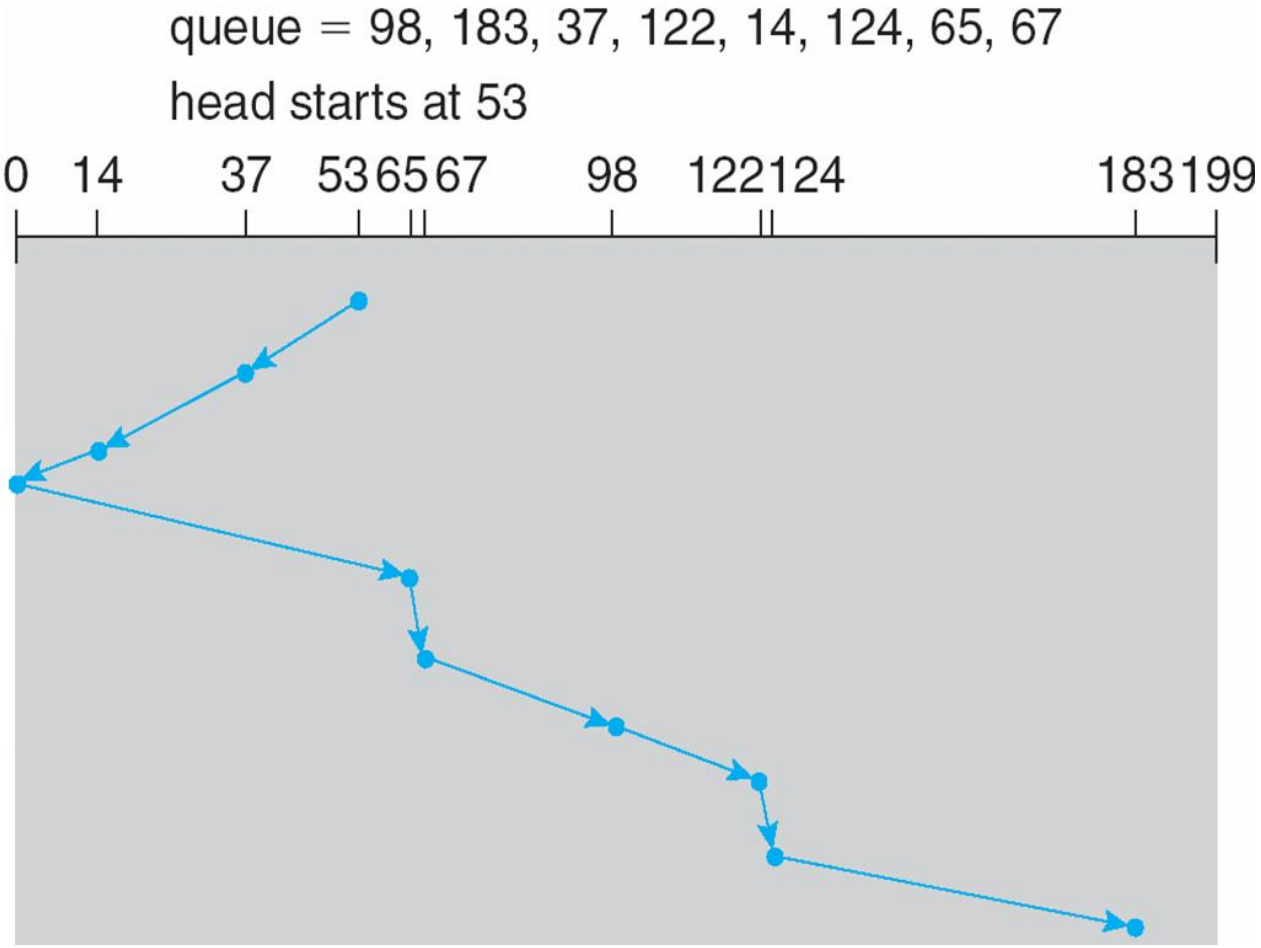


Colorado State University

SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other **end** of the disk, where the head movement is reversed, and servicing continues.
- **SCAN algorithm** Sometimes called the **elevator algorithm**
- But note that if requests are uniformly dense, largest density at the other end of disk and those wait the longest
- Variation: **Look**: may not go to the very edge

SCAN (Cont.)



LOOK will not go to 0

Total 53+ 183= **236** cylinders

FAQ

- LAN: local area network
- WAN: wide area network consisting of many LANs
- $\text{Page}_{\text{memory}}$ vs $\text{blocks/sectors}_{\text{disk}}$
- Difference among a file, its inode, and inode number?
  - inode number is the index of the inode in the inode table
- Hard links vs symbolic links:
  - Hard links refer to the same inode
  - Symbol link file is a pointer