

CS370 Operating Systems

Colorado State University

Yashwant K Malaiya

Fall 2021 L22

File-system Implementation



Slides based on

- Text by Silberschatz, Galvin, Gagne
- Various sources

FAQ

Partition vs volume

- A single disk can have multiple partitions with different file systems
- A volume can span multiple disks and still have a single file system: logical disk drive
- A partition with a file system can be referred to as volume.

How to partition a hard disk

- Locate Disk Management. Specify where you want the new partition etc.
- Find information on the web or Youtube.

Journaling

- Log (Journal) of partial changes until the overall change is *committed*. Makes changes *atomic*.
- Example: deleting a file
 1. Removing its directory entry.
 2. Releasing the inode (metadata) to the pool of free inodes.
 3. Returning any blocks used to the pool of free disk blocks.
 - Crash after 1: Orphan inode. After 2 Orphan blocks
 - Recovery: read journal and reapply changes
 - Implementations vary

FAQ

- Why can't I transfer a 4GB or larger file to my USB flash drive or memory card?
 - Files larger than 4GB can NOT be stored on a FAT32 volume. Formatting the flash drive as exFAT or NTFS will resolve this issue.
 - Compatible with both PC and Mac? FAT32 (most supported) or exFAT
- Partitioning: MBR (Master Boot Record: older, up to 2TB disks, max 4 partitions) or GPT (GUID Partition Table: newer, redundant partition/boot data, *globally unique* identifiers for components for developers. Late 1990s. Docu 2600 pages).
- Can windows mount files from linux; and vice versa?
 - Samba; auto mount depending on Linux version.

File-System Implementation

- Based on several on-disk and in-memory structures.
- On-disk
 - Boot control block (per volume) *boot block in unix*
 - Volume control block (per volume) *master file table in UNIX*
 - Directory structure (per file system) *file names and pointers to corresponding FCBs*
 - File control block (per file) *inode in unix*
- In-memory
 - Mount table about mounted volumes
 - The open-file tables (system-wide and per process)
 - Directory structure cache
 - Buffers of the file-system blocks

Volume: logical disk drive, perhaps a partition

Block Allocation Methods

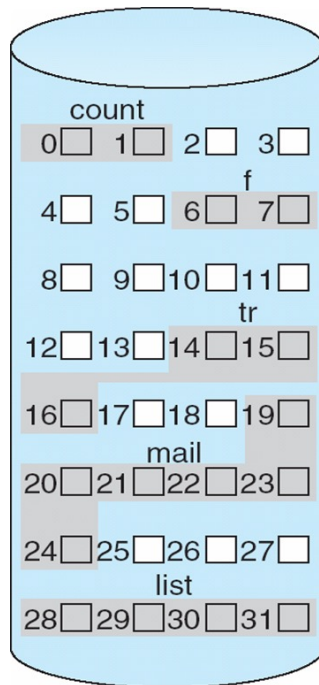
An allocation method refers to how disk blocks are allocated for files:

- Contiguous (not common, except for DVDs etc.)
- Linked, based on Linked (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.
Contrast these with allocation for processes in memory

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

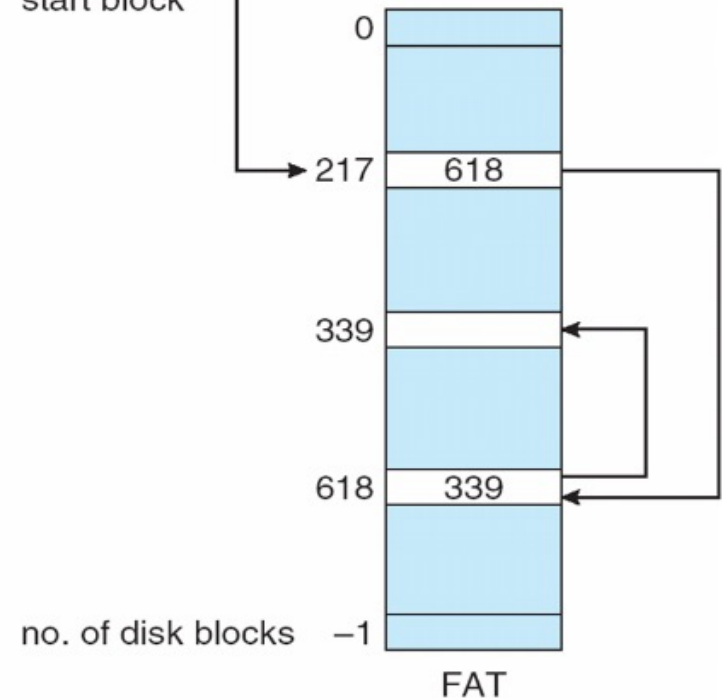
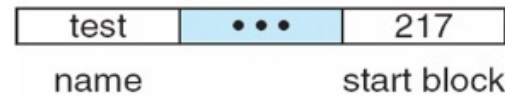
ii. Linked allocation – each file a linked list of blocks

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

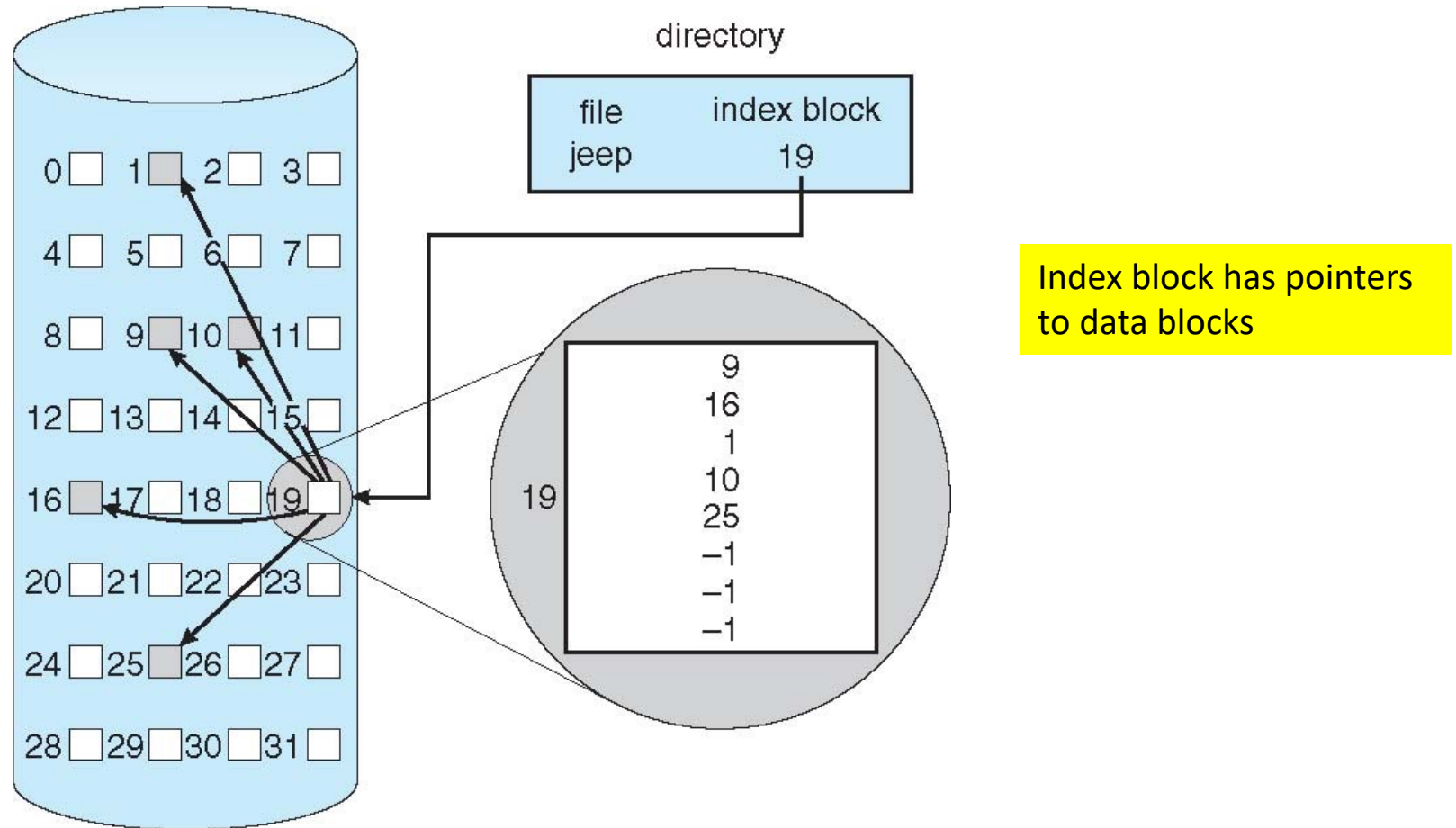
Free space management system called when new block needed

- 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

directory entry



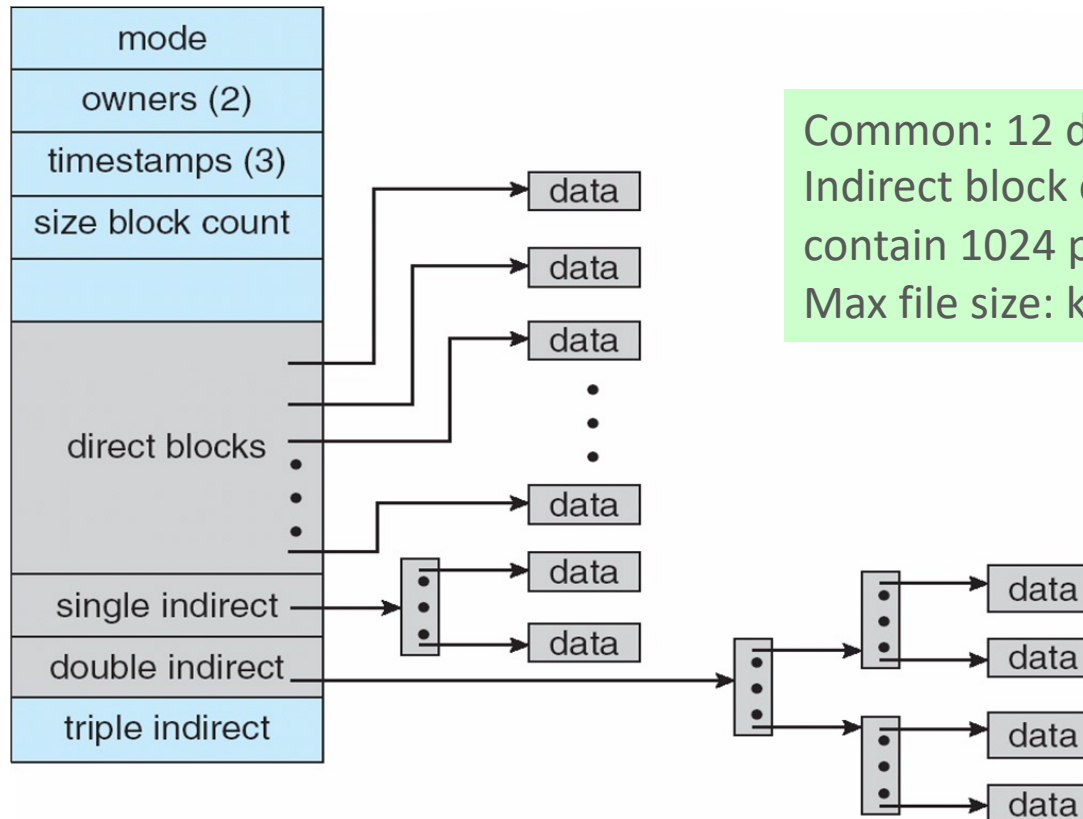
Example of Indexed Allocation



Indexed Scheme: UNIX inodes

Assume 4K bytes per block, 32-bit addresses

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

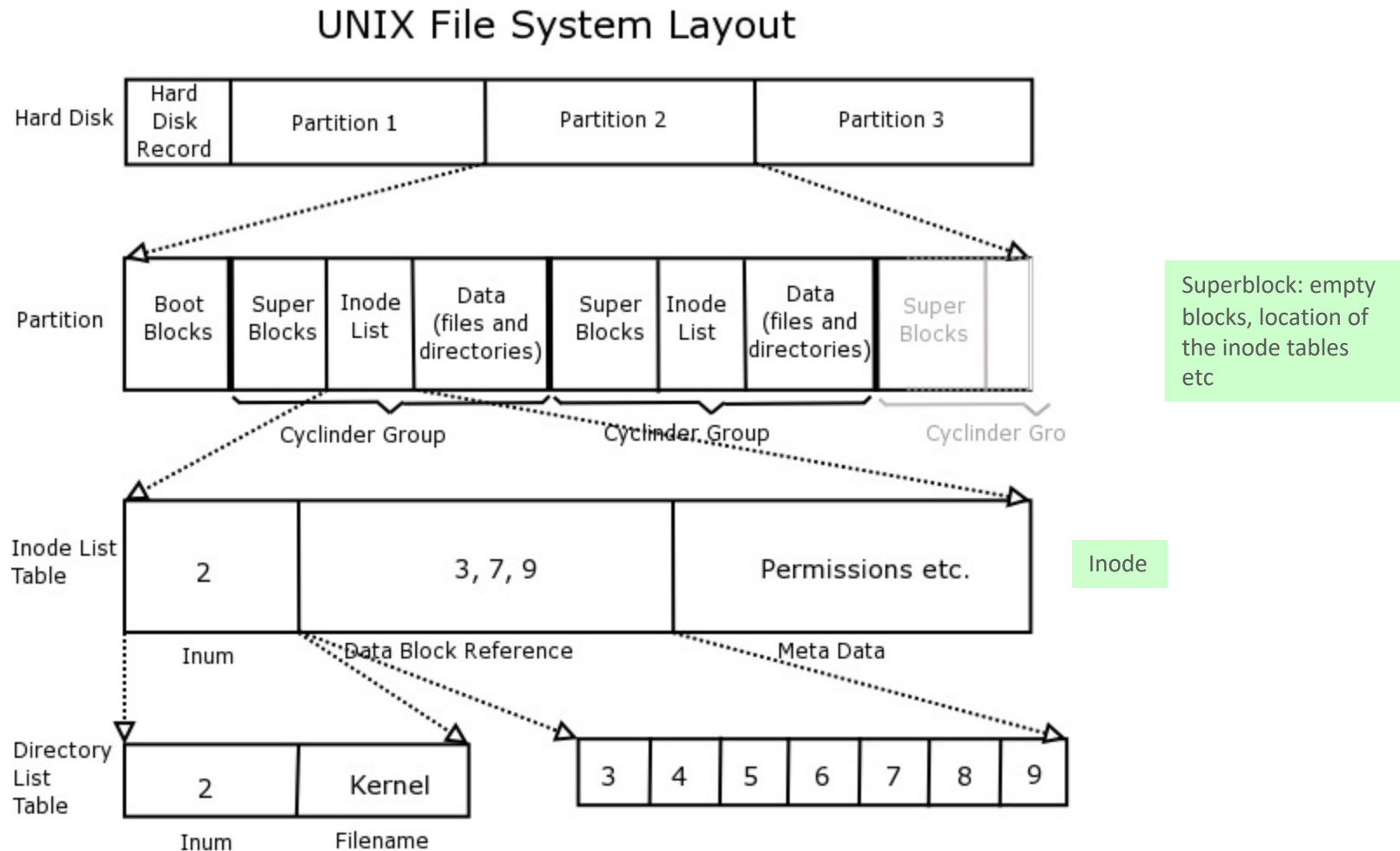


Common: 12 direct+3.
Indirect block could
contain 1024 pointers.
Max file size: k.k.k.4k (triple)+

Ext4: uses extents
(pointer+ length)

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

On-disk layout of a typical UNIX file system



Performance

- Best method depends on file access type
 - Contiguous great for sequential and random
 - Linked good for sequential, not random
 - Indexed more complex
 - Single block access could require 0-3 index block reads then data block read
 - Clustering can help improve throughput, reduce CPU overhead

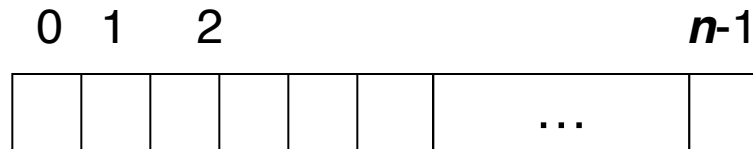
Cluster: set of contiguous sectors

Performance (Cont.)

- Adding instructions to the execution path to save one disk I/O is reasonable
 - Intel Core i7 Extreme Edition 990x (2011) at 3.46Ghz
= 159,000 MIPS
 - http://en.wikipedia.org/wiki/Instructions_per_second
 - Typical disk drive at 250 I/Os per second
 - $159,000 \text{ MIPS} / 250 = 630$ million instructions during one disk I/O
 - Fast SSD drives provide 60,000 IOPS
 - $159,000 \text{ MIPS} / 60,000 = 2.65$ 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**
- **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

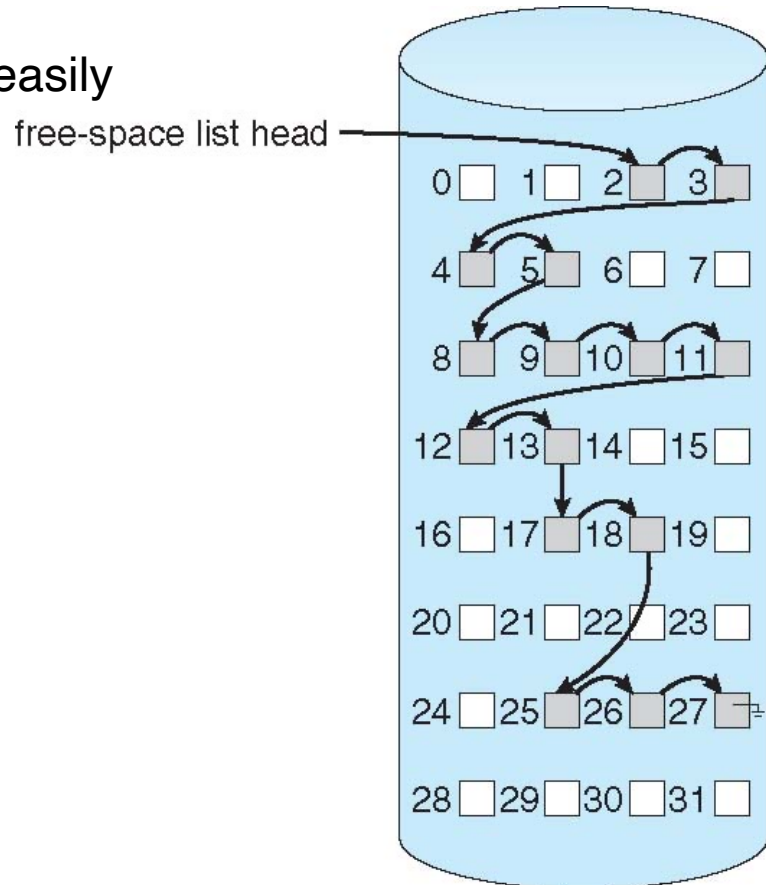
Free-Space Management (Cont.)

- 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`
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`

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 -li`

13320302	diskusage.txt
2408538	Documents/
680003	downloads/

CS370 Operating Systems

Colorado State University

Yashwant K Malaiya

Spring 2021. 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



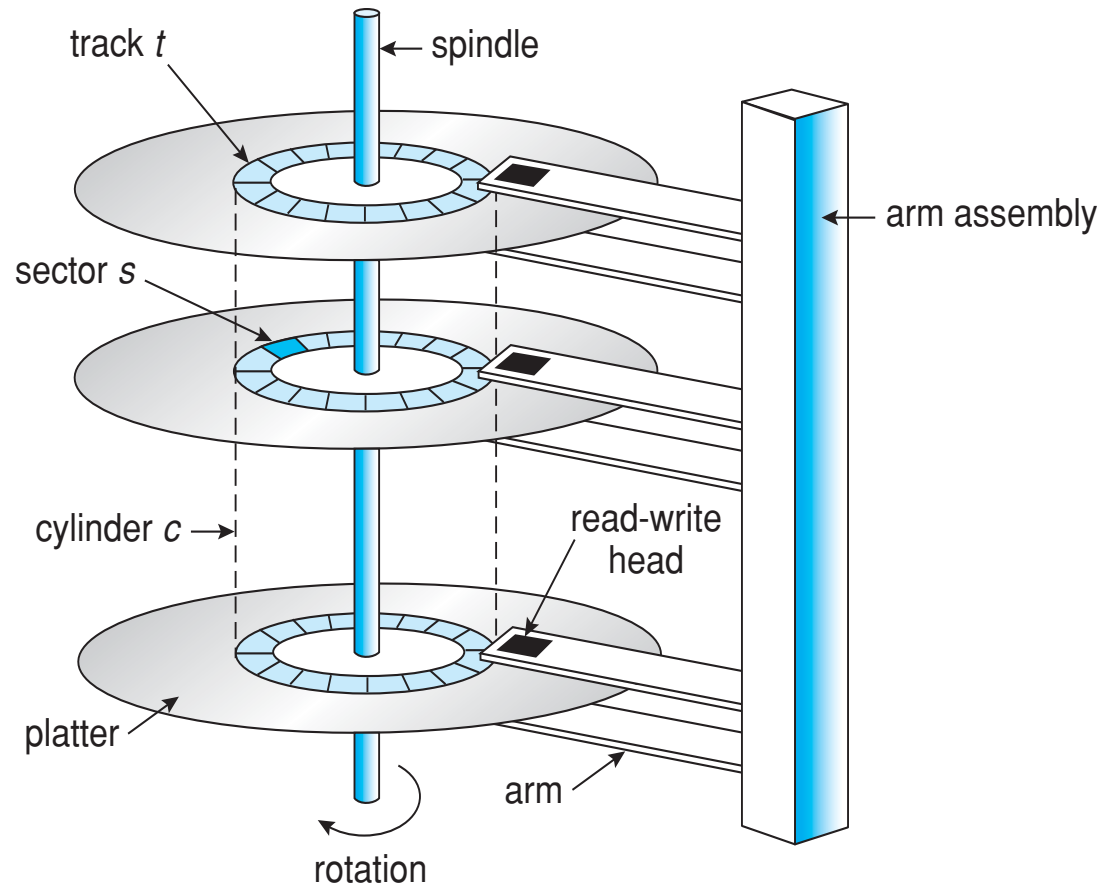
Objectives

- 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

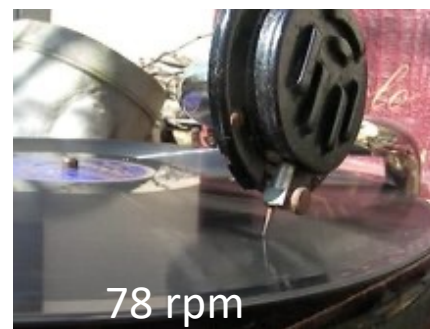


Hard Disks

- 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 Gb/sec
 - Effective Transfer Rate – real – 1Gb/sec (about 150 MB/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
 - $1 / (\text{RPM} / 60) = 60 / \text{RPM}$
 - Average latency = $\frac{1}{2}$ 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 = $4 \times 8K / G = 0.031$ 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

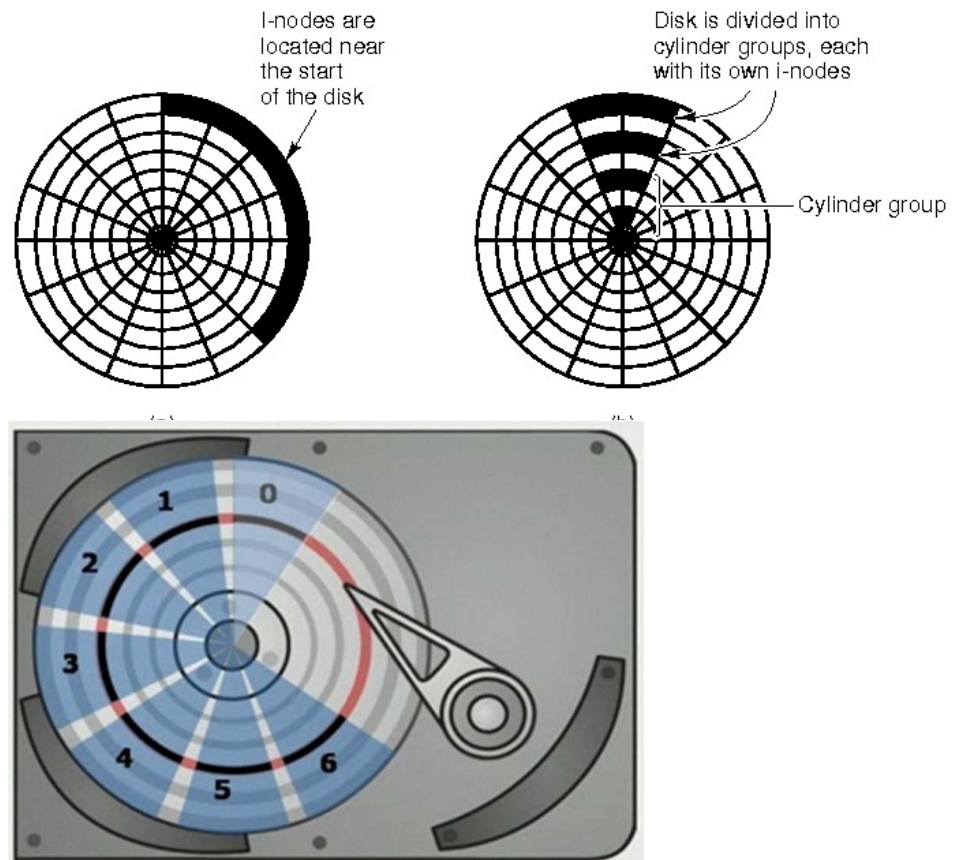
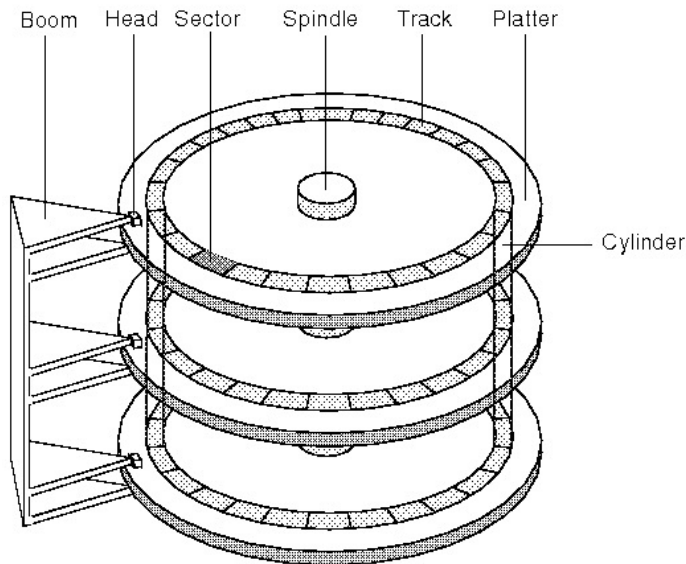
5M (7 bit) characters
50 x 24" platters
Access time = < 1 second

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?



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



Search ID: bfrn431
" MY DOG ATE THE FLASH DRIVE WITH MY
HOMEWORK ON IT...BUT I'M HOPING
TO GET IT BACK REAL SOON! "

SSD Architecture

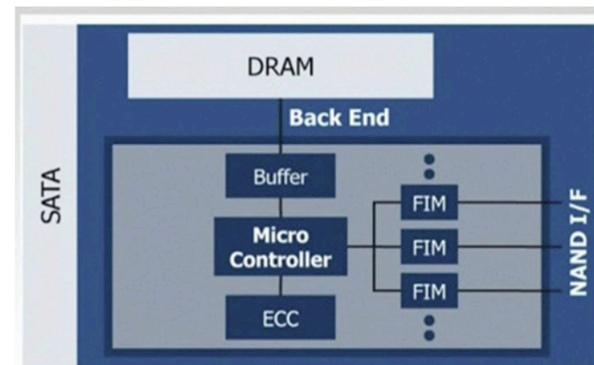
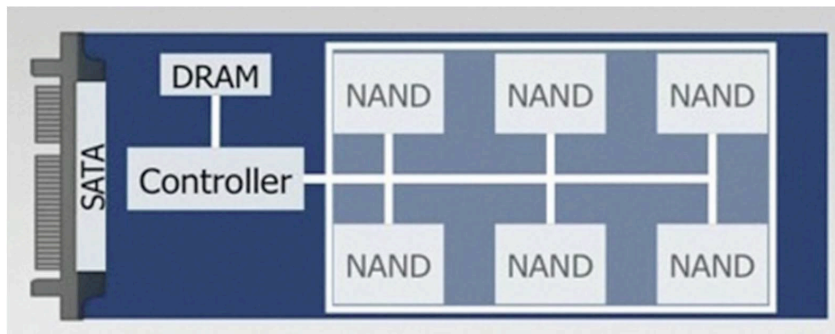
Controller

- 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

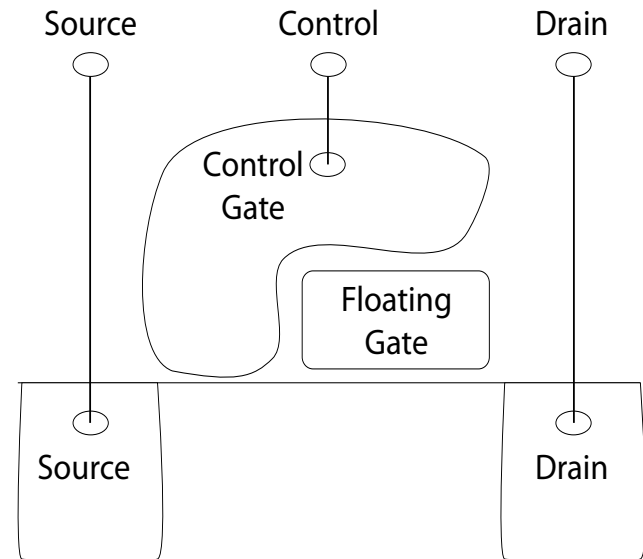
- Correcting errors
- Manages mapping
- Putting data into the flash or retrieving it from the flash

DRAM Cache – Reasonable amount of very low latency



Flash Memory

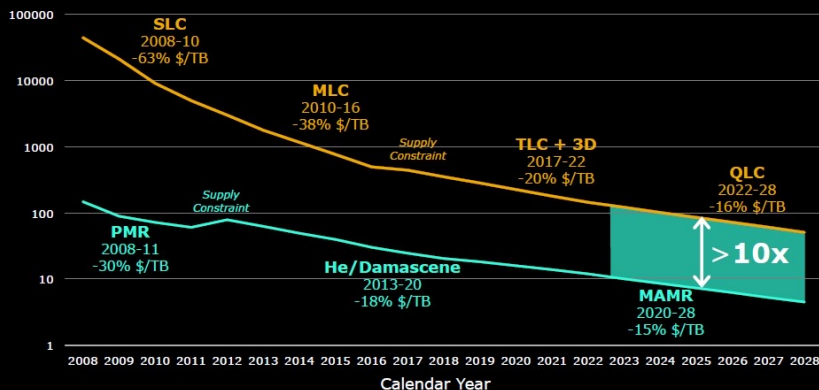
- Writes must be to “clean” cells; no update in place
 - Large block erasure required before write
 - Erasure block: 128 – 512 KB
 - Erasure time: Several milliseconds
- Write/read page (2-4KB)
 - 50-100 usec



SSD vs HDD

HDD vs. Flash SSD \$/TB Annual Takedown Trend

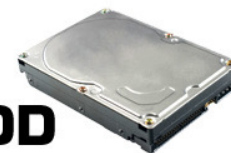
MAMR will enable continued \$/TB advantage over Flash SSDs



Western Digital

©2017 Western Digital Corporation or its affiliates. All rights reserved.

Source: WDC Analysis



SSD vs HDD

Usually 10 000 or 15 000 rpm SAS drives

0.1 ms

Access times

SSDs exhibit virtually no access time

5.5 ~ 8.0 ms

SSDs deliver at least

6000 io/s

Random I/O Performance

SSDs are at least 15 times faster than HDDs

HDDs reach up to

400 io/s

SSDs have a failure rate of less than

0.5 %

Reliability

This makes SSDs 4 - 10 times more reliable

HDD's failure rate fluctuates between

2 ~ 5 %

SSDs consume between

2 & 5 watts

Energy savings

This means that on a large server like ours, approximately 100 watts are saved

HDDs consume between

6 & 15 watts

SSDs have an average I/O wait of

1 %

CPU Power

You will have an extra 6% of CPU power for other operations

HDDs' average I/O wait is about

7 %

the average service time for an I/O request while running a backup remains below

20 ms

Input/Output request times

SSDs allow for much faster data access

the I/O request time with HDDs during backup rises up to

400 ~ 500 ms

SSD backups take about

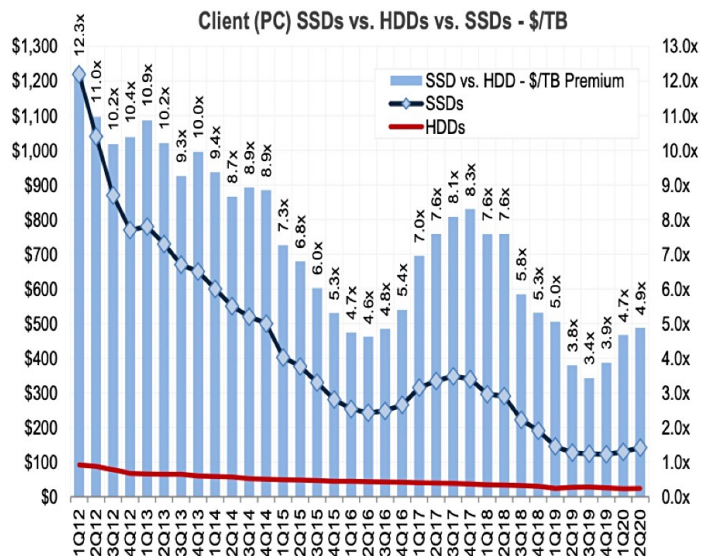
6 hours

Backup Rates

SSDs allows for 3 - 5 times faster backups for your data

HDD backups take up to

20 ~ 24 hours



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

Colorado State University

HDD vs SSD

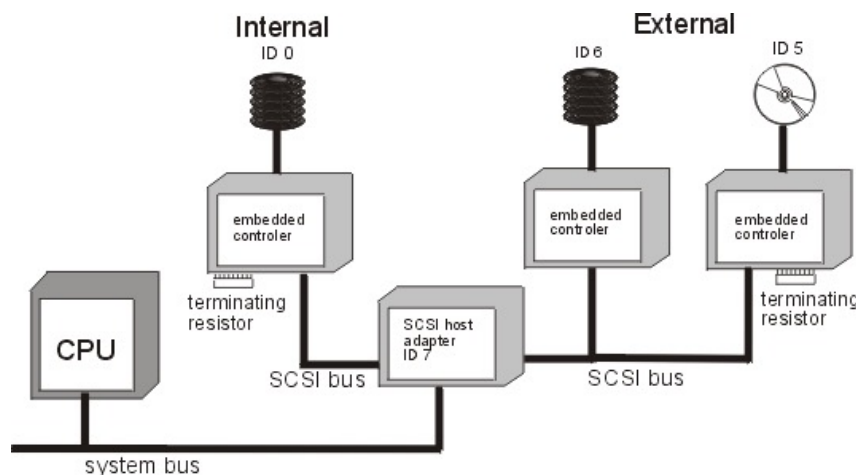
	HDD	SSD
	WD VelociRaptor	OCZ Vertex 3
Storage Capacity	600GB	120GB-360GB
Price for storage	48¢/ GB	2.08\$/GB x4
Seek Time/Rotational Speed	7ms/157 MB/s	
MTBF	1.4 million hours?	2 million hours?
Sequential Read/Write	1 MB/s	413.5/371.4 MB/s
Random Read	1 MB/s	68.8 MB/s
Random Write	1 MB/s	332.5 MB/s
IOPS	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 (SANs)** in which many hosts attach to many storage units

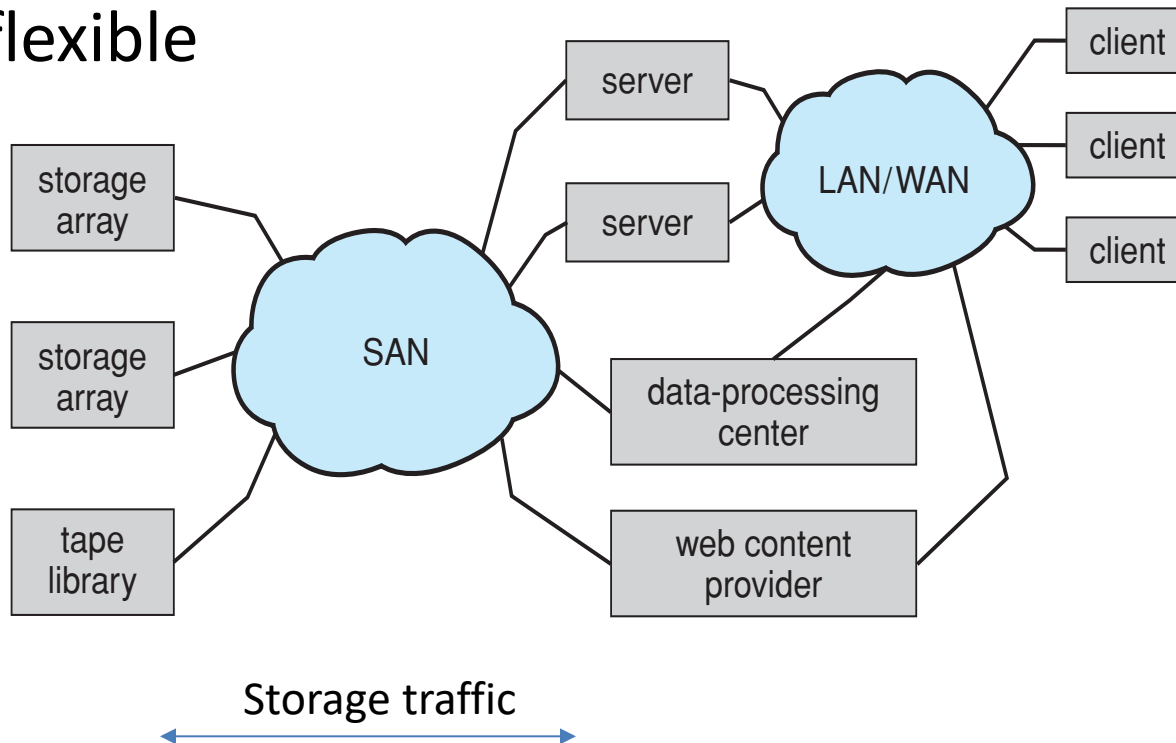


Storage Array

- 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

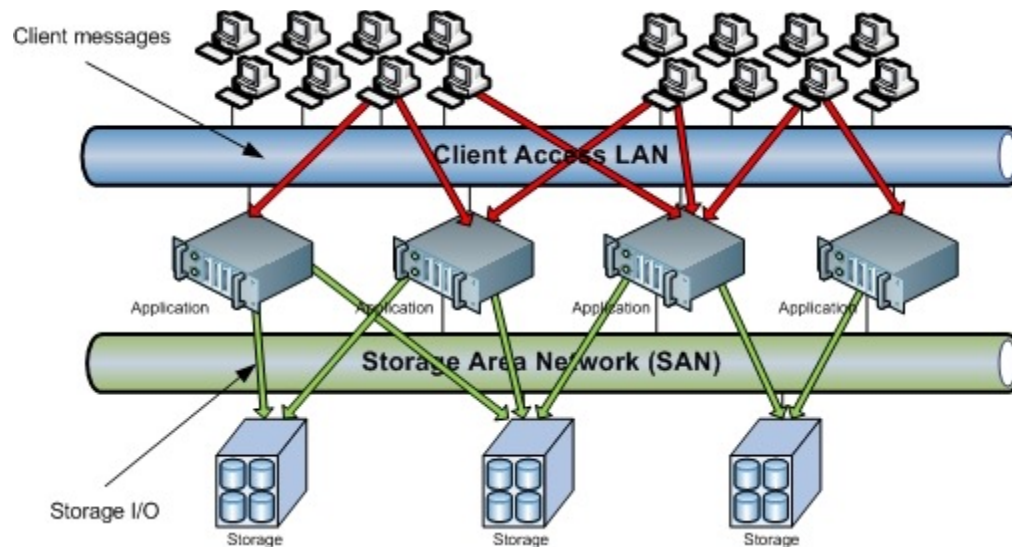
Storage Area Network

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



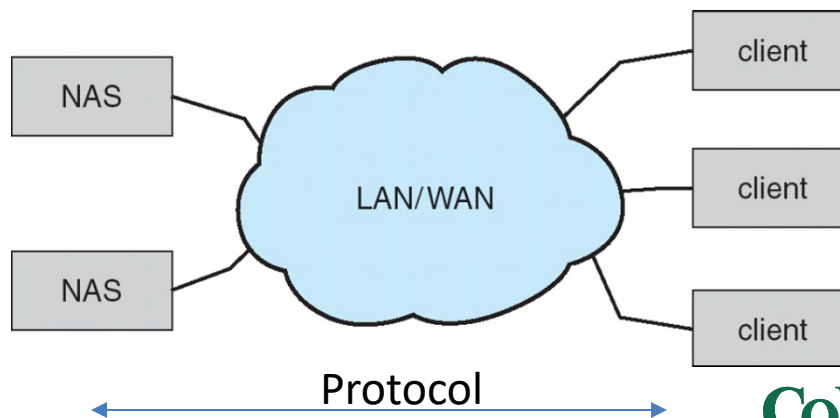
Storage Area Network (Cont.)

- 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 \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