Mass Storage Systems

Operating Systems

School of Data & Computer Science
Sun Yat-sen University

Lecture Notes: os_sysu@163.com
Instructor: Guoyang Cai
email: isscgy@mail.sysu.edu.cn

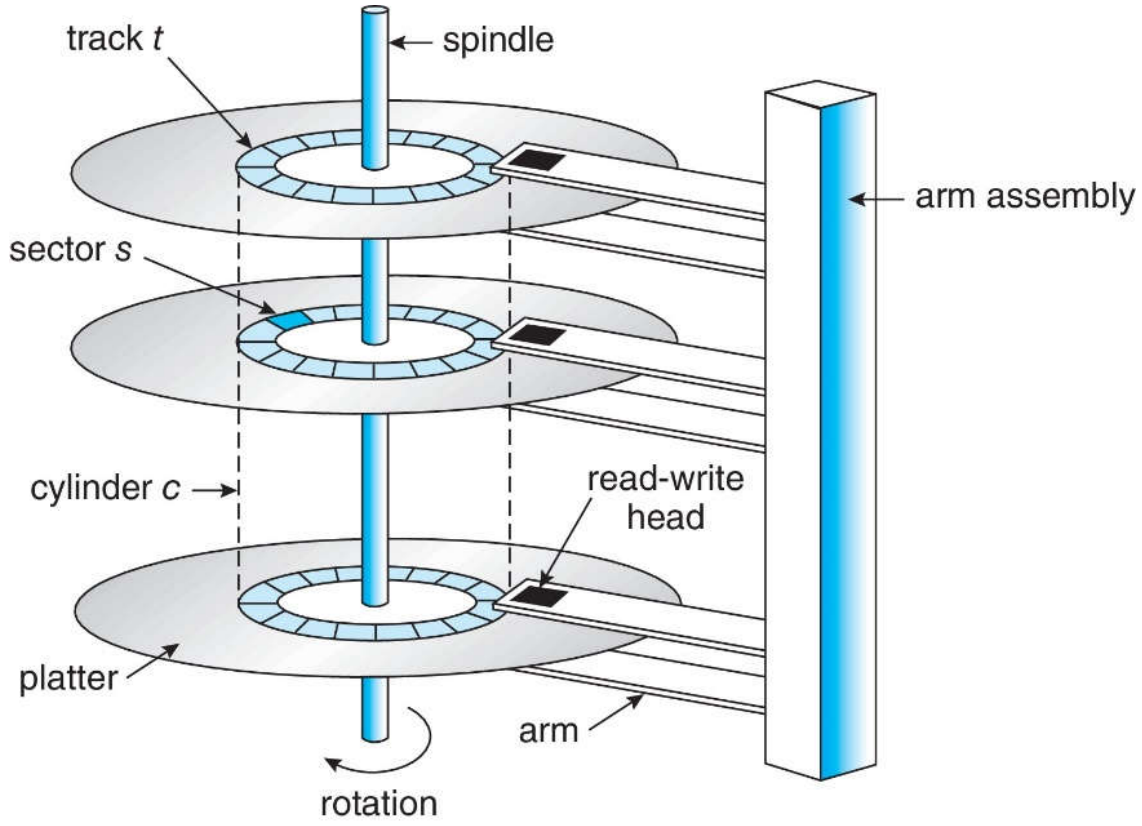
Contents

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 the rate at which data flow between the drive and the computer.
  - *Positioning time* (random-access time) is time to move the disk arm to the desired cylinder (seek time) and time for the desired sector to rotate under the disk head (rotational latency)
  - *Head crash* results from some disk head making contact with the disk surface. It cannot be repaired normally.
- 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.

Hard Disks

- Moving-head Disk Mechanism.



Hard Disks

- Platters sizes range from .85” to 14” (historically)
  - Commonly 3.5”, 2.5”, and 1.8”
- Capacities range from 420M to 3TB or bigger per drive.
- Performance
  - Transfer Rate – theoretical – 6 Gb/sec
  - Effective Transfer Rate – real – 1Gb/sec
  - Seek time from 3ms to 12ms – 9ms common for desktop drives
  - Average seek time measured or calculated based on 1/3 of tracks
  - Max latency based on spindle speed (period of rotation in sencond)
    - 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

Hard Disks

- Hard Disk Performance

$$\begin{aligned} \text{access\_latency} &= \text{average\_access\_time} \\ &= \text{average\_seek\_time} + \text{average\_latency} \end{aligned}$$

  - For slow disk (4200rpm) 9ms + 5.56ms = 14.56ms
  - For fastest disk (15000rpm) 3ms + 2ms = 5ms
  - SSD (non-magnetic, no moving parts) may drop to 0.05ms

$$\begin{aligned} \text{average\_I/O\_time} = \text{average\_access\_time} \\ + (\text{amount\_to\_transfer} / \text{transfer\_rate}) \\ + \text{controller\_overhead} \end{aligned}$$

Hard Disks

- Hard Disk Performance
  - Example.
    - Transfer a 4KB block on a 7200rpm disk with a 5ms average seek time, 1Gb/sec transfer rate with a 0.1ms controller overhead.

```
average_seek_time = 5ms
average_latency = 60000/7200/2 = 4.17ms
average_access_time = 5ms + 4.17ms = 9.17ms
controller_overhead = 0.1ms
amount_to_transfer = 4KB = 32Kb
transfer_rate = 1Gb/sec = 2^20 Kb/sec
transfer time = 32Kb / 2^20 Kb/sec = 3.1× 10^-5 s = 0.031ms
average_I/O_time = 9.17 + 0.031 + 0.1 = 9.301ms
```

Hard Disks

- The First Commercial Disk Drive
  - 1956, IBM RAMDAC computer included the IBM Model 350 disk storage system.
    - 5M (7 bit) characters
    - 50 × 24” platters
    - Access time ≤ 1 second

Solid-State Disks

- Solid-State Disk (SSD) is nonvolatile memory (非易失存储器) used like a hard drive.
  - Many technology variations
- Can be more reliable than HDDs
- More expensive per MB
- Maybe have shorter life span
- Less capacity
- Busses can be too slow
  - connect directly to PCI
- No moving parts, so no seek time or rotational latency
  - much faster

Magnetic Tape

- Magnetic Tape was early secondary-storage medium
  - 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 magnetic disk
  - 140MB/sec and greater
- 200GB to 1.5TB typical storage
- Common technologies are LTO-{3,4,5} and T10000

Hard Disk Structure

- Hard 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 logical blocks on physical media.
- 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 number of sectors per track via constant angular velocity (恒定角速度)

Disk Attachment

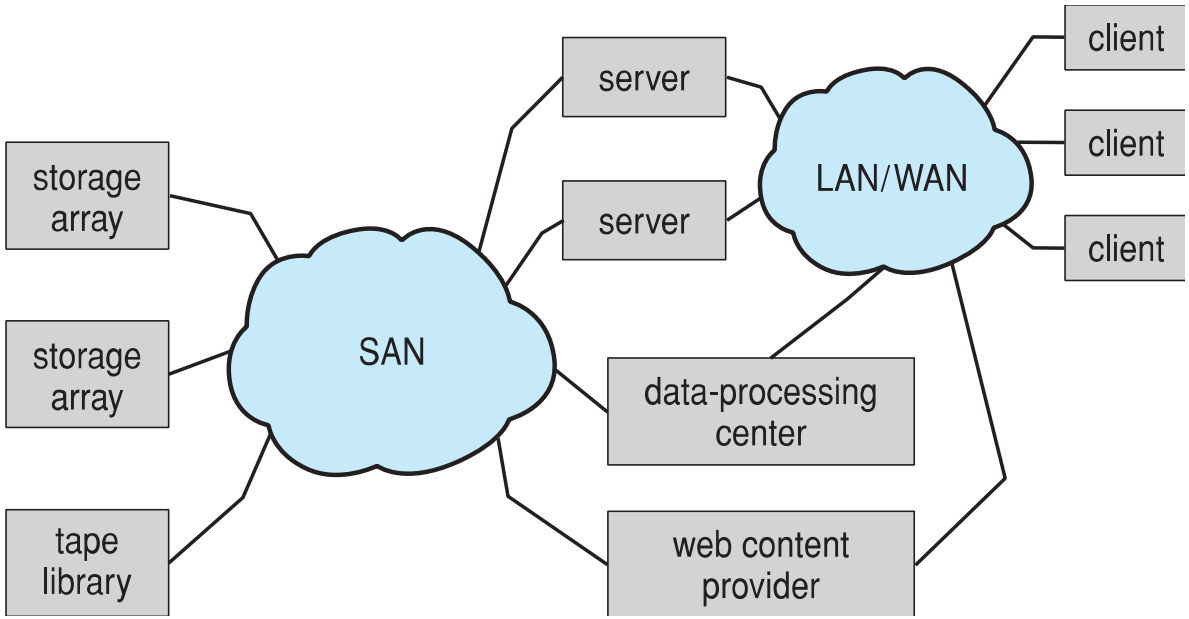
- Host-attached storage is accessed through I/O ports
  - talking to I/O busses.
- SCSI itself is a bus, up to 16 devices on one cable, *SCSI initiator* requests operation and *SCSI targets* perform tasks.
  - Each target can have up to 8 *logical units* (disks attached to device controller).
- Fibre Channel (FC) 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.
- I/O directed to bus ID
  - device ID, logical unit (LUN).

Disk Attachment

- Storage Array
  - Can just attach disks, or arrays of disks
  - Storage Array has controller(s), provides features to attached host(s).
    - Ports to connect hosts to array
    - Memory, controlling software (sometimes NVRAM, etc.)
    - A few to thousands of disks
    - RAID, hot spares, hot swap
    - Shared storage
      - more efficiency
    - Features found in some file systems
      - Snaphots, clones, thin provisioning, replication, deduplication, etc.

Storage Area Network

- Storage Area Network (SAN) is common in large storage environments.
- Multiple hosts are attached to multiple storage arrays – flexible.

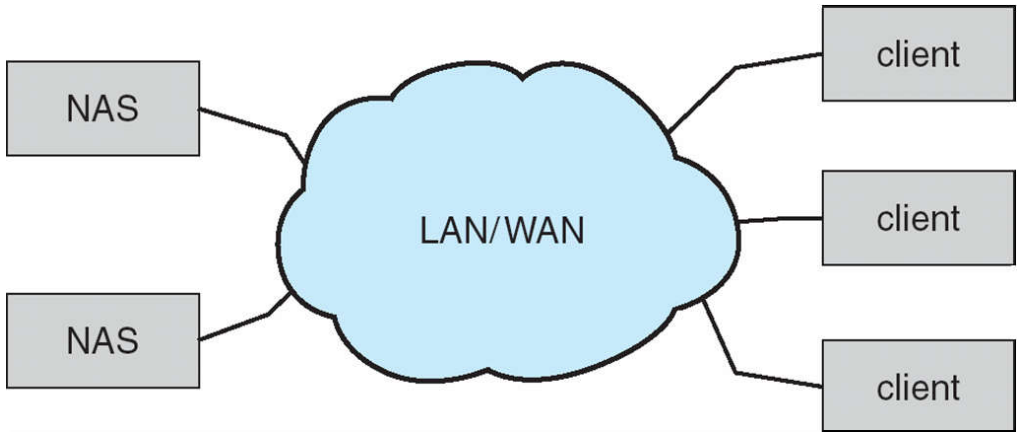


Storage Area Network

- SAN is one or more storage arrays.
  - Connected to one or more Fibre Channel switches
- Hosts also attach to the switches.
- Storage is made available via *LUN Masking* 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
- Why have separate storage networks and communications networks?
  - Consider iSCSI, FCOE

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 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)



HDD Scheduling

- Operating System is responsible for using hardware efficiently.
  - For the disk drives, this means having a fast access time and disk bandwidth.
- Two parts of Access Time
  - Seek time is the time for the disk to move the heads to the cylinder containing the desired sector.
  - Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head.
  - Assume that Seek time ≈ seek distance.
- 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.

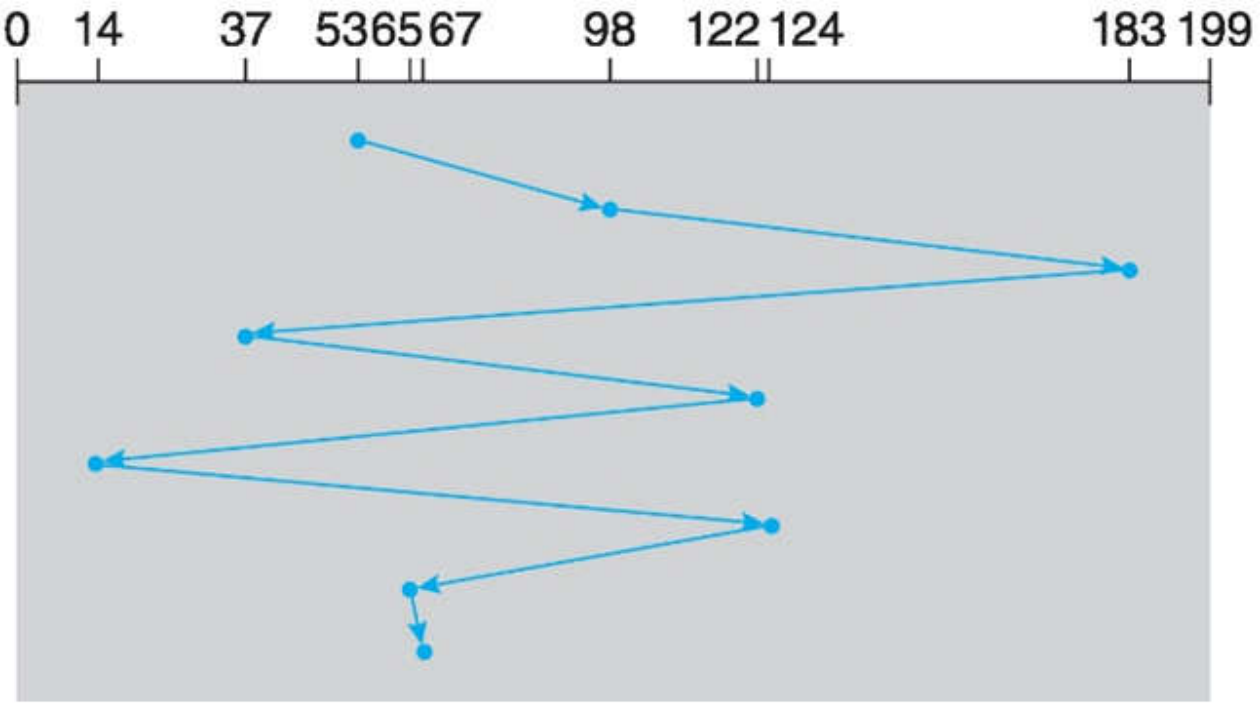
HDD Scheduling

- 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 the requests must queue up for services.
  - Drive controllers have small buffers and can manage a queue of I/O requests
  - Optimization algorithms only make sense *when a queue exists*.

First Come First Serve (FCFS)

- Illustration shows total head movement of 200 cylinders.
  - Assume that the initial head position is in cylinder 53. The queue with requests for cylinders is

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



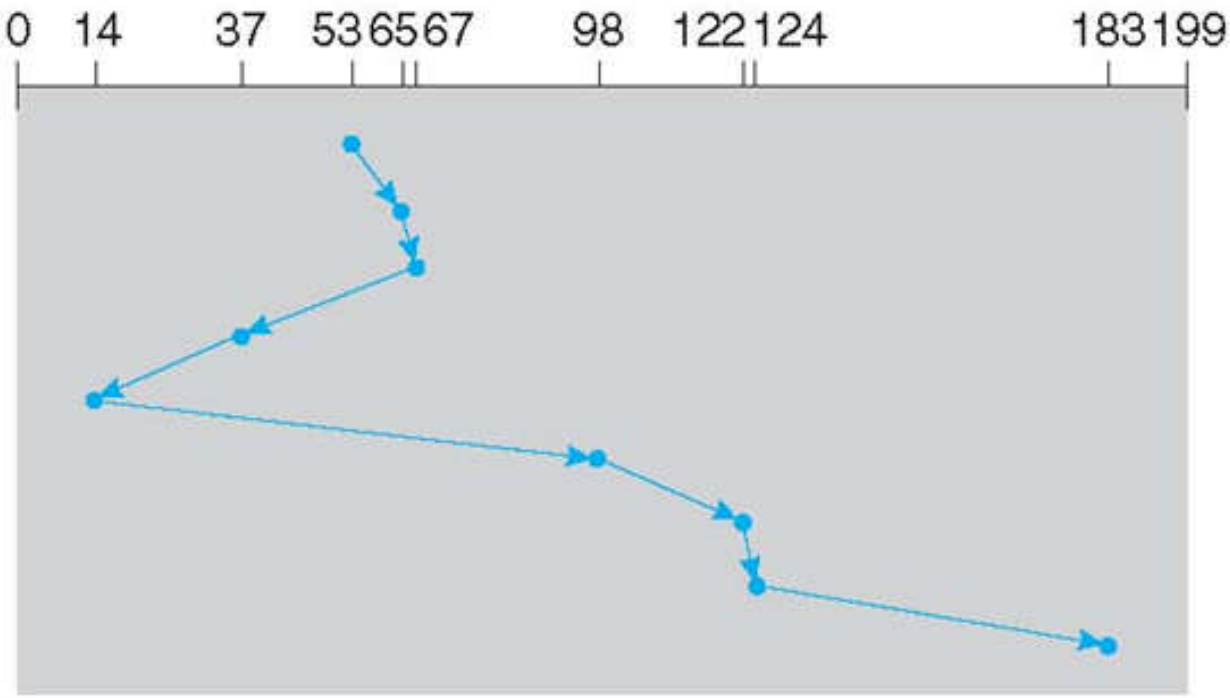
First Come First Serve (FCFS)

- Comments on FCFS
  - Handle I/O requests sequentially
  - Fair to all processes
  - Approaches random scheduling in performance if there are many processes/requests
  - Suffers from global zigzag effect (之字形效应).

Shortest Seek Time First (SSTF)

- Illustration shows total head movement of 200 cylinders.
  - Assume that the initial head position is in cylinder 53. The queue with requests for cylinders is

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



Shortest Seek Time First (SSTF)

- Comments on SSTF
  - SSTF selects the request with the minimum seek time from the *current* head position.
    - Also called Shortest Seek Distance First (SSDF) – It's easier to compute distances.
    - A Nearest-Neighbor algorithm (最近邻法).
  - It's biased in favor of the middle cylinders requests.
  - SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.

Elevator Algorithms

- Elevator algorithms are based on the common *elevator scheduling principle*.
- Four combinations of Elevator algorithms:
  - Service in both directions or in only one direction.
  - Go until the last cylinder or until the last I/O request.

Direction \ Go until	Go until the last cylinder	Go until the last request
Service in both directions	SCAN	LOOK
Service in only one direction	C-SCAN	C-LOOK

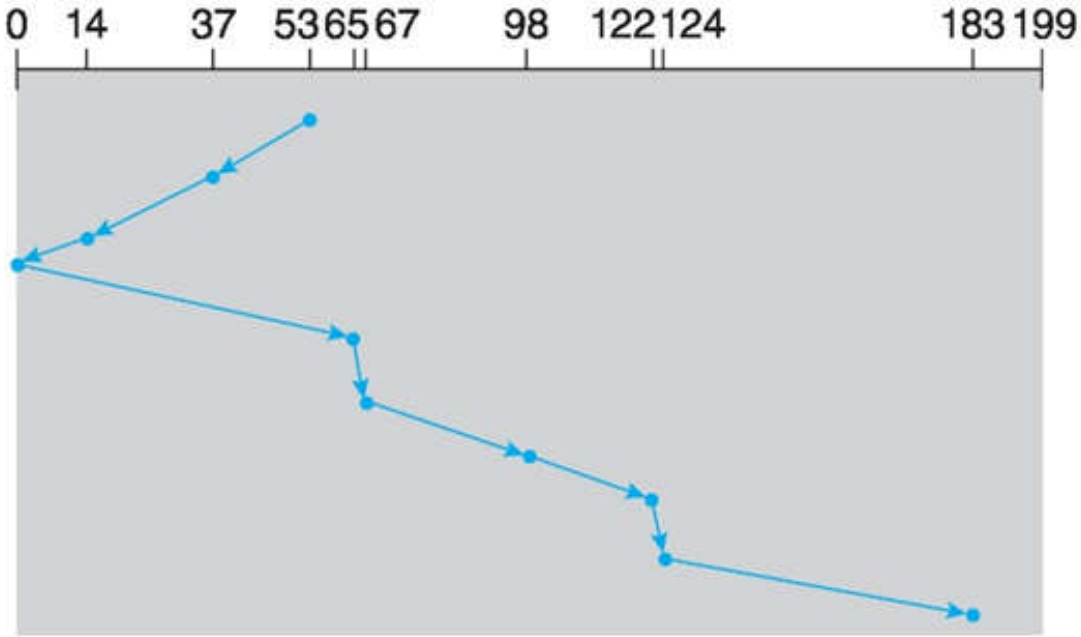
Elevator Algorithms

- SCAN Scheduling
  - The disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
    - The disk arm behaves just like an elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.
  - It tends to stay more at the ends so more fair to the extreme cylinder requests.
  - Note that in a uniform distribution of requests for cylinders, when the head reaches one end and reverses direction, relatively few requests are immediately in front of the head since these cylinders have recently been serviced. The heaviest density of requests is at the other end of the disk. These requests have also waited the longest.

Elevator Algorithms

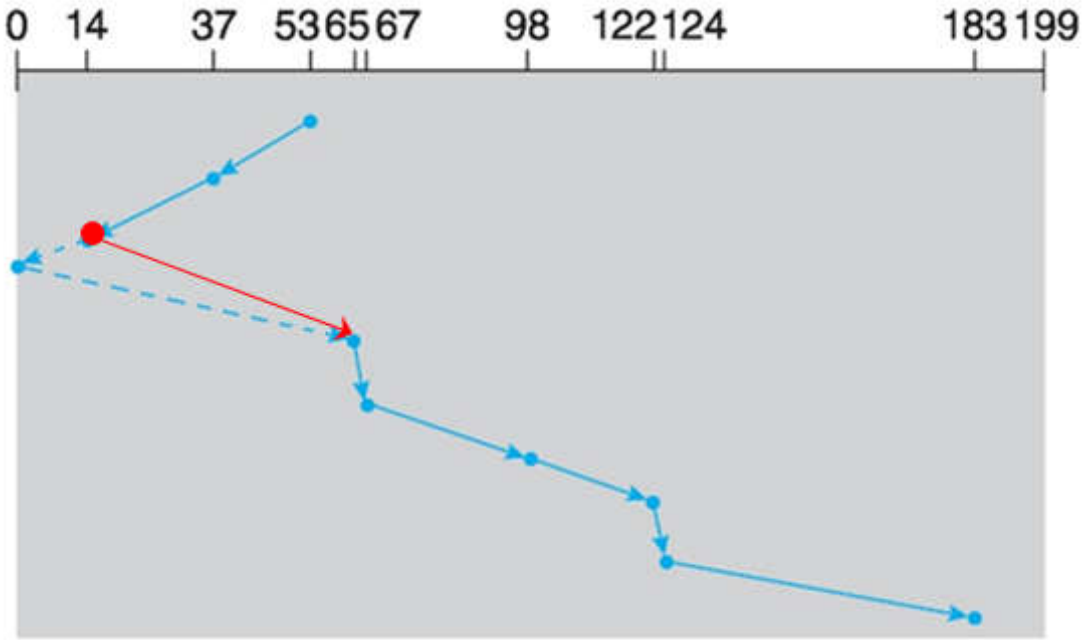
- SCAN Scheduling
  - Illustration shows total head movement of 200 cylinders.
    - Assume that the disk arm is moving toward 0 and that the initial head position is in cylinder 53. The queue with requests for cylinders is

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



Elevator Algorithms

- LOOK Scheduling
  - The disk arm starts at the first I/O request on the disk, and moves toward some end, servicing requests as it reaches each cylinder, until it gets to the last I/O request on that direction, where the head movement is reversed and servicing continues.
    - It moves in both directions.
    - more inclined to serve the middle cylinder requests.



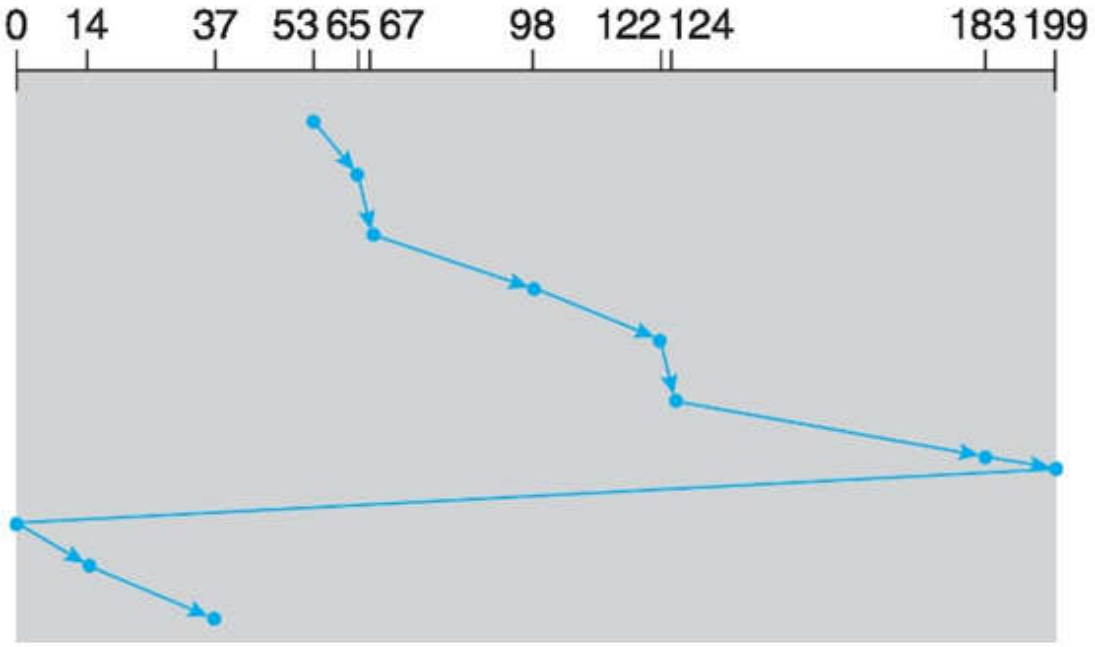
Elevator Algorithms

- C-SCAN Scheduling
  - Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk *without servicing* any requests on the return trip.
    - It treats the cylinders as a circular list that wraps around from the last cylinder to the first one.

Elevator Algorithms

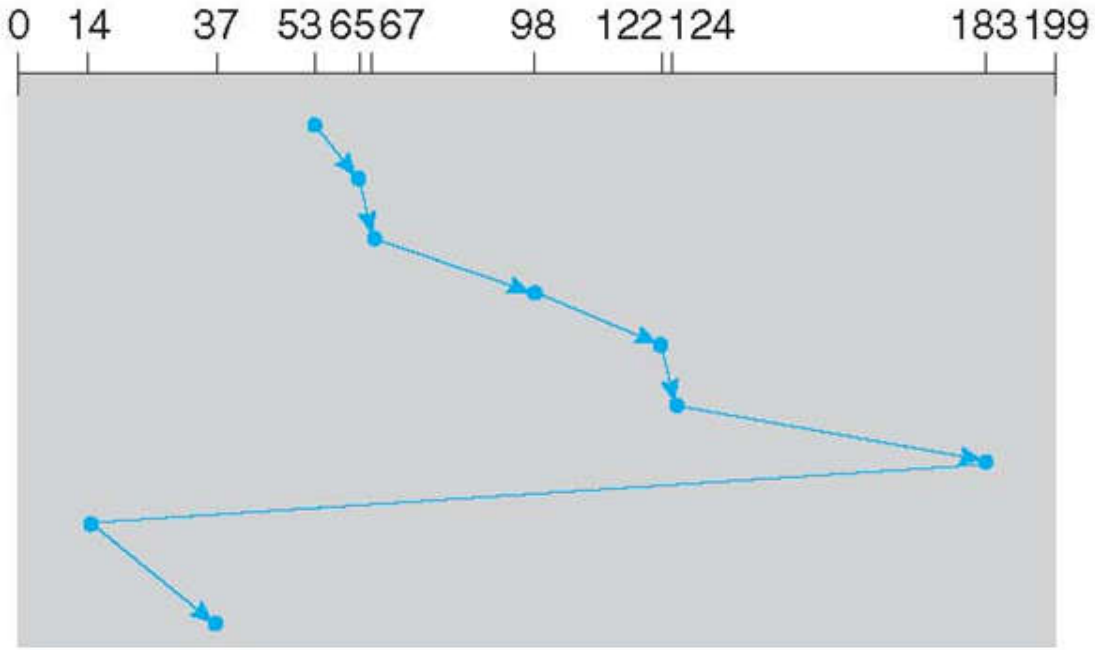
- C-SCAN Scheduling
  - Illustration shows total head movement of 200 cylinders.
    - Assume that the disk arm is moving toward 0 and that the initial head position is in cylinder 53. The queue with requests for cylinders is

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



Elevator Algorithms

- C-LOOK Scheduling
  - Circular LOOK (C-LOOK) scheduling is the LOOK version of C-SCAN.
  - In general, Circular versions are more fair but pay with a larger total seek time.
  - SCAN versions have a larger total seek time than the corresponding LOOK versions.



Disk Scheduling

- Examples. (William Stallings)

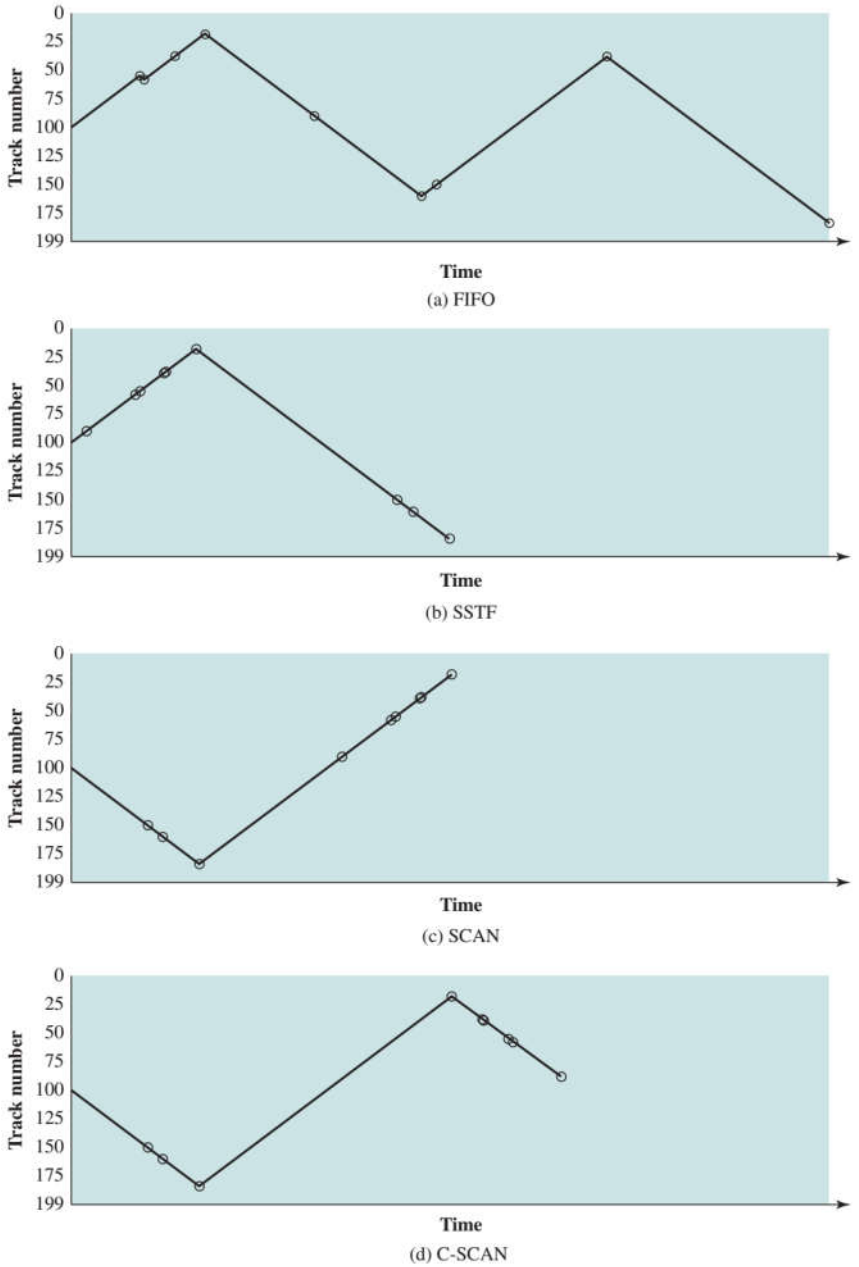
Table 11.2 Comparison of Disk Scheduling Algorithms

(a) **FIFO** (starting at track 100)		(b) **SSTF** (starting at track 100)		(c) **SCAN** (starting at track 100, in the direction of increasing track number)		(d) **C-SCAN** (starting at track 100, in the direction of increasing track number)	
Next track accessed	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	**Average seek length**	27.5	**Average seek length**	27.8	**Average seek length**	35.8

Request sequence

Disk Scheduling

- Examples. (William Stallings)



Disk Scheduling

- Other Disk Scheduling Policies
  - Pickup
    - A combination of FCFS and LOOK.
    - Goes to next I/O request by FCFS but services all existing requests on the way to it.
  - Priority
    - Goal is not to optimize disk use but to meet other objectives.
    - Short batch jobs may have higher priority.
    - Provide good interactive response time.

- **Disk Scheduling**
  - SCAN Algorithm Variations
    - FScan
      - Use two queues.
      - One queue is empty to receive new requests.
    - N-step-Scan
      - Segments the disk request queue into subqueues of length N.
      - Subqueues are processed one at a time, using SCAN.
      - New requests added to other queue when a certain queue is processed.

Selecting a Disk-Scheduling Algorithm

- With low load on the disk, It's FCFS anyway.
- SSTF is common and has a natural appeal.
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
  - Less starvation.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method and metadata layout.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

Selecting a Disk-Scheduling Algorithm

- From William Stallings.

Table 11.3 Disk Scheduling Algorithms

Name	Description	Remarks
Selection according to requestor		
Random	Random scheduling	For analysis and simulation
FIFO	First-in-first-out	Fairest of them all
PRI	Priority by process	Control outside of disk queue management
LIFO	Last-in-first-out	Maximize locality and resource utilization
Selection according to requested item		
SSTF	Shortest-service-time first	High utilization, small queues
SCAN	Back and forth over disk	Better service distribution
C-SCAN	One way with fast return	Lower service variability
N-step-SCAN	SCAN of *N* records at a time	Service guarantee
FSCAN	*N*-step-SCAN with N = queue size at beginning of SCAN cycle	Load sensitive

Disk Management

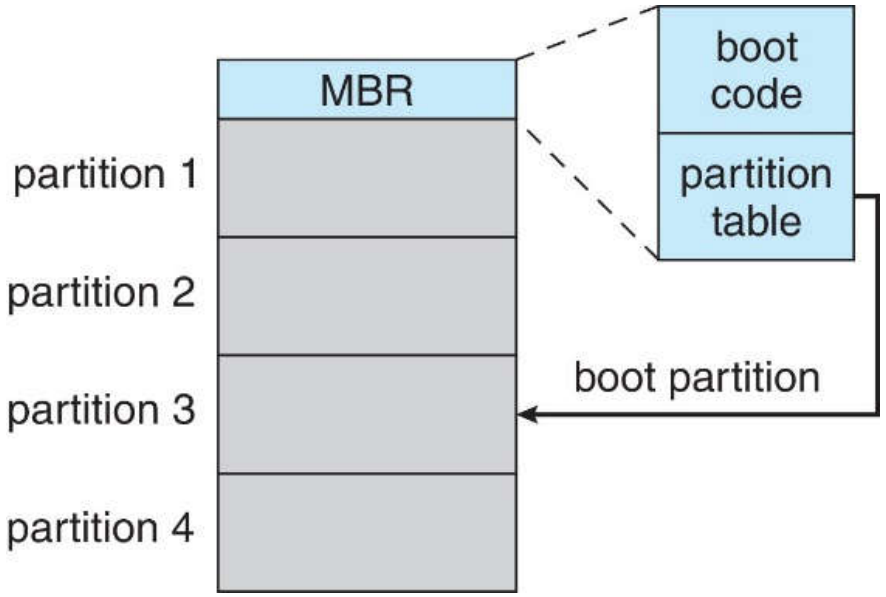
- *Low-level formatting*, or *physical formatting*
  - Low-level formatting divides a disk into sectors that the disk controller can read and write.
  - Each sector can hold header information, plus data, plus error correction code (*ECC*).
  - A sector usually contains 512 bytes of data but can be selectable.
- To use a disk to hold files, the operating system still needs to record its own data structures on the disk.
  - *Partitioning* the disk into one or more groups of cylinders, each treated as a logical disk
  - *Logical formatting* or “making a file system”
- To increase efficiency most file systems group blocks into *clusters.*
  - Disk I/O done in blocks
  - File I/O done in clusters

Disk Management

- Raw disk access
  - Applications that want to do their own block management keep OS out of the way.
    - E.g., DBMS.
- Boot block initializes the system.
  - *The bootstrap* was stored in ROM.
  - *Bootstrap loader* program was stored in boot blocks of boot partition.
- Methods such as *sector sparing* is used to handle bad blocks.

Disk Management

- Booting from a Disk in Windows.



Disk Management

- Booting Processes in Linux
  - (1) BIOS (Basic Input Output System)
    - Perform Power On Self Testing, POST
  - (2) MBR (Master Booting Record, 1st sector in the disk)
    - Load GRUB2
  - (3) GRUB2 (GRand Unified Boot-loader version 2)
    - Load the vmlinuz kernel image
    - Extract the contents of initramfs image
  - (4) KERNEL
    - Loads necessary driver modules from initrd image; Systemd starts systems first process.
    - Kernel initialization highlights include:
      - initialize CPU components, e.g., MMU
      - initialize the scheduler (PID 0)
      - mount the root filesystem in rw mode
      - fork off the init process (PID 1)

Disk Management

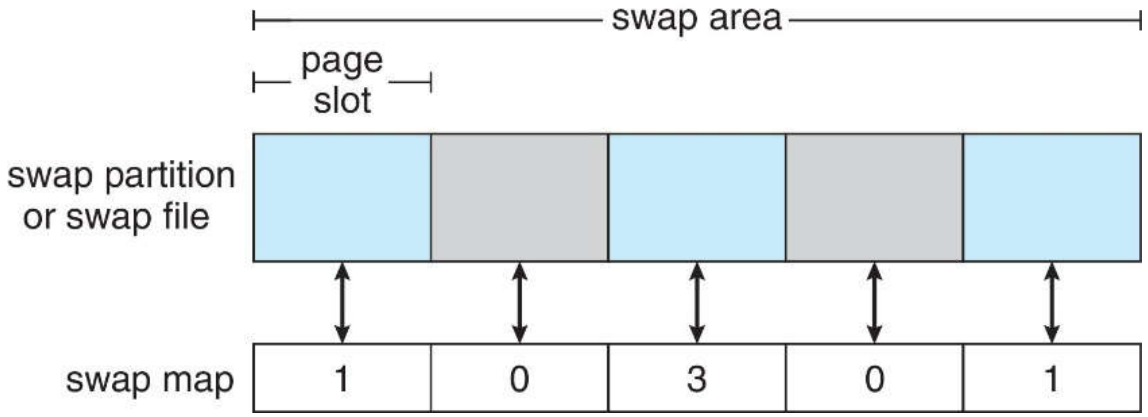
- Booting Processes in Linux
  - (4) KERNEL
    - In essence, kernel initialization does two things:
      - Start the core system of shared resource managers (RAM, processor and mass storage).
      - Starts a single process, /sbin/init.
    - Init process (sbin/init) is the very first process which loads all the various daemons and mounts all the partitions which are listed under /etc/fstab.
  - (5) SYSTEMD
    - Read conf files from /etc/system
    - Read file linked by /etc/systemd/system/default.target
    - Brings the system to the state defined by the system target.

Swap-Space Management

- Swap-space
  - Virtual memory uses disk space as an extension of main memory.
  - Less common now due to memory capacity increases.
- Swap-space can be carved out of the normal file system, or, more commonly, it can be in a separate disk partition (raw).
- 4.3BSD allocates swap-space when process starts.
  - The space holds text segment (the program) and data segment.
  - Kernel uses *swap maps* to track swap-space use.
- Solaris 2 allocates swap-space only when a dirty page is forced out of physical memory, not when the virtual memory page is first created.
  - File data written to swap-space until write to file system requested
  - Other dirty pages go to swap-space due to no other home
  - Text segment pages thrown out and reread from the file system as needed.
- What if a system runs out of swap-space?
  - Some systems allow multiple swap-spaces

Swap-Space Management

- Data Structures for Swapping on Linux.



RAID

- RAIDs
  - "Redundant Arrays of Independent Disks"
    - "Redundant Arrays of Inexpensive Disks" in the old days.
  - multiple disk drives provides reliability via *redundancy.*
- *Mean time between failure* – MTBF, exposure time when another failure could cause data loss. (平均失效时间)
- *Mean time to repair* – the time it takes to replace a failed drive and to restore the data on it. (平均修复时间)
- *Mean time to data loss* based on above factors.
- If mirrored disks fail independently, consider 100,000 hours of MTBF and 10 hours of mean time to repair for a single disk.
  - The mean time to data loss of such a mirrored system is

    $$100000^2/(2\times10) = 500\times10^6 \text{ (hours)},$$

    - or 57,000 years!
- Frequently combined with *NVRAM* to improve write performance
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.

RAID

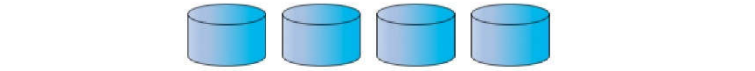
- Data Striping (数据分拆)
  - Bit-level Striping (按位分拆)
    - Bit-level Striping splits the bits of each byte across multiple drives.
    - E.g., for an array of eight drives, bit i of each byte is written to drive i. The storage array can be treated as a single drive with sectors that are eight times the normal size and have eight times the access rate.
  - Block-level Striping (按块分拆)
    - Block-level Striping is the only commonly available striping. It strips blocks of a file across multiple drives.
    - E.g., with n drives, block i of a file goes to drive (i mod n)+1.
  - Other levels of striping
    - bytes of a sector; sectors of a block.
  - Parallelism in a storage system achieved through striping is to:
    - Increase the throughput of multiple small accesses (that is, page accesses) by load balancing.
    - Reduce the response time of large accesses.

RAID

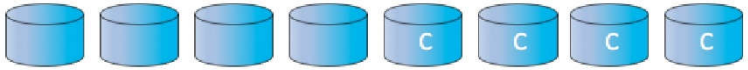
- RAID is arranged into six different levels
  - *Mirroring* or *shadowing* (RAID 1) keeps duplicate of each disk.
  - Striped mirrors (RAID 1+0) or mirrored stripes (RAID 0+1) provides high performance and high reliability.
  - *Block interleaved parity* (RAID 4, 5, 6) uses much less redundancy.
- RAID within a storage array can still fail if the array fails, so automatic *replication* of the data between arrays is common.
  - Frequently, a small number of *hot-spare* disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them.

RAID

- RAID Levels.



(a) RAID 0: non-redundant striping.

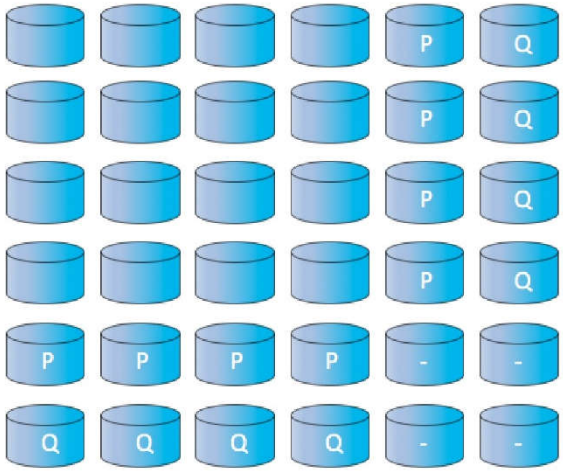


(b) RAID 1: mirrored disks.

(c) RAID 4: block-interleaved parity.

(d) RAID 5: block-interleaved distributed parity.

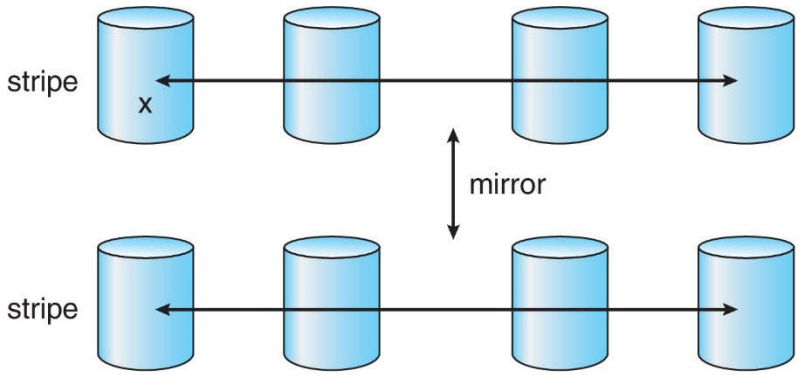
(e) RAID 6: P + Q redundancy.



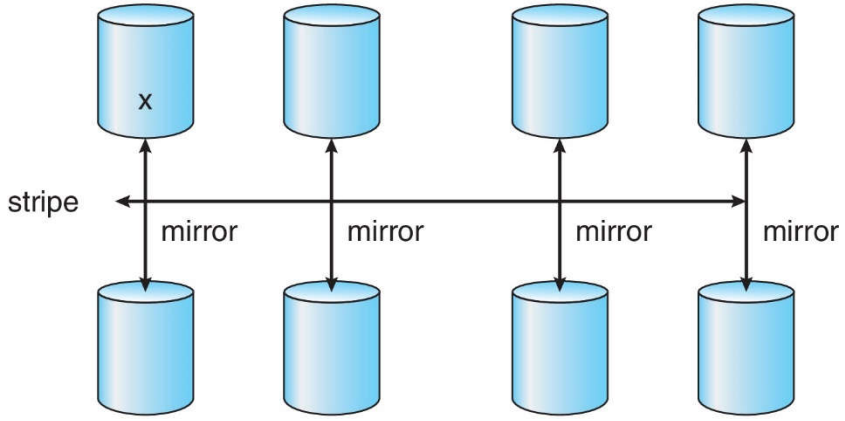
(f) Multidimensional RAID 6.

RAID

- RAID (0 + 1) and (1 + 0)



a) RAID 0 + 1 with a single disk failure.



b) RAID 1 + 0 with a single disk failure.

Other Features

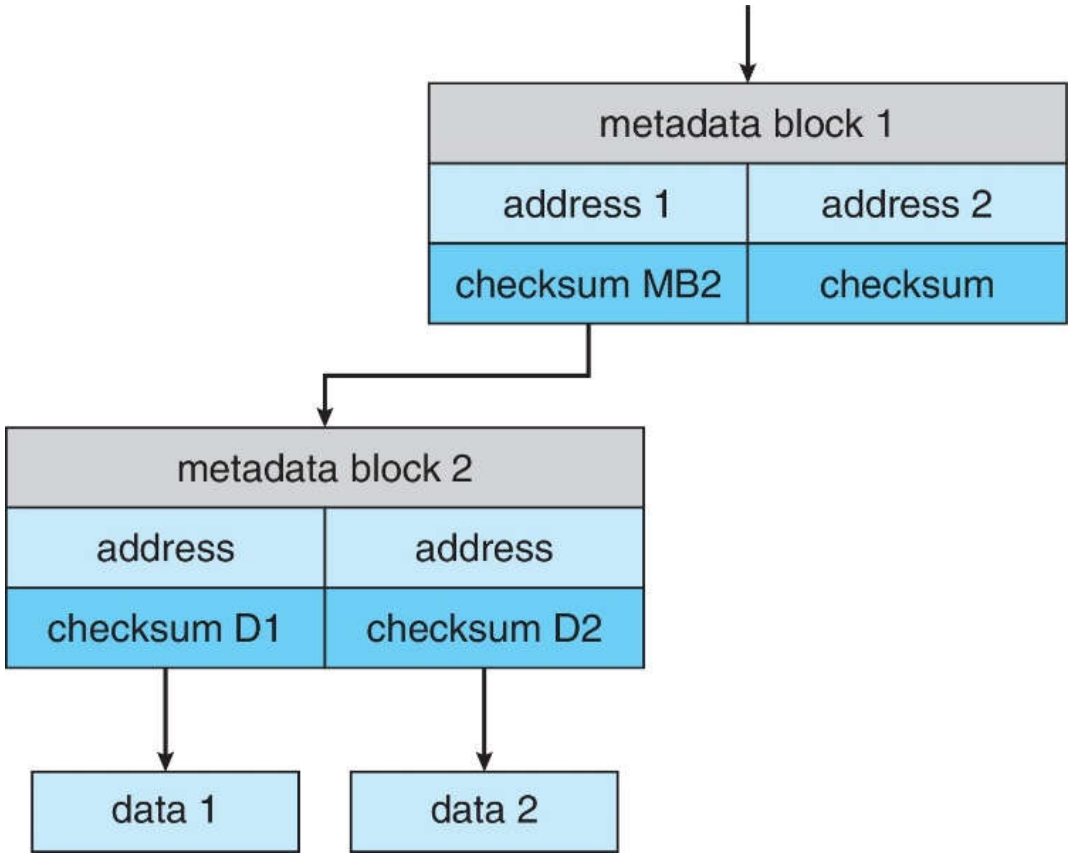
- Regardless of where RAID implemented, other useful features can be added.
- *Snapshot* is a view of file system before a set of changes take place.
  - i.e. at a point in time
- *Replication* is automatic duplication of writes between separate sites.
  - For redundancy and disaster recovery
  - Can be synchronous or asynchronous
- *Hot spare* disk is not used for data but is configured to be used as a replacement in case of drive failure.
  - It is automatically used by RAID production to replace the failed disk and rebuild the RAID set if possible.
  - Hot spare decreases the mean time to repair.
- *RAID alone* does not prevent or detect data corruption or other errors, just disk failures

Other Features

- Solaris ZFS adds *checksums* of all data and metadata.
  - Checksums are kept with pointer to object, to detect if object is the right one and whether it changed.
  - Checksums can be used to detect and correct data and metadata corruption.
  - ZFS also removes volumes, partitions.
    - Disks allocated in *pools*
    - File systems with a pool share that pool, use and release space like malloc() and free() memory allocate / release calls.

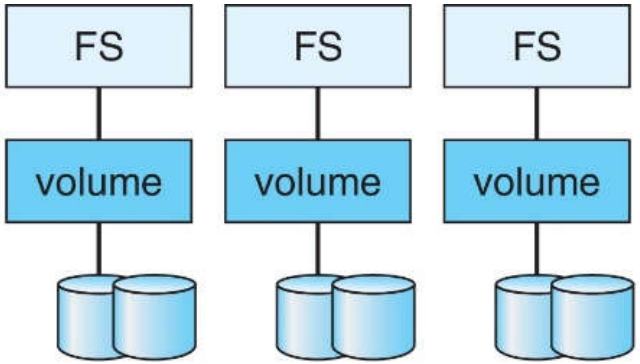
Other Features

- ZFS Checksums All Metadata and Data.

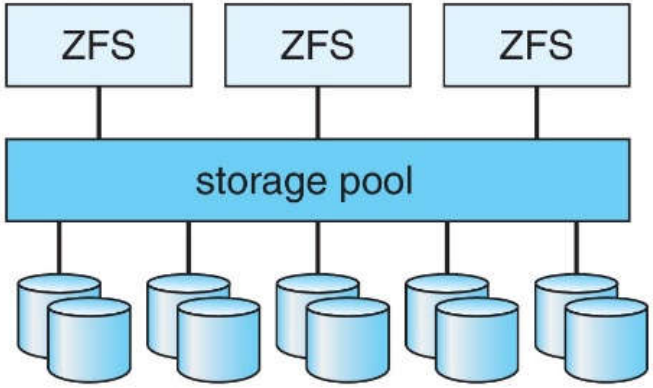


Other Features

- Traditional and Pooled Storage.



(a) Traditional volumes and file systems.



(b) ZFS and pooled storage.

Stable-Storage Implementation

- Write-ahead log scheme requires stable storage.
- Stable storage means data is never lost (due to failure, etc.)
- To implement stable storage:
  - Replicate information on more than one nonvolatile storage media with independent failure modes
  - Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery.
- Disk write has 1 of 3 outcomes.
  - Successful completion - The data were written correctly on disk.
  - Partial failure - A failure occurred in the midst of transfer, so only some of the sectors were written with the new data, and the sector being written during the failure may have been corrupted.
  - Total failure - The failure occurred before the disk write started, so the previous data values on the disk remain intact.

Stable-Storage Implementation

- If failure occurs during block write, recovery procedure restores block to consistent state.
  - System maintains 2 physical blocks per logical block and does the following:
    (1) Write to 1st physical
    (2) When successful, write to 2nd physical
    (3) Declare complete only after second write completes successfully.
  - Systems frequently use NVRAM as one physical to accelerate.