PART 4. STORAGE MANAGEMENT

Chapter 12: Mass Storage Structure

WHAT'S AHEAD:

- Overview of Mass Storage Structure
  - Disk Structure
  - Disk Attachment
  - Disk Scheduling
  - Disk Management
  - Swap-Space Management
  - RAID Structure
  - Stable-Storage Implementation

WE AIM:

- To describe the physical structure of secondary storage devices and its effects on the device usage
- To explain the performance aspects of mass-storage devices
- To evaluate disk scheduling algorithms
- To discuss OS services provided for mass storage, including RAID

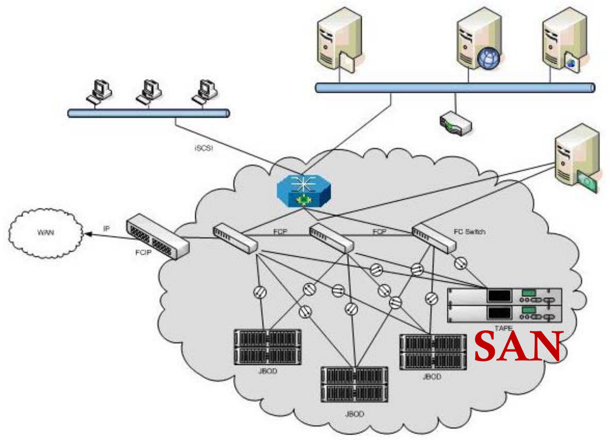
Note: These lecture materials are based on the lecture notes prepared by the authors of the book titled *Operating System Concepts*, 9e (Wiley)

핵심• 요점 이슈와 해결방안

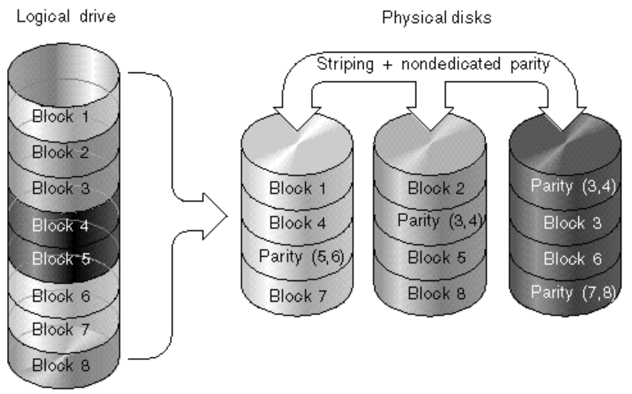
이슈: 어떻게 대량의 디스크 데이터를 효율적• 안정적으로 관리할 것인가

해결방안: 고속 저장장치 버스/네트워크, 효율적인 디스크 스케줄링, RAID

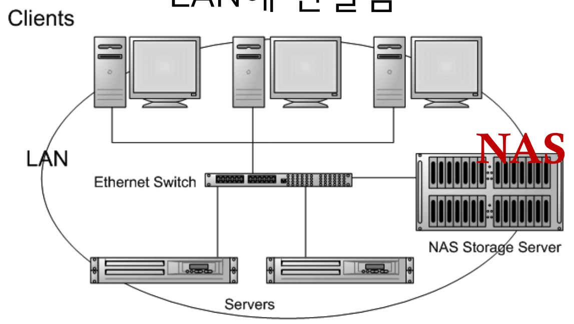
Storage area network (SAN): 저장장치가 별도의 고속 네트워크에 연결됨



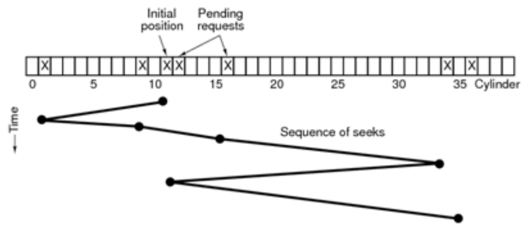
RAID: 하나의 저장장치로 동작하는 다수의 값싼 디스크의 중복 배열

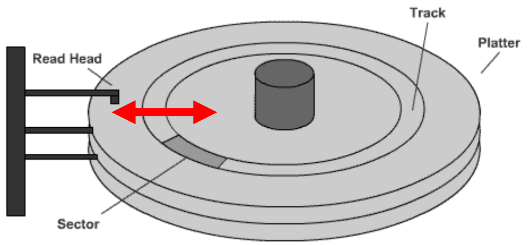


Network attached storage (NAS): 저장장치가 호스트와 함께 LAN에 연결됨



디스크 스케줄링: 디스크 헤드가 트랙 사이를 이동할 때 요청순서를 어떻게 정렬할 것인가



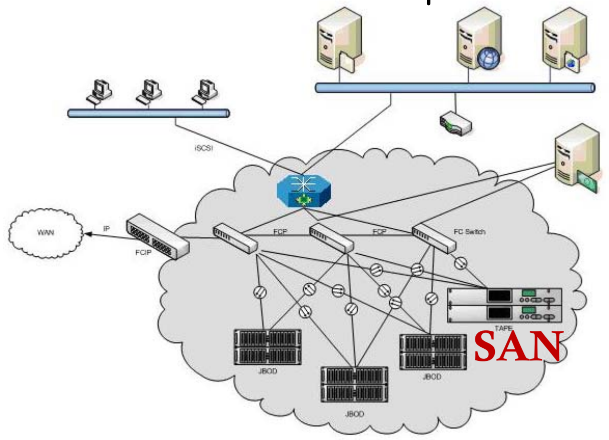


Core Ideas Issues and Approaches

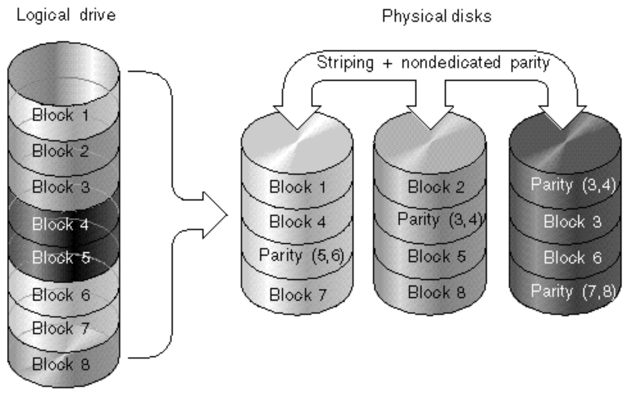
Issues: How to manage massive data on disks *efficiently and reliably*

Approaches: Fast storage bus/networks, Efficient disk scheduling, RAID

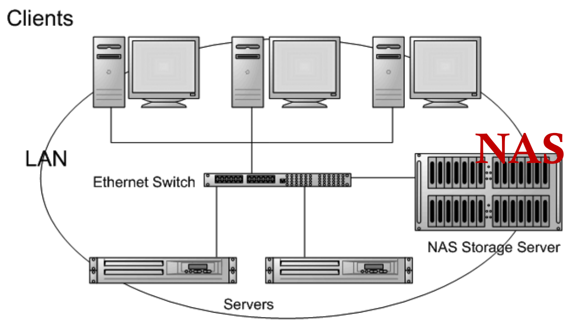
Storage area network (SAN): Storage devices are on a separate fast network



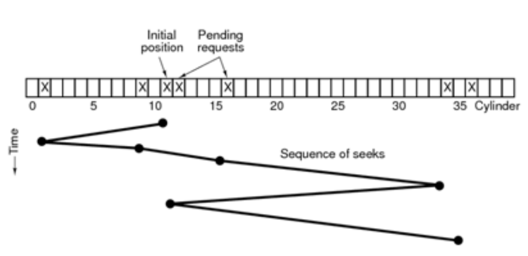
RAID: An array of disks working as single storage device

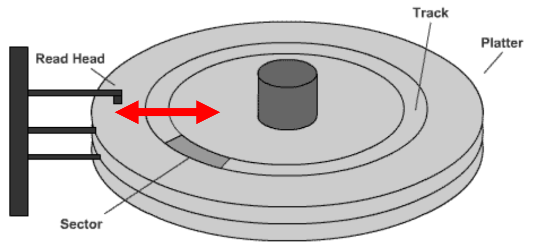


Network attached storage: Storage devices are also attached to LAN



Disk scheduling: In what request order should the disk head move from one track to another?



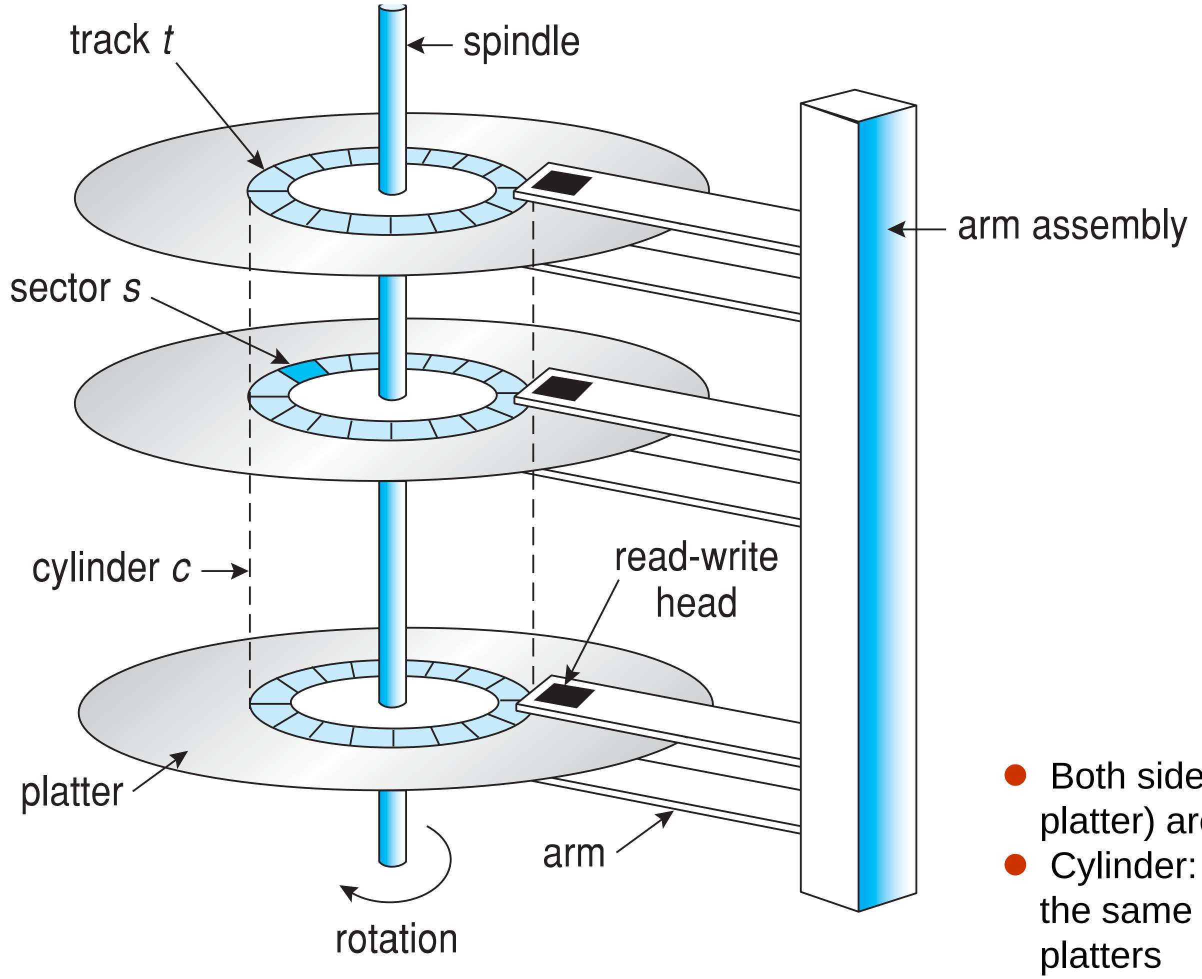


Overview of Mass Storage Structure

- **Magnetic disks** provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 250 times per second, or 5400, 7200, 10000 and 15000 rpm
  - **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!

- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE**, **ATA**, **SATA**, **USB**, **Fibre Channel**, **SCSI**, **SAS**(Serial Attached SCSI), **Firewire**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

Moving-head Disk Mechanism

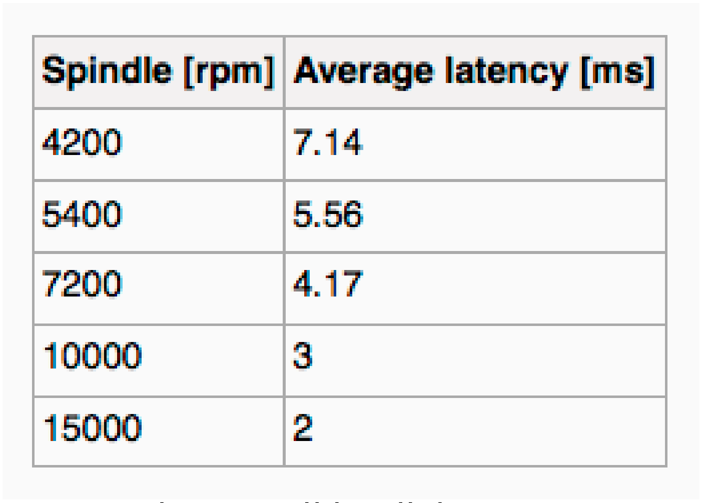


- Both sides of each disk (i.e., platter) are used.
- Cylinder: a set of tracks of the same track number on platters

Magnetic Disks

- Platters range from 0.85" to 14" (historically)
  - Commonly 3.5", 2.5", and 1.8"
- Range from 30GB to 3TB per drive
- Performance
  - Transfer Rate – theoretical – 6 Gb/sec
  - Effective Transfer Rate – real – 1 Gb/sec
  - Seek time from 3ms 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) sec
  - Average latency = ½ latency

At the 2015 Flash Memory Summit in California this past week, Samsung unveiled the largest capacity hard drive in the world. It's a 2.5-inch SSD that can hold 16 *terabytes* of data (15.36TB).
http://petapixel.com/2015/08/15/samsung-16tb-ssd-is-the-worlds-largest-hard-drive/



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

(From Wikipedia)

Magnetic Disk Performance

- *Access Latency* = *Average access time* = average seek time + average rotational 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
- For example, to transfer a 4KB block on a 7200 RPM disk with a 5ms seek time, 1Gb/sec transfer rate with a .1ms controller overhead, average I/O time is estimated to be

  = 5ms + 4.17ms + 4KB / (1Gb/sec) + 0.1ms

  = 9.27ms + 4 / 131072 sec

  = 9.27ms + .03ms = 9.30ms

The First Commercial Disk Drive

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

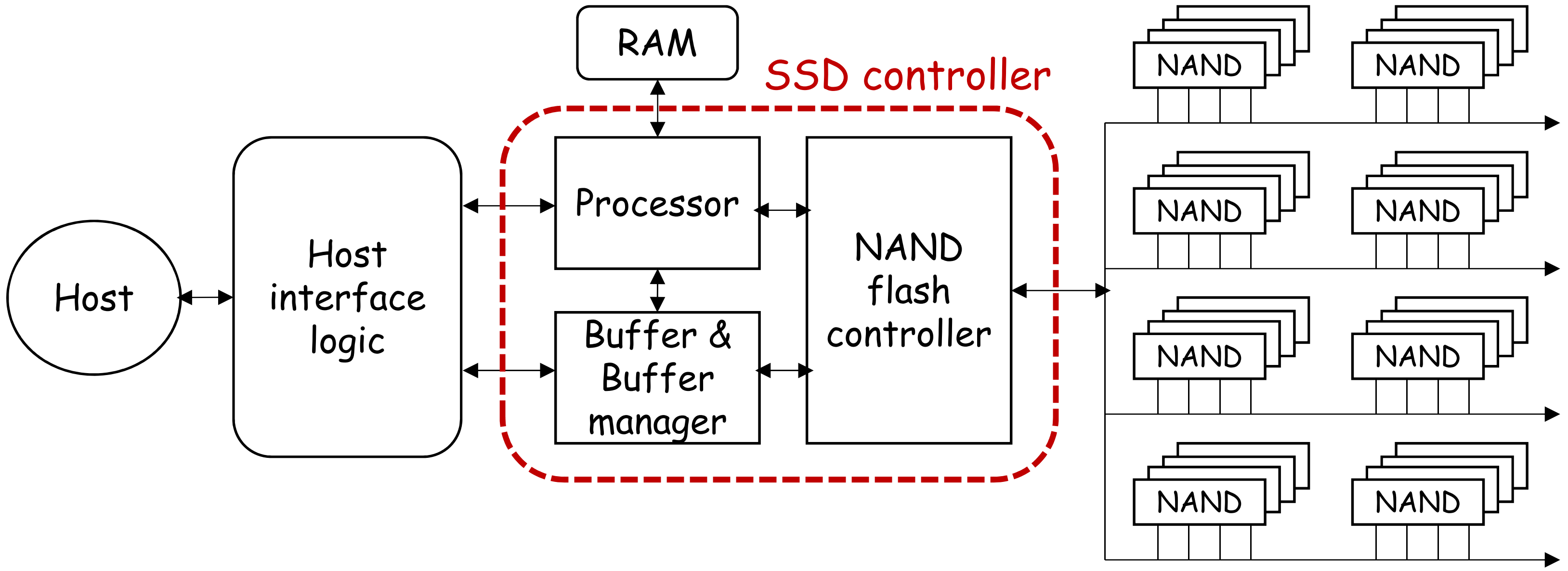
5M (7 bit) characters
50 x 24” platters
Access time = < 1 second

Solid-State Disks (SSDs)

- Nonvolatile memory used like a hard drive
  - Many technology variations, typically based on NAND flash memory
- More expensive per MB, e.g., \$3/GB vs. \$0.15/GB
- Maybe have shorter life span
  - wear-out limit per block (maximum write/erase cycles, e.g., 100,000 times)
- Less capacity, less power consumption, but much faster
- Busses can be too slow

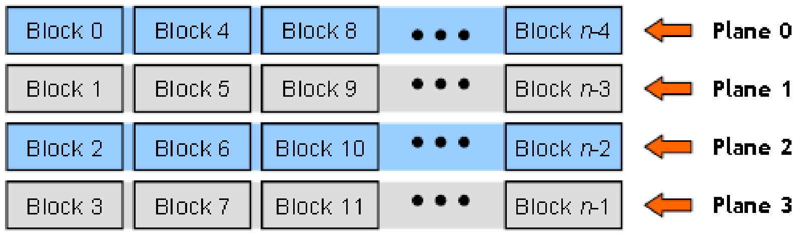
  → connect directly to e.g., PCI, Serial ATA, Fiber Channel
- No moving parts, so no seek time or rotational latency

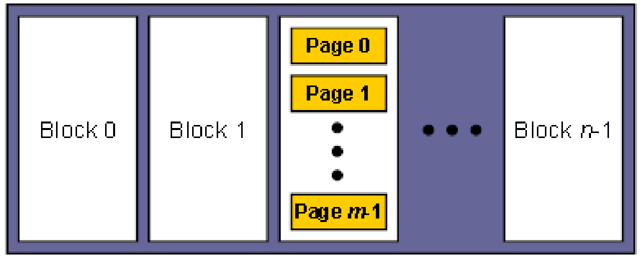
  → Deterministic read performance and more reliable in mobile environments

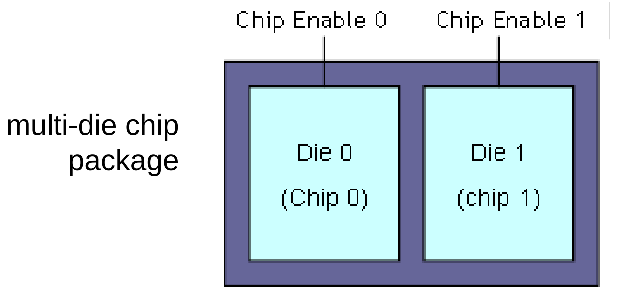


Example of SSD

- Structure
  - 1 die = 4 planes
  - 1 plane = 2048 blocks
  - 1 block = 64 pages
  - 1 page = 4KB
  - Dies can operate independently







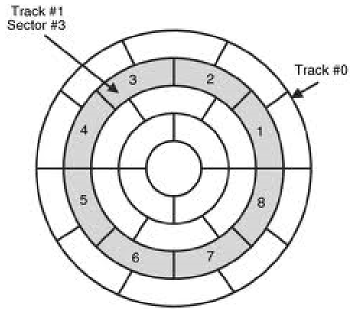
- Operations
  - Reading and programming is performed on a page basis, erasure can only be performed on a block basis
  - Read page (chip #, block #, page #)
    - ~20µs from page to data register
    - 100µs transfer in the serial line
  - Write (program) page (chip #, block #, page #)
    - Page granularity
    - Sequentially within a block
    - Block must be erased before writing
    - ~200µs from register into flash cells
  - Erase block (chip #, block #)
    - ~2 ms
    - Finite number of erase-write cycles
  - Cleaning:
    - Erase out-of-date pages
    - Garbage collection

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 disk
  - 140MB/sec and greater
- 200GB to 1.5TB typical storage
- Common technologies are LTO-{3,4,5} and T10000
  - LTO: Linear Tape-Open, a magnetic tape storage technology
  - T10000: StorageTek cartridge tape

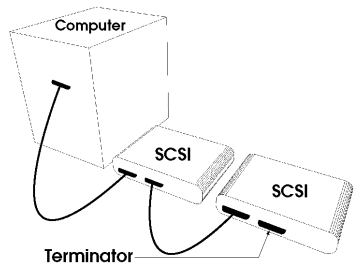
Disk Structure

- 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 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 (Track 0) 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 mapping should be easy
    - Except for bad sectors
    - Non-constant # of sectors per track via constant angular velocity



Disk Attachment

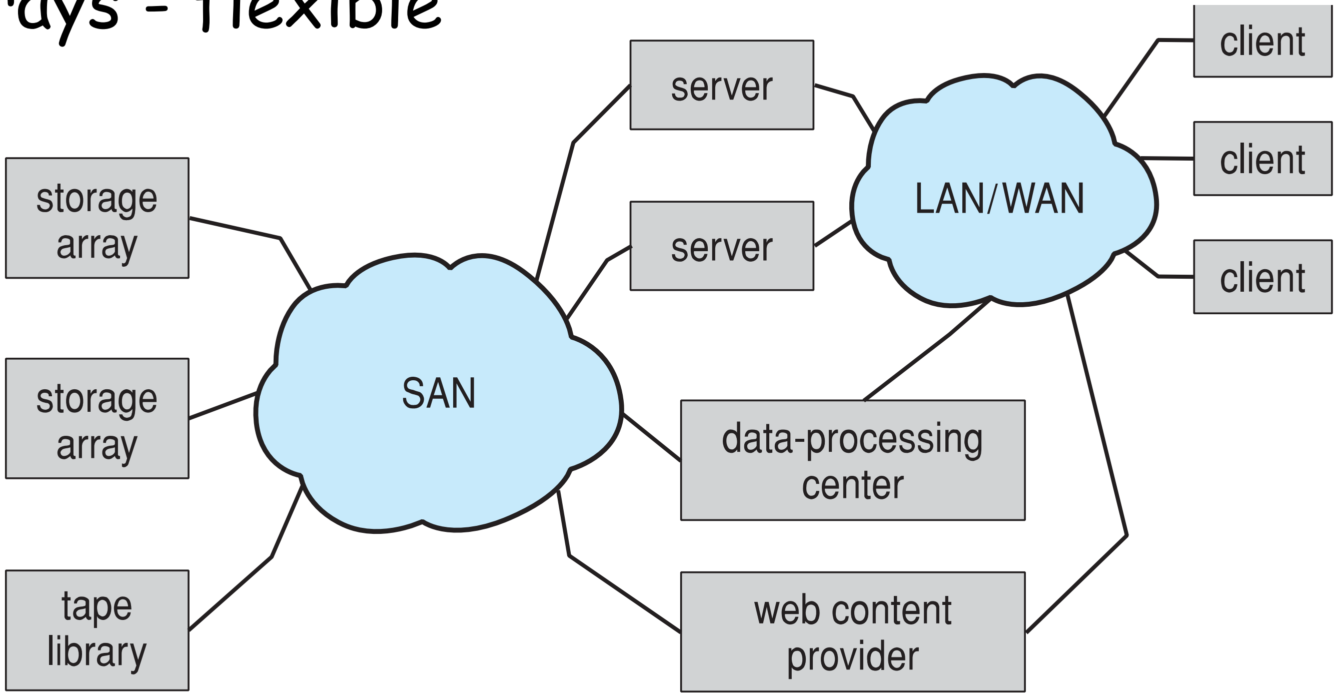
- Host-attached storage accessed through I/O ports talking to I/O busses
- SCSI itself is a bus, up to 16 devices on one cable, **SCSI initiator** (typically, computer) requests operation and **SCSI targets** (typically, storage devices) perform tasks
  - Each target can have up to 8 **logical units** (disks attached to device controller)
  - SCSI initiator and target: Analogous to client and server



- 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)

Storage Area Network

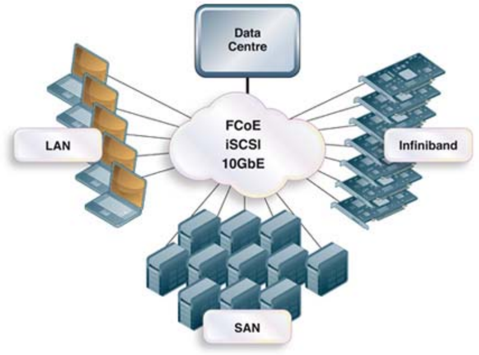
- Common in large storage environments
- Multiple hosts attached to multiple storage arrays - flexible

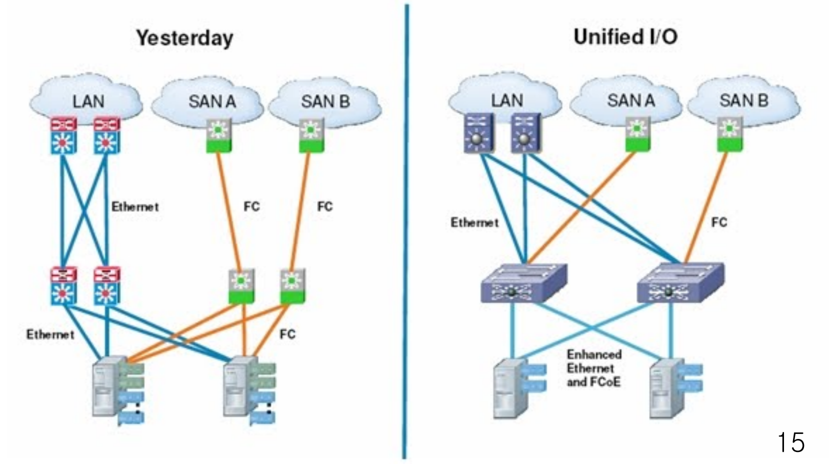


Storage Area Network (Cont.)

- SAN is one or more storage arrays
  - Connected to one or more Fibre Channel switches
- Hosts also attach to the switches
- Storage made available via **LUN masking** from specific arrays to specific servers
  - LUN masking: provide access controls between logical unit numbers (LUNs) and individual servers at the storage controller
- 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? - Flexibility
  - Consider iSCSI, FCoE

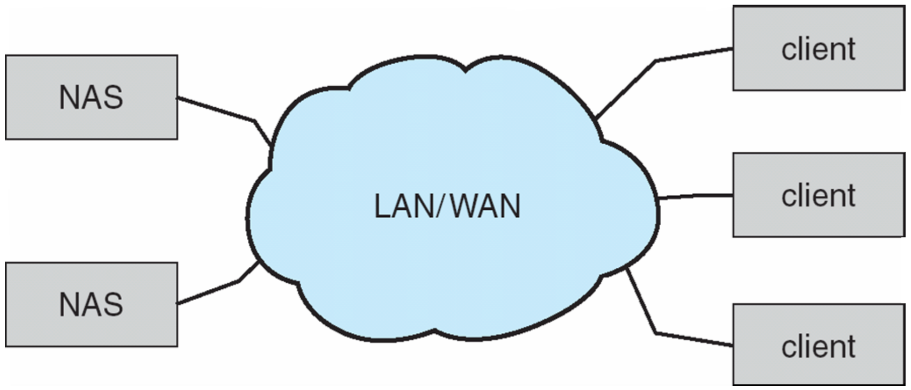
- Fiber channel over Ethernet (FCoE)
  - Encapsulates Fibre Channel frames over Ethernet networks.
  - Allows FC to use 10 Gigabit Ethernet while preserving the FC protocol.
  - Need not run parallel network infrastructures for their LANs and SANs





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)



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 ≈ 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
- There are many sources of disk I/O request
  - OS
  - System processes
  - Users processes

Disk Scheduling (Cont.)

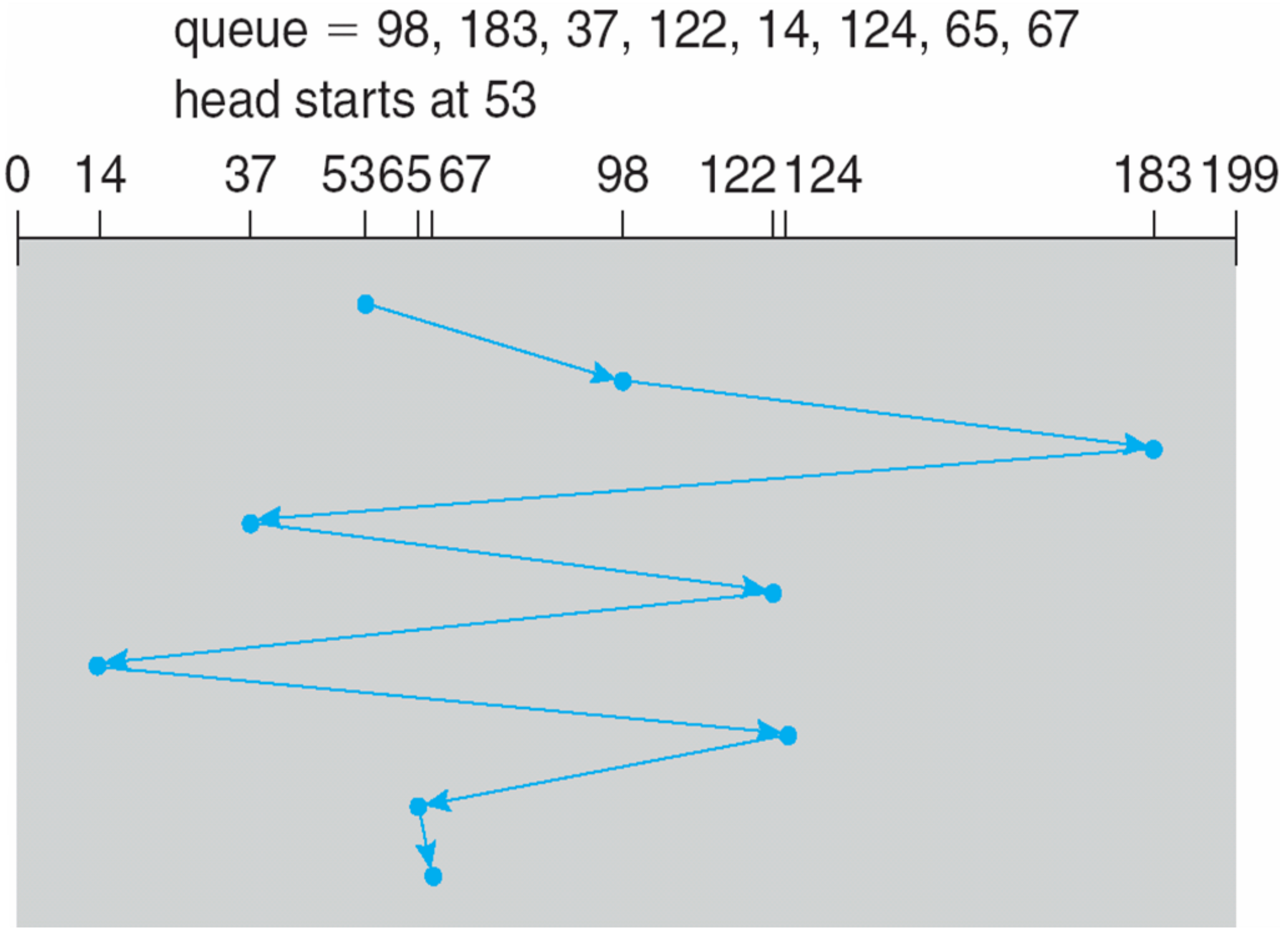
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
  - Optimization algorithms only make sense when a queue exists
- 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 (track number: 0-199)

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

  Head pointer 53

FCFS

- Illustration shows total head movement of 640 cylinders (or tracks)

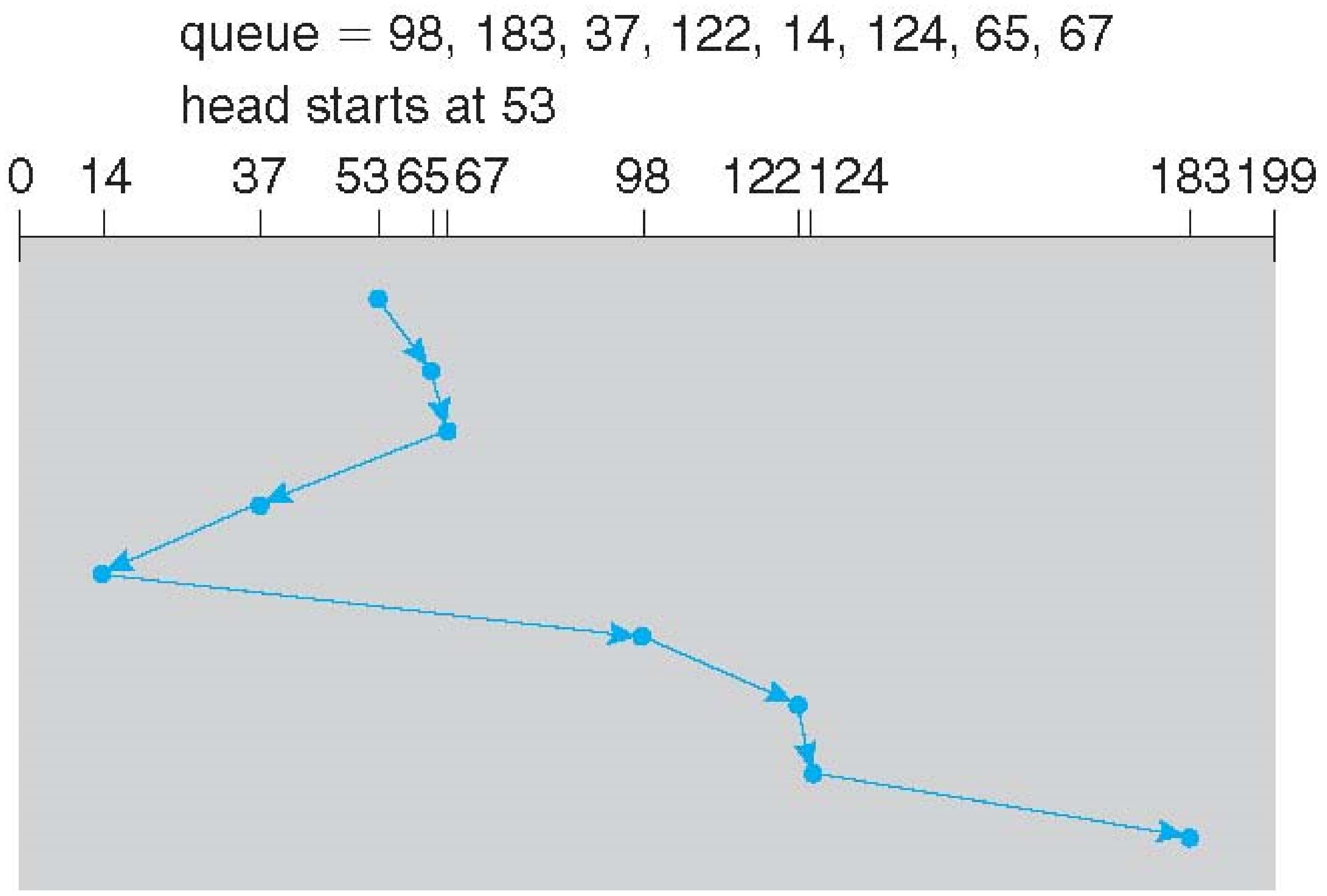


SSTF

- 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
- Illustration shows total head movement of 236 cylinders

SSTF (Cont.)

- Illustration shows total head movement of 236 cylinders

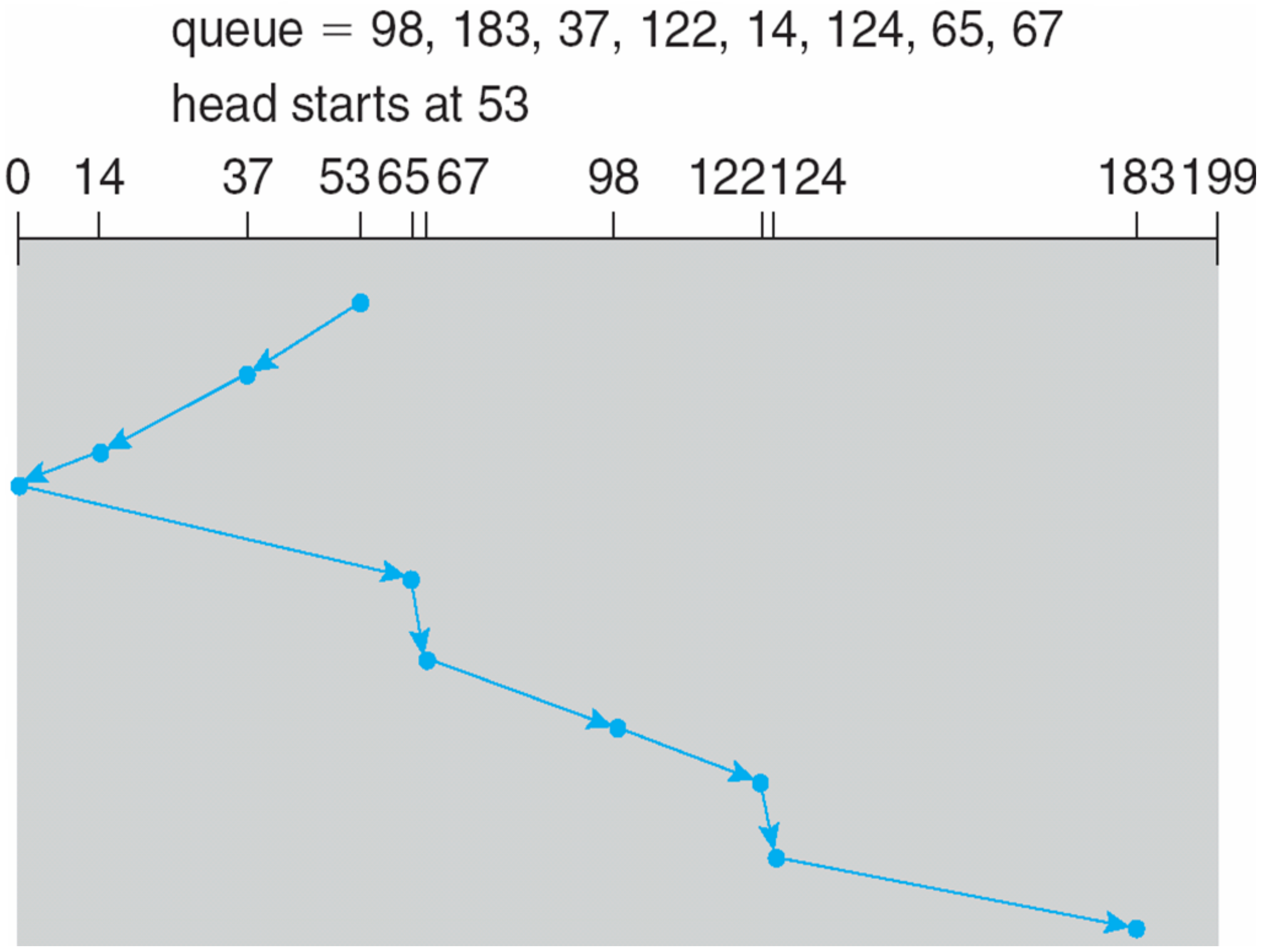


SCAN

- 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**
- Illustration shows total head movement of 208 cylinders
- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

SCAN (Cont.)

- Illustration shows total head movement of 208 cylinders

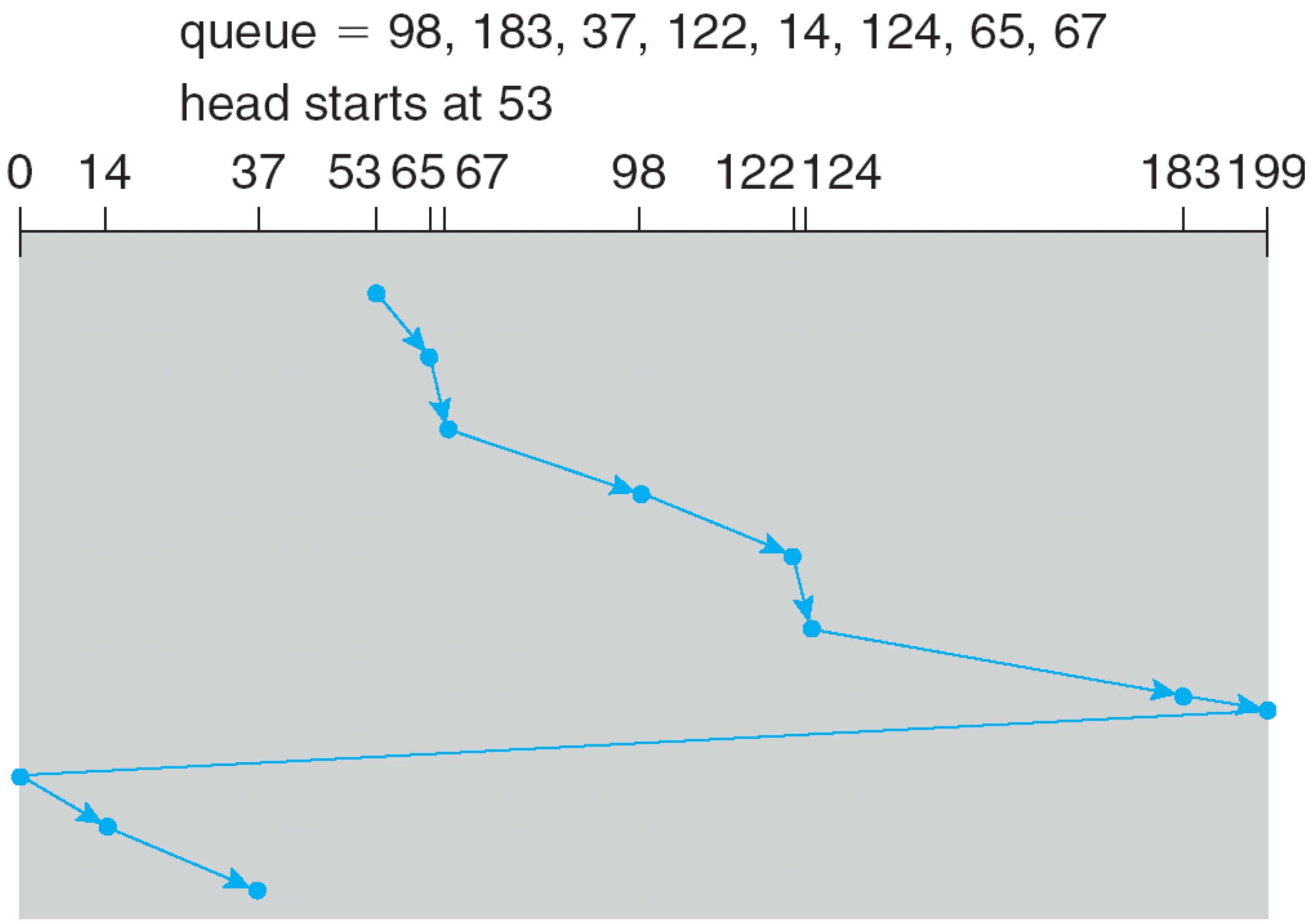


C-SCAN

- Provides a more uniform wait time than SCAN
- The head moves from one end of the disk to the other, servicing requests as it goes
  - When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one
- Total number of cylinders?

C-SCAN (Cont.)

- Illustration shows total head movement of 382 cylinders (or tracks)



C-LOOK

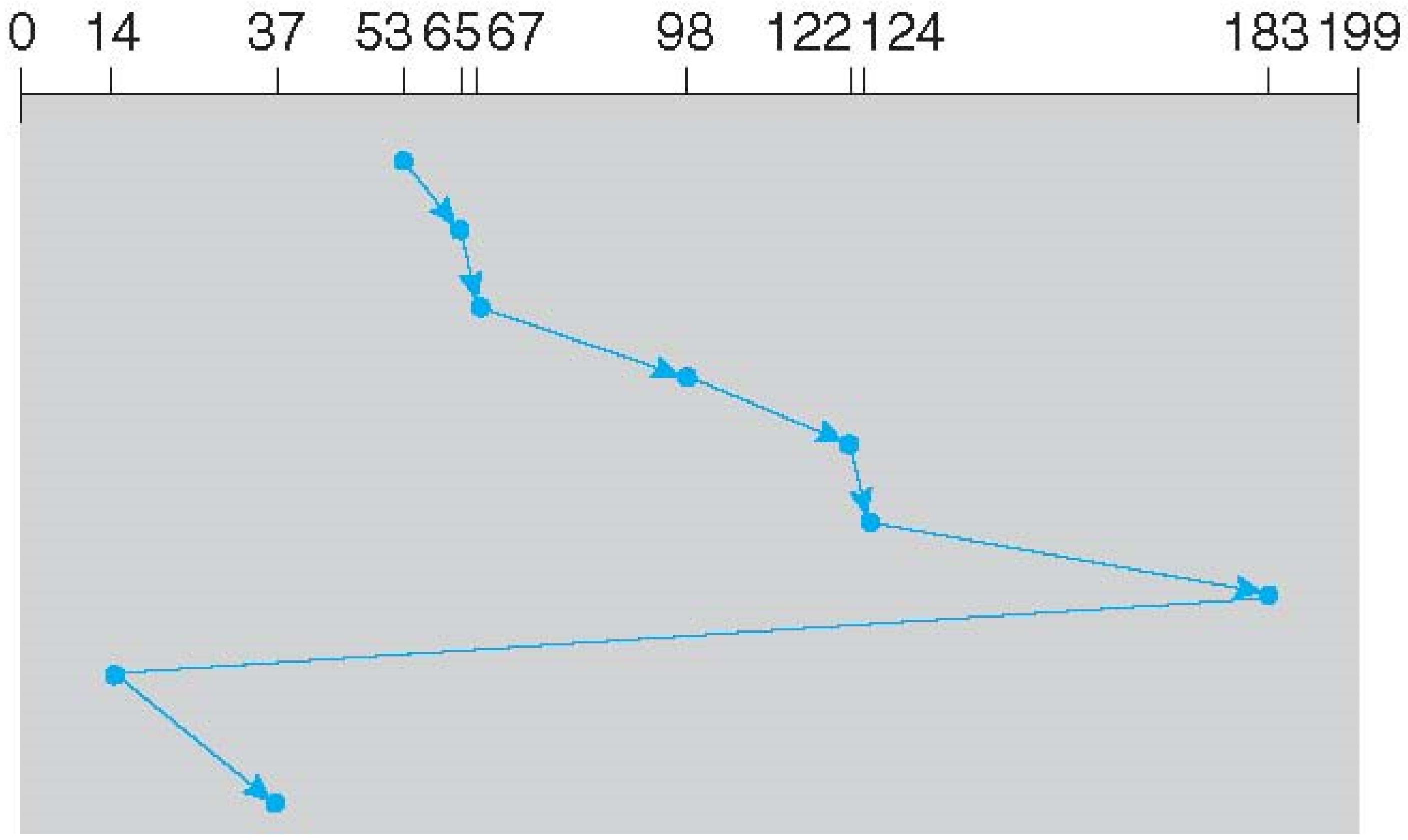
- LOOK a version of SCAN, C-LOOK a version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk
- Total number of cylinders?

C-LOOK (Cont.)

- Illustration shows total head movement of 322 cylinders (or tracks)

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

head starts at 53



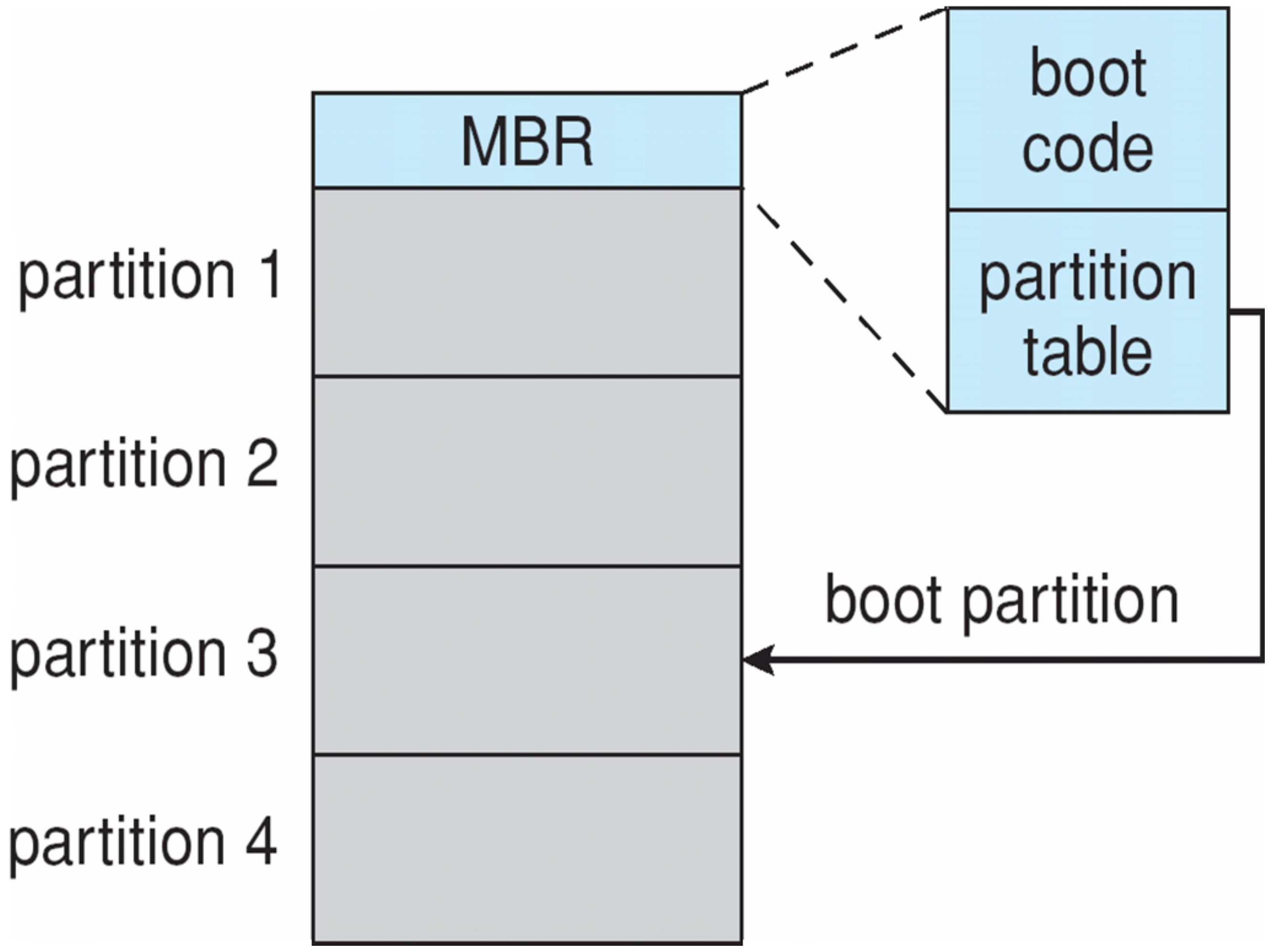
Selecting a Disk-Scheduling Algorithm

- 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
- What about rotational latency?
  - Difficult for OS to calculate
- How does disk-based queuing affect OS queue ordering efforts?

Disk Management

- Low-level formatting, or physical formatting — Dividing 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)
  - Usually 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
  - Partition 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
- Raw disk access for apps that want to do their own block management, keep OS out of the way (databases for example)
- Boot block initializes system
  - The bootstrap is stored in ROM
  - Bootstrap loader program stored in boot blocks of boot partition
- Methods such as sector sparing used to handle bad blocks

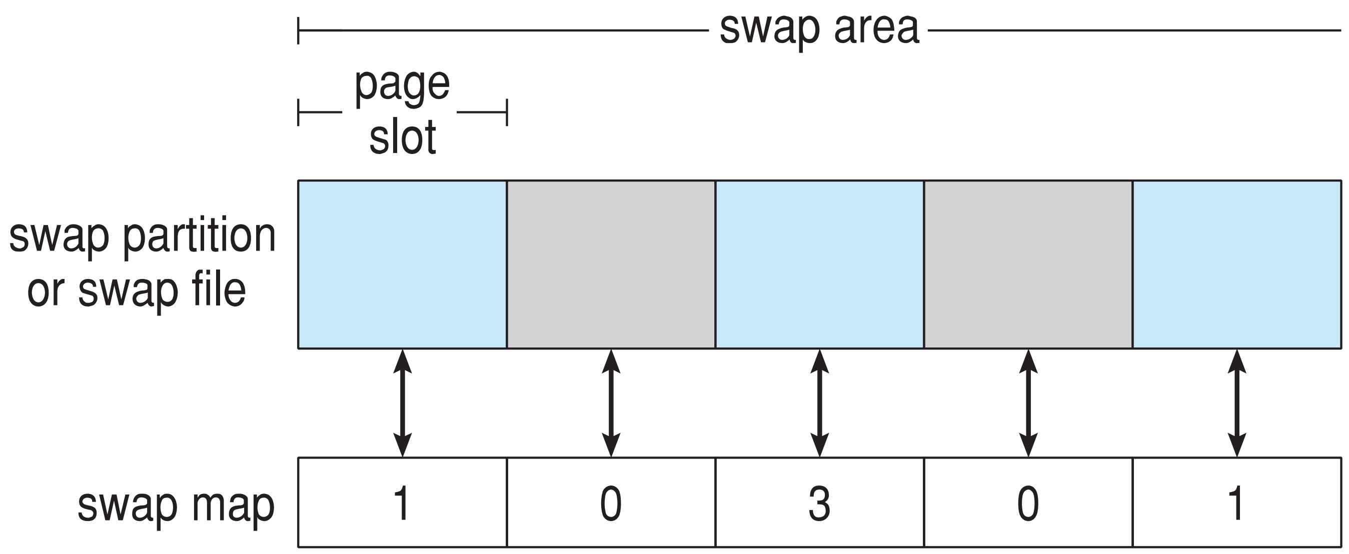
Booting from a Disk in Windows



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)
- Swap-space management
  - 4.3BSD allocates swap space when process starts; 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

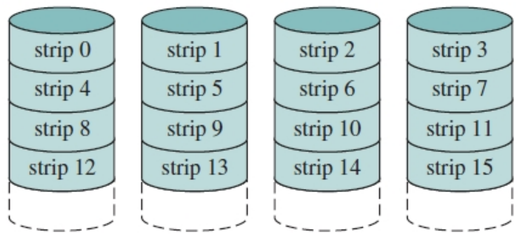
Data Structures for Swapping on Linux Systems



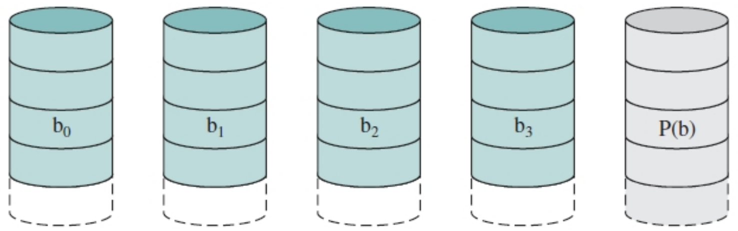
RAID Structure

- RAID – redundant array of inexpensive disks
  - multiple disk drives provides reliability via **redundancy**
- Increases the **mean time to failure** (MTTF)
- RAID is arranged into six different levels
- Disk **striping** uses a group of disks as one storage unit
  - RAID level 0: a disk array with striping at the block level, but without any redundancy, not a true member of the RAID family
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
  - **Mirroring** or **shadowing** (**RAID 1**) keeps duplicate of each disk
  - **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

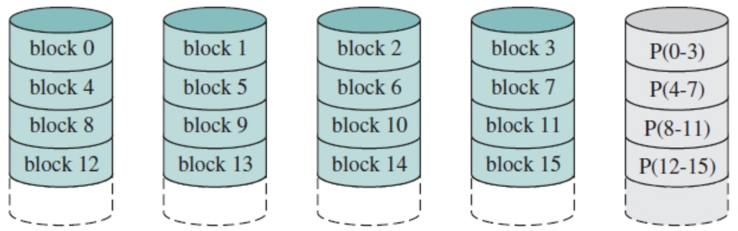
RAID Levels



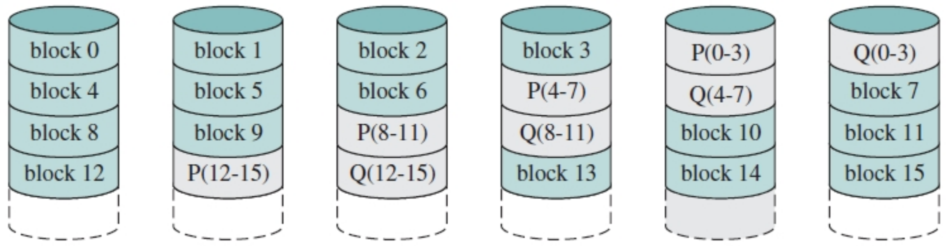
RAID 0 (nonredundant)



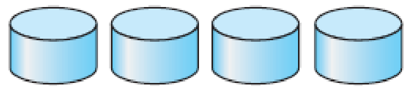
RAID 3 (bit-interleaved parity)



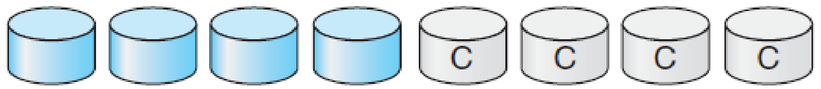
RAID 4 (block-level parity)



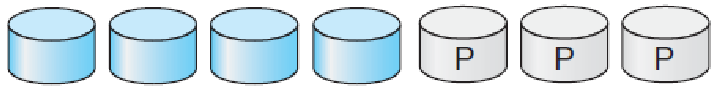
RAID 6 (dual redundancy)



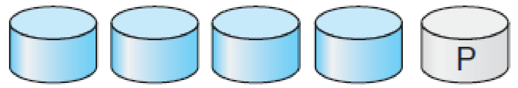
(a) RAID 0: non-redundant striping.



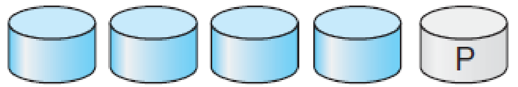
(b) RAID 1: mirrored disks.



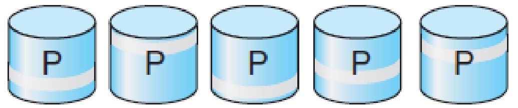
(c) RAID 2: memory-style error-correcting codes.



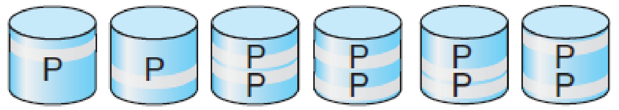
(d) RAID 3: bit-interleaved parity.



(e) RAID 4: block-interleaved parity.



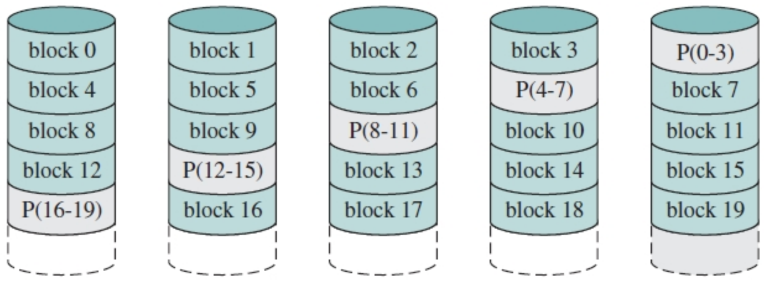
(f) RAID 5: block-interleaved distributed parity.



(g) RAID 6: P + Q redundancy.

Other Features

- Regardless of where RAID is 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)
  - More in Ch 11
- Replication is automatic duplication of writes between separate sites
  - For redundancy and disaster recovery
  - Can be synchronous or asynchronous
- Hot spare disk is unused, automatically used by RAID production if a disk fails to replace the failed disk and rebuild the RAID set if possible
  - Decreases mean time to repair

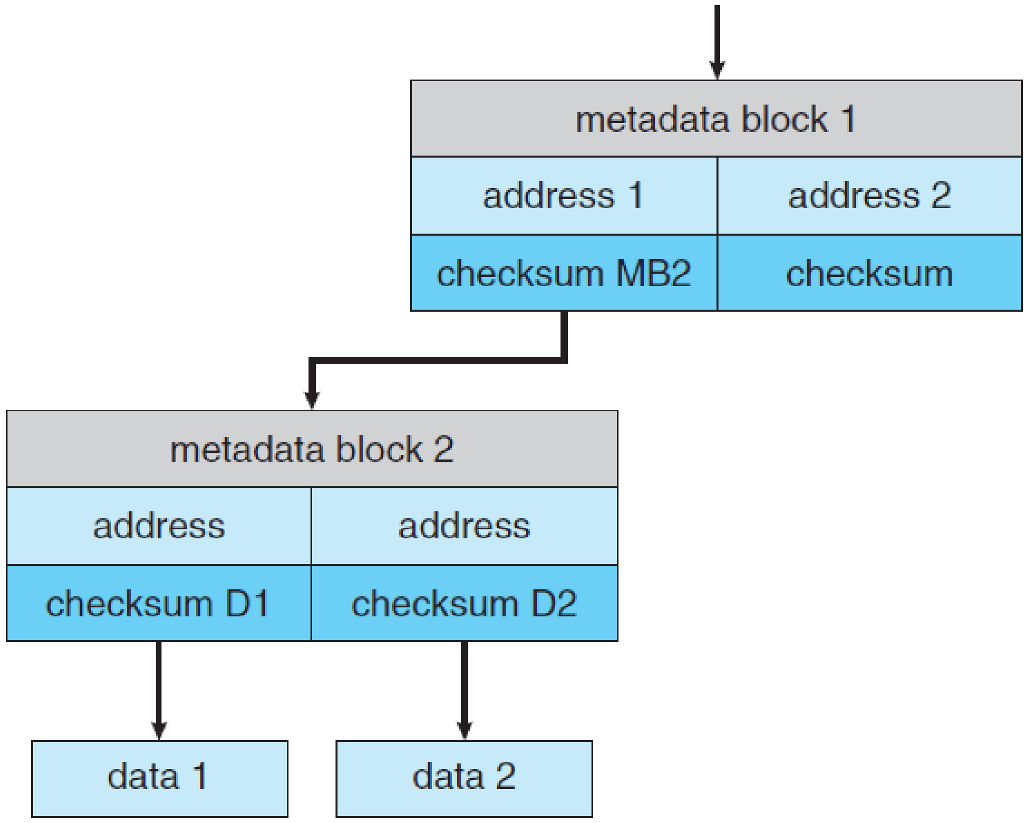


RAID 5 (block-level distributed parity)

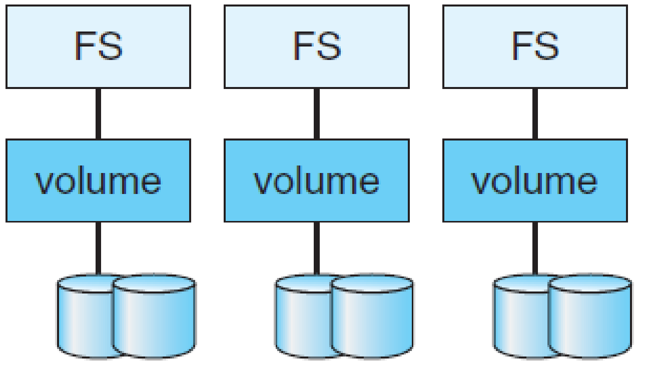
Extensions – Solaris ZFS

- RAID alone does not prevent or detect data corruption or other errors, just disk failures
- Solaris ZFS adds **checksums** of all data and metadata
- Checksums kept with pointer to object, to detect if object is the right one and whether it changed
- Can detect and correct data and metadata corruption
- ZFS also removes volumes, partitions
  - Disks allocated in **pools**
  - Filesystems with a pool share that pool, use and release space like "malloc" and "free" memory allocate / release calls

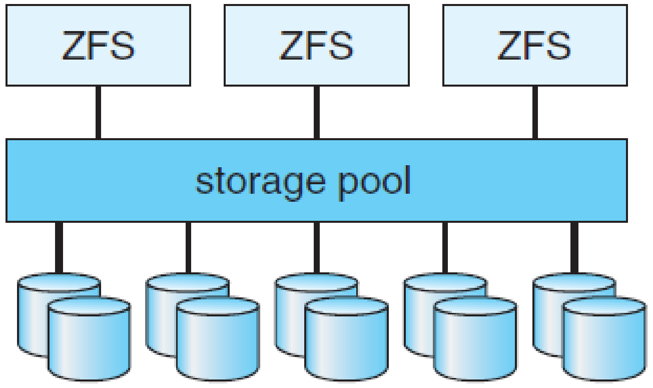
ZFS



ZFS Checksums All Metadata and Data



(a) Traditional volumes and file systems.



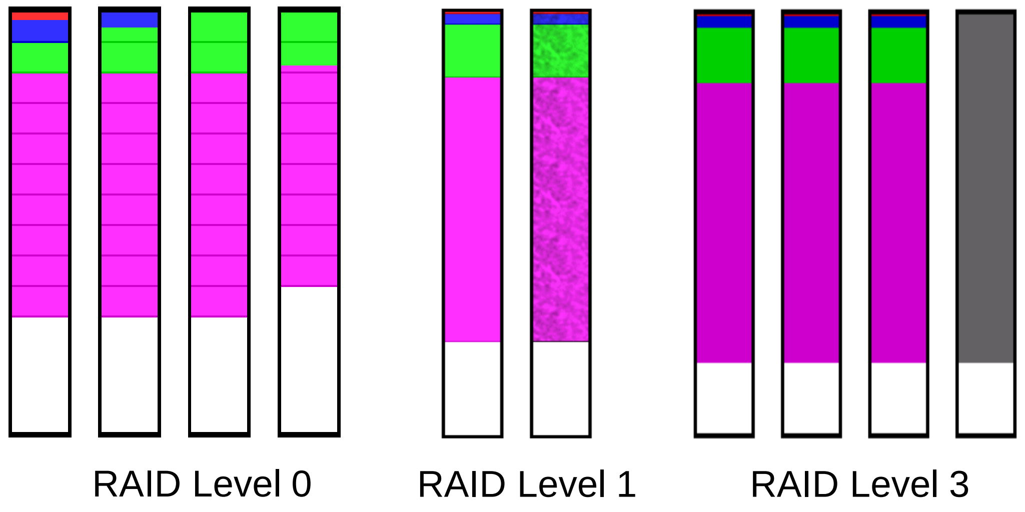
(b) ZFS and pooled storage.

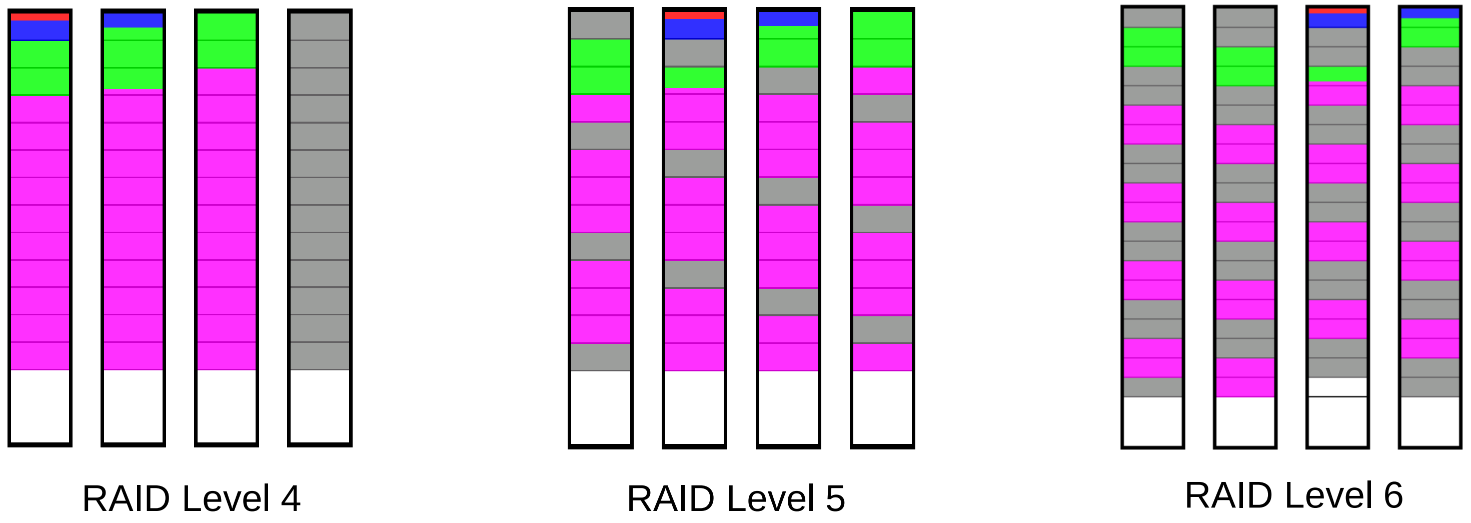
A ZFS pool and file systems.

Note

RAID Configurations - Illustrations

- To show how files of different sizes are distributed on four-disk, 16-KB stripe size RAID arrays.
  - File sizes:
    - 🟥 4 KB (red)
    - 🟦 20 KB (blue)
    - 🟩 100 KB (green)
    - 🟪 500 KB (magenta)
  - Parity block: gray ⬜





http://www.pcguide.com/ref/hdd/perf/raid/levels/single.htm

Note

Category	Level	Description	Required data availability	Large I/O data transfer capacity	Small I/O request rate
Striping	0	Non-redundant	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity; N-way striping of data → N times faster	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Indepen-dent access	4	Block-interleaved parity; A block read accesses only one disk; Parity disk may become a bottleneck	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; Higher than single disk for write; For multiple R/W accesses in parallel → higher rates	Similar to RAID 0 for read; significantly lower than single disk for write ← read-modify-write cycle
	5	Block-interleaved distributed parity	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

Note

The Case of RAID Level 4

- Independent access
  - Each member disk operates independently, so that separate I/O requests can be satisfied in parallel.
  - More suitable for applications that require high I/O request rates, and especially for a multiprocessing system
- Small independent writes
  - Smaller than a block - cannot perform in parallel (See the case of 4-KB file in the preceding slide)
  - Requires that the block be read, modified with the new data, and written back, and that parity block be updated as well → read-modify-write cycle
  - Results in four disk accesses, two reads and two writes
- Larger size reads and writes
  - Performed in parallel → high transfer rates (For writes, data and parity can be written in parallel, too)
- In any case, every write operation must involve the parity disk, which therefore can become a **bottleneck**.

$X4(i) = X3(i) \oplus X2(i) \oplus X1(i) \oplus X0(i)$ {X4(i): parity bit}

After the update, {modified bits: prime symbol}

$$\begin{aligned} X4'(i) &= X3(i) \oplus X2(i) \oplus X1'(i) \oplus X0(i) \\ &= X3(i) \oplus X2(i) \oplus X1'(i) \oplus X0(i) \oplus X1(i) \oplus X1(i) \\ &= X3(i) \oplus X2(i) \oplus X1(i) \oplus X0(i) \oplus X1(i) \oplus X1'(i) \\ &= X4(i) \oplus X1(i) \oplus X1'(i) \end{aligned}$$

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
  1. **Successful completion** - The data were written correctly on disk
  2. **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
  3. **Total failure** - The failure occurred before the disk write started, so the previous data values on the disk remain intact

Stable-Storage Implementation (Cont.)

- 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

Write-ahead logging (WAL): Changes to data files are made only after those changes are completely logged, so that we may ensure the recovery after system crash. Useful for, e.g., DB systems and transaction processing.

Summary

- 대용량 저장장치 개관
  - 자기 디스크의 구성 및 특성, 성능
  - 전송율, 접근시간, 탐색(seek)시간, 회전지연시간
  - 기타 대용량 저장장치: SSD, 자기 테입
- 디스크의 구조
  - cylinder, track, sector, platter
  - 논리 블록 → 물리블록
- 디스크 연결구조
  - SCSI, FC, iSCSI
  - Host-attached, SAN, NAS
- 디스크 스케줄링
  - seek time, 즉 탐색거리의 최소화
  - FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK
  - 디스크 스케줄링 알고리즘의 선택은 여러가지의 요소를 고려해야 함: 디스크 입출력 요청의 유형 및 수, 파일 할당방법, 디렉토리와 소속 파일의 위치관계, 등

- Overview of mass storage structure
  - structure, characteristics, performance of magnetic disk
  - transfer rate, access time, seek time, rotational latency
  - other mass storage devices: SSDs, magnetic tapes
- Disk structure
  - cylinder, track, sector, platter
  - logical block → physical block
- Disk attachment
  - SCSI, FC, iSCSI
  - Host-attached, SAN, NAS
- Disk scheduling
  - to minimize the seek time
  - FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK
  - need to consider various factors when selecting a disk scheduling algorithm: types and number of disk I/O requests, file allocation methods, relative location of directories and associated files, etc.

Summary (Cont.)

- 디스크 관리
  - 파티션, 포맷 수행, 부트 블록, 부트 파티션
  - 문제 블록(bad block)의 관리: 예비 섹터
- 교체공간 관리
  - 가상기억 시스템이나 페이징 시스템에서 성능향상을 위해 디스크에 특별한 공간을 설정
  - 임시 데이터나 코드, 데이터 세그먼트를 저장
- RAID 구조
  - 각 레벨(Level-0 – Level-6)의 구성 및 특성
  - 장점과 단점 – 단점의 보완(ZFS의 경우)
- 안전 저장장치(stable storage)의 구현
  - stable storage: 시스템의 비상정지나 고장 상태에서도 데이터의 저장이 안전하게 이루어지는 저장장치
  - 적절한 데이터의 복제와 갱신절차, 그리고 복구기법을 이용함

- Disk management
  - partitioning, formatting, boot block, boot partition
  - bad block management: spare sectors
- Swap-space management
  - set up a special disk space to enhance the performance in virtual memory or paging system
  - save temporary data and code and data segment
- RAID structure
  - structure and characteristics of RAID levels (Level-0 – Level-6)
  - merits and demerits – how to remedy the demerits (ZFS)
- Stable storage implementation
  - stable storage: a storage system in which data are safely saved without data loss
  - appropriate techniques, such as data replications, update control procedures and recovery methods are used