東南大學
SOUTHEAST UNIVERSITY

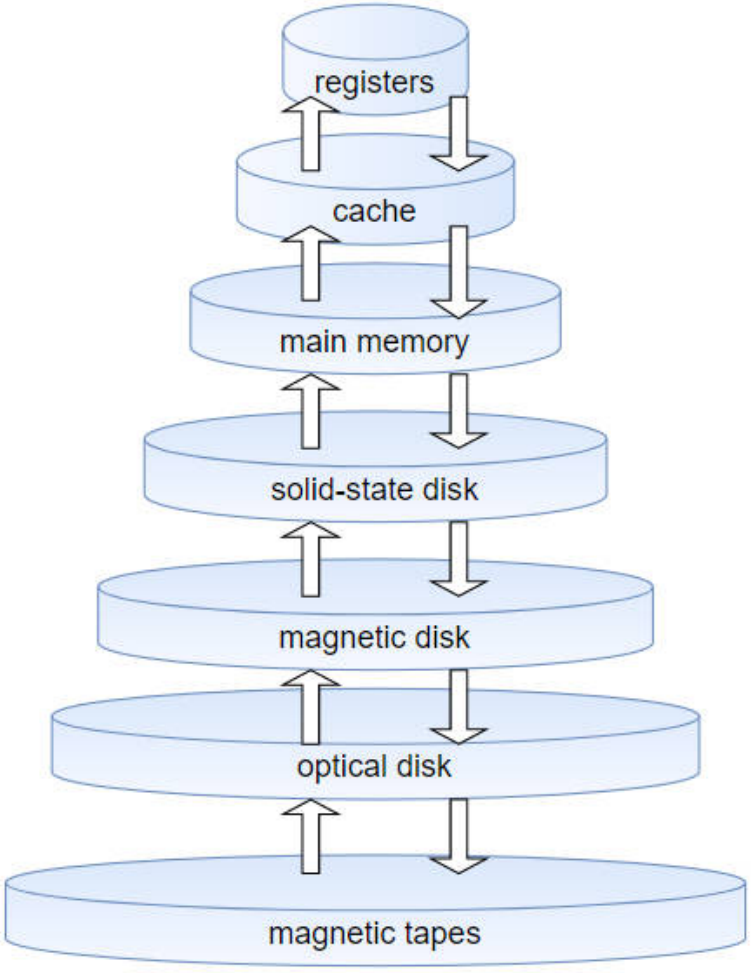
OPERATING SYSTEM CONCEPTS

Chapter 10. Mass-Storage Systems

A/Prof. Kai Dong

Warm-up

Storage Device Hierarchy



Warm-up

Non-Volatile Random-Access Memory

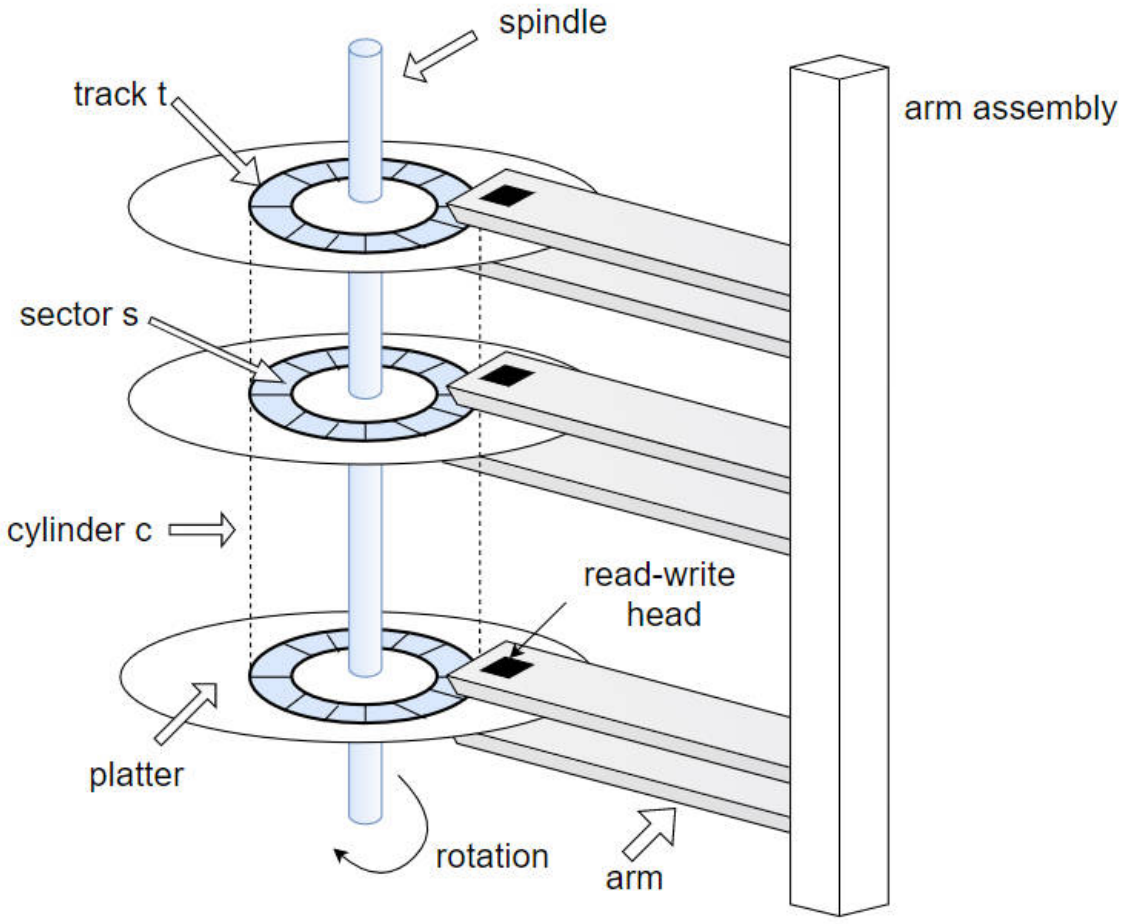
Objectives

- To describe 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

Contents

Disk Structure

Moving-head Disk Mechanism



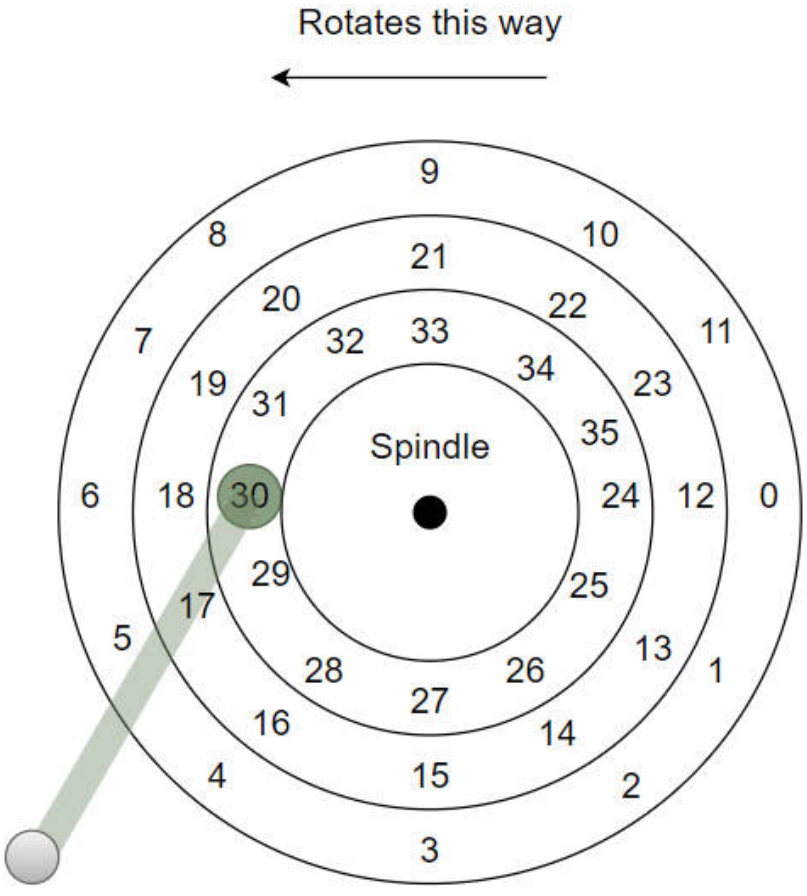
Disk Structure

Disk Interface

- Disk drives are addressed as large 1-dimensional arrays of (512-byte) 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
- Multi-sector operations are possible; indeed, many file systems will read or write 4KB at a time (or more)
- A single 512-byte write is atomic (i.e., it will either complete in its entirety or it won't complete at all

Disk Structure

A Simple Disk Drive



Disk Structure

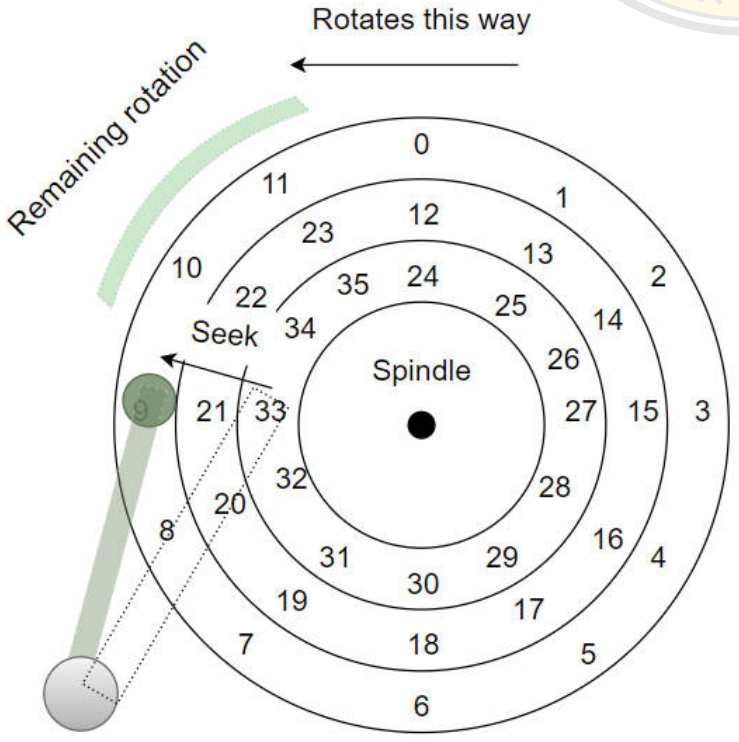
Disk I/O

- The complete picture of I/O time:
- First a **seek**, then a remaining **rotation**, and finally the **transfer**.

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

- The rate of I/O is often more easily used for comparison between drives.

$$R_{I/O} = \frac{Size_{Transfer}}{T_{I/O}}$$



Disk Structure

Seek Time

- Phases in a seek:
  1. First an acceleration phase as the disk arm gets moving;
  2. Then coasting as the arm is moving at full speed;
  3. Then deceleration as the arm slows down;
  4. Finally settling as the head is positioned over the correct track.
     - » The settling time is often quite significant, e.g., 0.5 to 2 ms, as the drive must be certain to find the right track.
- **Seek Time**
  - Time of moving the disk arm to the correct track.
  - Average disk-seek time is roughly one-third of the full seek time.

Disk Structure

Rotational Delay

- **Rotational delay**
  - Time of waiting for the desired sector to rotate under the disk head.
  - Average rotational delay is half of the full rotational delay.
- Rotations and seeks (**NOT** transfer) are the most costly disk operations.

Disk Structure

Workload Assumptions

- Assume two workloads
  - Random workload, issues small (e.g., 4KB) reads to random locations on the disk.
  - Sequential workload, simply reads a large number of sectors consecutively from the disk, without jumping around.

Disk Structure

Comparing Disks

- A couple of modern disks from *Seagate*
  - The *Barracuda*^{TM} *ST4000DM004*
  - The *IronWolf*^{TM} *ST4000VN008*

	Barracuda	*IronWolf*
Price (@JingDong)	CNY 509	CNY 839
Capacity	4 TB	4 TB
RPM	5400	5900
Average Seek	15 ms	9 ms
Max Transfer	190 MB/s	180 MB/s
Cache	256 MB	64 MB

Disk Structure

Computing $T_{I/O}$ and $R_{I/O}$

	Barracuda	IronWolf
RPM	5400	5900
Average Seek	15 ms	9 ms
Max Transfer	190 MB/s	180 MB/s

- Random workload

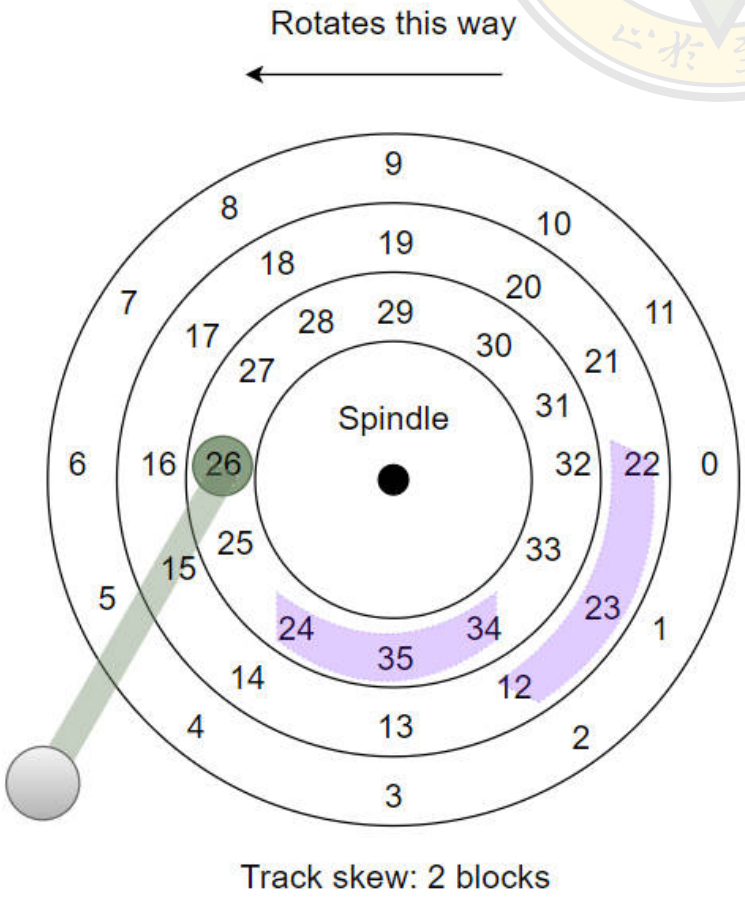
	computation	Barracuda	IronWolf
T_{seek}	$= Average\ Seek$	$15\ ms$	$9\ ms$
$T_{rotation}$	$= 0.5\ R/(RPM/60)$	$5.56\ ms$	$5.08\ ms$
$T_{transfer}$	$= 4\ KB/Max\ Transfer$	$0.02\ ms$	$0.02\ ms$
$T_{I/O}$	$= T_{seek} + T_{rotation} + T_{transfer}$	$20.58\ ms$	$14.10\ ms$
$R_{I/O}$	$= 4\ KB/T_{I/O}$	$0.19\ MB/s$	$0.28\ MB/s$

- Sequential workload, can you compute $T_{I/O}$ and $R_{I/O}$?
- There is a huge gap in drive performance between random and sequential workloads.

Disk Structure

Track skew

- **Track skew** — to make sure that sequential reads can be properly serviced even when crossing track boundaries.



Track skew: 2 blocks

Disk Structure

Some Other Details

- **Multi-zoned** disk drives — Outer tracks have more sectors than inner tracks. The disk is organized into multiple zones, and where a zone is consecutive set of tracks on a surface. Each zone has the same number of sectors per track, and outer zones have more sectors than inner zones.
  - constant angular velocity (CAV) and constant linear velocity (CLV)
- **Cache** or track buffer — Some small amount of memory (usually around 64 or 256 MB) which the drive can use to hold data read from or written to the disk.

Contents

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

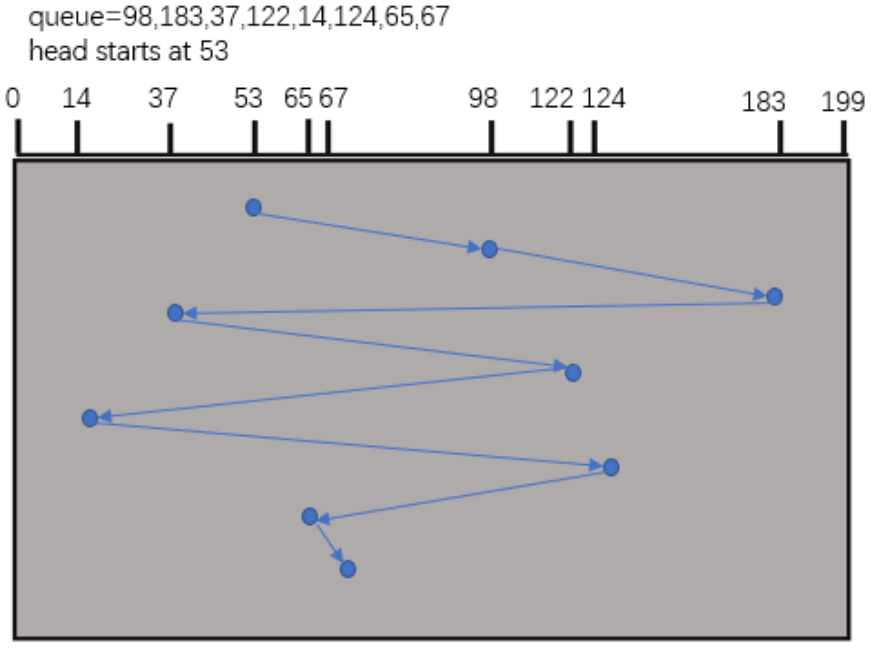
Disk Scheduling

- 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 work must queue
  - Optimization algorithms only make sense when a queue exists
- We illustrate scheduling algorithms with a request queue (0-199):
  98, 183, 37, 122, 14, 124, 65, 67, with head pointer at 53

Disk Scheduling

First Come First Serve (FCFS)

- **First Come First Serve (FCFS)**

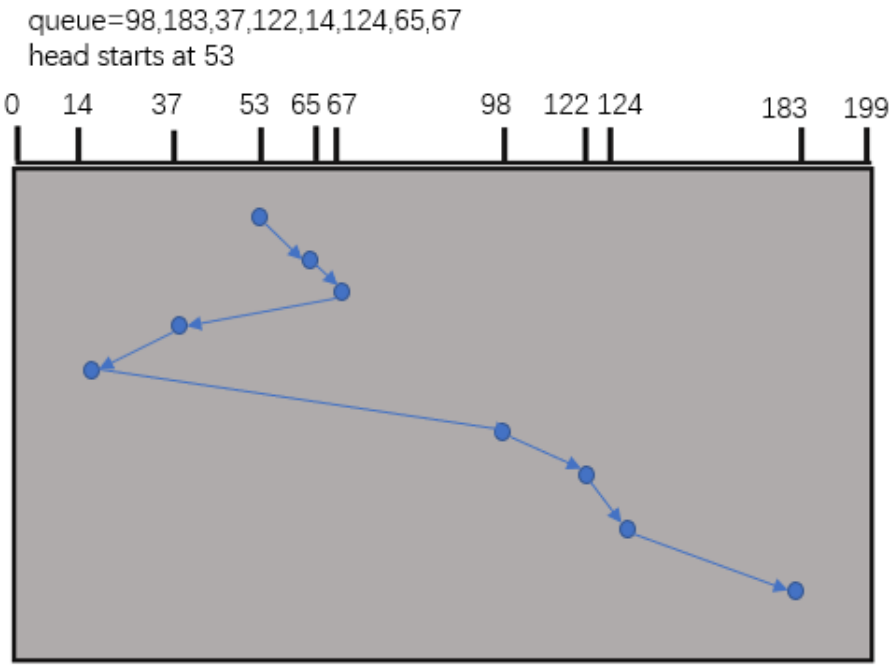


- Illustration shows total head movement of 640 cylinders.

Disk Scheduling

Shortest Seek Time First (SSTF)

- **Shortest Seek Time First** (**SSTF**)
  - Selects the request with the minimum seek time from the current head position
  - A form of SJF scheduling; may cause starvation of some requests

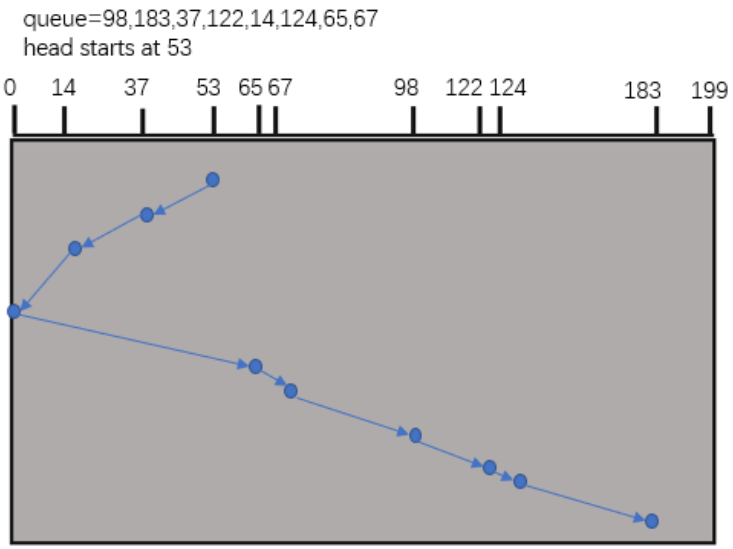


- Illustration shows total head movement of 236 cylinders.

Disk Scheduling

SCAN

- **SCAN algorithm** (a.k.a., the **elevator** algorithm)
  - The head 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.
  - we need to know the direction of head movement.
    - » Assuming that the disk arm is moving toward 0.

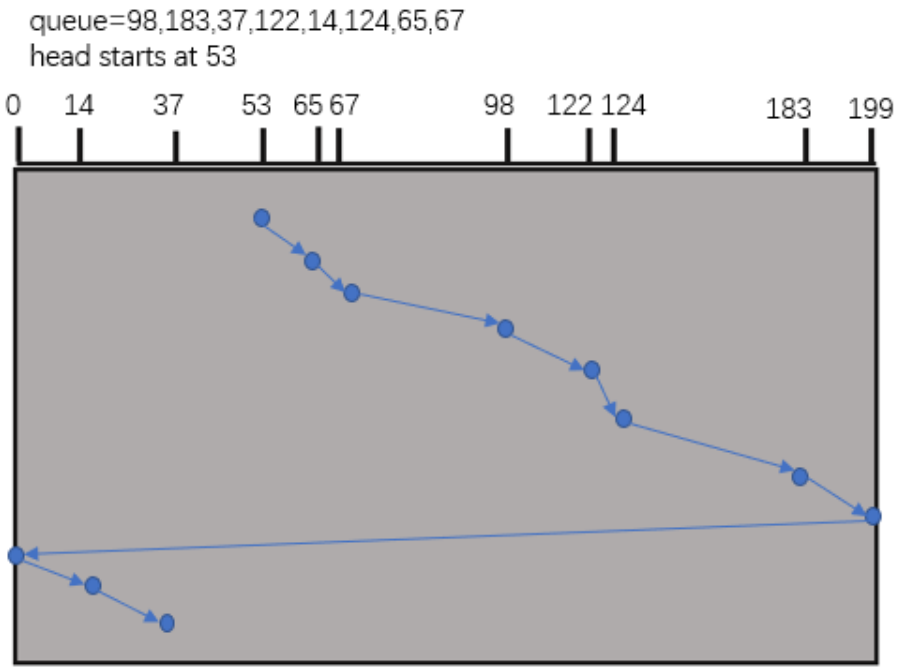


- Illustration shows total head movement of 236 cylinders.

Disk Scheduling

C-SCAN

- **C-SCAN algorithm** 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.

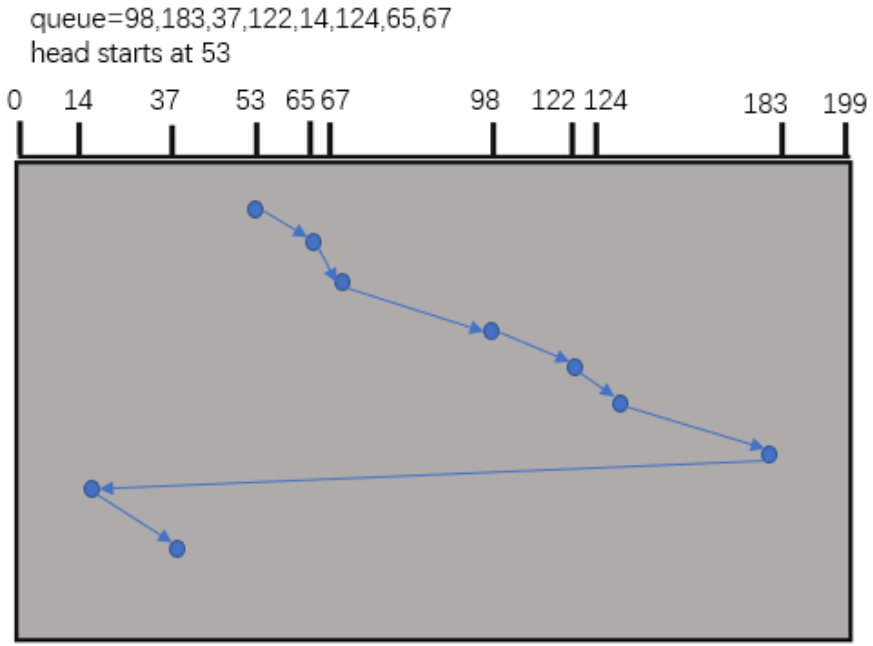


- Illustration shows total head movement of 382 cylinders.

Disk Scheduling

LOOK and C-LOOK

- **LOOK** is a version of SCAN, **C-LOOK** is a version of C-SCAN
  - The head 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.



- Illustration shows total head movement of 322 cylinders.

Disk Scheduling

Shortest Positioning Time First (SPTF)

- Remaining problem: SCANs (or LOOKs) do not represent the best scheduling technology (they ignore rotation).
- **Shortest Positioning Time First** (**SPTF**) — Usually performed inside a drive.

Disk Scheduling

In Class Exercise

Suppose that a disk drive has 200 cylinders, numbered 0 to 199. The drive is currently serving a request at cylinder 123, and the previous request was at cylinder 175. The queue of pending requests, in FIFO order, is 86, 147, 180, 28, 95, 151, 12, 77, 30. Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms respectively?

1. FCFS
2. SSTF
3. C-SCAN
4. LOOK

Disk Scheduling

Selecting a Disk-Scheduling Algorithm

- SSTF (or SPTF) 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
- Where is disk scheduling performed?
  - In older systems, the OS did all the scheduling.
  - In modern systems, the OS scheduler performs I/O merging; the disk then uses its internal knowledge of head position and detailed track layout information to service said requests in the best possible (SPTF) order.
- The OS has other constraints on the service order for requests.

Contents

RAID Structure

- **Redundant Arrays of Independent Disks** (**RAIDs**)
- A technique to use multiple disks in concert to build a faster, bigger, and more reliable disk system.
  - Externally, a RAID looks like a disk: a group of blocks one can read or write.
    - This transparency improves deploy ability of RAID.
  - Internally, a RAID consists of multiple disks, memory (both volatile and non-volatile), and one or more processors to manage the system.
    - RAIDs offer various advantages.

RAID Structure

Why Using RAID instead of Single Disk?

- **Performance**: Using multiple disks in parallel can greatly speed up I/O times.
- **Capacity**: Large data sets demand large disks
- **Reliability**: With some form of redundancy, RAIDs can tolerate the loss of a disk and keep operating as if nothing were wrong

RAID Structure

RAID Level 0: Striping

- **RAID Level 0** — Stripe blocks across the disks

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

- is actually **NOT** a RAID level at all, in that there is no redundancy.
- serves as an excellent upper-bound on performance and capacity and thus is worth understanding.

RAID Structure

RAID Level 0: Striping (contd.)

- A variation

Disk 0	Disk 1	Disk 2	Disk 3
0	2	4	6
1	3	5	7
8	10	12	14
9	11	13	15

- Chunk size mostly affects performance of the array.
  - A small chunk size implies that many files will get striped across many disks, thus increasing the parallelism of reads and writes to a single file.
  - However, the positioning time to access blocks across multiple disks increases, because the positioning time for the entire request is determined by the maximum of the positioning times of the requests across all drives.
  - Vice versa.

RAID Structure

RAID Level 0: Striping (contd.)

- RAID-0 Analysis
- Capacity:
  - Perfect, given N disks each of size B blocks, striping delivers $N \times B$ blocks of useful capacity.
- Reliability:
  - Perfect but in the bad way, any disk failure will lead to data loss.
- Performance:
  - Excellent, all disks are utilized, often in parallel, to service user I/O requests.
  - More in detail

RAID Structure

RAID Level 0: Striping (contd.)

- Performance
- Single-request latency
  - The latency of a single-block request should be just about identical to that of a single disk, since RAID-0 will simply redirect that request to one of its disks.
- Steady-state throughput: We will assume that a disk can transfer data at *S MB/s* under a sequential workload, and *R MB/s* when under a random workload (*S* >> *R*)
  - Sequential
    - » $N \times S$ *MB/s*
  - Random
    - » $N \times R$ *MB/s*

RAID Structure

RAID Level 1: Mirroring

- **RAID Level 1** — Makes more than one copy of each block in the system; each copy should be placed on a separate disk, to tolerate disk failures.

Disk 0	Disk 1
0	0
1	1
2	2
3	3

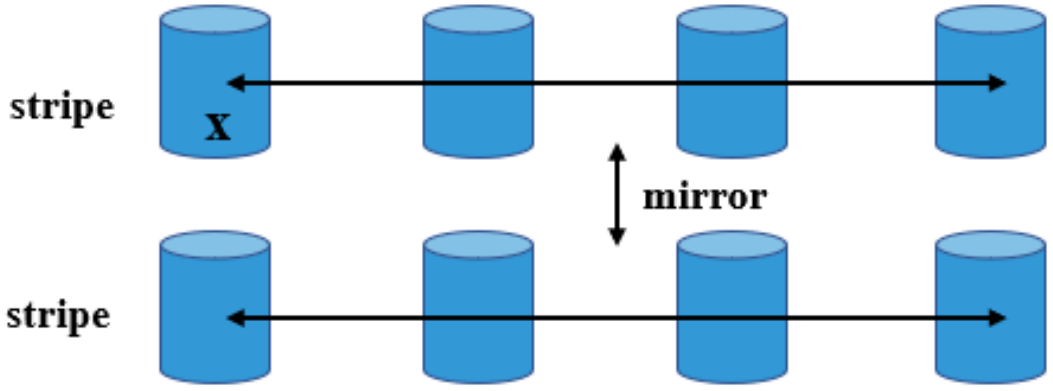
- One common arrangement is **RAID-10** or (RAID 1+0) because it uses mirrored pairs (RAID-1) and then stripes (RAID-0) on top of them.

Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

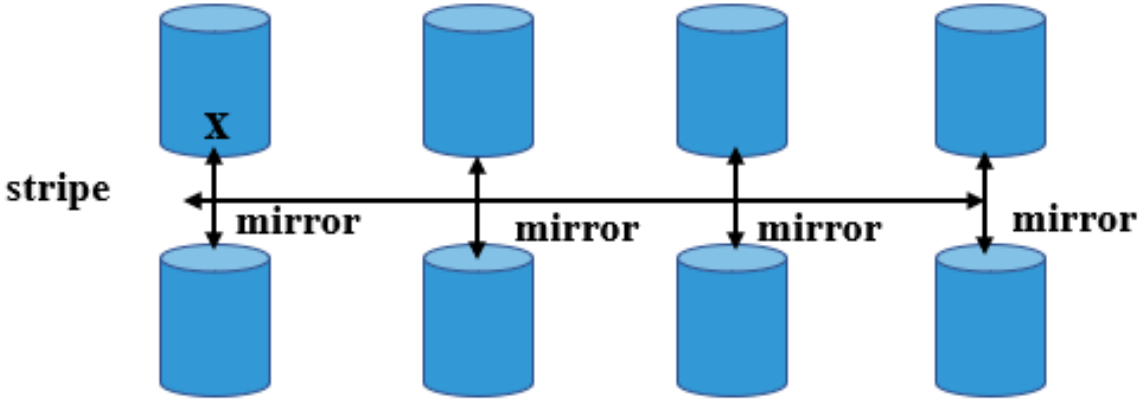
- You can imagine another arrangement, RAID-01.

RAID Structure

RAID Level 1: Mirroring (contd.)



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



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

RAID Structure

RAID Level 1: Mirroring (contd.)

- RAID-1 Analysis
- Capacity:
  - Expensive, with the mirroring level = 2, we only obtain half of our peak useful capacity.
- Reliability:
  - Good, it can tolerate the failure of any one disk, and maybe more, up to $N/2$ disk failure.

RAID Structure

RAID Level 1: Mirroring (contd.)

- Performance:
- Single-request latency
  - Read: The same as the latency on a single disk.
  - Write: Two writes in parallel. Thus suffers the worst-case seek and rotational delay of the two.
- Steady-state throughput
  - Sequential
    - $N \times S/2$ *MB/s*, for both writes and reads.
  - Random
    - $N \times R$ *MB/s* for reads, and $N \times R/2$ *MB/s* for writes.

RAID Structure

RAID Level 4: Saving Space with Parity

- **RAID Level 4** — For each stripe of data, add a single parity block that stores the redundant information for that stripe of blocks.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

- Parity is computed from the stripe of blocks by using a mathematical function XOR

RAID Structure

RAID Level 4: Saving Space with Parity (contd.)

- RAID-4 Analysis
- Capacity:
  - $N - 1$, since 1 disk is used for parity information.
- Reliability:
  - Tolerate 1 disk failure and no more.

RAID Structure

RAID Level 4: Saving Space with Parity (contd.)

- Performance:
- Single-request latency
  - Read — The same as the latency on a single disk.
  - Write — Twice as the latency on a single disk.
- Steady-state throughput
  - Sequential
    - $(N-1) \times S$ *MB/s*, for both reads and writes.
  - Random
    - $(N-1) \times R$ *MB/s* for reads, $R/2$ *MB/s* for writes.

RAID Structure

RAID Level 4: Saving Space with Parity (contd.)

- Full-stripe writes are the most efficient way for RAID-4 to write to disk.
- For random writes
  - Additive parity: To compute the value of the new parity block, read in all of the other data blocks in the stripe in parallel.
  - **Subtractive parity**: use only the modified block.

$$Parity_{new} = Data_{old} \oplus Data_{new} \oplus Parity_{old}$$

- Using the subtractive method, for each write, the RAID has to perform 4 physical I/Os (two reads from *old* and two writes to *new*).

RAID Structure

RAID Level 4: Saving Space with Parity (contd.)

- Now imagine there are lots of writes submitted to the RAID; how many can RAID-4 perform in parallel?

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4 *new*	5	6	7	P1 *new*
8	9	10	11	P2
12	13 *new*	14	15	P3 *new*

- Small-write problem: Even though small-writes to disk 0 and 1 can be performed in parallel, the problem that arises is with the parity disk.

RAID Structure

RAID Level 5: Rotating Parity

- **RAID Level 5** — Rotate the parity block across drives

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

RAID Structure

RAID Level 5: Rotating Parity (contd.)

- RAID-5 Analysis
- Capacity:
  - $N - 1$, since 1 disk is used for parity information.
- Reliability:
  - Tolerate 1 disk failure and no more.
- The same as RAID-4

RAID Structure

RAID Level 5: Rotating Parity (contd.)

- Performance:
- Single-request latency
  - Read — The same as the latency on a single disk.
  - Write — Twice as the latency on a single disk.
- Steady-state throughput
  - Sequential
    - $(N-1) \times S$ *MB/s*, for both reads and writes.
  - Random
    - $N \times R$ *MB/s* for reads
    - $(N/4) \times R$ *MB/s* for writes.

RAID Structure

RAID Level 5: Rotating Parity (contd.)

- Random read performance is a little better, because we can utilize all of the disks. ($N \times R$ *MB/s*)
- Random write performance improves noticeably over RAID-4, as it allows for parallelism across requests. ($(N/4) \times R$ *MB/s*)
  - Imagine a write to block 1 and a write to block 10; this will turn into requests to disk 1 and disk 4 (for block 1 and its parity) and requests to disk 0 and disk 2 (for block 10 and its parity). Thus, they can proceed in parallel.
  - In fact, we can generally assume that given a large number of random requests, we will be able to keep all the disks about evenly busy.

RAID Structure

RAID Analysis

	RAID-0	RAID-1	RAID-4	RAID-5
Capacity	$N \times B$	$N \times B/2$	$(N-1) \times B$	$(N-1) \times B$
Reliability	0	1 (for sure) $N/2$ (if lucky)	1	1
Throughput				
Sequential Read	$N \times S$	$(N/2) \times S$	$(N-1) \times S$	$(N-1) \times S$
Sequential Write	$N \times S$	$(N/2) \times S$	$(N-1) \times S$	$(N-1) \times S$
Random Read	$N \times R$	$N \times R$	$(N-1) \times R$	$N \times R$
Random Write	$N \times R$	$(N/2) \times R$	$(1/2) \times R$	$(N/4) \times R$
Latency				
Read	T	T	T	T
Write	T	T	2T	2T