CS 342302 Operating Systems

Fall Semester 2021

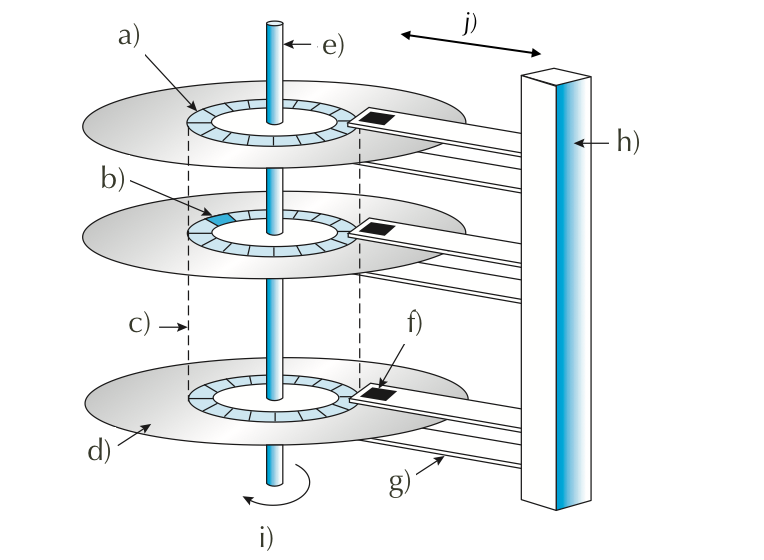
Prof. Pai H. Chou

Weekly Review 12

Scope: Chapter 11, Mass-Storage Systems

## 1. Definitions and Short Answers

1. Name the following parts of a disk (a - h) and the motions (i - j).



A: a) track b) sector c) cylinder – Set of tracks at the same radius

d) platter e) rotator or spindle f) read-write head

g) arm h) arm-assembly

i) rotation j) seek

1. When accessing data on a magnetic disk,
   1. What is the **rotational latency**?

A: It is defined as the time required to rotate to a desired sector.

* 1. What is the **seek time**?

A: Time to move disk arm to a desired cylinder.

* 1. What is the **positioning time**?

A: Consider the equation: Positioning time = Seek time + Rotational latency

* 1. What is another word for the positioning time?

A: Random-access time.

* 1. What are the two components of positioning time in magnetic disk access?

A: Seek time and rotational latency.

1. What is the difference between a cylinder and a track?

A: A single track is a concentric ring on a given platter while a cylinder is the set of tracks on all platters at a given arm position (radius).

1. For flash memory terminology, assume NAND flash,
   1. What is the minimum unit of reading?

A: A page.

* 1. What is the minimum unit of writing?

A: A page.

* 1. What is a block?

A: It is the minimum erasable unit. Contains several pages. Could be 8,16,32, etc.

* 1. What happens during an erase?

A: During an erase we set or reset the desired bits to 1. On the other hand, during a write bits are set to 0.

* 1. If you must modify one byte, what are the steps involved?

A: If a single byte is to be modified,

1. What is **wear-leveling**, and why is it important for flash memory?

A: The hardware controller must be aware to how often it is writing to a particular physical block. Wear-leveling avoids writing excessively to these physical “hot” blocks/pages and thus avoids rendering the device unusable. Example – Directory information – Used to find the rest of the data.

1. What is the primary action that an OS can schedule to improve the performance of a hard disk drive?

A: The OS can minimize the seek time (seek distance) to improve the performance of the hard disk drive.

1. Of the different disk scheduling algorithms,
   1. is FIFO in general a good policy for HDD? for SSD?

A: No, it is not a good policy for HDD in general. However, it is the primary algorithm used with SSD’s.

* 1. is STSF in general a good policy for HDD? What kind of problem does it have? What about for SSD?

A: It may cause starvation in some requests because it tends to favor middle cylinders over inner and outermost ones. It is a reasonable choice for HDD. FCFS is the better choice for SSD.

* 1. What is the difference between SCAN and C-SCAN?

A: The main difference is that in C-SCAN requests are served only in one direction. That is, cylinders are treated as a circular list that wraps around from the last cylinder to the first one. This achieves a more uniform wait time than SCAN.

* 1. What is the difference between SCAN and LOOK?

A: LOOK is an optimized version of SCAN where the disk arm only goes as far as the last request in each direction, not the end of the disk.

* 1. Why does SCAN have more predictable behavior than LOOK?

A: Because SCAN moves the disk arm uniformly regardless of the extremes in the queue. In LOOK, the extremes of the queue may change and thus it is harder to predict its behavior.

1. If NVM scheduling does not need to consider rotation or seek time, what does it need to consider?

A: Since writing is so much slower in NVM scheduling, one must plan and minimize the amount of writing to be done and avoid erase latency. Also, garbage collection and wear leveling are required. Write amplification must be avoided – That is one write causes more writes to happen.

1. What is a **spare** sector?

A: A spare sector is a sector that is set aside by the low-level formatter to replace defective sectors over time. Usually hidden but the OS has access to them.

1. What is **sector-slipping**?

A: During sector slipping, spare sectors are allocated such that the spare preserves contiguity when the remaining sectors are shifted.

~~a~~bcdwxs -> ~~a~~abcdwx where the spare sector s was used!

1. Can a regular file system be used for swap space? What are the advantages and disadvantages?

A: Yes, a regular file system can be used for swap space. It may be easier for the user to set up, but it is NOT as efficient for the OS (More on this on later chapters).

1. What is a **swap partition**, and why is it a good idea?

A: A swap partition or raw partition is a partition dedicated to performing swapping. It is a good idea because a swap-space storage manager is used to allocate and deallocate blocks from the raw partition and thus may use dedicated algorithms to optimize speed rather than storage efficiency. Recall that that the swap-space is used much more frequently than the file system.

1. How do you pronounce SCSI?

A: It is pronounced as “skuzzy”.

1. What does **RAID** stand for?

A: Redundant Array of Independent Disks. Originally “I” stood for *Inexpensive.*

1. How can RAID achieve higher reliability?

A: Having the redundancy provided by RAID, if a disk controller fails, chances are that other disk controllers won’t be affected and thus independence is achieved, and RAID can provide reliability.

1. How can RAID achieve higher disk performance? In what metric?

A: Through striping or data parallelism. Split your data in various disks to achieve data parallelism and improve performance.

1. What is the meaning of the following about disks?
   1. **mean time to failure**

A: Average time expected for a disk to fail.

* 1. **mean time to repair**, and is it related to mean time to failure?

A: Average time to discover and repair or replace or restore the failed disk.

* 1. **mean time to data loss**

A: Average time to lose data assuming failure of disk and its copy.

1. What is the meaning of?
   1. **mirroring**?

A: It is the simplest approach to introducing redundancy and consist in duplicating every drive. Logical disks consist of two physical drives and every write is carried on both drives.

* 1. **data striping**? Is it related to mirroring?

A: Data striping splits data onto multiple disks such that it can be accessed in parallel.

* 1. What is the difference between **bit-level** and **block-level** striping? which is more common?

A: Bit-level striping accesses different bits on different disks while block-level striping access different blocks on different disks.

* 1. What is the meaning of **striped mirror**? **mirrored stripes**? Which one is a better choice?

A: Striped Mirror – RAID 1+0 -> First do striping and for each split part make a mirror. Allows for all parts of data to be mirrored.

(Better choice).

Mirrored Striped – RAID 0+1 -> Make an entire mirror of the striped system.

## 2. Programming Exercise

## 3. Disk Scheduling Algorithms [25 points]

You are to implement the disk (seek) scheduling algorithms covered in Chapter 11.

Use the following template ([download](https://drive.google.com/file/d/1R5EUj-fHPWaGPdlTFmtO1Jwyn-bOTnJc/view?usp=sharing) and rename as disk.py):

class DiskScheduler:

\_POLICIES = ['FCFS', 'SSTF', 'SCAN', 'LOOK', 'C-SCAN', 'C-LOOK']

def \_\_init\_\_(self, nCylinders):

self.nCylinders = nCylinders

def schedule(self, initPos, requestQueue, policy, direction):

'''

request is the list of cylinders to access

policy is one of the strings in \_POLICIES.

direction is 'up' or 'down' and applies to (C-)SCAN/LOOK only.

returns the list for the order of cylinders to access.

'''

if policy == 'FCFS':

# return the disk schedule for FCFS

if policy == 'SSTF':

# compute and return the schedule for shortest seek time first

if policy in ['SCAN', 'C-SCAN', 'LOOK', 'C-LOOK']:

# sequentially one direction to one end (up or down),

# then sequentially in opposite direction.

# compute and return the schedule accordingly.

def totalSeeks(initPos, queue):

lastPos = initPos

totalMoves = 0

for p in queue:

totalMoves += abs(p - lastPos)

lastPos = p

return totalMoves

if \_\_name\_\_ == '\_\_main\_\_':

def TestPolicy(scheduler, initHeadPos, requestQ, policy, direction):

s = scheduler.schedule(initHeadPos, requestQ, policy, direction)

t = totalSeeks(initHeadPos, s)

print('policy %s %s (%d): %s' % (policy, direction, t, s))

scheduler = DiskScheduler(200)

requestQueue = [98, 183, 37, 122, 14, 124, 65, 67]

initHeadPos = 53

for policy **in** DiskScheduler.\_POLICIES:

if policy[:2] == 'C-' or policy[-4:] in ['SCAN', 'LOOK']:

TestPolicy(scheduler,initHeadPos, requestQueue, policy, 'up')

TestPolicy(scheduler,initHeadPos, requestQueue, policy, 'down')

else:

TestPolicy(scheduler, initHeadPos, requestQueue, policy, '')

print('more tests on SCAN and C-SCAN')

rQs = [[98, 37, 0, 122, 14], [98, 37, 199, 122, 14], [98,0,37,199,14]]

for q **in** rQs:

print('Q=%s' % q)

for policy in ['SCAN', 'C-SCAN']:

for direction in ['up', 'down']:

TestPolicy(scheduler, initHeadPos, q, policy, direction)

You can expect to get output like this:

$ python3 disk.py

policy FCFS (640): [98, 183, 37, 122, 14, 124, 65, 67]

policy SSTF (236): [65, 67, 37, 14, 98, 122, 124, 183]

policy SCAN up (331): [65, 67, 98, 122, 124, 183, 199, 37, 14]

policy SCAN down (236): [37, 14, 0, 65, 67, 98, 122, 124, 183]

policy LOOK up (299): [65, 67, 98, 122, 124, 183, 37, 14]

policy LOOK down (208): [37, 14, 65, 67, 98, 122, 124, 183]

policy C-SCAN up (382): [65, 67, 98, 122, 124, 183, 199, 0, 14, 37]

policy C-SCAN down (386): [37, 14, 0, 199, 183, 124, 122, 98, 67, 65]

policy C-LOOK up (322): [65, 67, 98, 122, 124, 183, 14, 37]

policy C-LOOK down (326): [37, 14, 183, 124, 122, 98, 67, 65]

more tests on SCAN and C-SCAN

Q=[98, 37, 0, 122, 14]

policy SCAN up (345): [98, 122, 199, 37, 14, 0]

policy SCAN down (175): [37, 14, 0, 98, 122]

policy C-SCAN up (382): [98, 122, 199, 0, 14, 37]

policy C-SCAN down (353): [37, 14, 0, 199, 122, 98]

Q=[98, 37, 199, 122, 14]

policy SCAN up (331): [98, 122, 199, 37, 14]

policy SCAN down (252): [37, 14, 0, 98, 122, 199]

policy C-SCAN up (382): [98, 122, 199, 0, 14, 37]

policy C-SCAN down (353): [37, 14, 0, 199, 122, 98]

Q=[98, 0, 37, 199, 14]

policy SCAN up (345): [98, 199, 37, 14, 0]

policy SCAN down (252): [37, 14, 0, 98, 199]

policy C-SCAN up (382): [98, 199, 0, 14, 37]

policy C-SCAN down (353): [37, 14, 0, 199, 98]