CS 342302 Operating Systems

Fall Semester 2021

Prof. Pai H. Chou

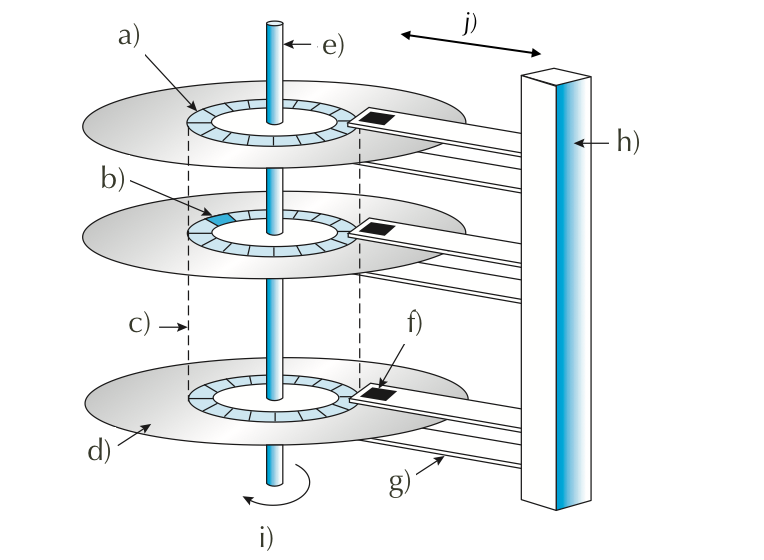
Weekly Review 12

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).

p7



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

time to rotate to desired sector

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

time to move disk arm to desired cylinder

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

seek time + rotational latency

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

random-access time

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

seek time + rotational latency

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

track = Concentric rings on a given platter

cylinder = 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?

page

* 1. What is the minimum unit of writing?

page

* 1. What is a block?

minimum erase unit, contains several page

* 1. What happens during an erase?

set to all 1's

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

p12

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

…

to even out wear on some "hot" blocks

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

minimize seek time ≈ seek distance

OS maintains queue of requests (per disk or device) => p19

1. Of the different disk scheduling algorithms,

p21~28

* 1. is FIFO in general a good policy for HDD? for SSD?
  2. is STSF in general a good policy for HDD? What kind of problem does it have? What about for SSD?
  3. What is the difference between SCAN and C-SCAN?
  4. What is the difference between SCAN and LOOK?
  5. Why does SCAN have more predictable behavior than LOOK?

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

Need to minimize write

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

set-aside by low level formatter to replace defective sectors over time

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

shift sector contents so the spare preserves contiguity

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

p37

yes

…

1. What is a **swap partition**, and why is it a good idea?
2. how do you pronounce SCSI?

"skuzzy"

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

Redundant Array of Independent Disks

1. How can RAID achieve higher reliability?

mirroring: Make exact copy of disk: every write => write same data to multiple disks

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

data striping: split data onto multiple disks, access in parallel

Bit-level, Block-level

1. What is the meaning of the following about disks?
   1. **mean time to failure**
   2. **mean time to repair**, and is it related to mean time to failure?

average time to (discover and) repair or replace+restore the failed disk

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

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

Make exact copy of disk: every write => write same data to multiple disks

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

split data onto multiple disks, access in parallel

no

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

bit: access different bits on different disks

block: access different blocks on different disks

block

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

striped mirror: RAID 1+0

mirrored stripes: RAID 0+1

striped mirror

## 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]