PERSISTENCE: RAID

Shivaram Venkataraman

CS 537, Spring 2020

ADMINISTRIVIA

Midterm grades → Grading complete
Uploading PDFs to Canvas

Pass / Fail options
↳ CS advising
advising@cs.wisc.edu

Grading Policy
↳ Exams

Project 4a: Spec updates, test cases

Project 4a: New due date April 3rd → 10pm

Office hours
Debug
Collaborate

AGENDA / LEARNING OUTCOMES

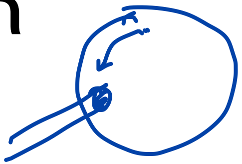
Why do we need more than one disk in a system?

How do we achieve resilience against disk errors?

RECAP

Disk Drives Geometry

SEEK, ROTATE, TRANSFER



Seek cost: Function of cylinder distance

- Not purely linear cost
- Must accelerate, coast, decelerate, settle
- Settling alone can take 0.5 - 2 ms

Entire seeks often takes 4 - 10 ms

Average seek = 1/3 of max seek

Depends on rotations per minute (RPM)

- 7200 RPM is common, 15000 RPM is high end

Average rotation – half of a rotation

Transfer

Pretty fast: depends on RPM and sector density.

100+ MB/s is typical for maximum transfer rate

I/O SCHEDULERS

Random expensive

Sequential

Given a stream of I/O requests, in what order should they be served?

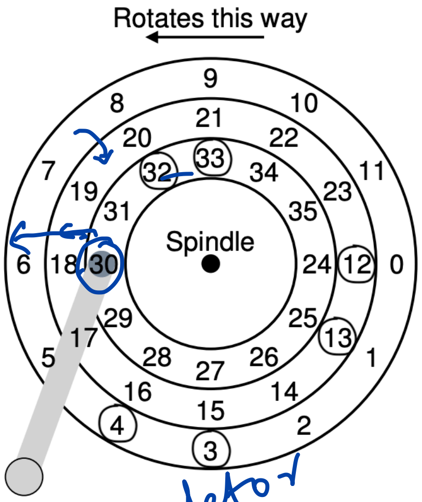
Much different than CPU scheduling

Position of disk head relative to request position matters more than length of job

QUIZ 22

https://tinyurl.com/cs537-sp20-quiz22

HW: Textbook



Disk Simulator
↳ start pause:

Disk accesses: 32, 12, 33, 3, 13, 4
Rotation Time = 2ms (non-adjacent reads)
Seek Time (for adjacent track) = 2ms.

What is the time taken to using (FCFS) scheduling?

32: 2ms rotate
12: 2ms seek + 2ms rotate
33: 2ms seek + 2ms rotate
3: 4ms seek + 2ms rotate
13: 2ms seek + 2ms rotate
4: " "

= 2 + 4 + 4 + 6 + 4 + 4 = 24ms

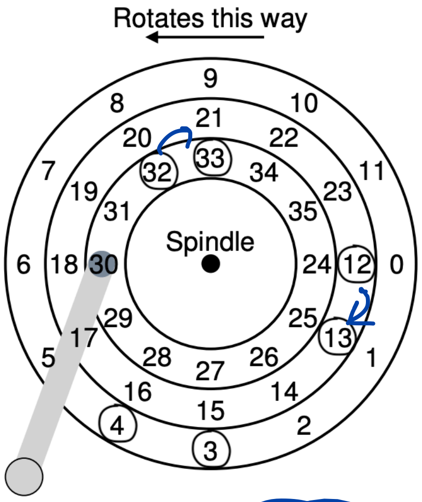
Order in which requests will be serviced for Shortest Seek Time First (SSTF)?

32, 33, 12, 13, 3, 4

1 track   1 track

QUIZ 22

https://tinyurl.com/cs537-sp20-quiz22



Disk accesses: 32, 12, 33, 3, 13, 4
Rotation Time = 2ms (non-adjacent reads)
Seek Time (for adjacent track) = 2ms.

Order in which requests will be serviced for Shortest Seek Time First (SSTF)?

32, 33, 12, 13, 3, 4

FCFS
24 ms

SSTF
10 ms

Time Taken

32: 2ms rotate
33: 0
12: 2ms seek + 2ms rotate
13: 0
3: 2ms seek + 2 ms rotate, 4's is 0

2 ms + 4 ms + 4 ms = 10ms

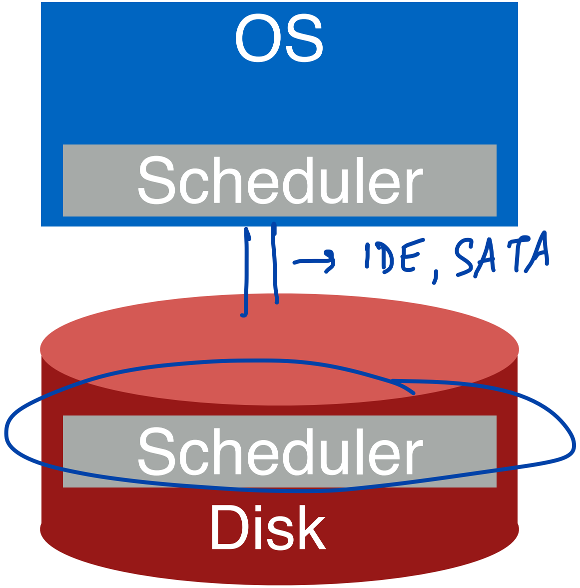
SCHEDULERS

Process is requesting →

→ Fairness across processes

→ Easy to upgrade

→ Disk geometry
seek time
rotation time



→ IDE, SATA

OS scheduler does scheduling sectors close to each other

Where should the scheduler go?

Disk can do further reading

WHAT HAPPENS?

Assume 2 processes each calling read() with C-SCAN

```
void reader(int fd) {
    char buf[1024];
    int rv;
    while((rv = read(fd, buf)) != 0) {
        assert(rv);
        // takes short time, e.g., 1ms
        process(buf, rv);
    }
}
```

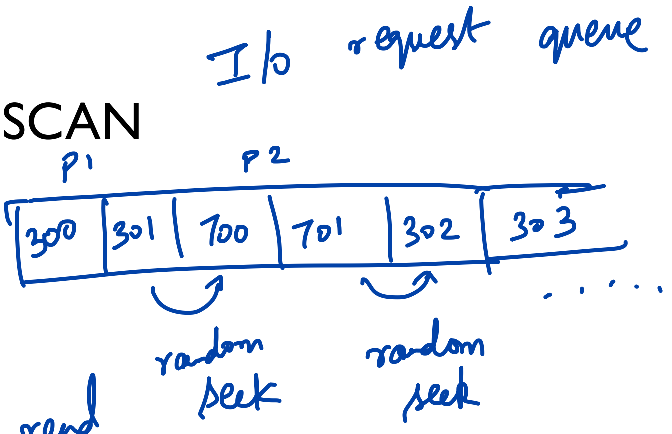
P1, P2

512 bytes   sector size

Shared OS

seq read

Processing



Even though process is reading sequentially disk reads may not be.

WORK CONSERVATION

Work conserving schedulers always try to do work if there's work to be done

Sometimes, it's better to wait instead if system anticipates another request will arrive

Possible improvements from I/O Merging

If request exists ⇒ work conserving always processes it] Device / resource busy

Not work conserving ⇒ wait before doing work

Pre-fetch = fetch more than asked

1024 bytes vs. 4096 bytes

DISKS SUMMARY

Disks: Specific geometry with platters, spindle, tracks, sectors

I/O Time: rotation_time + seek_time + transfer_time

Sequential throughput vs. random throughput

Advanced Techniques: Skewed layout, caching

Scheduling approaches: SSTF, SCAN, C-SCAN
Benefits of violating work conservation

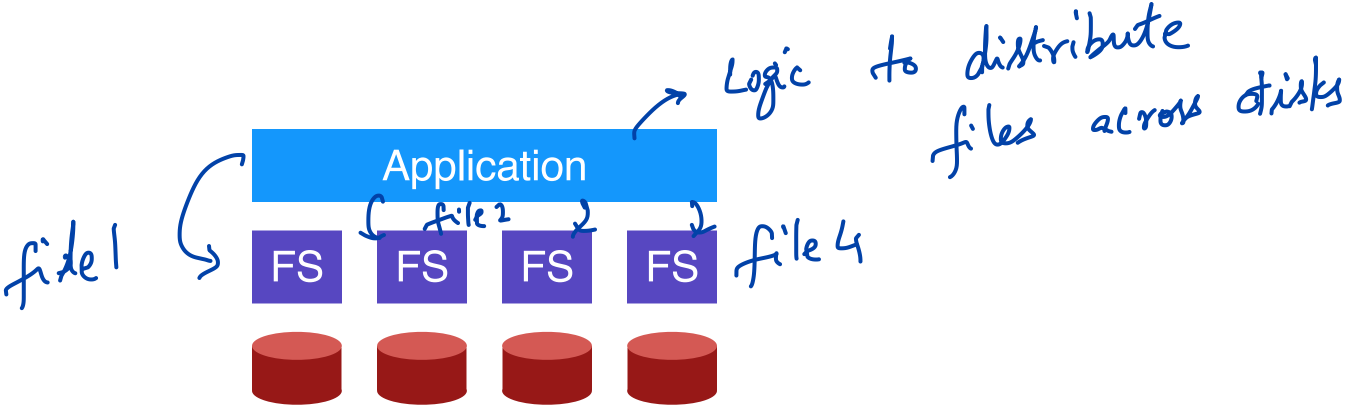
ONLY ONE DISK?

Sometimes we want many disks — why?

- capacity
- reliability → physical media wear & tear
- performance

Challenge: most file systems work on only one disk

SOLUTION 1: JBOD

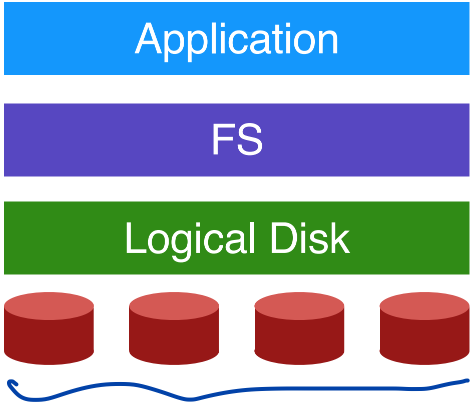


Application is smart, stores different files on different file systems.

JBOD: **J**ust **a** **B**unch **O**f **D**isks

SOLUTION 2: RAID

Build logical disk from many physical disks.



Logical disk gives

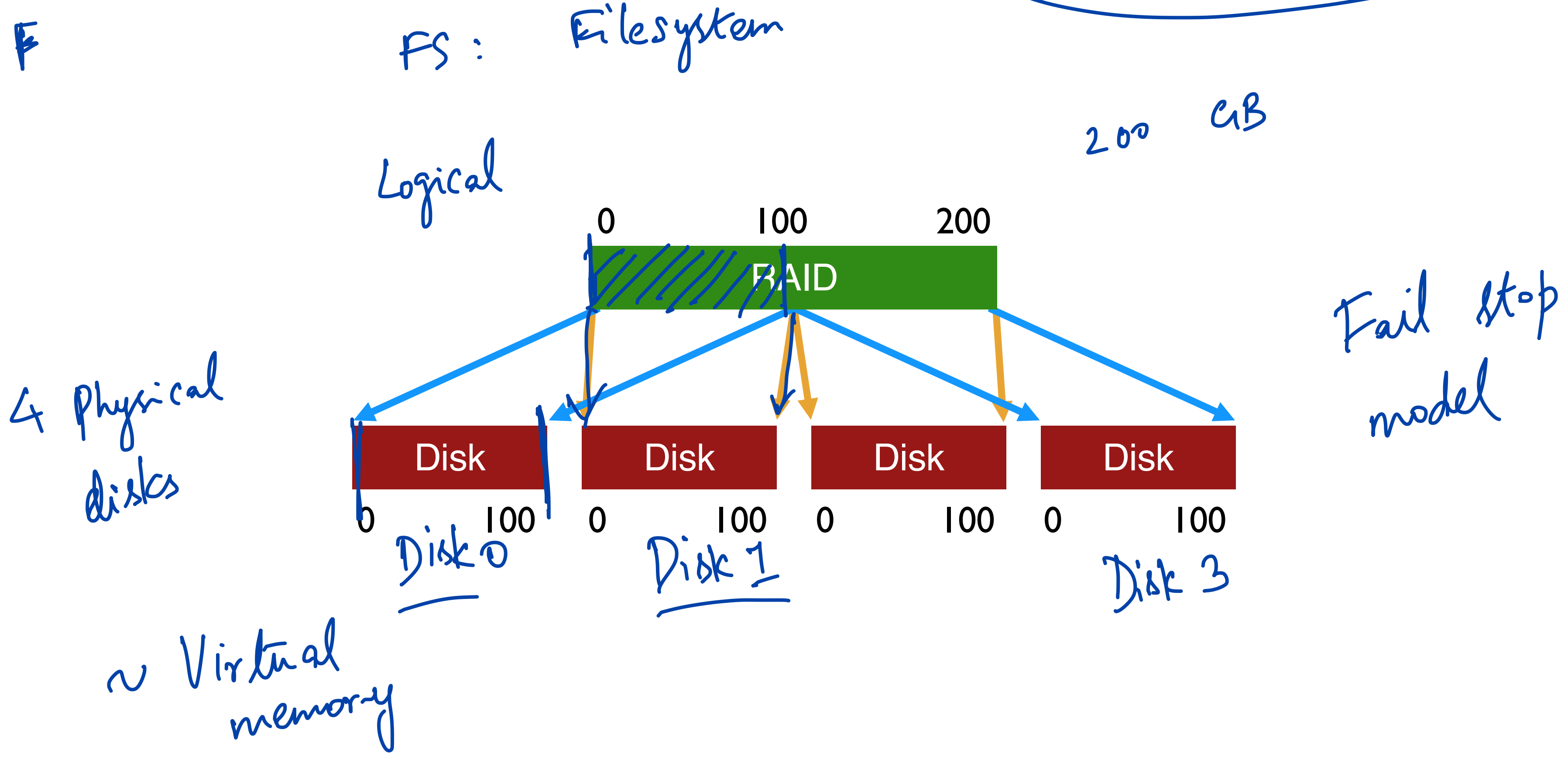
capacity,

performance,

reliability

RAID: **R**edundant **A**rray of **I**nexpensive **D**isks

GENERAL STRATEGY: MAPPING, REDUNDANCY



MAPPING

How should we map logical block addresses to physical block addresses?

1) **Dynamic** mapping: use data structure (hash table, tree)

Page table

2) **Static** mapping: use simple math

RAID schemes
also called levels → different strategies to do the mapping

WORKLOADS

Reads

- One operation
- Steady-state I/O
  - Sequential ← read 10 MB of data
  - Random ← read 4 KB of data
    1 KB of data

Writes

- One operation
- Steady-state I/O
  - Sequential
  - Random

METRICS

Capacity: how much space can apps use?

Reliability: how many disks can we safely lose? (assume fail stop)

Performance: how long does each workload take? (latency, throughput)

Normalize each to characteristics of one disk

Different **RAID levels** make different trade-offs

"Logical disk" in terms of physical disks

Disk with capacity C

N such disks

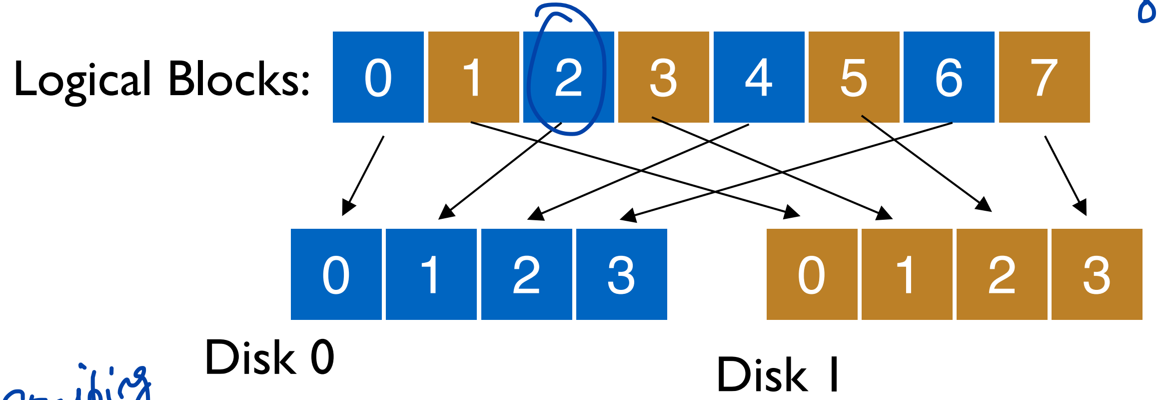
max. N * C

Actual capacity maybe < N * C

RAID-0: STRIPING

Optimize for capacity. No redundancy

8 blocks / sectors



logical block % 2

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

Striping

4 disks

D0	D1	D2	D3
0	1	2	3
4	5	6	7
8	9	10	11
. . .	. . .	. . .	. . .

RAID 0: STRIPES AND CHUNK SIZE

Fails ⇒ block 1, 5, 9, 13 will be inaccessible

Chunk size = 1

stripe: (row 4, 5, 6, 7)

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

assume chunk size of 1 for this lecture

Chunk size = 2

more contiguous data

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

RAID-0: ANALYSIS

What is capacity? N * C — as we use all the disks to store data

How many disks can we safely lose? 0

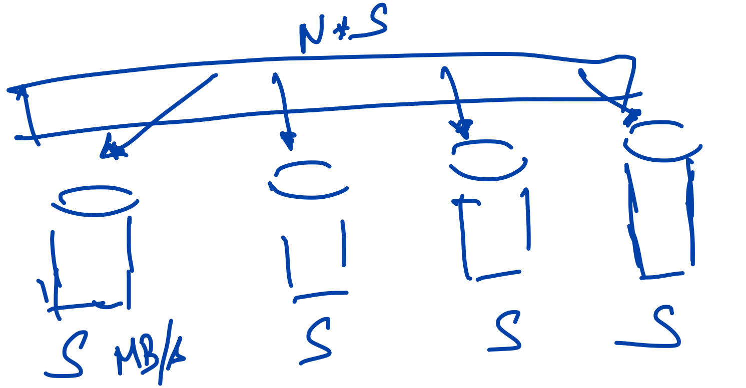
Latency (random) D → choose a disk where block is located ⇒ latency of read same as that disk

Throughput (~~sequential, random~~)? N*S , N*R

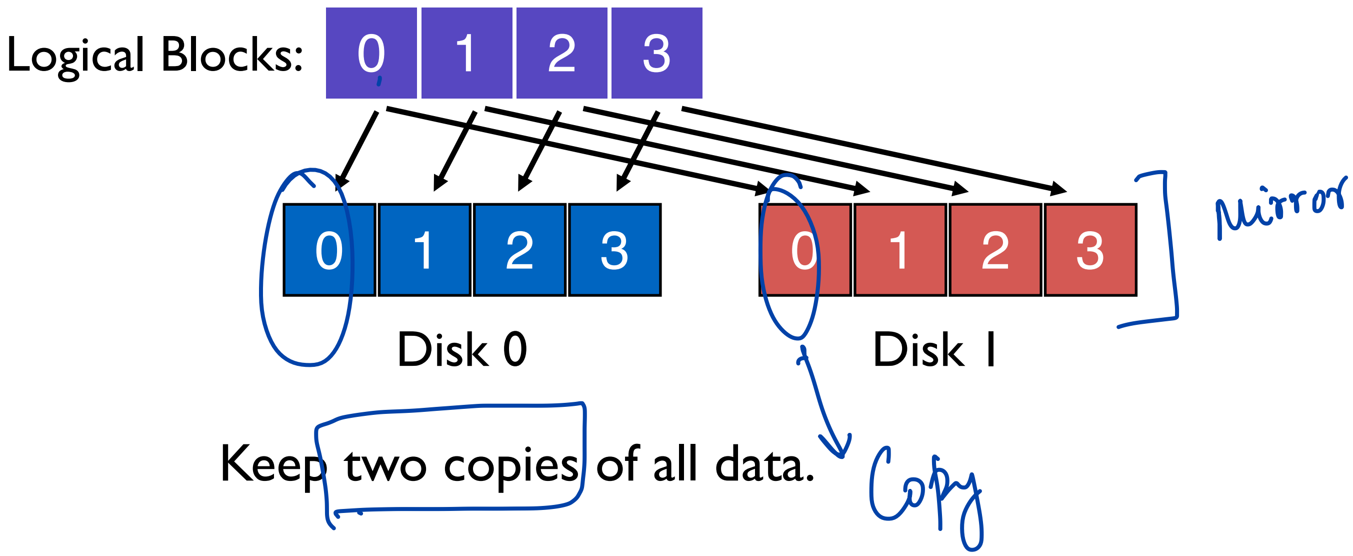
Notations Physical

N := number of disks
C := capacity of 1 disk
S := sequential throughput of 1 disk
R := random throughput of 1 disk
D := latency of one small I/O operation

Can send req to disks in parallel!



RAID-1: MIRRORING



Keep two copies of all data.

RAID-1 LAYOUT: MIRRORING

2 disks

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

2 copies

even number of disks

4 disks

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

RAID-1: ANALYSIS

What is capacity? $\frac{N}{2} * C$

How many disks can fail? 1 at least, sometimes more than 1 failure is also OK

Latency (read, write)? D

block 0

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

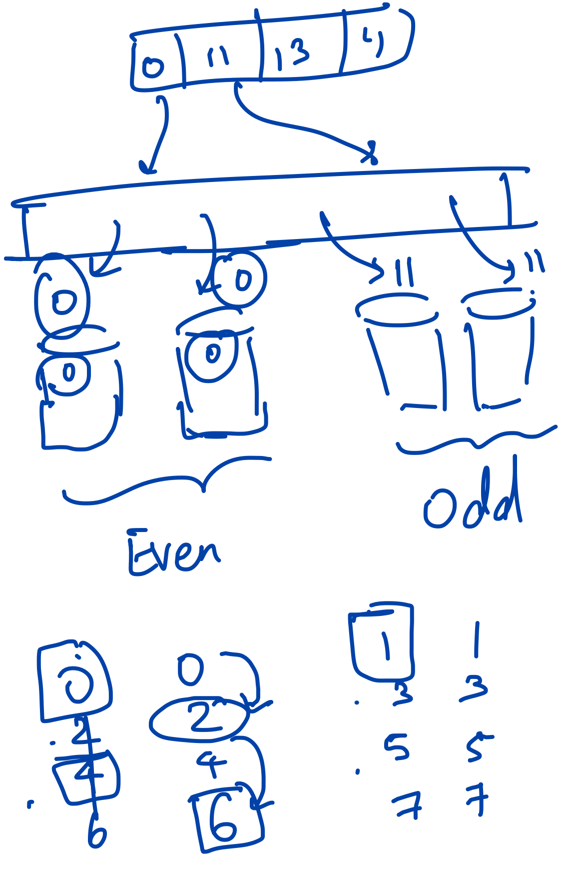
N := number of disks
C := capacity of 1 disk
S := sequential throughput of 1 disk
R := random throughput of 1 disk
D := latency of one small I/O operation

RAID-1: THROUGHPUT

What is steady-state throughput for

- random reads? $N * R$
- random writes? $\frac{N}{2} * R$ ← lesser Tput RAID-0
- sequential writes? $\frac{N}{2} * S$
- sequential reads? $\frac{N}{2} * S$

same logic as random writes
end up having skip sectors



QUIZ 23

https://tinyurl.com/cs537-sp20-quiz23

Disk characteristics: Average seek time = 7ms,
Average rotational time = 3ms, transfer rate = 50 MB/s

Sequential transfer of 10MB : $T_s + T_r + T_t$

$= 7ms + 3ms + \frac{10 MB}{50 MB} ms/s \times 1000 = 210 ms$

Effective BW $= 10MB / 210 ms = 47.62$ MB/s → not that bad!

Random transfer of 10KB

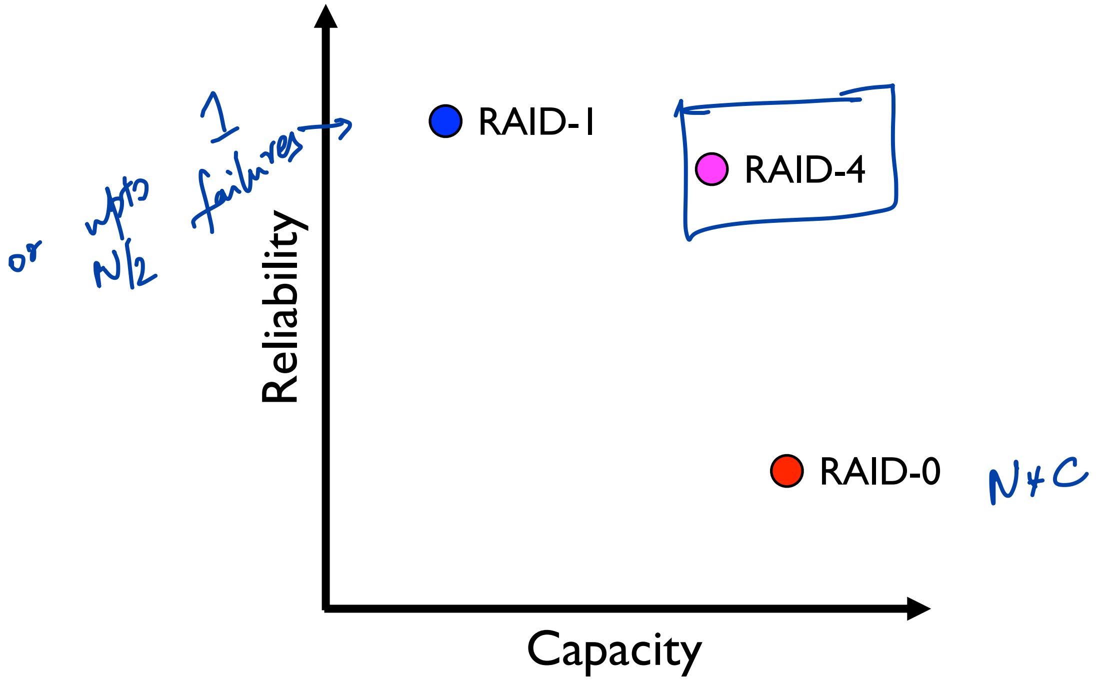
$: T_s + T_r + T_t$

$: 7ms + 3ms + \frac{10 KB}{50 MB \times 1000} \times 1000 ms = 10.2 ms$

Effective BW $= 10KB / 10.2ms = 0.981$ MB/s

RAID0, Random Writes → $N * R = 4 * 0.981 = 3.92$ MB/s

RAID1, Random Writes → $\frac{N}{2} * R = 2 * 0.981 = 1.96$ MB/s



Office hours
from 2.30
Blackboard

P4 a
April 3rd

RAID-4 STRATEGY

Use **parity** disk

If an equation has N variables, and N-1 are known,
you can solve for the unknown.

Treat sectors across disks in a stripe as an equation.

Data on bad disk is like an unknown in the equation.

RAID 4: EXAMPLE

	Disk0	Disk1	Disk2	Disk3	Disk4
Stripe:	3	0	1	2	
					parity

What functions can we use to compute parity?

RAID-4: ANALYSIS

What is capacity?

How many disks can fail?

Latency (read, write)?

Disk0	Disk1	Disk2	Disk3	Disk4 parity
1	0	1	1	1
0	1	1	0	0
1	1	0	1	1

N := number of disks
C := capacity of 1 disk
S := sequential throughput of 1 disk
R := random throughput of 1 disk
D := latency of one small I/O operation

RAID-4: THROUGHPUT

What is steady-state throughput for

- sequential reads?
- sequential writes?
- random reads?
- random writes? (next page!)

Disk0	Disk1	Disk2	Disk3	Disk4
3	0	1	2	6
				(parity)

RAID-4: ADDITIVE VS SUBTRACTIVE

C0	C1	C2	C3	P0
0	0	1	1	XOR(0,0,1,1)

Additive Parity

Subtractive Parity

$$P_{new} = (C_{old} \oplus C_{new}) \oplus P_{old}$$

RAID-5

Disk0	Disk1	Disk2	Disk3	Disk4
-	-	-	-	P
-	-	-	P	-
-	-	P	-	-

...

Rotate parity across different disks

RAID-5: ANALYSIS

What is capacity?

How many disks can fail?

Latency (read, write)?

Disk0	Disk1	Disk2	Disk3	Disk4
-	-	-	-	P
-	-	-	P	-
-	-	P	-	-

...

N := number of disks
C := capacity of 1 disk
S := sequential throughput of 1 disk
R := random throughput of 1 disk
D := latency of one small I/O operation

RAID-5: THROUGHPUT

What is steady-state throughput for RAID-5?

- sequential reads?
- sequential writes?
- random reads?
- random writes? (next page!)

Disk0	Disk1	Disk2	Disk3	Disk4
-	-	-	-	P
-	-	-	P	-
-	-	P	-	-

...

RAID-5 RANDOM WRITES

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 LEVEL COMPARISONS

	Reliability	Capacity	Read latency	Write Latency	Seq Read	Seq Write	Rand Read	Rand Write
RAID-0	0	C*N	D	D	N * S	N * S	N * R	N * R
RAID-1	1	C*N/2	D	D	N/2 * S	N/2 * S	N * R	N/2 * R
RAID-4	1	(N-1) * C	D	2D	(N-1)*S	(N-1)*S	(N-1)*R	R/2
RAID-5	1	(N-1) * C	D	2D	(N-1)*S	(N-1)*S	N * R	N/4 * R

SUMMARY

RAID: a faster, larger, more reliable disk system

One logical disk built from many physical disk

Different mapping and redundancy schemes

Present different trade-offs

Next steps: Filesystems on Thu
P4a due on Friday!