Operating Systems

Lecture 28: HDD + RAID

Nipun Batra

Nov 13, 2018

Device Protocol Variants

Status checks: polling *vs.* interrupts

Data: PIO *vs.* DMA

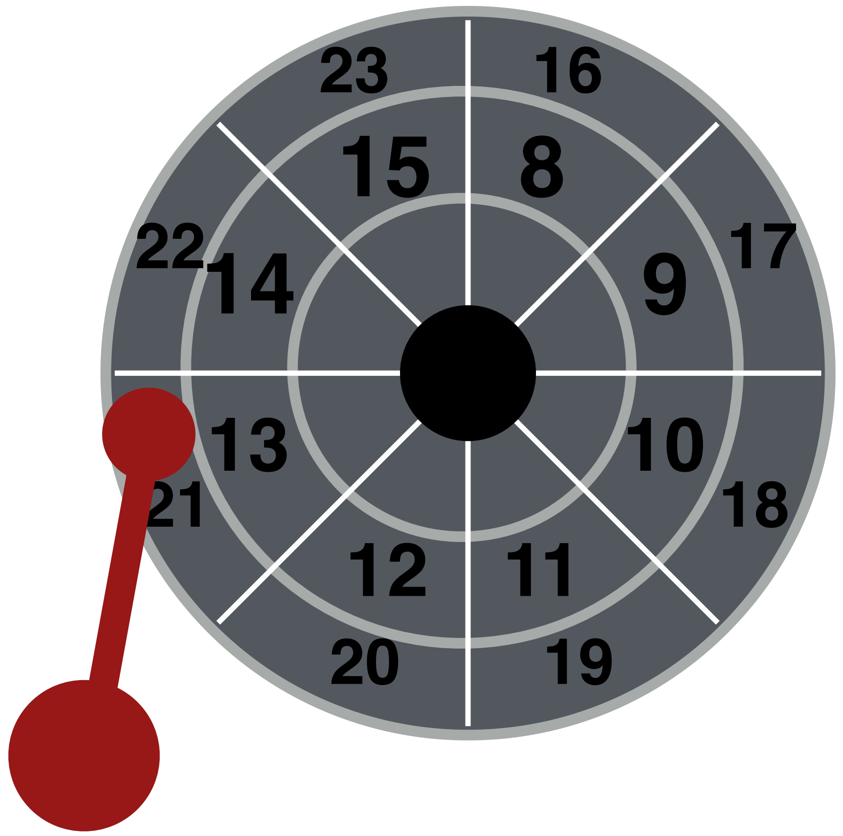
Control: special instructions *vs.* memory-mapped I/O

Disks

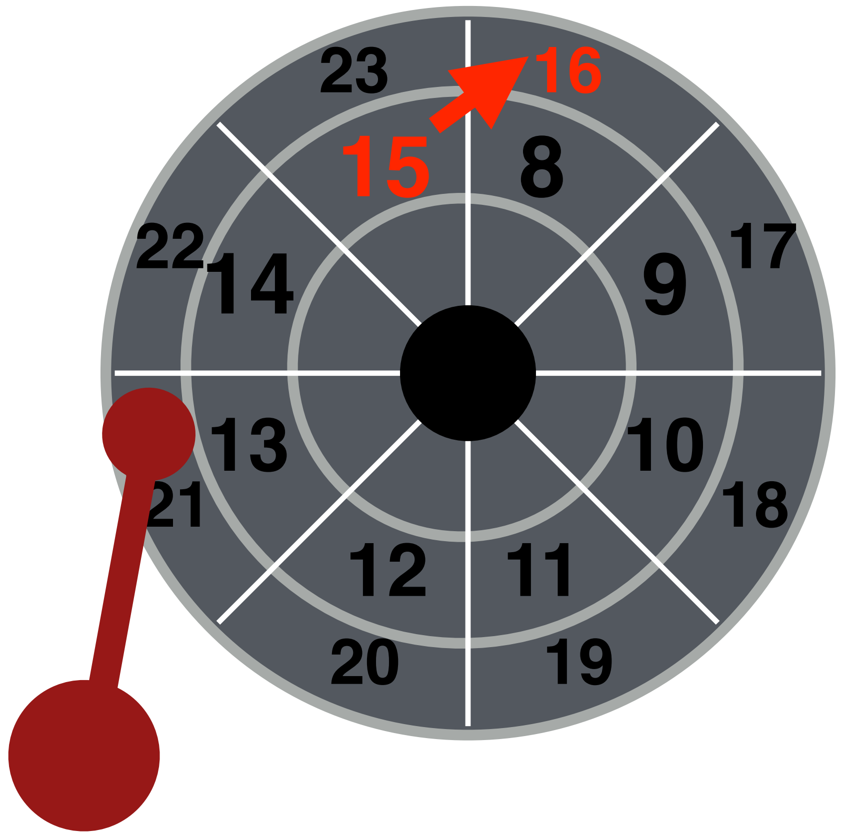
Doing an I/O requires:
- seek
- rotate
- transfer

What is expensive?

Track Skew

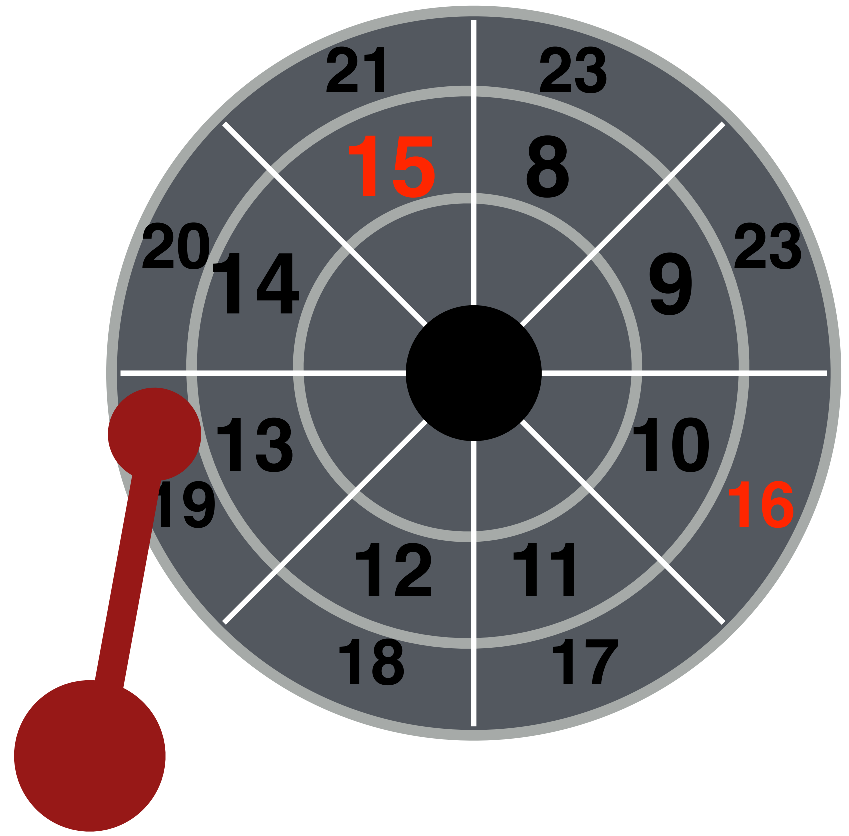


Track Skew

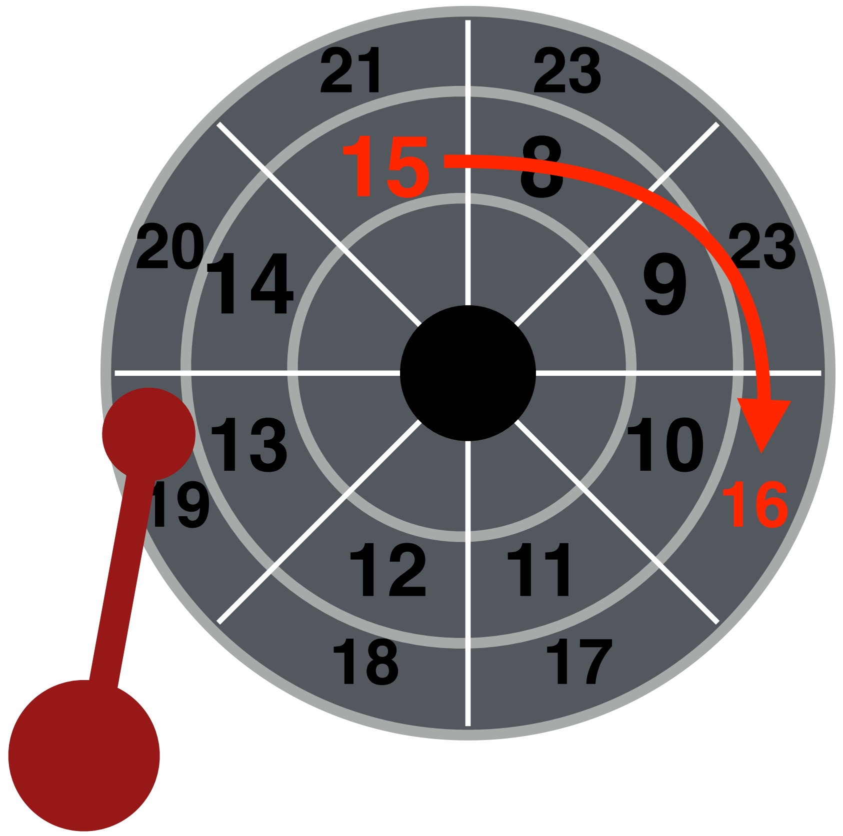


When reading 16 after 15, the head won't settle quick enough, so we need to do a rotation.

Track Skew



Track Skew



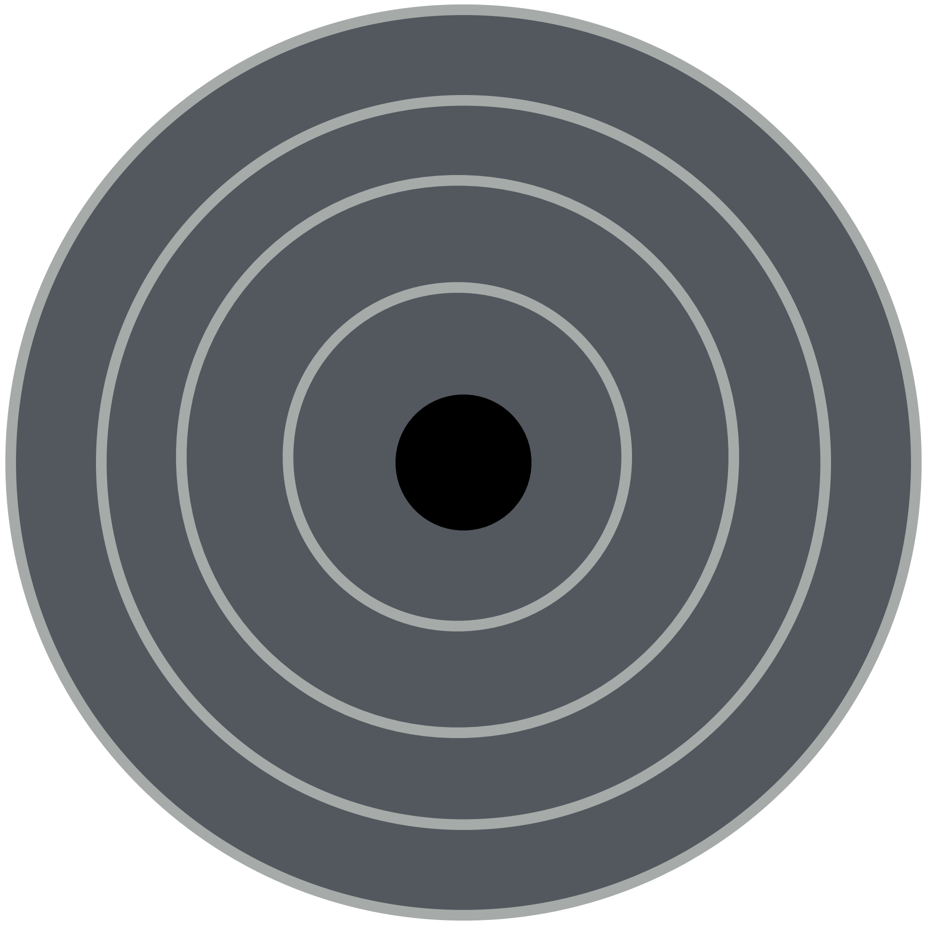
Other Improvements

Track Skew

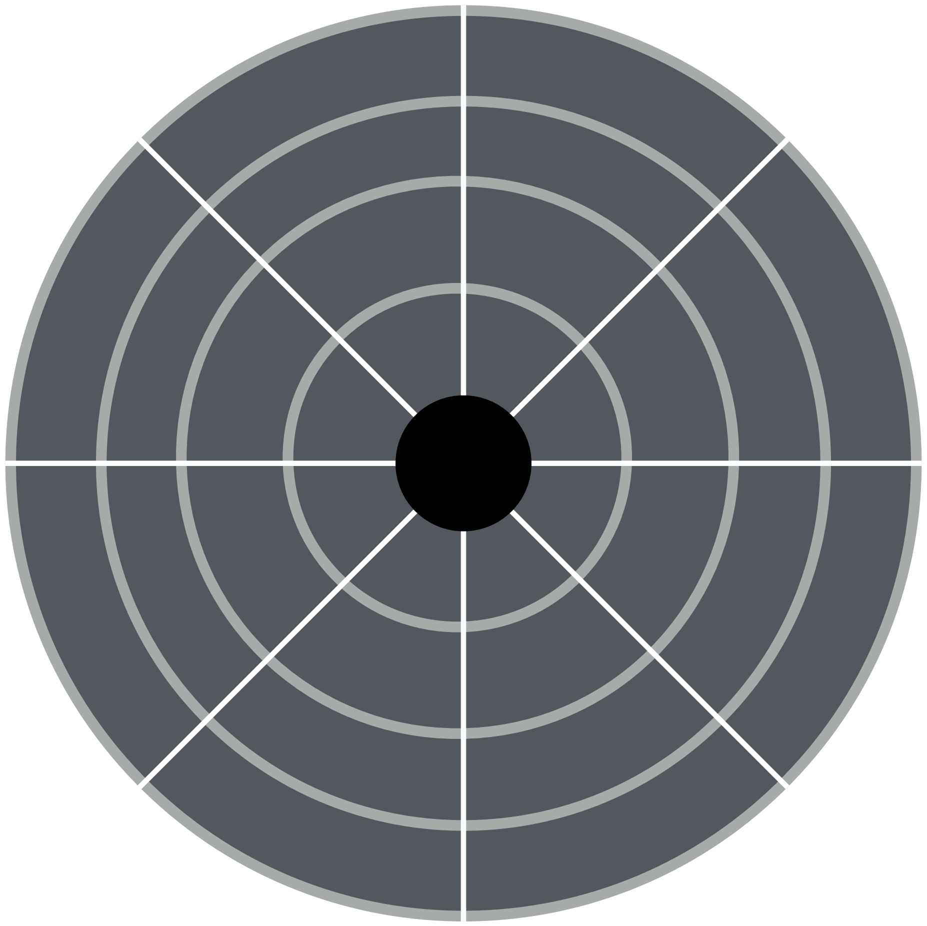
Zones

Cache

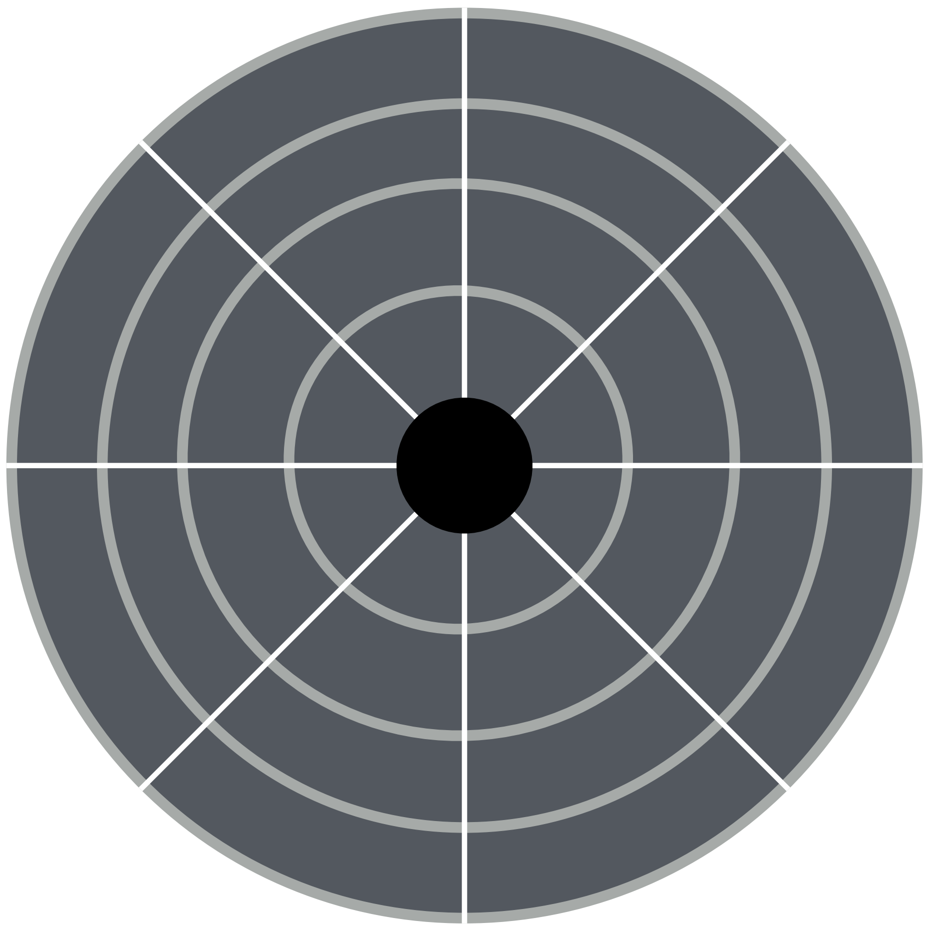
Zones



Zones

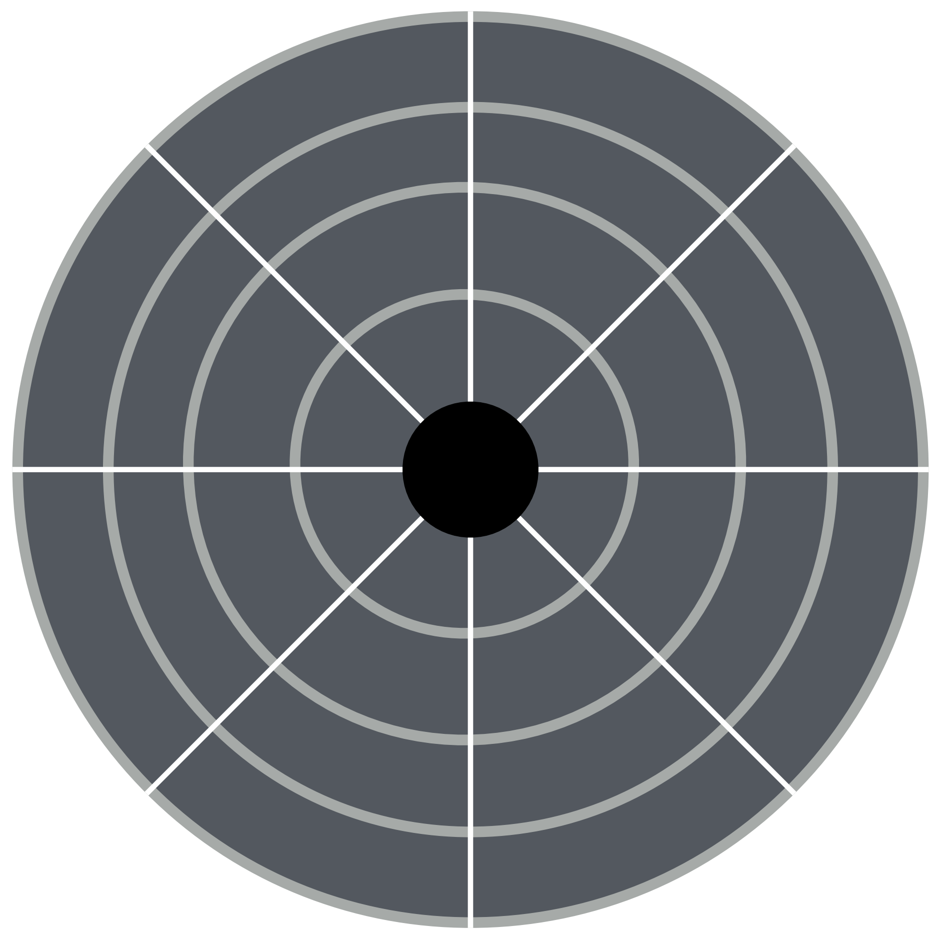


Zones



Should we have equal number of sectors for all tracks?

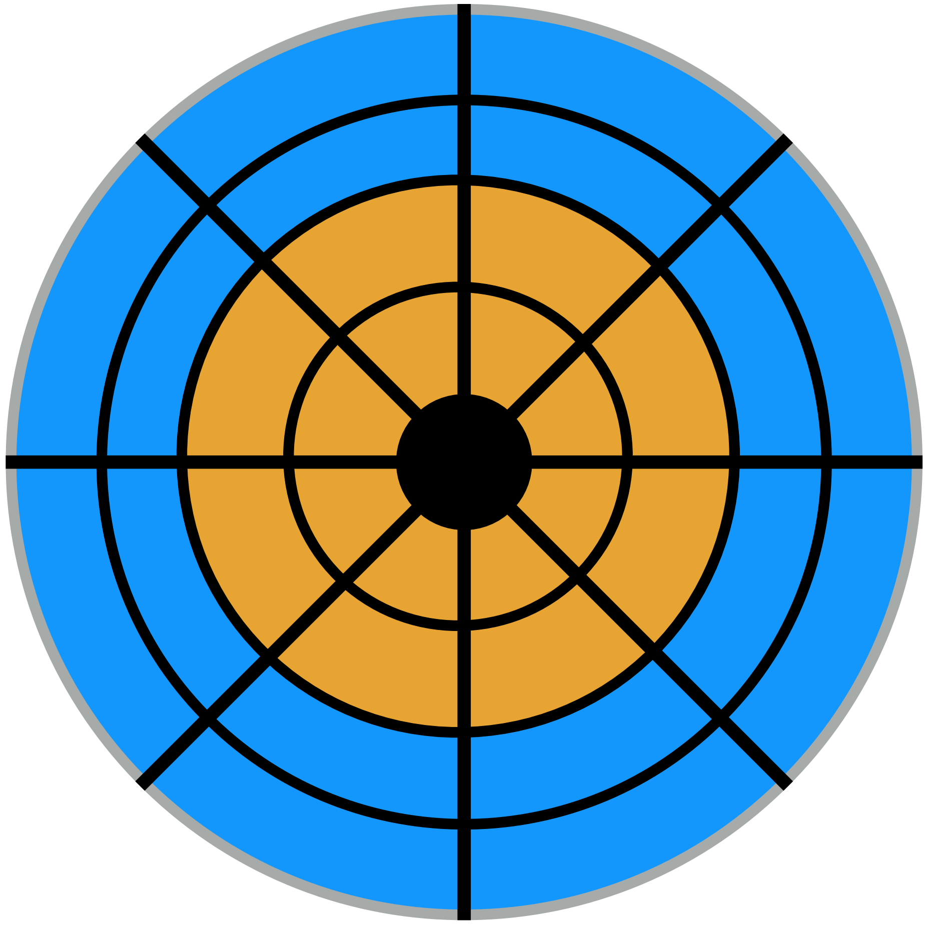
Zones



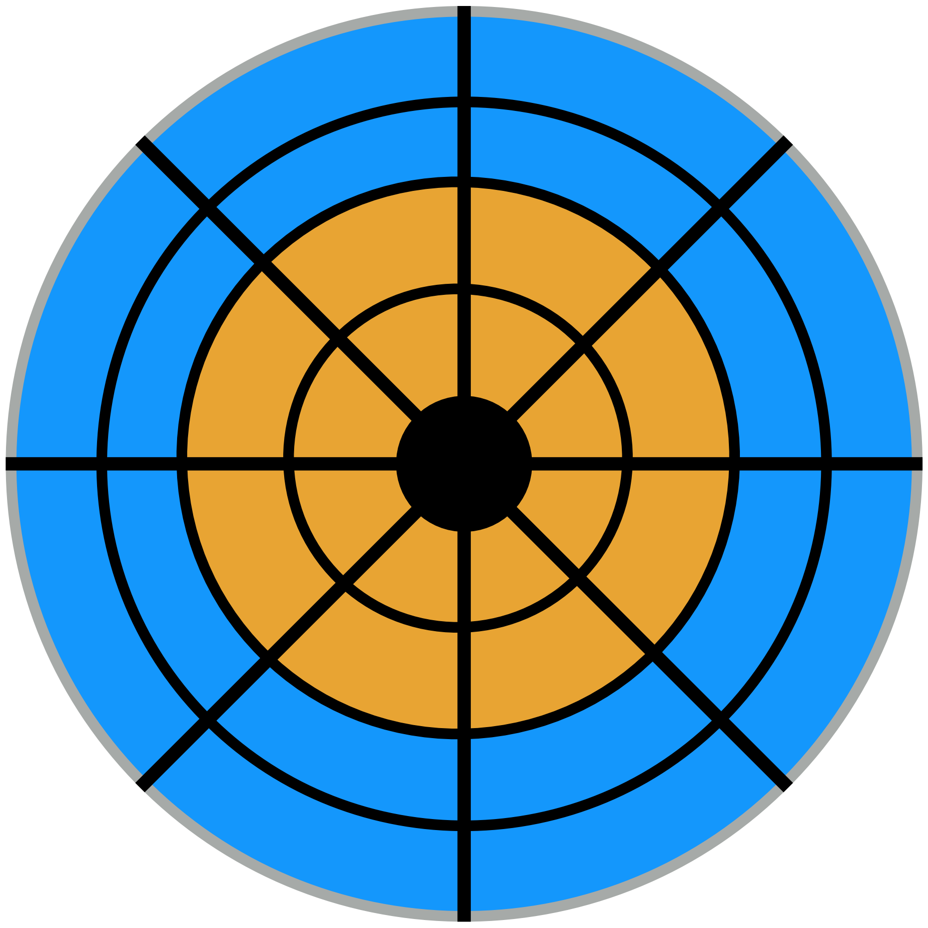
Should we have equal number of sectors for all tracks?

Sector density decreases with increasing radius?

Zones



Zones



Divide disk into zones — all tracks in a zone have equal number of sectors

Other Improvements

Track Skew

Zones

Cache

Drive Cache

- Drives may cache both reads and writes.

- What advantage does drive have for reads?
  - Anticipate disk reads and put them into cache
  - Return data from cache directly

- What advantage does drive have for writes?
  - Put data in cache —> tell OS that write done (write back caching or immediate reporting)

Schedulers

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

FCFS (First-Come-First-Serve)

Assume seek+rotate = 10 ms on average.
Assume transfer = 100 MB/s.

How long (roughly) does the below workload take?
The integers are sector numbers.

300001, 700001, 300002, 700002, 300003, 700003

FCFS (First-Come-First-Serve)

Assume seek+rotate = 10 ms on average.
Assume transfer = 100 MB/s.

How long (roughly) does the below workload take?
The integers are sector numbers.

300001, 700001, 300002, 700002, 300003, 700003 (~60ms)

FCFS (First-Come-First-Serve)

Assume seek+rotate = 10 ms on average.
Assume transfer = 100 MB/s.

How long (roughly) do the below workloads take?
The integers are sector numbers.

300001, 700001, 300002, 700002, 300003, 700003 (~60ms)
300001, 300002, 300003, 700001, 700002, 700003

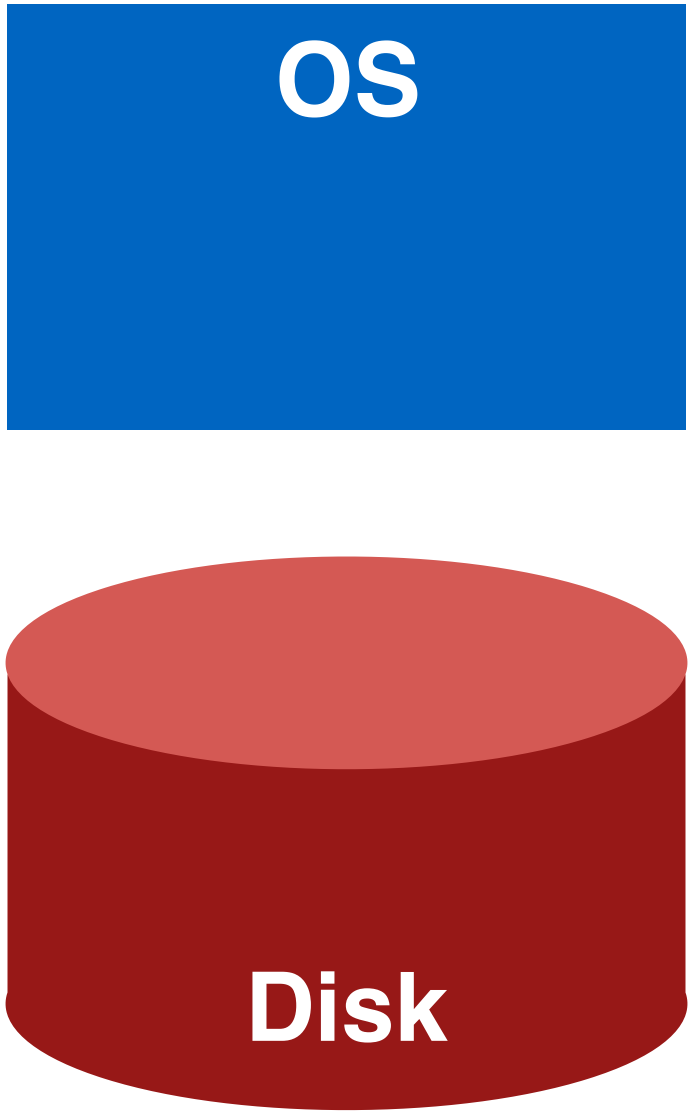
FCFS (First-Come-First-Serve)

Assume seek+rotate = 10 ms on average.
Assume transfer = 100 MB/s.

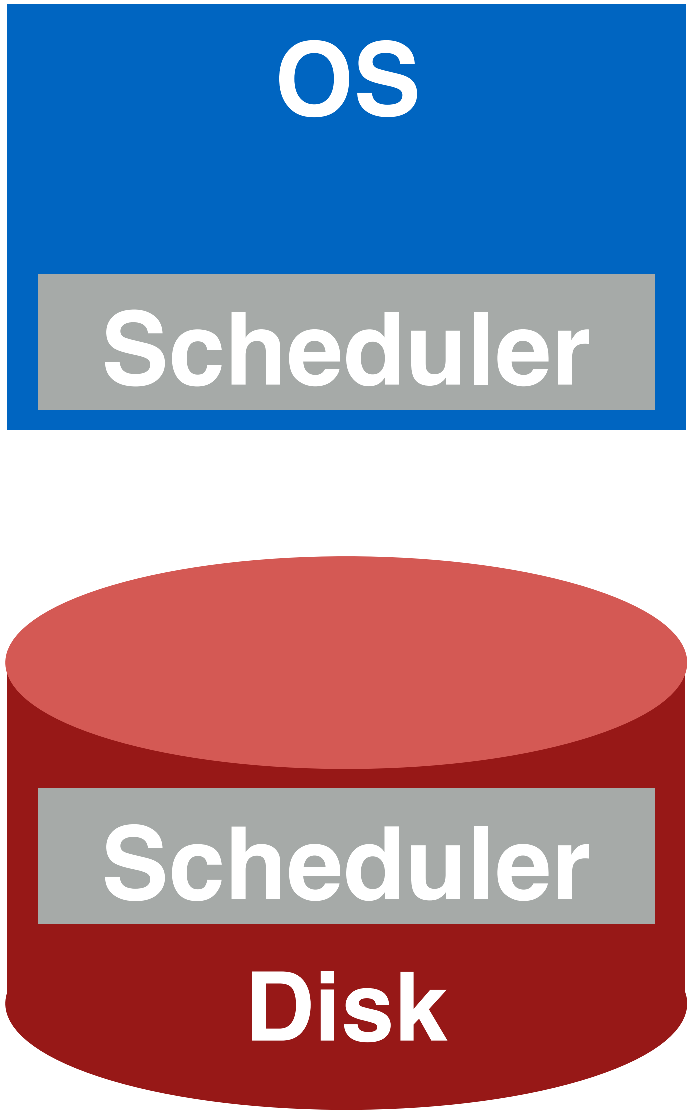
How long (roughly) do the below workloads take?
The integers are sector numbers.

300001, 700001, 300002, 700002, 300003, 700003 (~60ms)
300001, 300002, 300003, 700001, 700002, 700003 (~20ms)

Schedulers



Schedulers

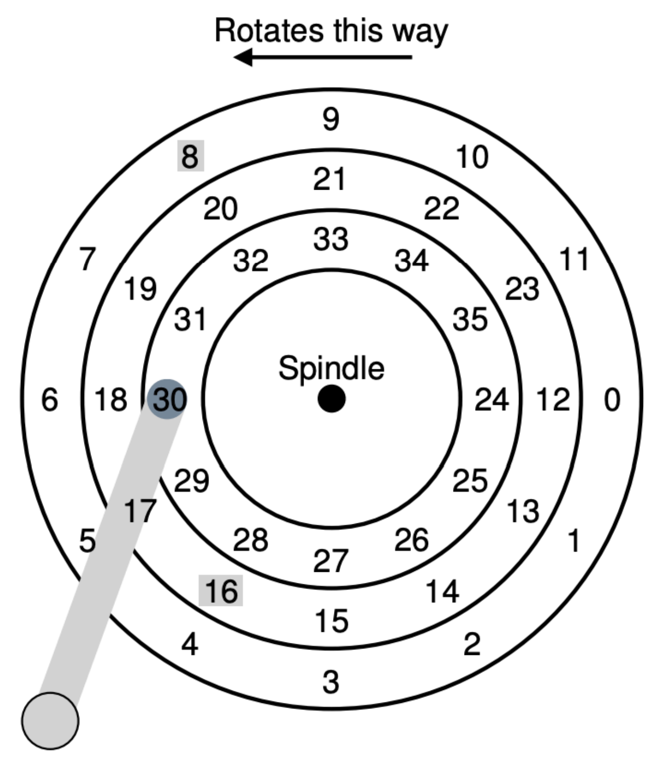


Shortest …Time First

Shortest Seek Time First: always choose the request that will take the least time for **seeking.**

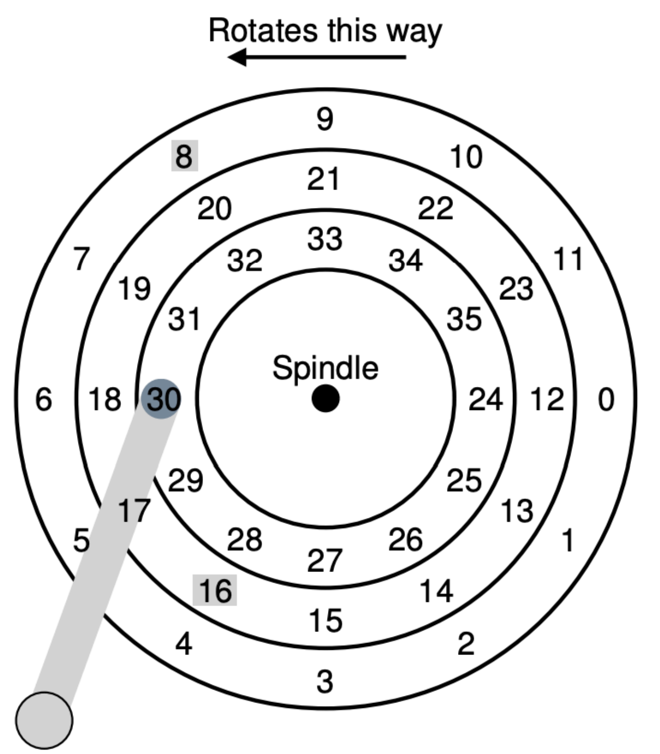
Shortest …Time First

Shortest Seek Time First: always choose the request that will take the least time for **seeking.**



Shortest …Time First

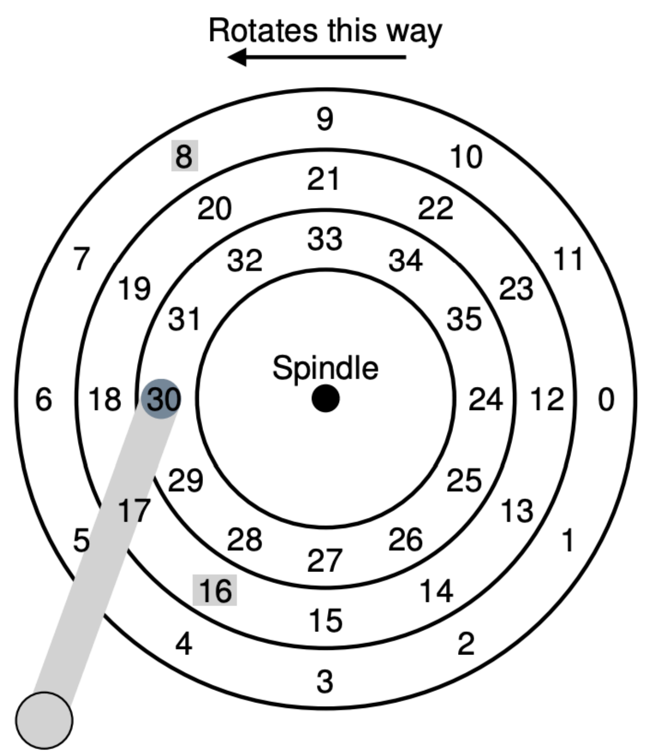
Shortest Seek Time First: always choose the request that will take the least time for **seeking.**



- Currently head is at 30, should next be 19 or 7?

Shortest …Time First

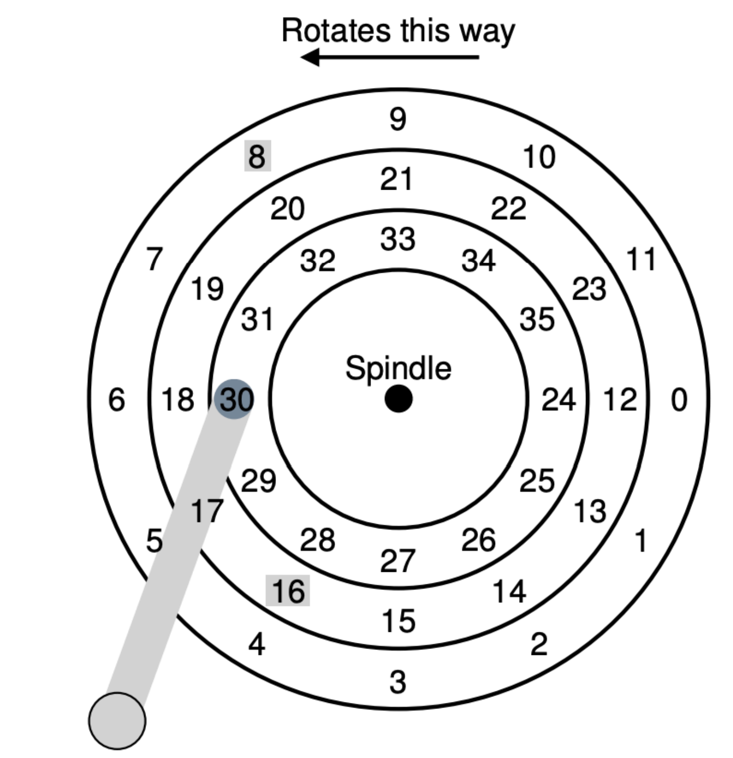
Shortest Seek Time First: always choose the request that will take the least time for **<u>seeking.</u>**



- Currently head is at 30, should next be 19 or 7?
- 19 would have lesser seek time

Shortest …Time First

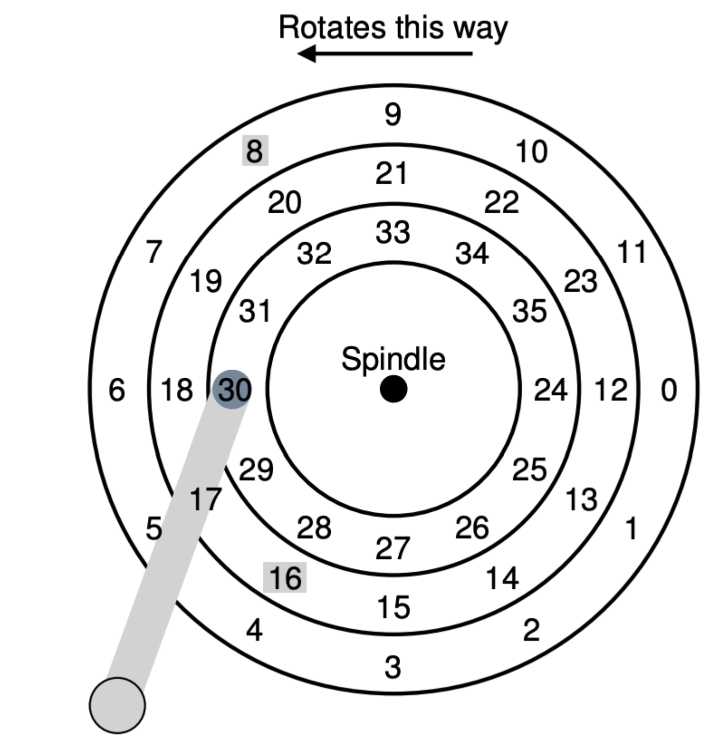
Shortest Seek Time First: always choose the request that will take the least time for **seeking.**



- Currently head is at 30, should next be 19 or 7?
- 19 would have lesser seek time
- Cons?
  - Starvation — if enough requests from close by sectors, then requests from far by sectors will be largely ignored

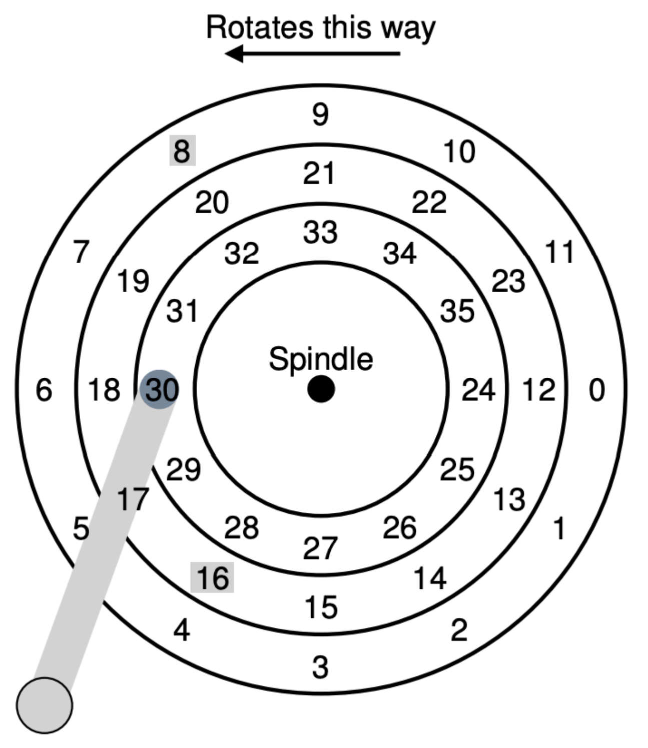
Shortest …Time First

Shortest Seek Time First: always choose the request that will take the least time for **seeking.**



Shortest …Time First

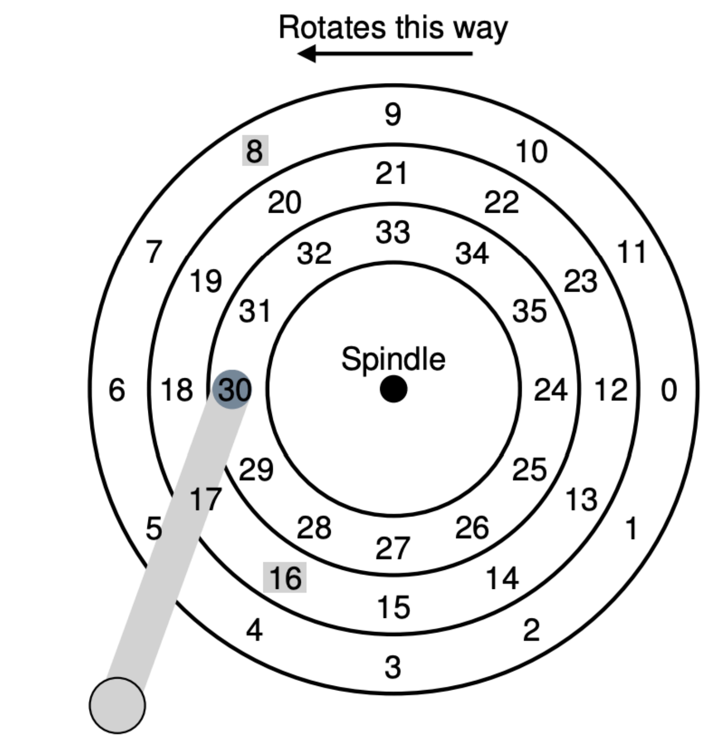
Shortest Seek Time First: always choose the request that will take the least time for **seeking.**



- What if currently head is at 30, next choice between 16 & 8

Shortest …Time First

Shortest Seek Time First: always choose the request that will take the least time for **seeking.**



- What if currently head is at 30, next choice between 16 & 8
- 16 would have lesser seek time, but more rotation time. Maybe 8 is a better choice!
  - Called Shortest Positioning Time First

SCAN (or Elevator)

Sweep back and forth, from one end of disk to the other, serving requests as you go.

Pros/Cons?

SCAN

- Sweep back and forth, from one end of disk to the other, serving requests as you go.

- Pros
    - Doesn't cause starvation

- Cons
    - Favours middle tracks more

- Better: C-SCAN (circular scan)
    - Only sweep in one direction

- Another variant F-Scan (Freeze scan)
    - Freeze the request queue when doing a sweep

Work Conservation

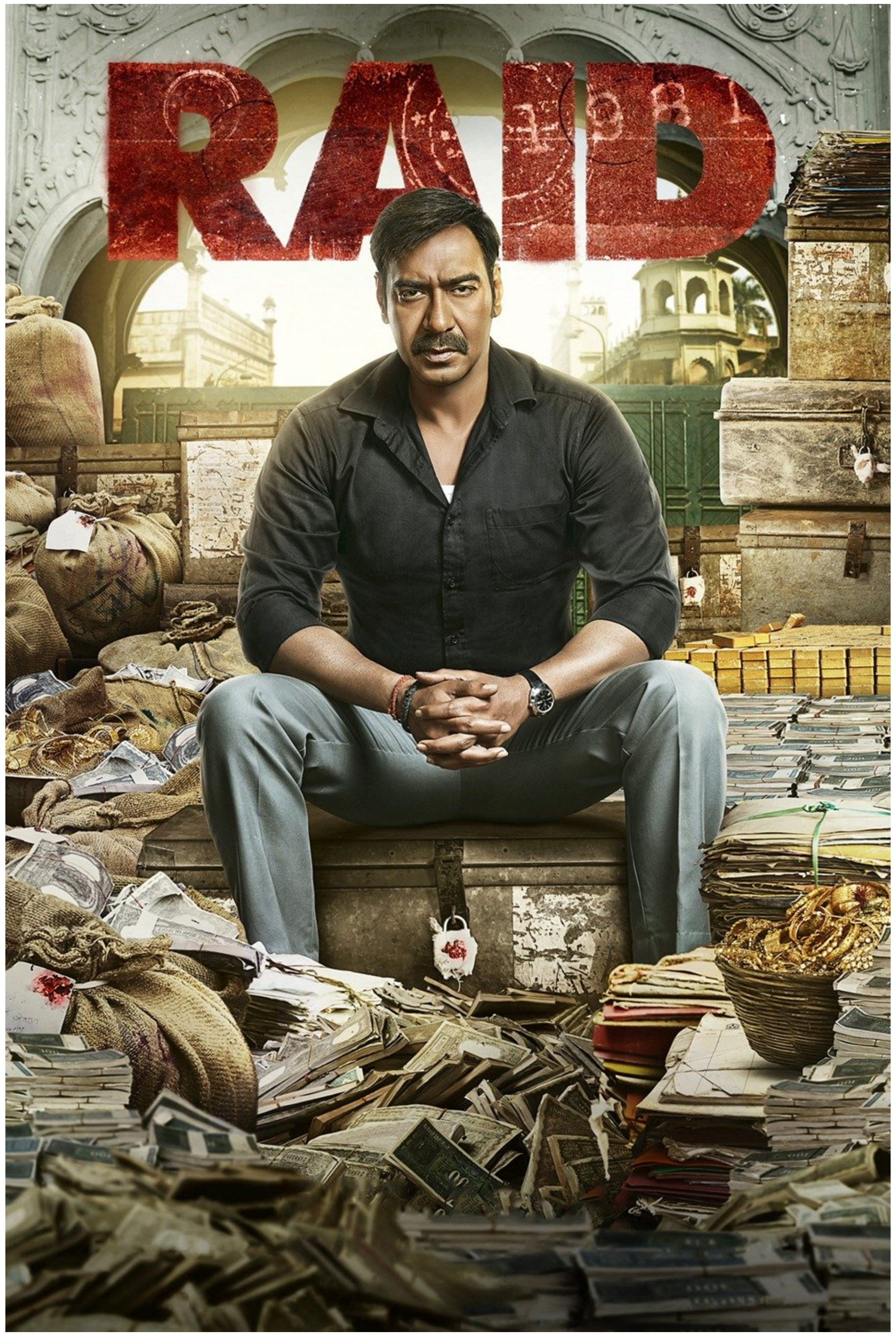
Work conserving schedulers always try to do I/O if there's I/O to be done.

Sometimes, it's better to wait instead if you anticipate another request will appear nearby.

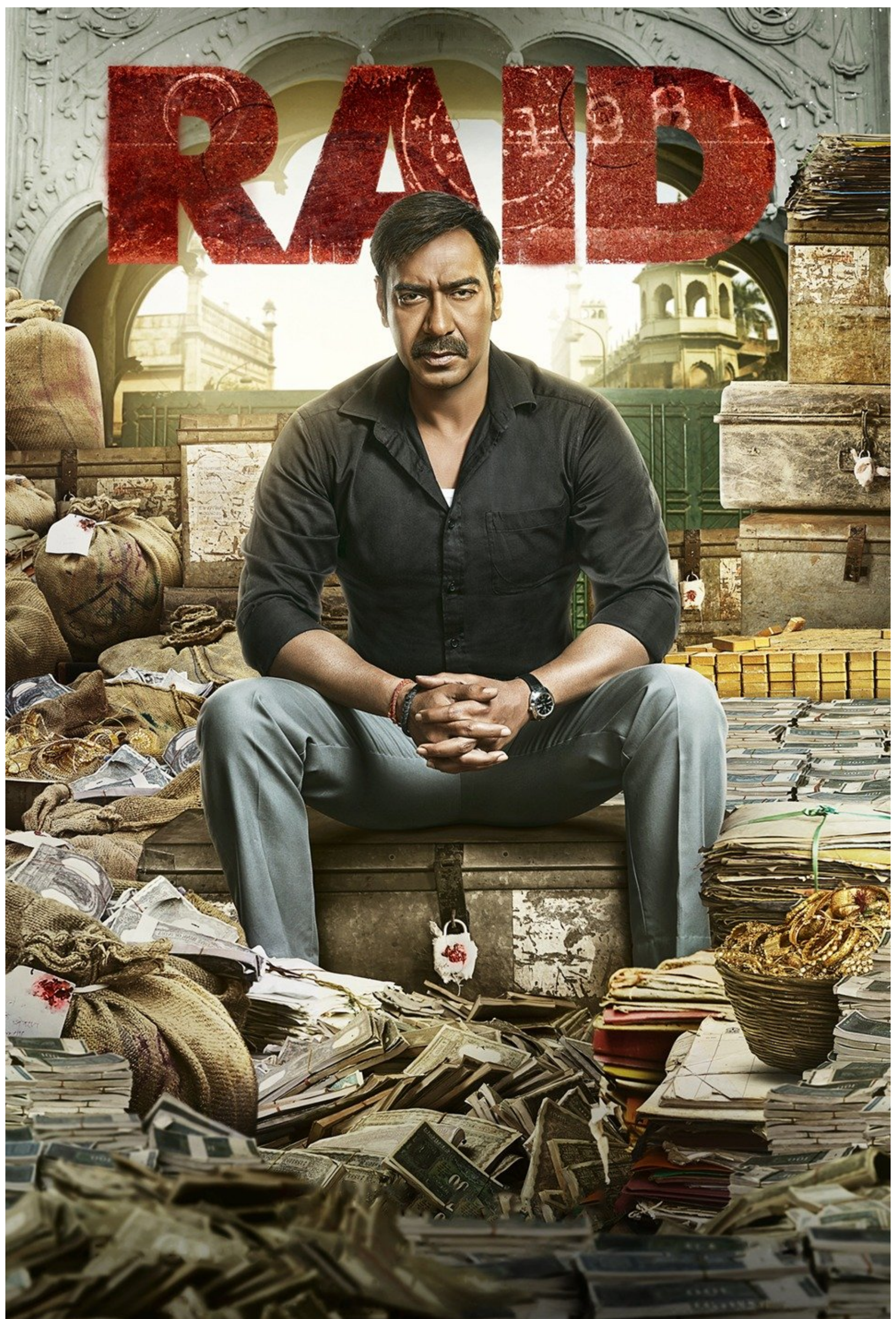
Such non-work-conserving schedulers are called anticipatory schedulers.

RAID

RAID

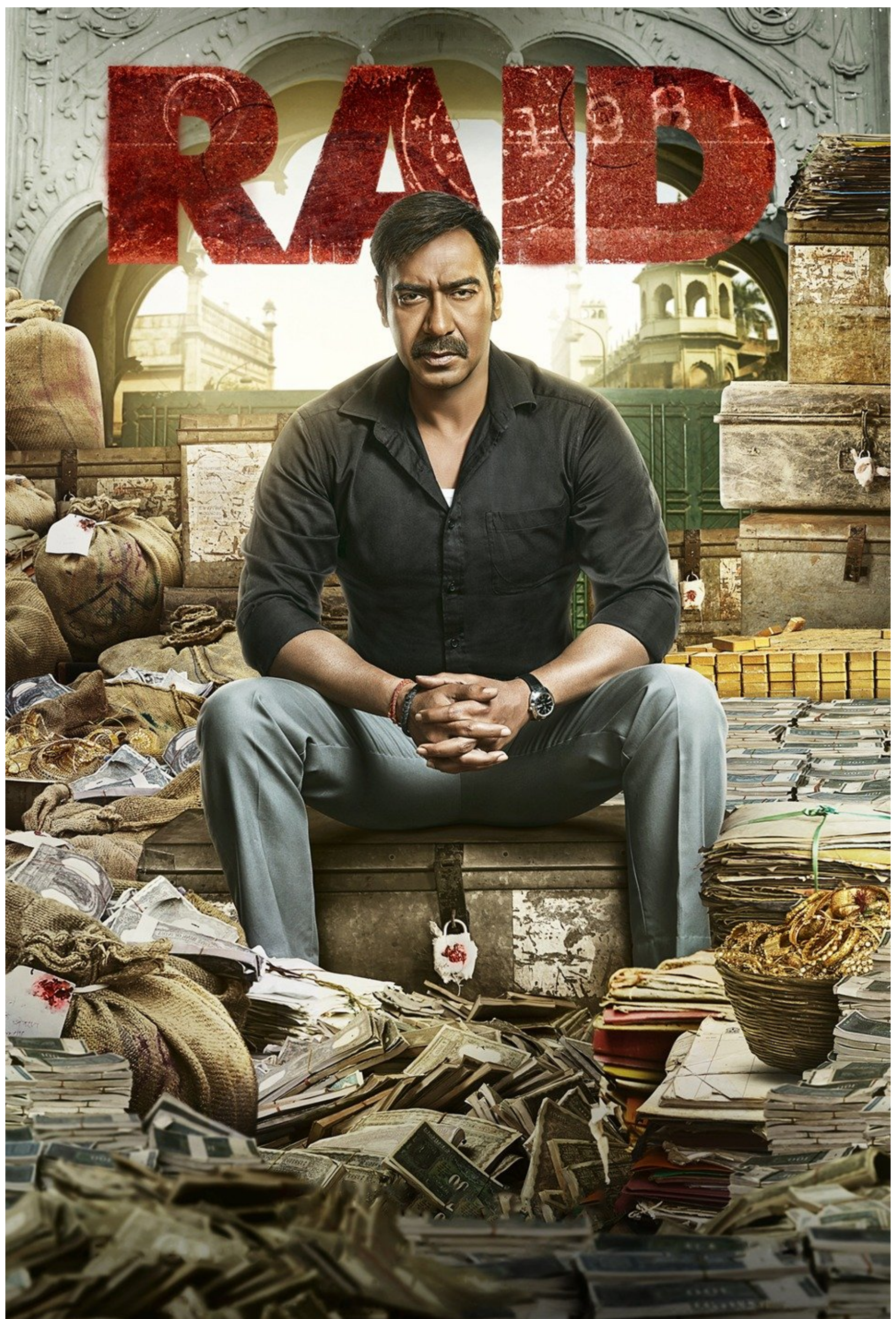


RAID


RAID


20oz VALUE SIZE
SC Johnson
A Family Company
Raid
Ant & Roach
KILLS ON CONTACT
ATTACK BUGS
No Lingering Chemical Odor
Lemon Scent
KEEP OUT OF REACH OF CHILDREN
CAUTION:
MANTENER FUERA DEL ALCANCE DE LOS NIÑOS
PRECAUCIÓN:

RAID


RAID


20oz VALUE SIZE
SC Johnson
A Family Company
Raid
Ant & Roach
KILLS ON CONTACT
ATTACK BUGS
No Lingering Chemical Odor
Lemon Scent
KEEP OUT OF REACH OF CHILDREN
CAUTION:
MANTENER FUERA DEL ALCANCE DE LOS NIÑOS
PRECAUCIÓN:
NET WT. 20 OZ.
(1.25 LB.) 567g


RocketStor

Only One Disk?

Sometimes we want many disks — why?

Only One Disk?

Only One Disk?

Sometimes we want many disks — why?

Only One Disk?

Sometimes we want many disks — why?
- capacity

Only One Disk?

Sometimes we want many disks — why?
- capacity
- performance

Only One Disk?

Sometimes we want many disks — why?

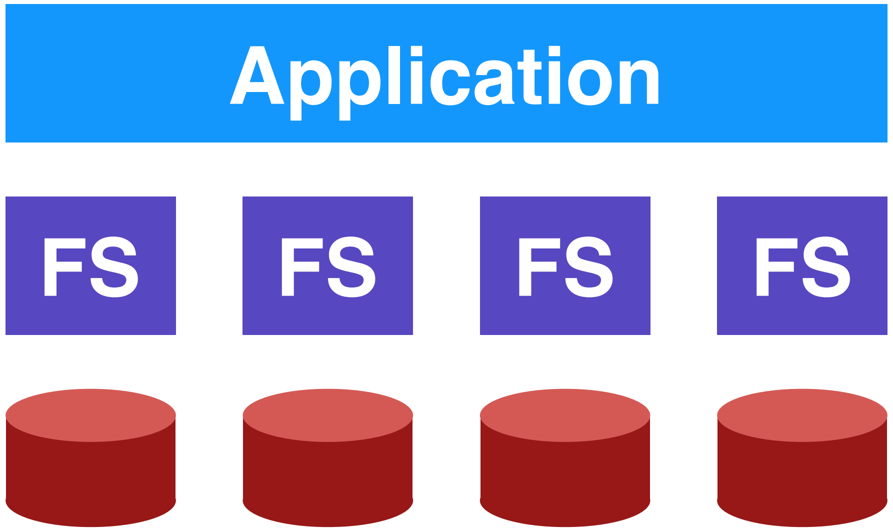
- capacity
- performance
- reliability

Only One Disk?

Sometimes we want many disks — why?
- capacity
- performance
- reliability

Challenge: most file systems work on only one disk.

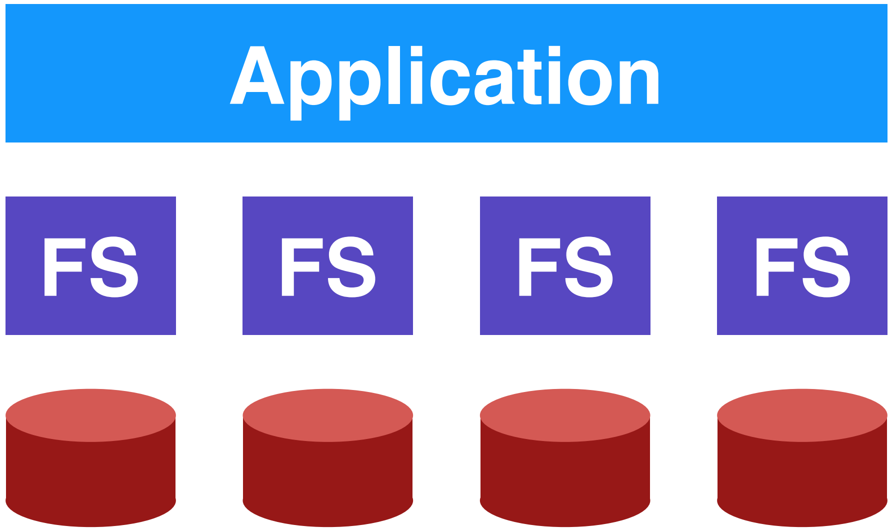
Solution 1: JBOD



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

Solution 1: JBOD

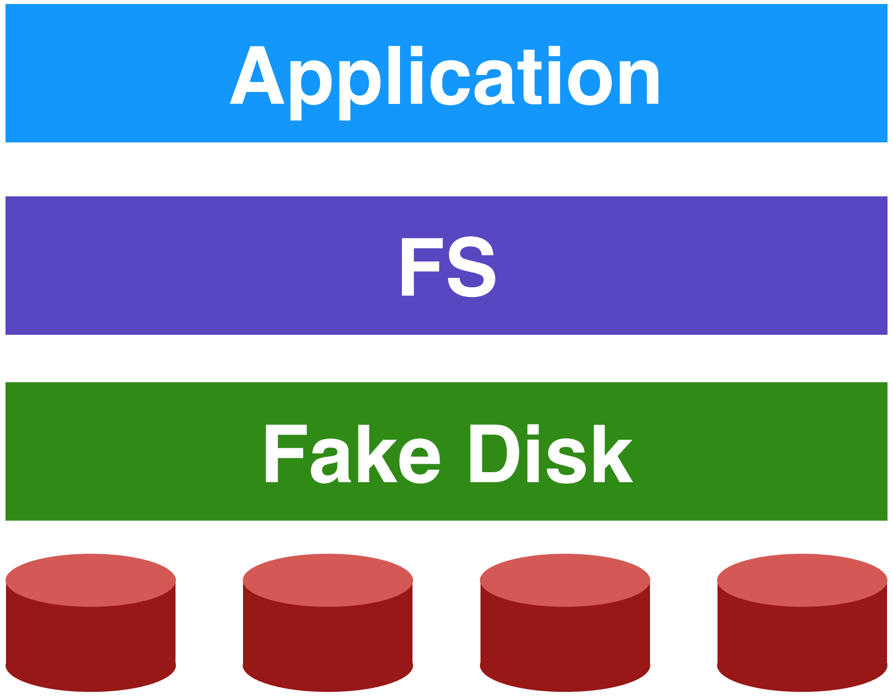
JBOD: **J**ust a **B**unch **O**f **D**isks



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

Solution 2: RAID

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

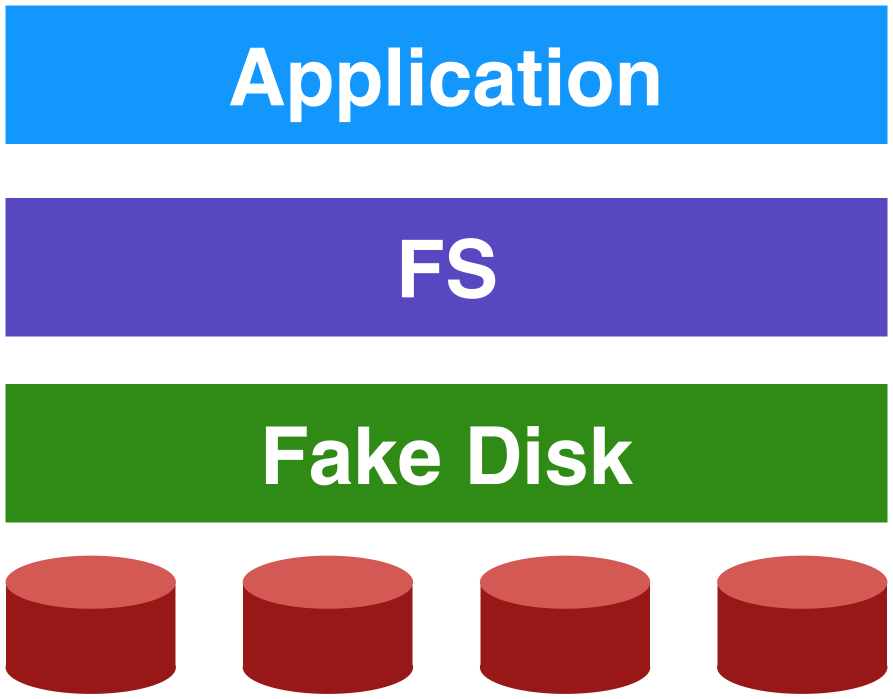


Build logical disk from many physical disks.

Solution 2: RAID

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

RAID is:
- transparent
- deployable

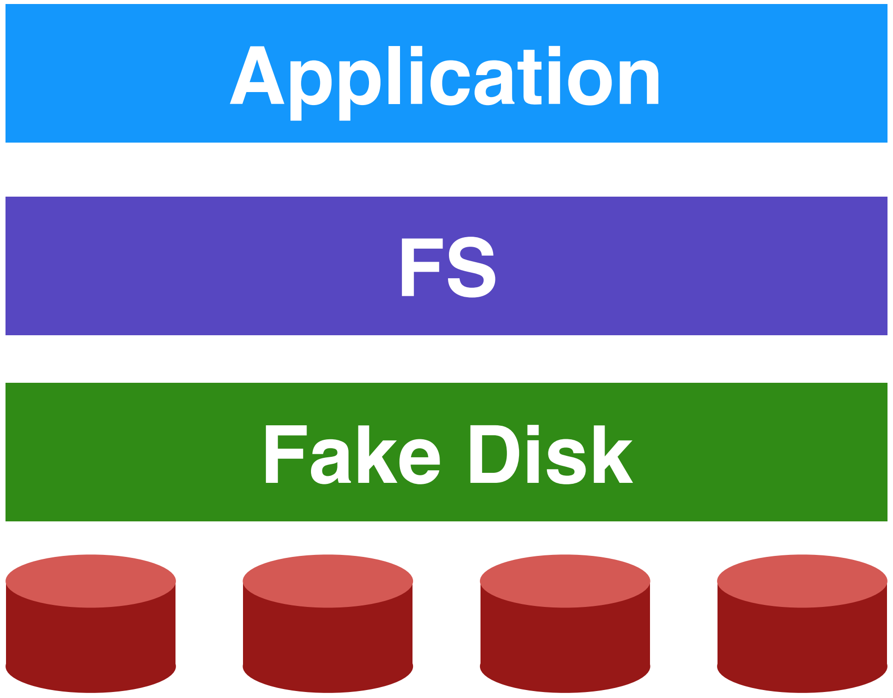


Build logical disk from many physical disks.

Solution 2: RAID

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

RAID is:
- transparent
- deployable



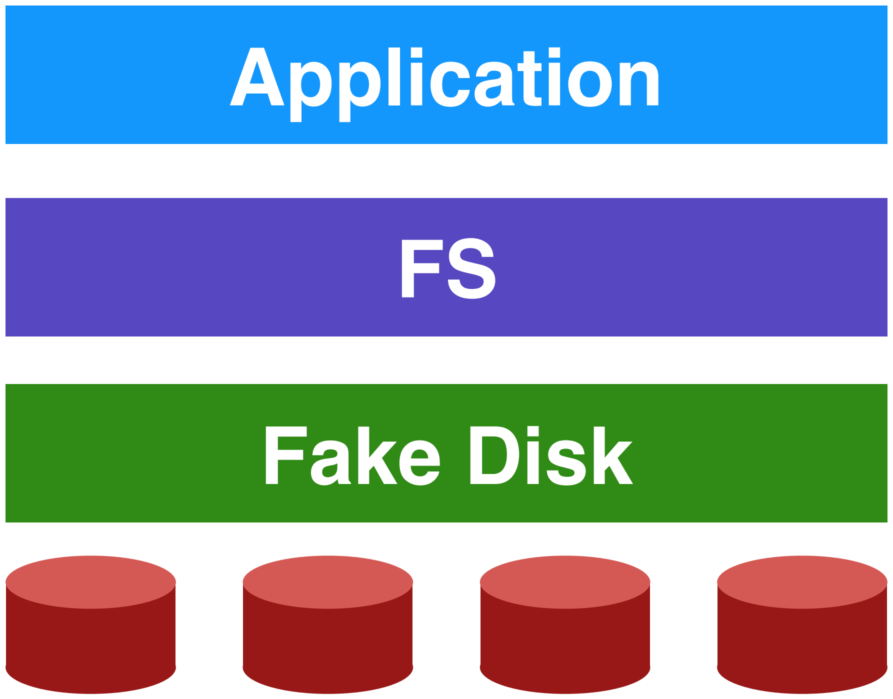
Logical disk gives

Build logical disk from many physical disks.

Solution 2: RAID

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

RAID is:
- transparent
- deployable



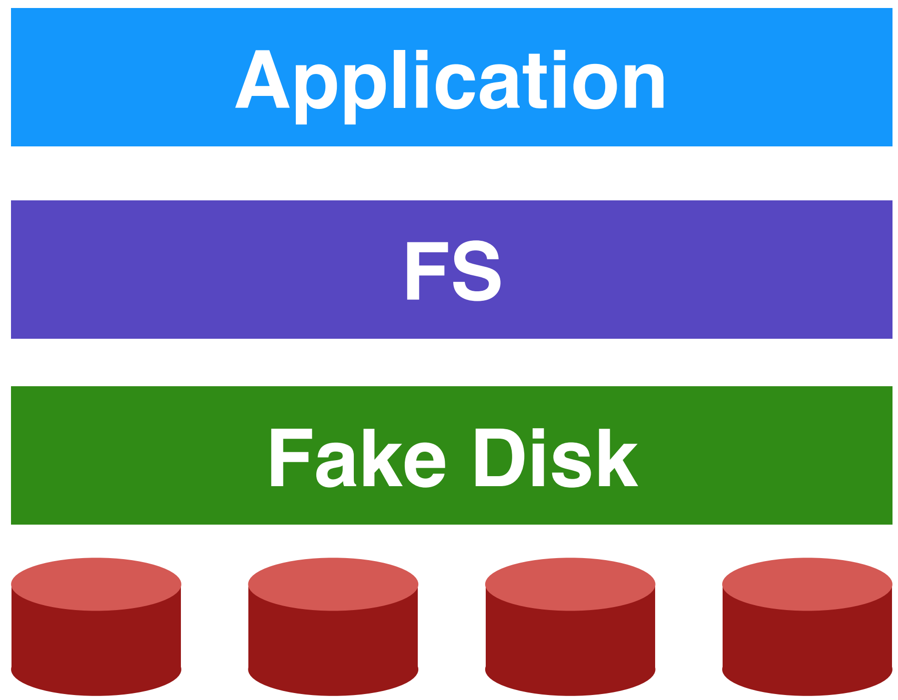
Logical disk gives
- capacity

Build logical disk from many physical disks.

Solution 2: RAID

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

RAID is:
- transparent
- deployable



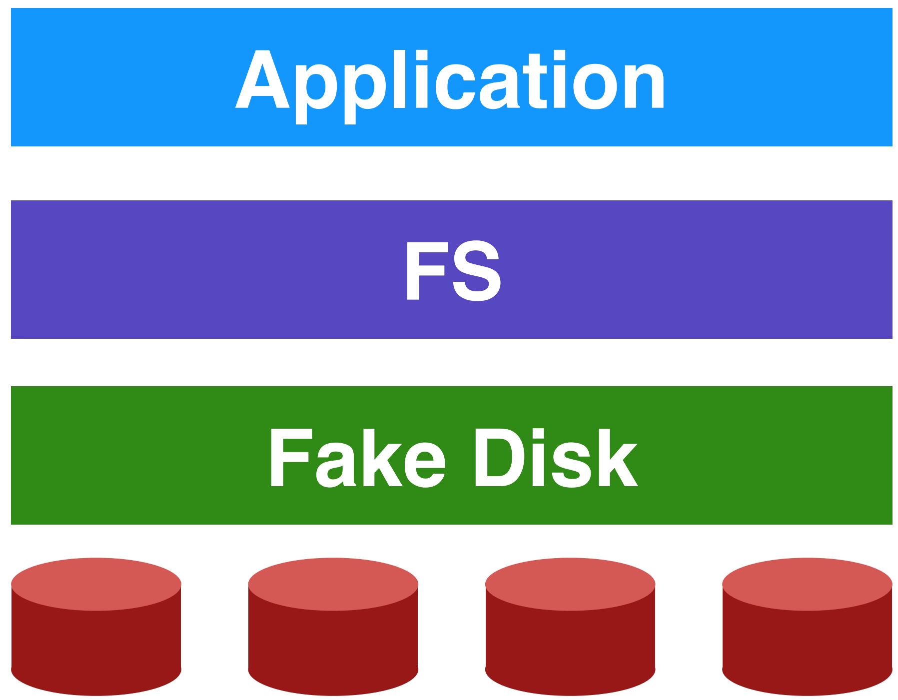
Logical disk gives
- capacity
- performance

Build logical disk from many physical disks.

Solution 2: RAID

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

RAID is:
- transparent
- deployable



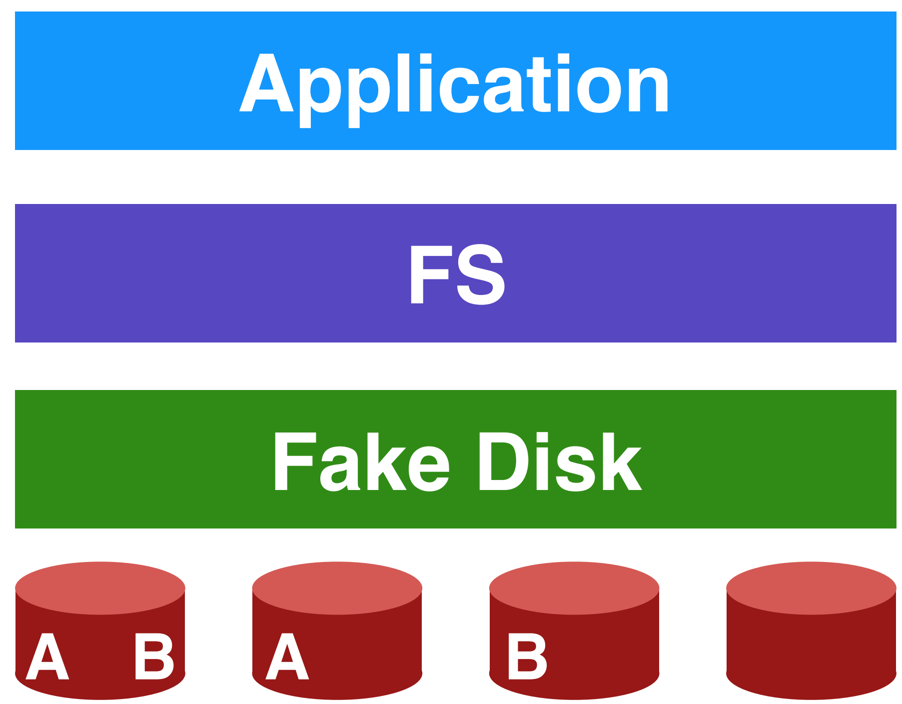
Logical disk gives
- capacity
- performance
- reliability?

Build logical disk from many physical disks.

Solution 2: RAID

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

RAID is:
- transparent
- deployable



Logical disk gives
- capacity
- performance
- reliability?

Build logical disk from many physical disks.

Why *Inexpensive* Disks?

Why *Inexpensive* Disks?

- Economies of scale! Cheap disks are popular.

Why *Inexpensive* Disks?

- Economies of scale! Cheap disks are popular.

Why *Inexpensive* Disks?

- Economies of scale! Cheap disks are popular.

- You can often get many commodity H/W components for the same price as a few expensive components.

Why *Inexpensive* Disks?

- Economies of scale! Cheap disks are popular.
- You can often get many commodity H/W components for the same price as a few expensive components.

Why *Inexpensive* Disks?

- Economies of scale! Cheap disks are popular.
- You can often get many commodity H/W components for the same price as a few expensive components.
- Strategy: write S/W to build high-quality logical devices from many cheap devices.

Why *Inexpensive* Disks?

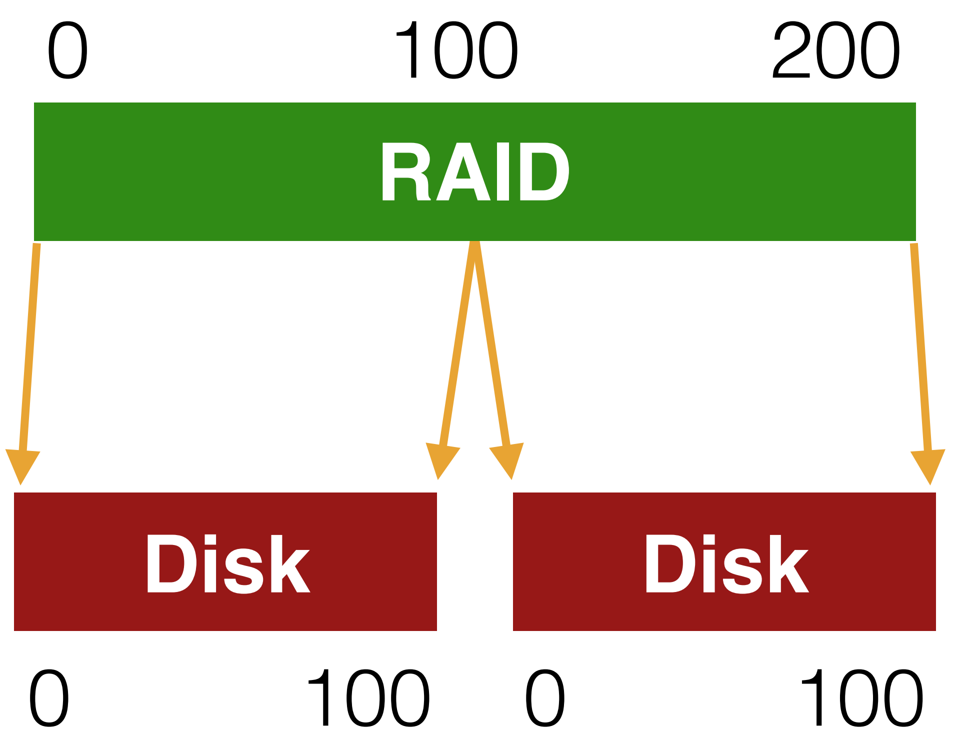
- Economies of scale! Cheap disks are popular.
- You can often get many commodity H/W components for the same price as a few expensive components.
- Strategy: write S/W to build high-quality logical devices from many cheap devices.

Why *Inexpensive* Disks?

- Economies of scale! Cheap disks are popular.
- You can often get many commodity H/W components for the same price as a few expensive components.
- Strategy: write S/W to build high-quality logical devices from many cheap devices.
- Alternative to RAID: buy an expensive, high-end disk.

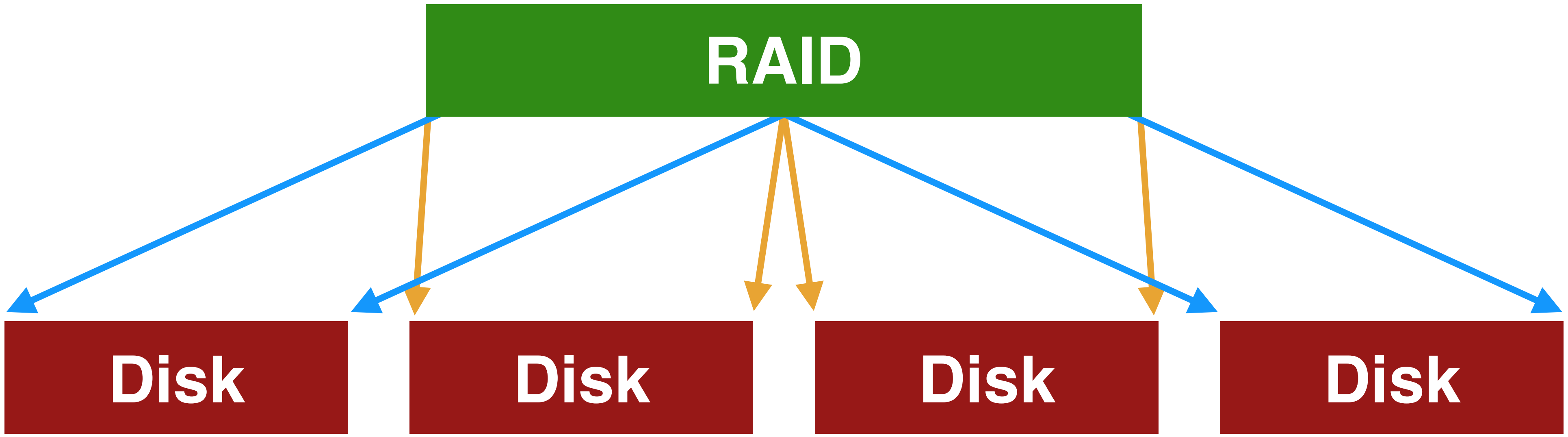
General Strategy

Build fast, large disk from smaller ones.



General Strategy

Add even more disks for reliability.



Mapping

How should we map logical to physical addresses?

How is this problem similar to virtual memory?

Mapping

How should we map logical to physical addresses?

How is this problem similar to virtual memory?

Dynamic mapping: use data structure (hash table, tree)
- paging

Static mapping: use math
- RAID

Mapping

How should we map logical to physical addresses?

How is this problem similar to virtual memory?

Dynamic mapping: use data structure (hash table, tree)
- paging

Static mapping: use math
- RAID

RAID volume is fixed-sized, dense

Redundancy

Redundancy: how many copies?

System engineers are always trying to increase or decrease redundancy.

Increase: replication (e.g., RAID)
Decrease: deduplication (e.g., code sharing)

Redundancy

Increase: improves reliability

Decrease: improves space efficiency

One strategy: reduce redundancy as much is possible. Then add back just the right amount.

Reasoning About RAID

Workload: types of reads/writes issued by app

RAID: system for mapping logical to physical addrs

Metric: capacity, reliability, performance

RAID "algebra", given 2 variables, find the 3

$$f(\mathbf{W}, \mathbf{R}) = \mathbf{M}$$

RAID Decisions

Which logical blocks map to which physical blocks?

How do we use extra physical blocks (if any)?

Workloads

Workloads

- Sequential loads

Workloads

- Sequential loads
  - transfer dominates, very little seeking and rotation

Workloads

- Sequential loads
  - transfer dominates, very little seeking and rotation
  - Disk speed: S MB/s

Workloads

- Sequential loads
  - transfer dominates, very little seeking and rotation
  - Disk speed: S MB/s

Workloads

- Sequential loads
  - transfer dominates, very little seeking and rotation
  - Disk speed: S MB/s

Workloads

- Sequential loads
  - transfer dominates, very little seeking and rotation
  - Disk speed: S MB/s

- Random loads

Workloads

- Sequential loads
  - transfer dominates, very little seeking and rotation
  - Disk speed: S MB/s
- Random loads
  - seeking and rotation have significant cost

Workloads

- Sequential loads
  - transfer dominates, very little seeking and rotation
  - Disk speed: S MB/s

- Random loads
  - seeking and rotation have significant cost
  - Disk speed: R MB/s

Workloads

- Sequential loads
  - transfer dominates, very little seeking and rotation
  - Disk speed: S MB/s

- Random loads
  - seeking and rotation have significant cost
  - Disk speed: R MB/s
  - R << S

Workloads

Workloads

- Average seek time - 7 ms

Workloads

- Average seek time - 7 ms
- Average rotational time - 3 ms

Workloads

- Average seek time - 7 ms
- Average rotational time - 3 ms
- Transfer rate of disk - 50 MB/s

Workloads

- Average seek time - 7 ms
- Average rotational time - 3 ms
- Transfer rate of disk - 50 MB/s

Workloads

- Average seek time - 7 ms
- Average rotational time - 3 ms
- Transfer rate of disk - 50 MB/s

- Sequential load

Workloads

- Average seek time - 7 ms
- Average rotational time - 3 ms
- Transfer rate of disk - 50 MB/s

- Sequential load
  - Transfer Size: 10 MB

Workloads

- Average seek time - 7 ms
- Average rotational time - 3 ms
- Transfer rate of disk - 50 MB/s

- Sequential load
  - Transfer Size: 10 MB
  - S = 10 MB/(7+3+(10/50)*1000)ms ~48 MB/s

Workloads

- Average seek time - 7 ms
- Average rotational time - 3 ms
- Transfer rate of disk - 50 MB/s

- Sequential load
  - Transfer Size: 10 MB
  - S = 10 MB/(7+3+(10/50)*1000)ms ~48 MB/s

Workloads

- Average seek time - 7 ms
- Average rotational time - 3 ms
- Transfer rate of disk - 50 MB/s

- Sequential load
  - Transfer Size: 10 MB
  - S = 10 MB/(7+3+(10/50)*1000)ms ~48 MB/s

- Random loads

Workloads

- Average seek time - 7 ms
- Average rotational time - 3 ms
- Transfer rate of disk - 50 MB/s

- Sequential load
  - Transfer Size: 10 MB
  - S = 10 MB/(7+3+(10/50)*1000)ms ~48 MB/s

- Random loads
  - Transfer Size: 10 KB

Workloads

- Average seek time - 7 ms
- Average rotational time - 3 ms
- Transfer rate of disk - 50 MB/s

- Sequential load
  - Transfer Size: 10 MB
  - S = 10 MB/(7+3+(10/50)*1000)ms ~48 MB/s

- Random loads
  - Transfer Size: 10 KB
  - R = 10 KB/(7+3+(10KB/50MB/s)) ~1 MB/s

Metrics

Capacity: how much space can apps use?

Reliability: how many disks can we safely lose?

Performance: how long does each workload take?

Metrics

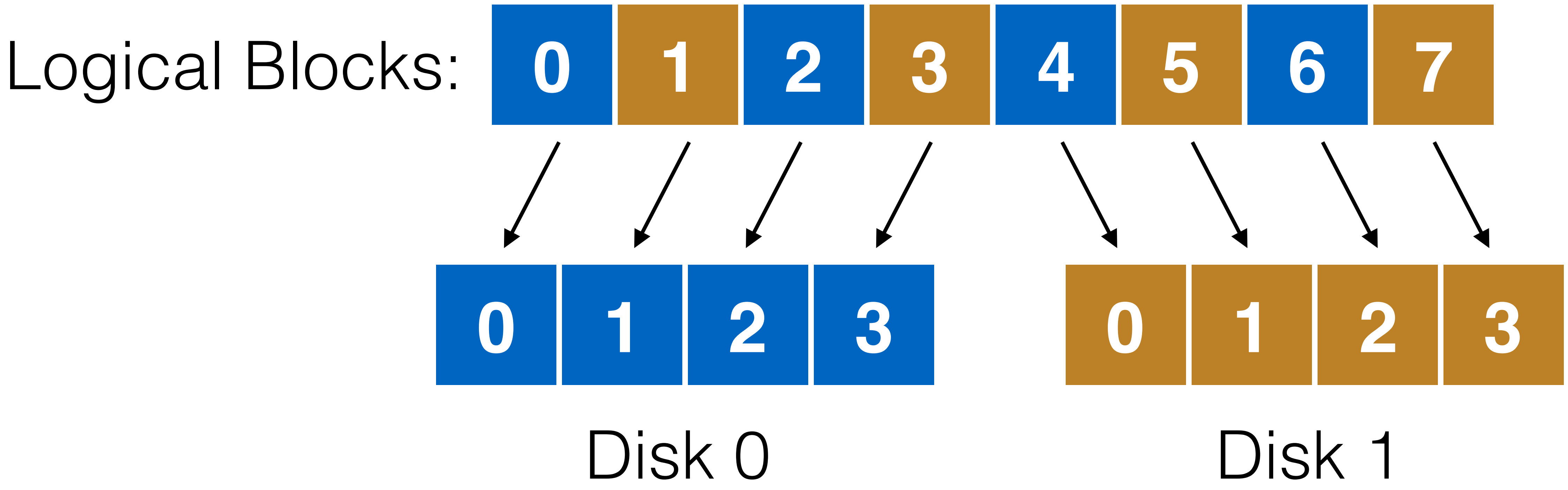
Capacity: how much space can apps use?

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

Performance: how long does each workload take?

RAID-0: Striping

Optimize for capacity. No redundancy (weird name).



Another View

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

4 disks

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

4 disks

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

How to Map

Given logical address A, find:
Disk = A % disk_count
Offset = A / disk_count

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

Chunk Size = 1

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

Chunk Size = 2

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

Chunk Size = 2

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

Chunk Size = 2

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

stripe: (rows 8–15)

Chunk Size = 2

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

We'll assume chunk size of 1 for today.
Sizes of 64KB are typical in deployment.

RAID-0: Analysis

RAID-0: Analysis

- What is capacity?

RAID-0: Analysis

- What is capacity?
  - N * C

RAID-0: Analysis

- What is capacity?
  - N * C
- How many disks can fail?

RAID-0: Analysis

- What is capacity?
  - N * C
- How many disks can fail?
  - 0

RAID-0: Analysis

- What is capacity?
  - N * C
- How many disks can fail?
  - 0
- Throughput?

RAID-0: Analysis

- What is capacity?
  - N * C
- How many disks can fail?
  - 0
- Throughput?
  - Sequential read and write — N*S

RAID-0: Analysis

- What is capacity?
  - N * C
- How many disks can fail?
  - 0
- Throughput?
  - Sequential read and write — N*S
  - Random read and write — N*R

RAID-0: Analysis

- What is capacity?
  - N * C
- How many disks can fail?
  - 0
- Throughput?
  - Sequential read and write — N*S
  - Random read and write — N*R
  - Latency?

RAID-0: Analysis

- What is capacity?
  - N * C
- How many disks can fail?
  - 0
- Throughput?
  - Sequential read and write — N*S
  - Random read and write — N*R
  - Latency?
    - D

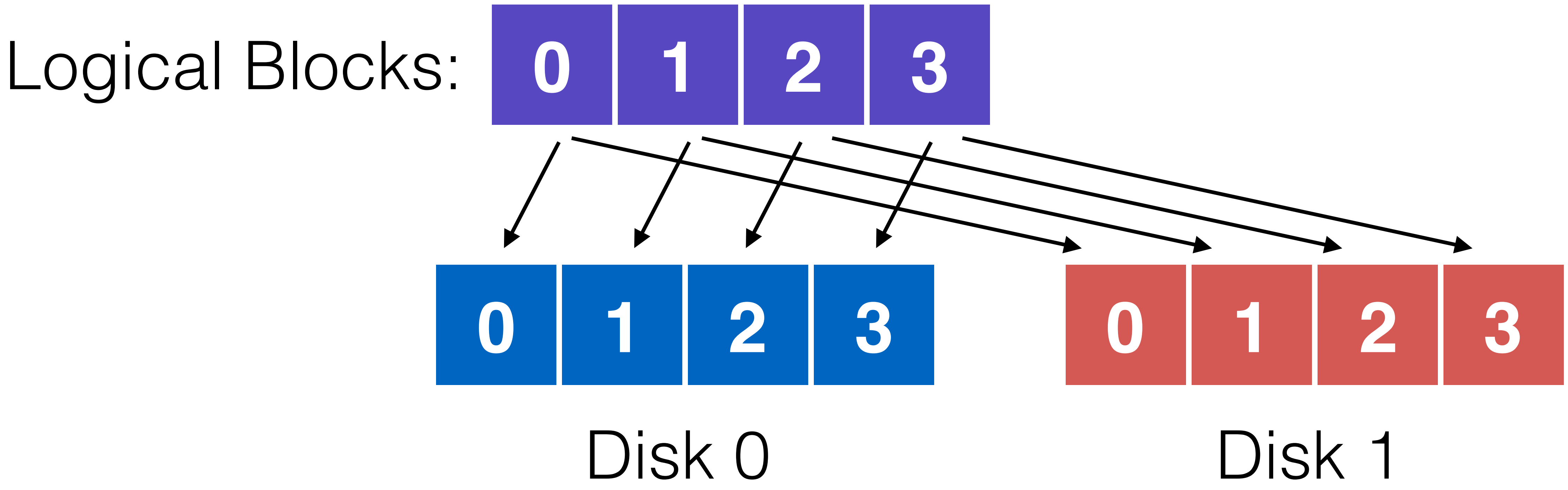
RAID-0: Analysis

- What is capacity?
  - N * C
- How many disks can fail?
  - 0
- Throughput?
  - Sequential read and write — N*S
  - Random read and write — N*R
  - Latency?
    - D

Buying more disks improves throughput, but not latency!

RAID-1: Mirroring

Keep two copies of all data.



Assumptions

Assume disks are **fail-stop**.
- they work or they don't
- we know when they don't

Tougher Errors:
- latent sector errors
- silent data corruption

2 disks

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

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

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

How many disks can fail?

RAID-1: Analysis

RAID-1: Analysis

- What is capacity?

RAID-1: Analysis

- What is capacity?
    - N/2 * C

RAID-1: Analysis

- What is capacity?
  - N/2 * C
- How many disks can fail?

RAID-1: Analysis

- What is capacity?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)

RAID-1: Analysis

- What is capacity?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?

RAID-1: Analysis

- What is capacity?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S

RAID-1: Analysis

- What is capacity?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S

RAID-1: Analysis

- What is capacity?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R

RAID-1: Analysis

- What is capacity?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)

RAID-1: Analysis

- What is capacity?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)
- Latency

RAID-1: Analysis

- What is capacity?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)
- Latency
  - D

RAID-1: Analysis

- What is capacity?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)
- Latency
  - D

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?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)
- Latency
  - D

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?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)
- Latency
  - D

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?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)
- Latency
  - D

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?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)
- Latency
  - D

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?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)
- Latency
  - D

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?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)
- Latency
  - D

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?
  - N/2 * C
- How many disks can fail?
  - 1 or N/2 (best case)
- Throughput?
  - Sequential write — (N/2)*S
  - Sequential read — (N/2)*S
  - Random write — (N/2)*R
  - Random read — (N*R)
- Latency
  - D