# Announcements

2nd Exam: Grades should be ready tomorrow

P3: Due last night

- Congratulations on finishing 3 out of 5!

P4: Threads (Part a and b) available soon…

- Can choose or be matched with new partner

Read as we go along!

- Chapter 38

UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 537
Introduction to Operating Systems

Andrea C. Arpaci-Dusseau
Remzi H. Arpaci-Dusseau

# Persistence: RAID

**Questions answered in this lecture:**

Why more than one disk?

What are the different RAID levels? (striping, mirroring, parity)

Which RAID levels are best for reliability? for capacity?

Which are best for performance? (sequential vs. random reads and writes)

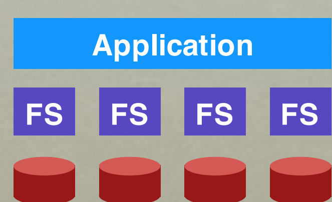
# Only One Disk?

Sometimes we want many disks — why?

- capacity
- reliability
- Performance

Challenge: most file systems work on only one disk

# Solution 1: JBOD



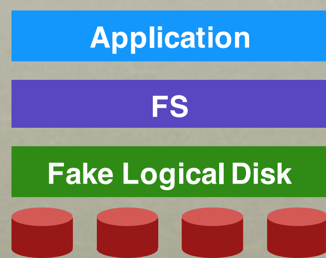

Application is smart, stores different files on different file systems.

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

# SOLUTION 2: RAID

RAID is:

- transparent
- deployable




Logical disk gives

- capacity
- performance
- reliability

Build logical disk from many physical disks.

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

# WHY *INEXPENSIVE* DISKS?

Economies of scale! Commodity disks cost less

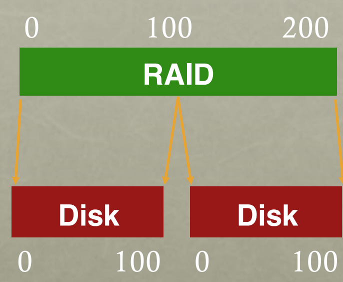
Can buy many commodity H/W components for the same price as few high-end components

Strategy: write S/W to build high-quality logical devices from many cheap devices

Alternative to RAID: buy an expensive, high-end disk

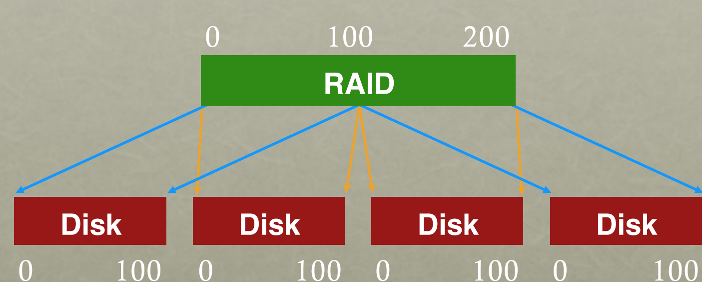
# General Strategy: Mapping

Build fast, large disk from smaller ones.




# General Strategy: Redundancy

Add even more disks for reliability.



# MAPPING

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

- Some similarity to virtual memory

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

- page tables

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

- RAID

# REDUNDANCY

Trade-offs to amount of redundancy

Increase number of copies:

- improves reliability (and maybe performance)

Decrease number of copies (deduplication)

- improves space efficiency

# Reasoning about RAID

**RAID**: system for mapping logical to physical blocks

**Workload**: types of reads/writes issued by applications (sequential vs. random)

**Metric**: capacity, reliability, performance

# RAID Decisions

Which logical blocks map to which physical blocks?

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

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

# WORKLOADS

- Reads
  - One operation
  - Steady-state I/O
    - Sequential
    - Random
- 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!)

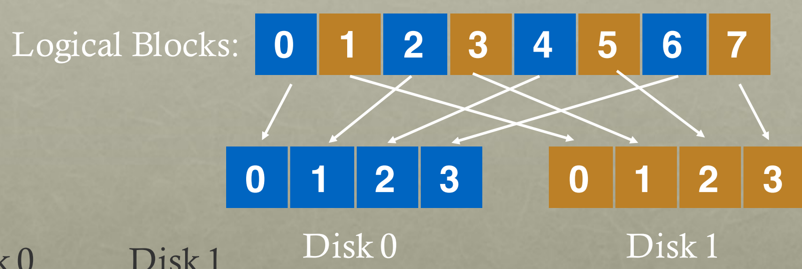
**Performance**: how long does each workload take?

Normalize each to characteristics of one disk

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-0: STRIPING

Optimize for capacity. No redundancy




| 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 |

Given logical address A, find:
Disk = …
Offset = …

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

# Chunk Size

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 |
| stripe: | 8 | 10 | 12 | 14 |
| | 9 | 11 | 13 | 15 |

assume chunk size of 1

# RAID-0: Analysis

What is capacity? N * C

How many disks can fail? 0

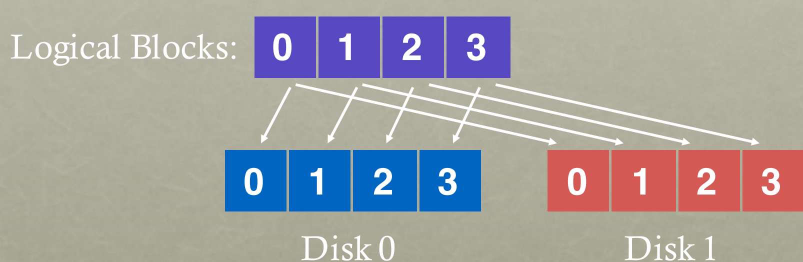
Latency D

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

Buying more disks improves throughput, but not latency!

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: Mirroring




Keep two copies of all data.

# RAID-1 Layout

2 disks

| Disk 0 | Disk 1 |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

4 disks

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

# RAID-1: 4 Disks

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

How many disks can fail?

Assume disks are **fail-stop**.
- each disk works or it doesn't
- system knows when disk fails

Tougher Errors:
- latent sector errors
- silent data corruption

# RAID-1: Analysis

What is capacity? **N/2 * C**

How many disks can fail? **1 (or maybe N / 2)**

Latency (read, write)? **D**

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

- sequential reads?
- sequential writes?
- random reads?
- random writes?

# RAID-1: Throughput

What is steady-state throughput for

- random reads? **N * R**
- random writes? **N/2 * R**
- sequential writes? **N/2 * S**
- sequential reads? **Book: N/2 * S (other models: N * S)**

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

# Crashes

| | Disk0 | Disk1 |
|---|---|---|
| 0 | A | A |
| 1 | B | B |
| 2 | C | C |
| 3 | D | D |

# CRASHES

| | Disk0 | Disk1 |
|---|---|---|
| 0 | A | A |
| 1 | B | B |
| 2 | C | C |
| 3 | D | D |

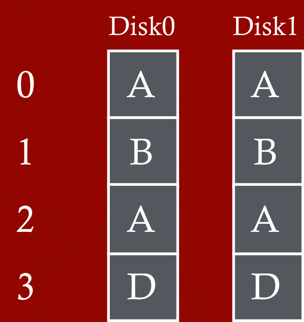
write(A) to 2

# CRASHES

| | Disk0 | Disk1 |
|---|---|---|
| 0 | A | A |
| 1 | B | B |
| 2 | A | C |
| 3 | D | D |

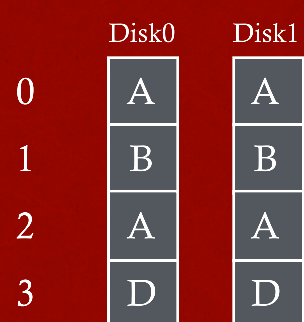
write(A) to 2

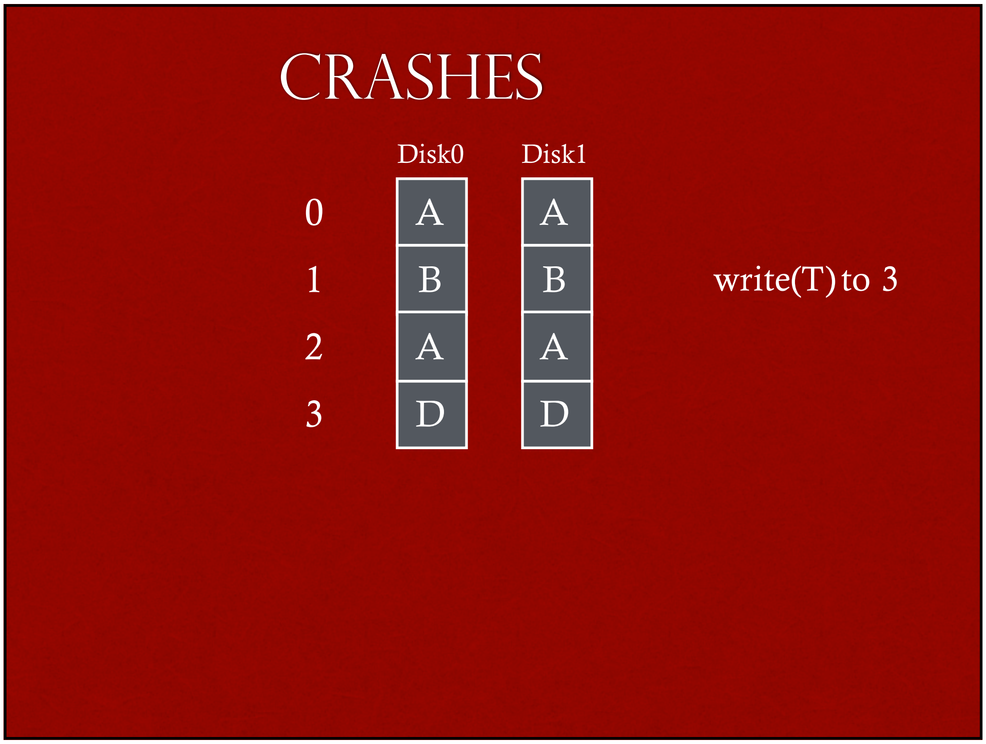
# CRASHES

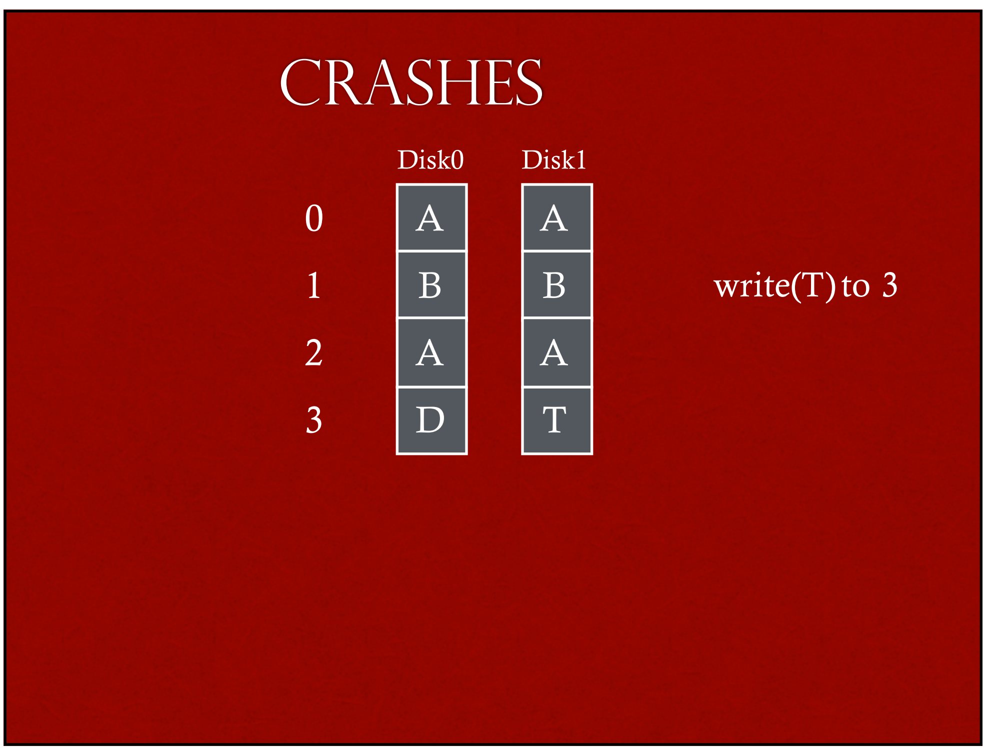


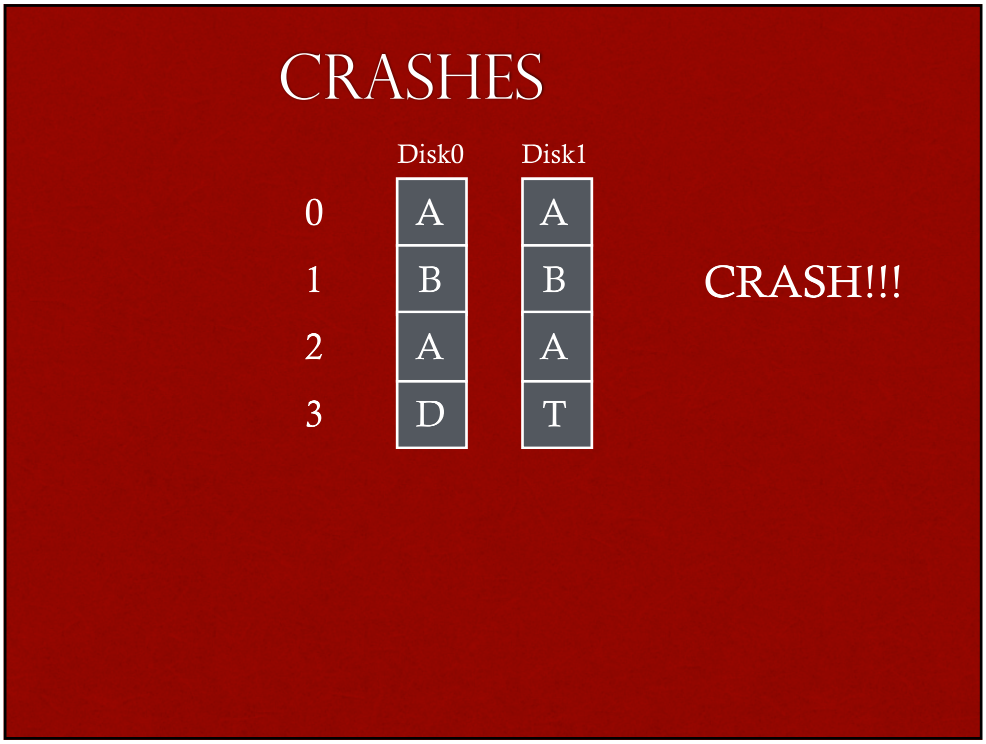

write(A) to 2

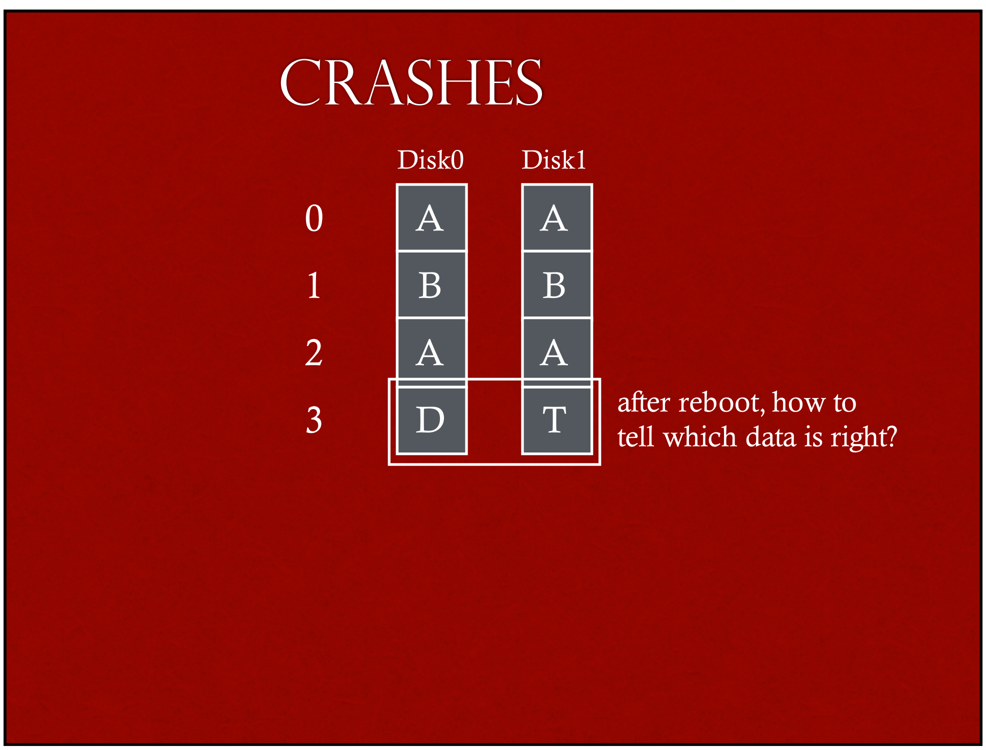
# CRASHES




CRASHES
Disk0
Disk1
0
1
2
3
A
B
A
D
A
B
A
D
write(T) to 3


CRASHES
Disk0
Disk1
0
1
2
3
A
B
A
D
A
B
A
T
write(T) to 3


CRASHES
Disk0
Disk1
0
1
2
3
A
B
A
D
A
B
A
T
CRASH!!!

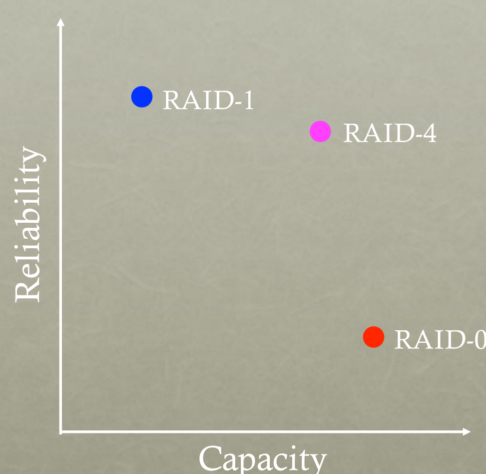

CRASHES
Disk0
Disk1
0
1
2
3
A
B
A
D
A
B
A
T
after reboot, how to
tell which data is right?

# H/W SOLUTION

Problem: Consistent-Update Problem

Use non-volatile RAM in RAID controller.

Software RAID controllers (e.g., Linux md) don't have this option



# RAID-4 Strategy

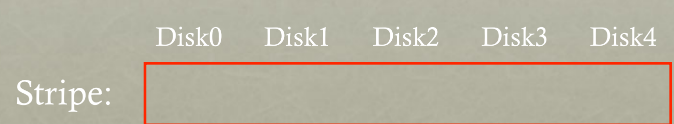
Use parity disk

In algebra, if an equation has N variables, and N-1 are known, you can often 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.

# Example



# EXAMPLE

| | Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|---|
| Stripe: | | | | | |
| | | | | | (parity) |

# EXAMPLE

| | Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|---|
| Stripe: | 5 | 3 | 0 | 1 | |
| | | | | | (parity) |

# EXAMPLE

| | Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|---|
| Stripe: | 5 | 3 | 0 | 1 | 9 |
| | | | | | (parity) |

# EXAMPLE

| | Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|---|
| Stripe: | 5 | X | 0 | 1 | 9 |
| | | | | | (parity) |

# EXAMPLE

| | Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|---|
| Stripe: | 5 | 3 | 0 | 1 | 9 |
| | | | | | (parity) |

# EXAMPLE

| | Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|---|
| Stripe: | 2 | 1 | 1 | X | 5 |
| | | | | | (parity) |

# EXAMPLE

| | Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|---|
| Stripe: | 2 | 1 | 1 | 1 | 5 |
| | | | | | (parity) |

# EXAMPLE

| | Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|---|
| Stripe: | 3 | 0 | 1 | 2 | X |
| | | | | | (parity) |

# Example

| | Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|---|
| Stripe: | 3 | 0 | 1 | 2 | 6 |
| | | | | | (parity) |

Which functions are used to compute parity?

# RAID-4: Analysis

What is capacity? **(N-1) * C**

How many disks can fail? **1**

Latency (read, write)? **D, 2*D (read and write parity disk)**

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

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? **(N-1) * S**
- sequential writes? **(N-1) * S**
- random reads? **(N-1) * R**
- random writes? **R/2 (read and write parity disk)**

how to avoid parity bottleneck?

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

# RAID-5

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

...

Rotate parity across different disks

# RAID-5: Analysis

What is capacity? **(N-1) * C**

How many disks can fail? **1**

Latency (read, write)? **D, 2*D (read and write parity disk)**

Same as RAID-4…

| 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

Steady-state throughput for RAID-4:

- sequential reads? **(N-1) * S**
- sequential writes? **(N-1) * S**
- random reads? **(N-1) * R**
- random writes? **R/2 (read and write parity disk)**

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

What is steady-state throughput for RAID-5?

- sequential reads? **(N-1) * S**
- sequential writes? **(N-1) * S**
- random reads? **(N) * R**
- random writes? **N * R/4**

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

…

# RAID Level Comparisons

| | Reliability | Capacity |
|---|---|---|
| RAID-0 | 0 | C*N |
| RAID-1 | 1 | C*N/2 |
| RAID-4 | 1 | (N-1) * C |
| RAID-5 | 1 | (N-1) * C |

# RAID Level Comparisons

| | Read Latency | Write Latency |
|---|---|---|
| RAID-0 | D | D |
| RAID-1 | D | D |
| RAID-4 | D | 2D |
| RAID-5 | D | 2D |

# RAID LEVEL COMPARISONS

| | Seq Read | Seq Write | Rand Read | Rand Write |
|---|---|---|---|---|
| RAID-0 | N * S | N * S | N * R | N * R |
| RAID-1 | N/2 * S | N/2 * S | N * R | N/2 * R |
| RAID-4 | (N-1)*S | (N-1)*S | (N-1)*R | R/2 |
| RAID-5 | (N-1)*S | (N-1)*S | N * R | N/4 * R |

RAID-5 is strictly better than RAID-4

# RAID LEVEL COMPARISONS

| | Seq Read | Seq Write | Rand Read | Rand Write |
|---|---|---|---|---|
| RAID-0 | N * S | N * S | N * R | N * R |
| RAID-1 | N/2 * S | N/2 * S | N * R | N/2 * R |
| RAID-5 | (N-1)*S | (N-1)*S | N * R | N/4 * R |

RAID-0 is always fastest and has best capacity (but at cost of reliability)

RAID-5 better than RAID-1 for sequential workloads

RAID-1 better than RAID-5 for random workloads

# Summary

Many engineering tradeoffs with RAID

- capacity, reliability, performance for different workloads

Block-based interface:

Very deployable and popular storage solution due to transparency