PERSISTENCE: RAID

Andrea Arpaci-Dusseau

CS 537, Fall 2019

ADMINISTRIVIA

Project 5: Due yesterday

- Working on grading…

Project 6: MapReduce with locking (not xv6) – Avail Thu or Fri

Midterm 2: Nov 11/6 (Wed) from 7:30-9:30pm

- Practice exams available
- Discussion section: Midterm Review
- Rooms:
- Mostly Concurrency
  + Some Virtualization (usually repeated from Midterm1)
- No Persistence
- Lab Hours -> Office Hours in CS 1207

LEARNING OUTCOMES

Why use 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)

PERSISTENCE

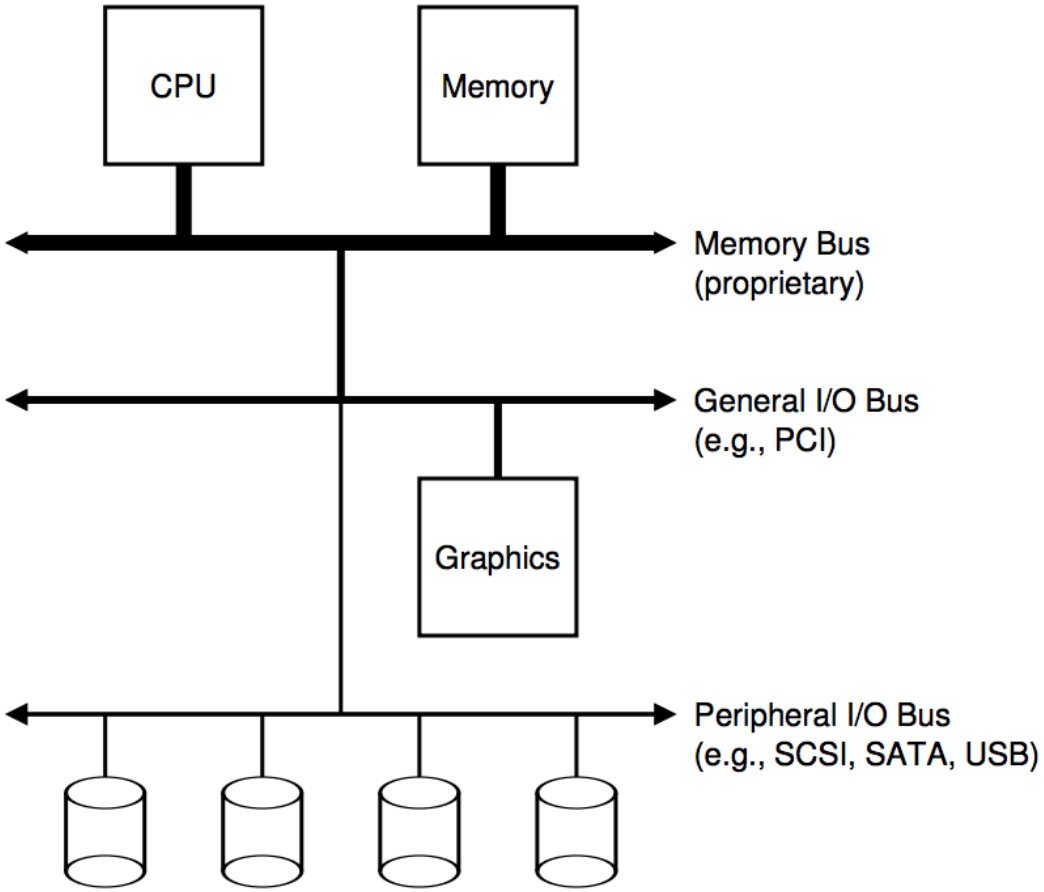
How to ensure data is available across reboots

- even after power outages, hardware failure, system crashes?

Topics:

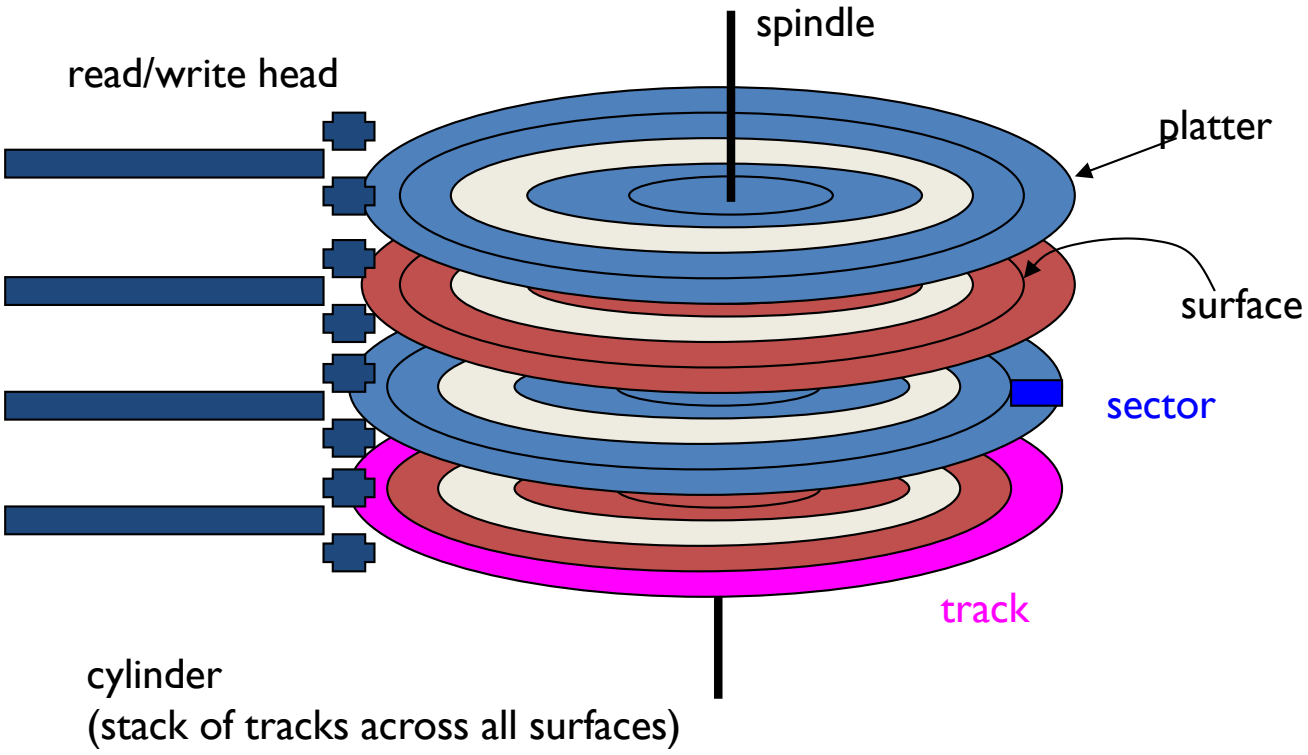
- Persistent storage devices (HDDs, RAID, SSDs)
- File API for processes
- FS implementation (meta-data structures, allocation policies)
- Crash recovery (journaling)
- Advanced Topics: Distributed systems?

HARDWARE SUPPORT FOR I/O



Hierarchical buses

DISK TERMINOLOGY



REVIEW POINTS

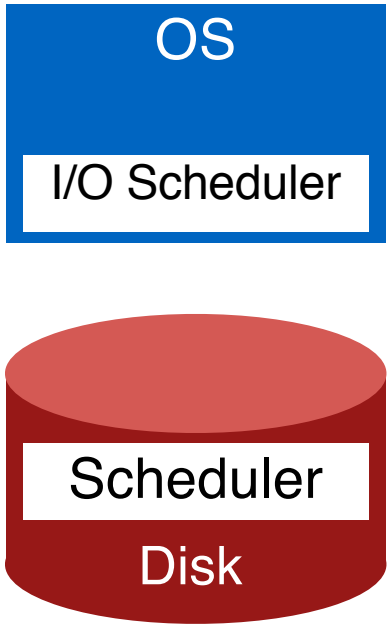
Disks: Linear array of sectors (512 bytes)
IO Time = Seek + Rotation + Transfer
Sequential bandwidth >> Random
Low-level details of disks:

- Track skew
- Zones
- Buffering/caching on disk

Disk Scheduling

- FCFS
- SSTF (shortest seek time first)
- SPTF (shortest positioning time first – only disk knows)
- Dealing with starvation
  - Elevator (Scan) or Circular Scan (C-Scan)
  - Bounded Window
- Greedy vs. Truly Optimal

I/O SCHEDULERS



Where should the I/O scheduler go?

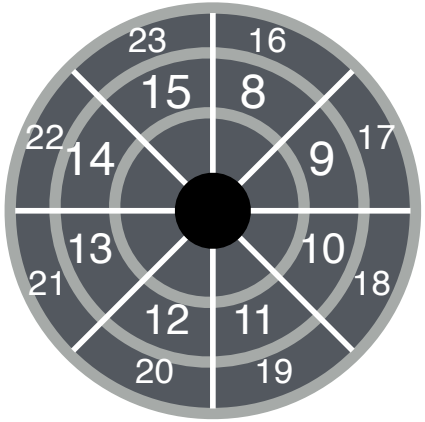
WHAT HAPPENS AT OS LEVEL?

Assume 2 processes A and B each call read() with C-SCAN

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

How will processes be scheduled?

Stream of requests seen by disk: ABABABA



WORK CONSERVATION

Work conserving schedulers always do work if work exists

- Principle applies to any type of scheduler (CPU too)

Could be better to wait if can **anticipate** another request will arrive

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

- Keeps resource idle while waiting for future requests

Better stream of requests for OS to give disk: AAAAAABBBBBBAAAAAA

I/O DEVICE SUMMARY

Overlap I/O and CPU whenever possible!

- Use interrupts, DMA

Storage devices provide common **block interface**

On a disk: Never do random I/O unless you must!

- Quicksort is a terrible algorithm on disk

Spend time to schedule on slow, stateful devices

Next: Other storage devices (RAIDs and SSDs/flash)

TODAY: RAID

ONLY ONE DISK?

Sometimes we want many disks — why?

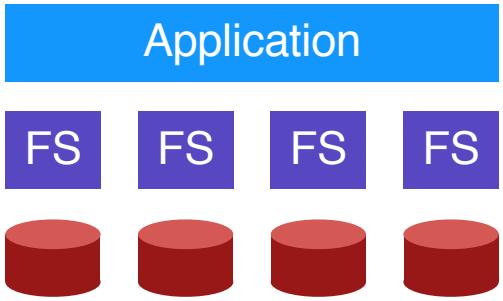
- Capacity
- Reliability
- Performance

Challenge:
Most file systems work on only one disk (assume linear array of blocks)

SOLUTION 1: JBOD

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

Application stores different files on different file systems



Disadvantages:

Application must manage multiple devices

Not portable

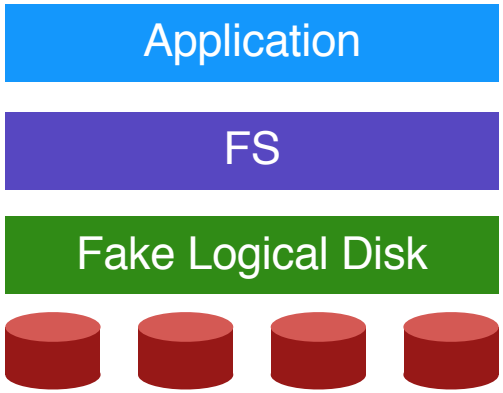
SOLUTION 2: RAID

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?

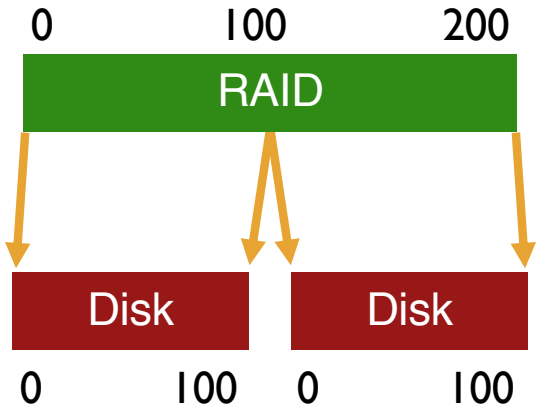
Alternative to RAID: buy an expensive, high-end disk

RAID Approach

- Economies of scale! Commodity disks cost less
- Can buy **many** commodity H/W components for same price as few high-end components
- Write software to build high-quality logical devices from many cheap devices

GENERAL STRATEGY: MAPPING

Build fast, large disk from smaller disks



RAID MAPPING

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

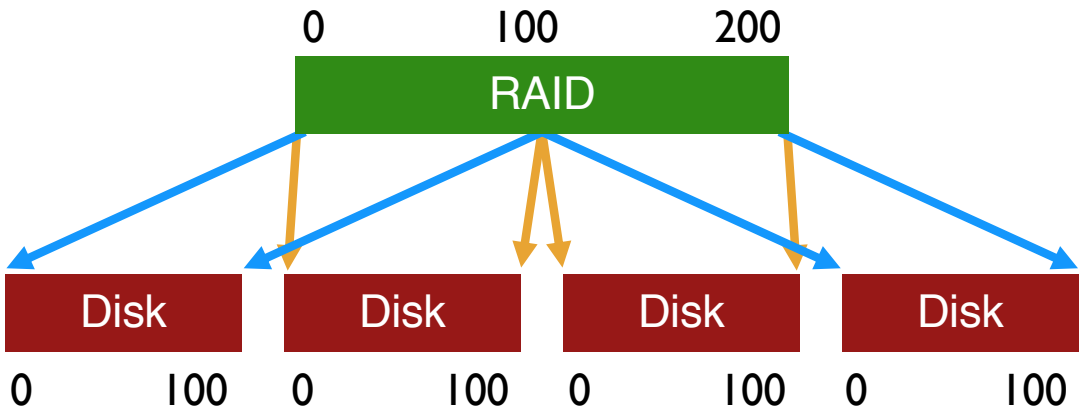
- Some similarity to virtual memory

1) **Dynamic** mapping (logical x sometimes maps to physical y and sometimes z):
   - Use data structure (array, hash table, tree)
   - Page tables

2) **Static** mapping (logical x always maps to physical y):
   - Use simple math
   - RAID

GENERAL STRATEGY: REDUNDANCY

Add even more disks for reliability



REDUNDANCY

How many physical **copies** should RAID keep for every logical block?

Increase number of copies:

- improves reliability (and maybe performance)

Decrease number of copies

- improves space efficiency

REASONING ABOUT RAID

1) RAID Level:

system for mapping logical to physical blocks

2) Workload:

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

3) Metric:

capacity, reliability, performance

1) RAID DECISIONS

Which logical blocks map to which physical blocks on disks?

How to use extra physical blocks (if any)?

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

- RAID 0: Striping
- RAID 1: Mirroring
- RAID 4: Parity
- RAID 5: Rotated Parity

2) WORKLOADS

- Reads
  - One operation (for latency)
  - Steady-state I/O (for throughput or bandwidth)
    - Sequential
    - Random
- Writes
  - One operation (for latency)
  - Steady-state I/O (for throughput or bandwidth)
    - Sequential
    - Random

3) METRICS

Capacity: how much space is available to higher levels?

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

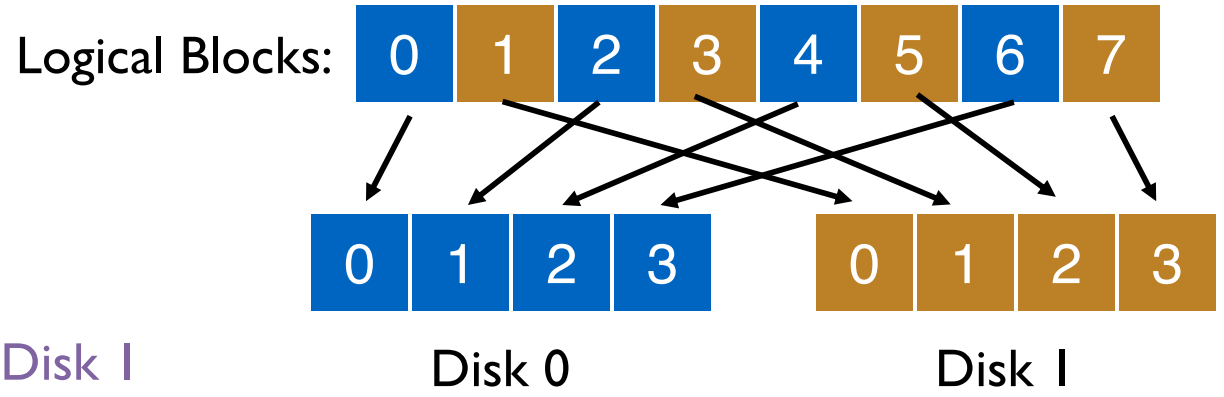
Performance: how long does each workload take?

Normalize each to characteristics of one disk

N := number of disks
C := capacity of 1 disk (500 GB?)
S := sequential throughput of 1 disk (100 MB/s?)
R := random throughput of 1 disk (5 MB/s?)
D := latency of one small I/O operation

RAID-0: STRIPING

Optimize for capacity. No redundancy

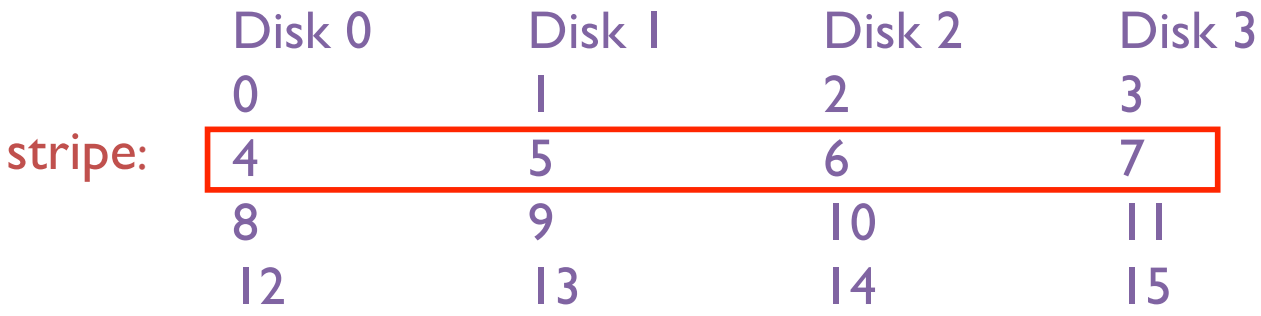


Disk 0	Disk 1
0	1
2	3
4	5
6	7

RAID-0: 4 DISKS

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

RAID-0: 4 DISKS



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

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

REAL SYSTEMS: CHUNK SIZE

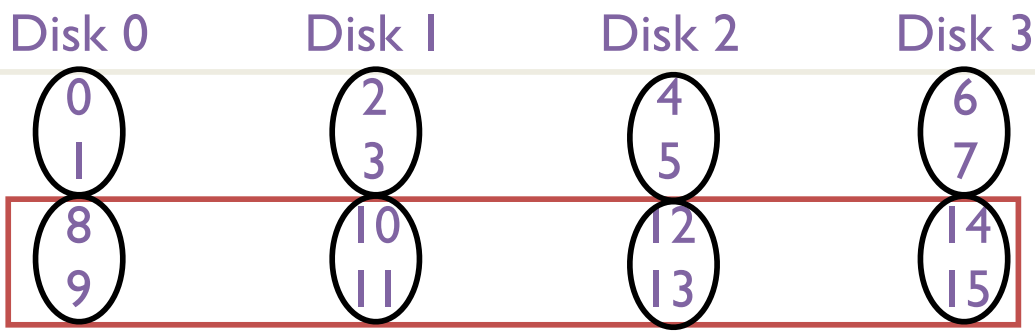
Chunk size = 1

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

Simplification: assume chunk size of 1

Chunk size = 2

stripe:



RAID-0 VISUALIZATION

1. ./example-stripe-one-request.csh
2. ./example-stripe-random-reads.py
3. ./example-stripe-random-writes.py

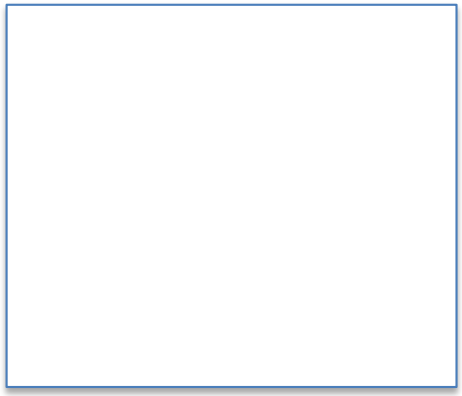
RAID-0: ANALYSIS

What is capacity?

How many disks can fail (no loss)?

Latency?

Throughput (sequential, random)?

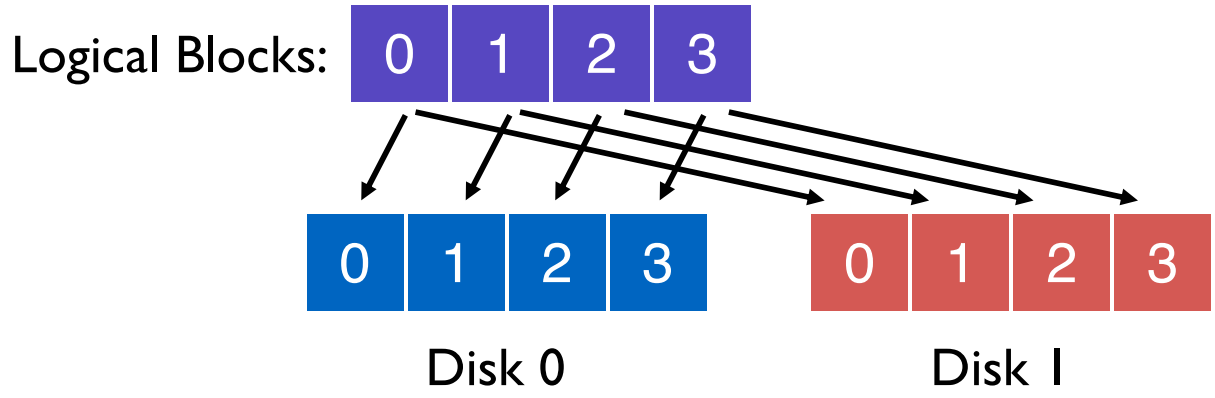


Buying more disks improves throughput, but not latency!

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

Disk 0	Disk 1	Disk 2	Disk3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

RAID-1: MIRRORING



Keep two copies of all data

RAID-10: MIRRORING + STRIPING

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

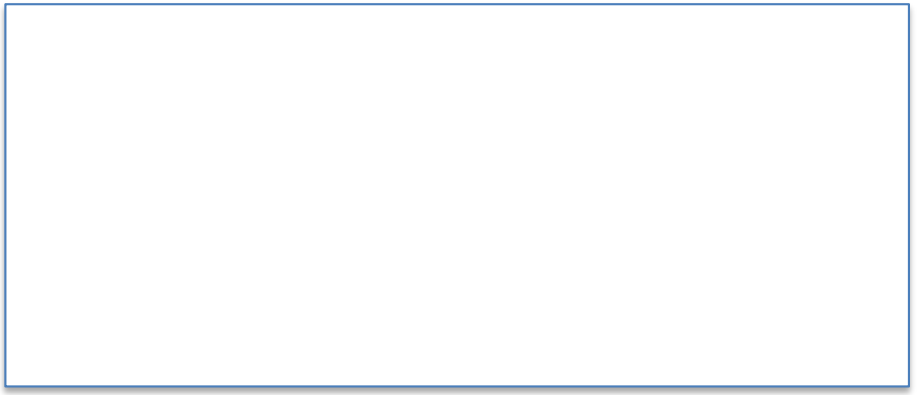
RAID-1: MIRRORING

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 without losing any data?

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 VISUALIZATION

4) ./example-mirror-random-reads.py

5) ./example-mirror-random-writes.py

6) ./example-mirror-seq-reads.csh

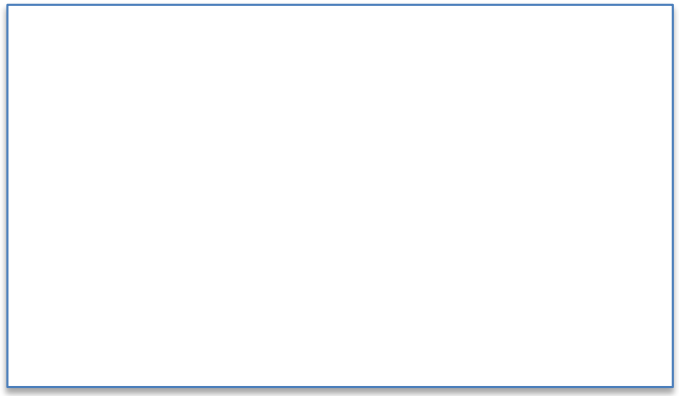
7) ./example-mirror-seq-reads-slomo.csh

RAID-1: ANALYSIS

What is capacity?

How many disks can fail?

Latency (read, write)?



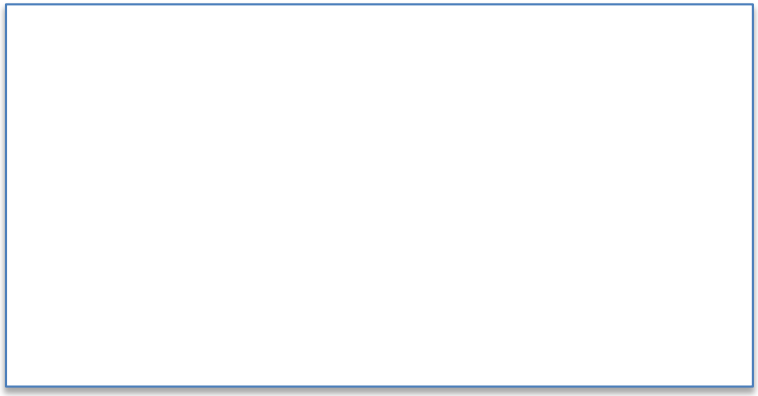
N := number of disks
C := capacity of 1 disk
S := sequential throughput of 1 disk
R := random throughput of 1 disk
D := latency of one small I/O operation

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: THROUGHPUT

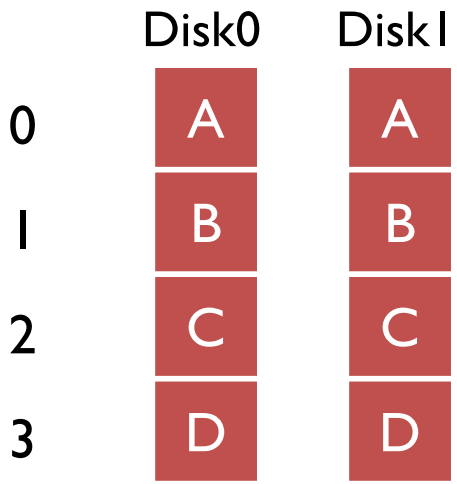
What is steady-state throughput for

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



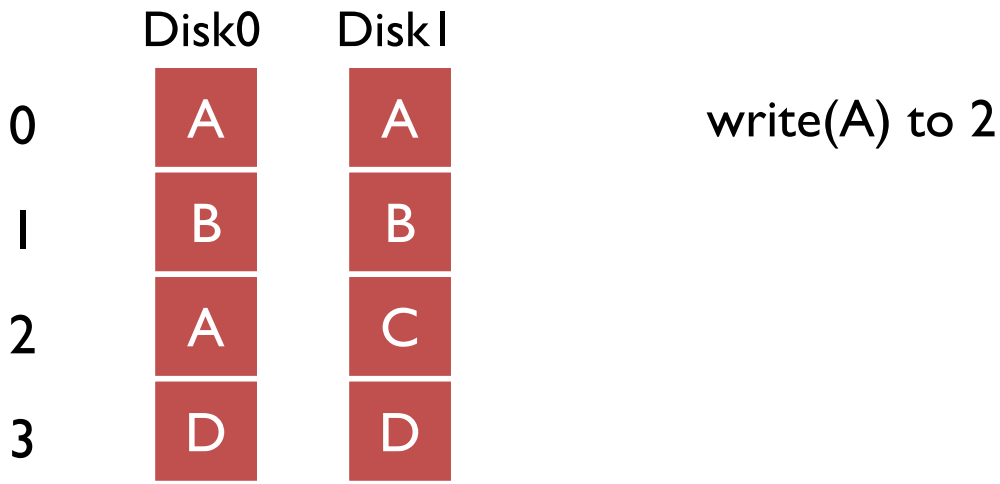
Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

SIDE ISSUE: CRASHES

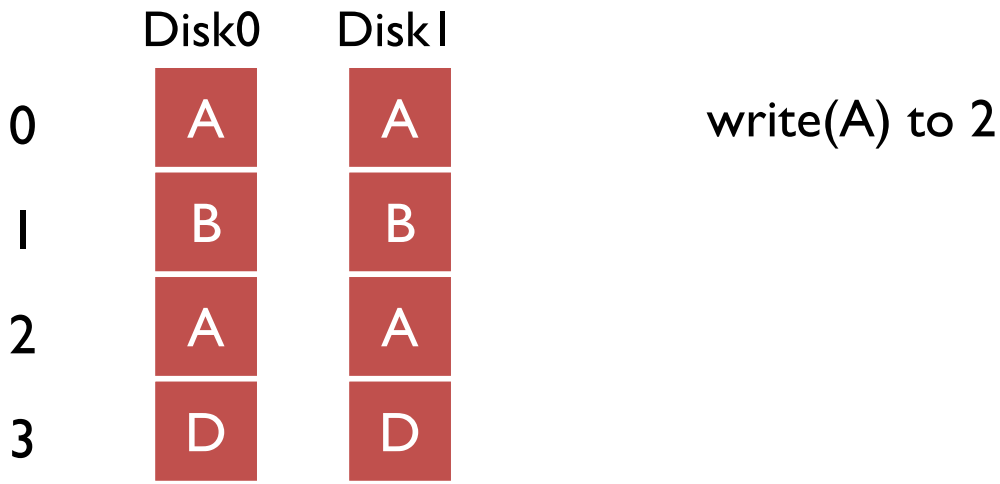


write(A) to 2

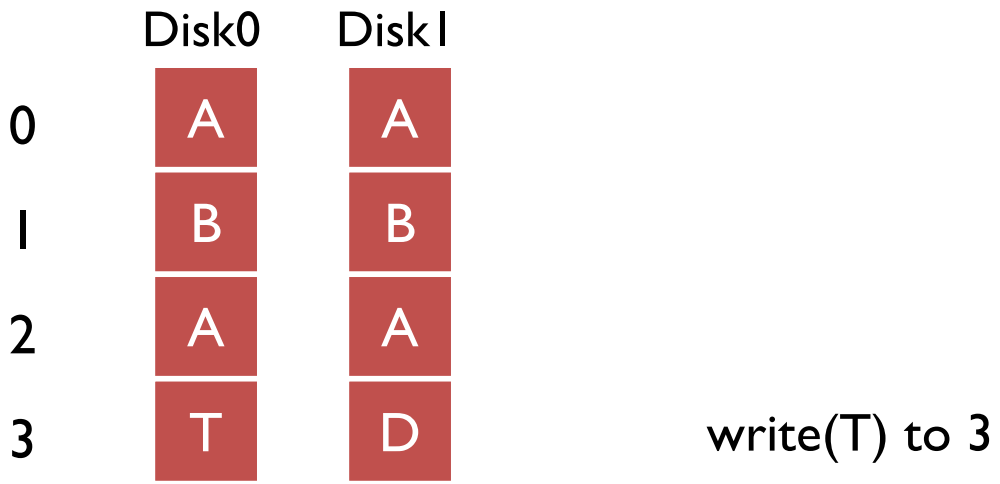
SIDE ISSUE: CRASHES



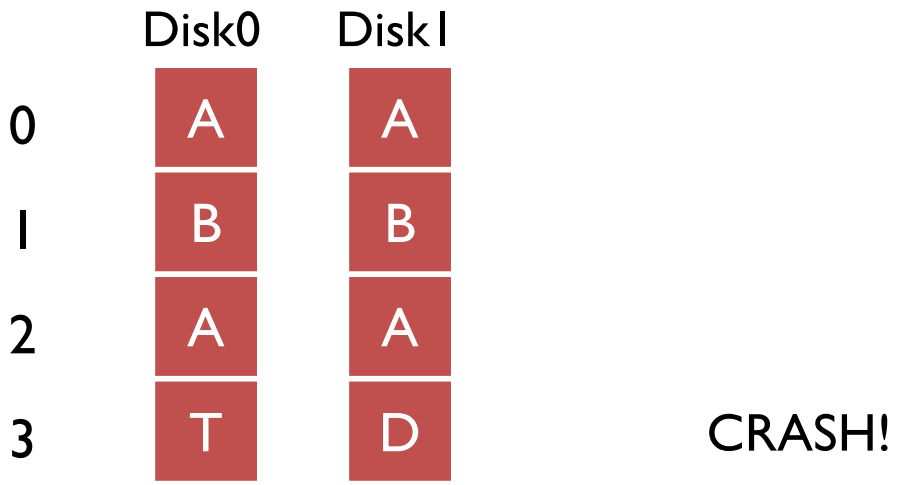
SIDE ISSUE: CRASHES



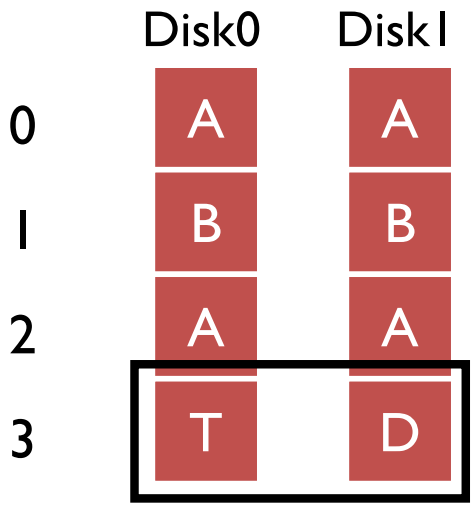
SIDE ISSUE: CRASHES



SIDE ISSUE: CRASHES



SIDE ISSUE: CRASHES



after reboot, how to
tell which data is right?

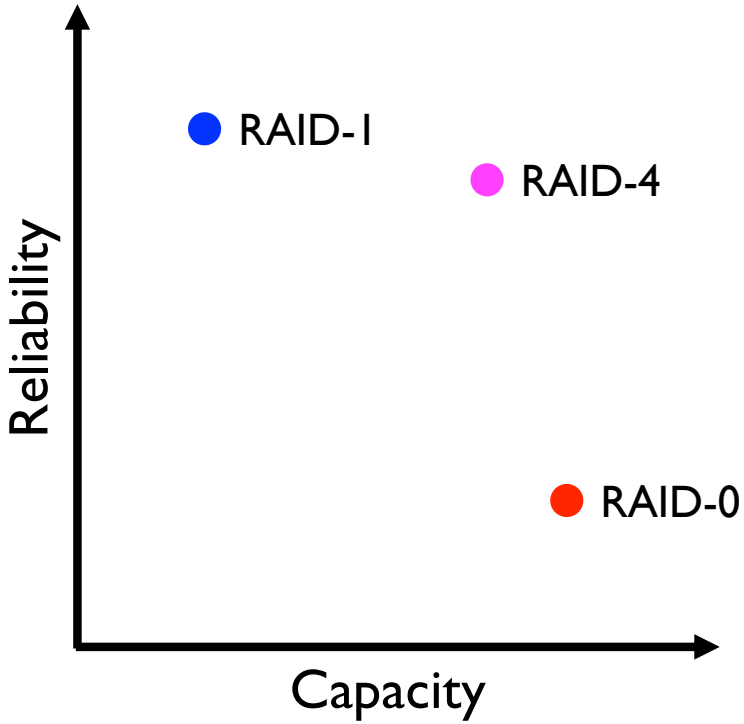
CRASHES: H/W SOLUTION

Problem: Consistent-Update Problem

Use non-volatile RAM in RAID controller

Can replay to ensure all copies are updated

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


RAID-1
RAID-4
RAID-0
Reliability
Capacity

RAID-4 STRATEGY

Use one disk for parity

In algebra:
Equation with N variables and N-1 are known, can often solve for unknown

Treat sectors across disks in a stripe as equation

Data on bad disk is the unknown in equation

RAID-4 WITH PARITY

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	P0
3	4	5	P1
6	7	8	P2
9	10	11	P3

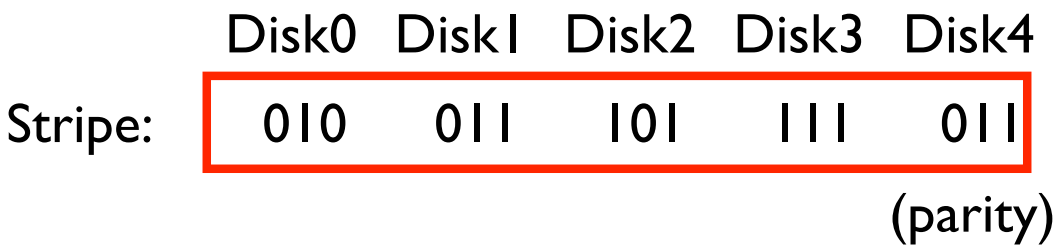
Data blocks on disks 0 – 2

Disk 3 for parity

Parity calculated over data blocks in stripe

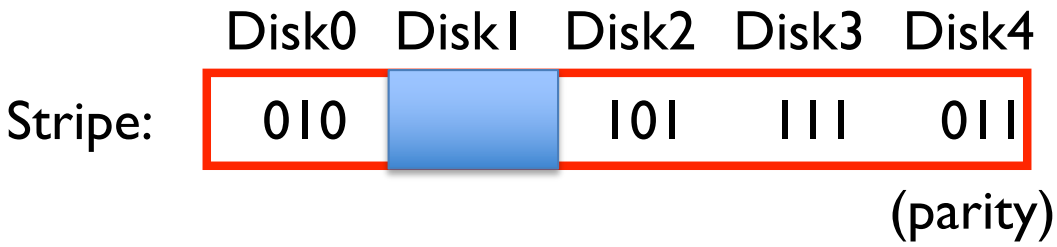
P0 = B0 XOR B1 XOR B2

PARITY EXAMPLE: 1



Calculate parity from data blocks
Parity = D0 XOR D1 XOR D2 XOR D3

PARITY EXAMPLE: 1



Can reconstruct blocks of lost disk by taking XOR

D1 = D0 XOR D2 XOR D3

UPDATING PARITY: XOR

If write "0110" to block 0, how should parity be updated?

One approach: read all other N-2 blocks in stripe and calculate new parity

Second approach: Read old value at block 0

1100

Read old value for parity

0101

Calculate new parity

1111

Write out new parity

→ 2 reads and 2 writes (1 read and 1 write to parity block)

RAID-4 VISUALIZATION

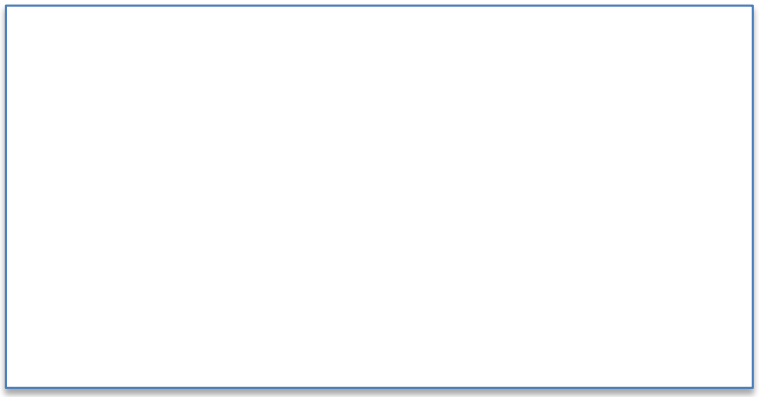
8) ./example-raid4-random-reads.csh
9) ./example-raid4-full-stripe-writes.csh
10) ./example-raid4-random-writes.csh

RAID-4: ANALYSIS

What is capacity?

How many disks can fail?

Latency (read, write)?



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

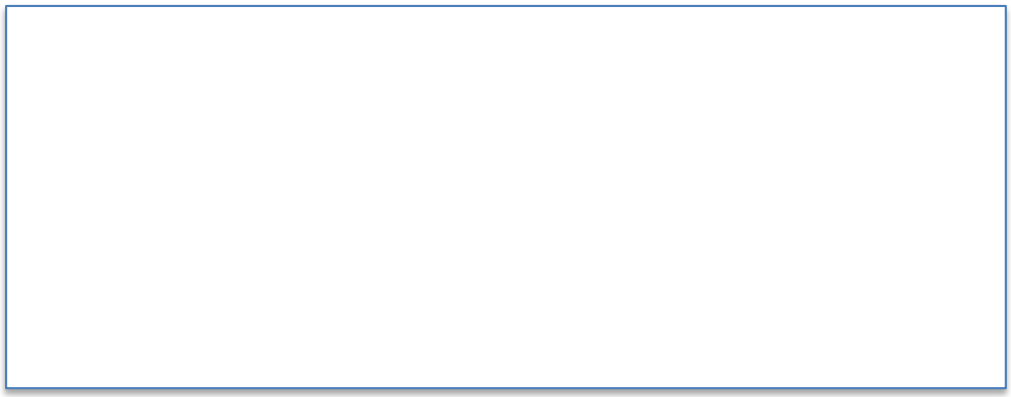
Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	P0
3	4	5	P1
6	7	8	P2
9	10	11	P3

RAID-4: THROUGHPUT

how to avoid parity bottleneck?

What is steady-state throughput for

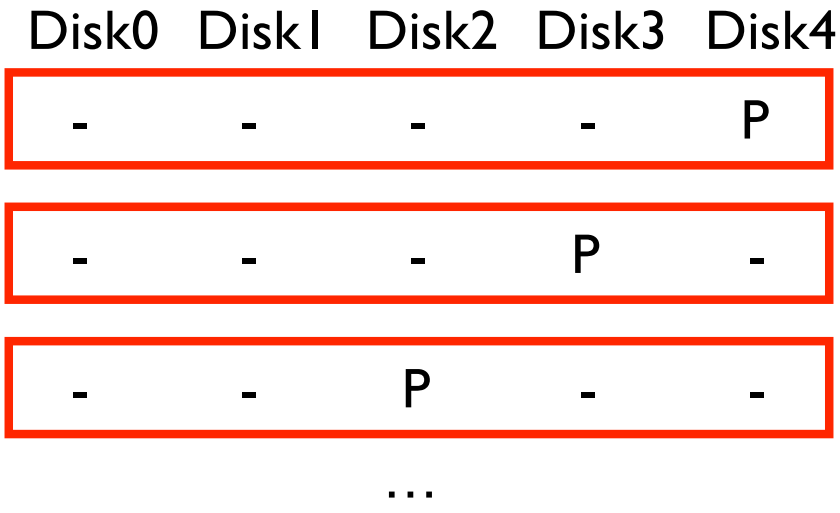
- sequential reads?
- sequential writes?
- random reads?
- random writes?



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

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	P0
3	4	5	P1
6	7	8	P2
9	10	11	P3

RAID-5



Rotate parity across different disks

Where exactly do individual data blocks go?

LEFT-SYMMETRIC RAID-5

D0	D1	D2	D3	D4
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

Pattern repeats…

RAID-5 VISUALIZATION

11) ./example-raid5-simple.csh

12) ./example-raid5-random-reads.csh

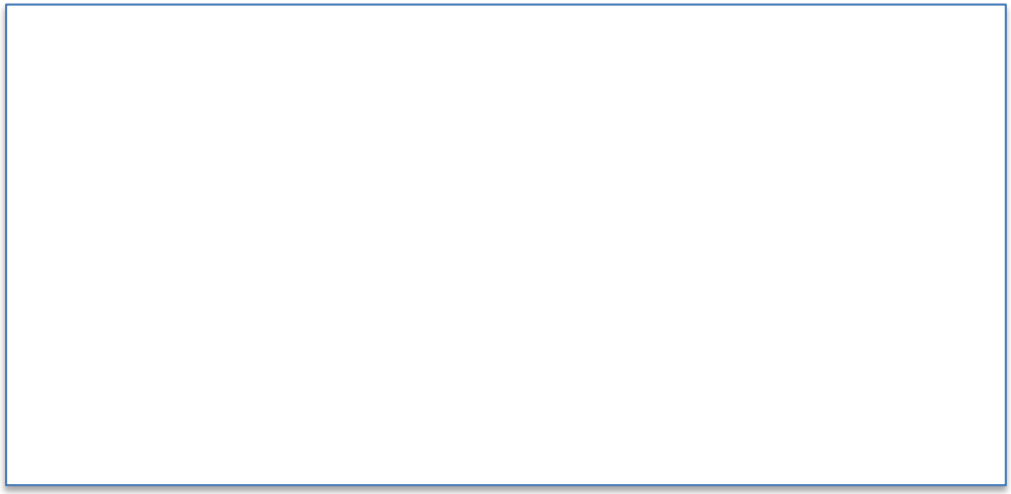
13) ./example-raid5-random-writes.csh

RAID-5: ANALYSIS

What is capacity?

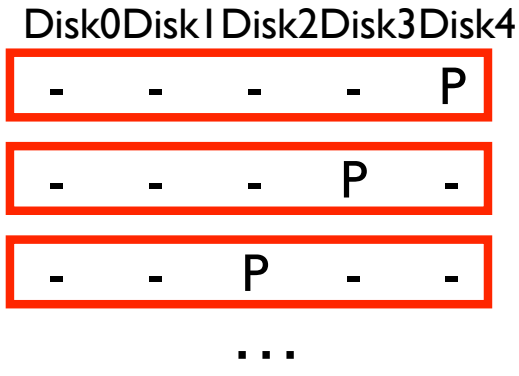
How many disks can fail?

Latency (read, write)?



These metrics same as RAID-4…

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



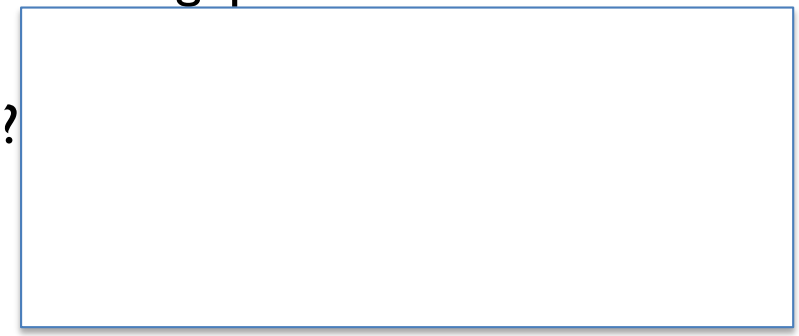
RAID-5: THROUGHPUT

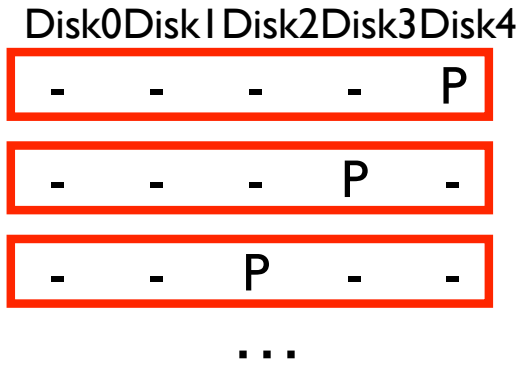
Steady-state throughput for RAID-4

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

What is steady-state throughput for RAID-5?

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





HIGHER RAID LEVELS

RAID-6 can handle more than 1 disk failure

Use multiple parity blocks

(Not covered in this course)

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-1 better than RAID-5 for random workloads

RAID-5 better than RAID-1 for sequential workloads

RAID SUMMARY

Block-based interface:
Very deployable and popular storage solution due to transparency

Many engineering tradeoffs with RAID

Capacity, reliability, performance for different workloads

Can build RAID over any other block-based storage device

SSDs instead of HDDs!