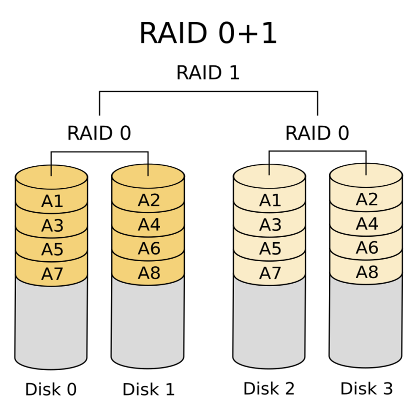
Distributed Systems

- *48 - Communication Basics*
- *49 - NFS*
- *50 - AFS*
- *GFS*
- *TRIO*
- Review

RAID Level 10 (1+0) *mirroring* and *striping*

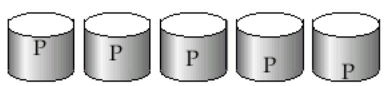
- Common to combine 0 and 1 to make "1+0", or "10"
  - two copies of each data block
  - if more disks, than *stripe* across the pairs
- For example, assume 4 disks, each 1TB, with bandwidth 1GB/sec:
  - 2 mirrors 0, 3 mirrors 1
  - capacity is 2 TB, because two copies of everything.
  - depending on workload, could have read rate up to 4 GB/sec:
    - stripe A1 read from disk 0
    - stripe A2 read from disk 1
    - stripe A3 read from disk 2
    - stripe A4 read from disk 3
  - write bandwidth capped at 2 GB/sec, because each block written two places



RAID Level 5 *redux*

If we assume many large, sequential writes, then caching does not help, and write bandwidth reduced by factor of 4

- Distributed parity "blocks" instead of bits
- Normal operation:
  - "Read" directly from single disk.
    - Load distributed across all 5 disks
  - "Write": Need to read and update the parity block
    - To update 9 to 9'
      - read 9 and P2
      - compute P2' = P2 *xor* 9 *xor* 9'
      - write 9' and P2'



(f) RAID 5: block-interleaved distributed parity

P0	0	1	2	3
4	P1	5	6	7
8	9	P2'	10	11
12	13	14	P3	15
16	17	18	19	P4

RH 7 *review*

Q1
5 Points

Given disk:

- 6000 RPM
- 200 sectors/track
- sector is 8KB
- avg. seek time 2 msec
- read/write no difference
- no track caching

Write only a number in each of the boxes for this question: no explanation, no units, no nothing.

Q1.1
1 Point

In msecs, what is the average rotational latency?

5

Explanation
6000 RPM ⇒ 100/sec ⇒ 10 msec per rotation. Average latency would be half of this, i.e. **5** msecs.

Q1.2
1 Point

In msecs, what is the average sector transfer time?

0.05

Explanation
10 msec/rot, 1/200th of a rotation

Q1.3
1 Point

In msecs, what is the average cost of a random 4k read?

7.05

Explanation
$= \text{seek} + \text{latency}_{rotational} + \text{transfer}_{sector}$
$= 2 + 5 + 0.05 = 7.05$ msec
you have to read an entire sector

RH 7 *review*

Q1.5
1 Point

In msecs, what would be the minimum expected cost of reading 10 sequentially ordered sectors?

7.5

Explanation

The minimum expected would be if they were all laid out in the same track, so we only pay seek time and rotational latency once. After that we just pay the transfer time for the rest of the sectors.

$7.05 + 9 * 0.05 = 7.5$ msec

RH 8 *review*

Q2
5 Points

List the writes that should occur when creating a 100-byte file `bar.c` in the directory `/foo` for a generic, non-journaling file system, in a correct order (there may be more than one):

Explanation

write data block bitmap (async)
write `bar.c` data (async)
write inode bitmap (async)
write `bar.c` inode
write `/foo` data
write `/foo` inode

Q3
5 Points

Instead, how many disk writes would a log-structured file system ideally issue?

(enter just an integer)

1

done 4/16

RH 9 *review*

Q4
1 Point

During creation of file f in directory d, the following write ordering would be appropriate for FFS:

- [ ] $\text{data}_d < \text{inode}_d < \text{data}_f < \text{inode}_f$
- [ ] $\text{data}_d < \text{data}_f < \text{inode}_d < \text{inode}_f$
- [x] $\text{data}_f < \text{inode}_f < \text{data}_d < \text{inode}_d$
- [ ] $\text{data}_f < \text{data}_d < \text{inode}_f < \text{inode}_d$

Explanation
The order of the synchronized writes must be:
$\text{inode}_f < \text{data}_d < \text{inode}_d$.
Only the third choice meets this criteria.

Q5
1 Point

During creation of file f in directory d, the following log order would be appropriate for LFS:

- [ ] $\text{data}_d < \text{inode}_d < \text{data}_f < \text{inode}_f$
- [ ] $\text{data}_d < \text{data}_f < \text{inode}_d < \text{inode}_f$
- [x] $\text{data}_f < \text{inode}_f < \text{data}_d < \text{inode}_d$
- [x] $\text{data}_f < \text{data}_d < \text{inode}_f < \text{inode}_d$

Explanation
Both the last two options are correct, as neither writes pointers to the log before they can be accessed. The goal is to *never allow a pointer to become visible before the data that the pointer specifies is completely initialized*.

In this LFS example, nothing of the other changes are visible until the new **inode**$_d$ is visible. Therefore, as long as **inode**$_d$ is written last, the ordering of the other writes is irrelevant.

RH 9 *review*

Q9
3 Points

How is an SSD like a log?

Explanation
The system erases blocks asynchronously, and logically orders them on the front of the "log". New writes go into the next available page in the log, so all new writes go sequentially into the "log".

Q11
1 Point

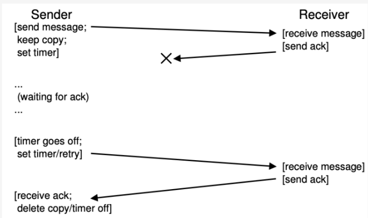
If we assume blocks have checksums stored with them on disk, how can file systems detect when the wrong logical block is returned? (i.e., the system misdirected another write).

Explanation
Include the physical block number / sector in the data to be checksummed.

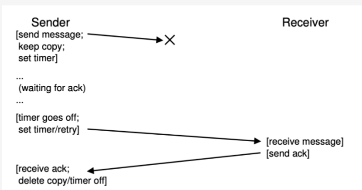
RH 10 *review*

Q1
1 Point

How does a sender in a reliable protocol distinguish between the following two cases?



and



- ○ acks
- ○ retries
- ○ sequence numbers
- ◉ it doesn't

Q2
1 Point

Why is it important that sequence numbers increase monotonically?

- [x] to identify lost packets
- [x] to identify duplicate packets
- [x] to reduce state overhead

Q3
1 Point

A programmer defining a new RPC protocol, and app with which to use it, is responsible for defining which software bits?

- [x] interface definition
- [x] calling the RPC
- [ ] creating and wiring in the client stub
- [ ] creating and wiring in the server stub
- [x] defining the remote procedure

RH 10 *review*

Q8
1 Point

Assume two NFS v2 clients are reading and modifying the file `foo`, initially containing blocks w/ contents A, B, C, and D (each letter defines an entire block of data). The following sequence of operations occurs:

- $client_1$ reads `foo`
- $client_1$ overwrites B, C w/ X, Y
- $client_2$ reads `foo`
- $client_2$ overwrites C, D w/ I, J
- $client_2$ closes `foo`
- $client_1$ closes `foo`

What are the final contents of the file? (Note that there are two possiblities, choose either.)

- ○ A, B, C, D
- ◉ A, X, I, J
- ○ A, X, Y, J ← also
- ○ A, B, Y, J

Q5
1 Point

How do NFS v2 clients detect server failures?

- ○ sequence numbers
- ○ timeouts
- ◉ they don't

RH 10

Q11
1 Point

Assume two AFS clients are reading and modifying the file `foo`, initially containing blocks w/ contents A, B, C, and D (each letter defines an entire block of data). The following sequence of operations occurs:

- $client_1$ reads `foo`
- $client_1$ overwrites B, C w/ X, Y
- $client_2$ reads `foo`
- $client_2$ overwrites C, D w/ I, J
- $client_2$ closes `foo`
- $client_1$ closes `foo`

What is the final contents of the file?

- ○ A, B, C, D
- ◉ A, X, Y, D
- ○ A, X, I, D
- ○ A, B, I, J

RH 10

Q11
1 Point

Assume two AFS clients are reading and modifying the file `foo`, initially containing blocks w/ contents A, B, C, and D (each letter defines an entire block of data). The following sequence of operations occurs:

- $client_1$ reads `foo`
- $client_1$ overwrites B, C w/ X, Y
- $client_2$ reads `foo`
- $client_2$ overwrites C, D w/ I, J
- $client_2$ closes `foo`
- $client_1$ closes `foo`

What is the final contents of the file?

- ○ A, B, C, D
- ○ A, X, Y, D
- ○ A, X, I, D
- ◉ A, B, I, J

RH 10

Q11
1 Point

Assume two AFS clients are reading and modifying the file `foo`, initially containing blocks w/ contents A, B, C, and D (each letter defines an entire block of data). The following sequence of operations occurs:

- $client_1$ reads `foo`
- $client_1$ overwrites B, C w/ X, Y
- $client_2$ reads `foo`
- $client_2$ overwrites C, D w/ I, J
- $client_2$ closes `foo`
- $client_1$ closes `foo`

What is the final contents of the file?

- ( ) A, B, C, D
- ( ) A, X, Y, D
- ( ) A, X, I, D
- (•) A, B, I, J

Exam 3 *review*

- exam topics:
  - disk performance: perf from latencies
  - disk scheduling: SSTF, CSCAN, SCAN, LOOK, CLOOK
  - RAID 0, 1, 10, 5
  - FFS: advantages, order constraints
  - journaling, meta-data journaling, order constraints
  - LFS
  - SSDS: simple mapping table, hybrid mapping table
  - Distributed Systems
    - Communication Basics - RPC
    - end-to-end argument
    - NFS - understand the stateless protocol
    - AFS - understand update visibility and stale caches
    - GFS - under data movement, division of responsibility
    - TRIO - secure sharing and integrity verification w/o impacting performance
- what to review
  - quizzes 7-10
  - mid2s24 (piazza)
  - lecture slides

done.