IT308: Operating Systems

Redundant Arrays of Inexpensive Disks (RAID)

Redundancy

- We have talked about how to use journal to recover from a crash during write, by replaying the logged writes
- this relies on the assumption that the disk is still usable after rebooting

Redundancy

- We have talked about how to use journal to recover from a crash during write, by replaying the logged writes
- this relies on the assumption that the disk is still usable after rebooting
- In reality, disks can just break, and the data on that disk is simply lost

Redundancy

- We have talked about how to use journal to recover from a crash during write, by replaying the logged writes
- this relies on the assumption that the disk is still usable after rebooting
- In reality, disks can just break, and the data on that disk is simply lost
- The solution: have more than one copies of the data, i.e., redundancy

Goals of RAID

- Performance: use multiple disks in parallel to speed up I/O performance

Goals of RAID

- Performance: use multiple disks in parallel to speed up I/O performance
- Capacity: more disks combined provide more space

Goals of RAID

- Performance: use multiple disks in parallel to speed up I/O performance
- Capacity: more disks combined provide more space
- Reliability: with data spread across multiple disks with redundancy, RAID can tolerate the loss of a disk – keep operating as if nothing were wrong.

Goals of RAID

- Performance: use multiple disks in parallel to speed up I/O performance
- Capacity: more disks combined provide more space
- Reliability: with data spread across multiple disks with redundancy, RAID can tolerate the loss of a disk – keep operating as if nothing were wrong.
- Transparency: from outside, a RAID just looks like a big disk with good performance, large capacity and great reliability.
  - You can easily unplug a regular disk and replace it with a RAID.

Fault Model

- In our discussion, we assume the following fail-stop fault model
  - a disk has only two possible states: working or failed
  - when a disk fails, we can detect it immediately, i.e., the fail is not silent

Fault Model

- In our discussion, we assume the following fail-stop fault model
  - a disk has only two possible states: working or failed
  - when a disk fails, we can detect it immediately, i.e., the fail is not silent
- Real-world faults can be more complicated than this

RAID Level 0: Striping

No redundancy yet, simply stripe the blocks across different 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

Each row is called a stripe

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

Another way of striping. Difference?

RAID-0 reliability issue: data is lost whenever a disk fails

RAID Level 1: Mirroring

Have more than one copy of each block

RAID Level 1: Mirroring

Have more than one copy of each block
When reading block 5, can choose either Disk 2 or Disk 3

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

Issue with RAID-1: bad disk space utilization, what you can store is only half of the total physical capacity

Interlude: Parity Bit

- Assume A, B and C are either 0 or 1.
- What is the Boolean expression that indicates whether there are an odd number of 1's in A, B and C ?
  - P = A xor B xor C
- Suppose A = 1, B = 0, P = 1. Do we know what C is?
  - C = 0
- What if P = 0?
  - C = 1

RAID Level 4: Save space with parity

Use less space to achieve redundancy, compared to mirroring.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

A dedicated disk for parity checks

RAID Level 4: Save space with parity (2)

- Each parity bit keeps, among the corresponding bits in the blocks of the same stripe, whether there are an odd or even number of 1's. (Done by XOR).

RAID Level 4: Save space with parity (2)

- Each parity bit keeps, among the corresponding bits in the blocks of the same stripe, whether there are an odd or even number of 1's. (Done by XOR).
- If one disk on the stripe fails, we can recover the lost bit.

RAID Level 4: Save space with parity (2)

- Each parity bit keeps, among the corresponding bits in the blocks of the same stripe, whether there are an odd or even number of 1's. (Done by XOR).
- If one disk on the stripe fails, we can recover the lost bit.
- If the parity of the remaining bits is the same as before, then we lost a 0, otherwise we lost a 1.

RAID Level 4: a problem

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
*4	5	6	7	+P1
8	9	10	11	P2
12	*13	14	15	+P3

- Suppose we are writing to Block 4 and Block 13 simultaneously, since they are on different disks we expect good performance due to parallelism. But no!

RAID Level 4: a problem

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
*4	5	6	7	+P1
8	9	10	11	P2
12	*13	14	15	+P3

- Suppose we are writing to Block 4 and Block 13 simultaneously, since they are on different disks we expect good performance due to parallelism. But no!
- Both writes cause writes on the parity blocks on Disk 4, which spoiled the parallelism.

RAID Level 4: a problem

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
*4	5	6	7	+P1
8	9	10	11	P2
12	*13	14	15	+P3

- Suppose we are writing to Block 4 and Block 13 simultaneously, since they are on different disks we expect good performance due to parallelism. But no!
- Both writes cause writes on the parity blocks on Disk 4, which spoiled the parallelism.
- The disk with parity blocks is so frequently written that it becomes the bottleneck

RAID Level 5: Rotating Parity

- Distribute the frequently-written parity blocks to different disks.
- More even workload across disks, better performance.

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