Operating Systems: Mass Storage Structure – Part II

Neerja Mhaskar

Department of Computing and Software, McMaster University, Canada

Acknowledgements: Material based on the textbook Operating Systems Concepts (Chapter 11)

Storage Array

Storage Array is an array of SSDs and/or HDDs that operate independently and in parallel.

<u>Advantages:</u>

- Improved performance achieved via parallelism.
  - **Separate I/O requests can be handled in parallel** as long as the data required reside on separate disks.
  - A **single I/O request can be executed in parallel** if the block of data to be accessed is distributed across multiple disks
- Improved reliability achieves via redundancy.
  - **Data mirroring**: Duplicating identical data on multiple disks

Storage Array

Disadvantages:

- Set of multiple disks increases the probability of failure.
- **Mean time between failures (MTBF):** The statistical mean time that a device is expected to work correctly before failing.
- Suppose that the **mean time between failures** (MTBF) of a single disk is 100,000 hours.
- Then the MTBF of some disk in an array of 100 disks will be 100,000/100 = 1,000 hours.

Image of storage array

RAID – Redundant Array of Independent Disks

- **RAID** - disk-organization techniques used to improve performance and reliability in a system *using an array of disks*.

- The different RAID levels (discussed here) share the below characteristics:

  1. Each RAID level consists of a set of physical disk drives viewed by the operating system as a *single logical drive*.

  2. Data are distributed across the physical drives of an array in a scheme known as **striping**.

  3. **Redundant disk capacity** is used to store duplicate data or parity information

     - Guarantees data recoverability in case of a disk failure.

Data Striping

- **Data striping** is of two types:
  - **Bit-level striping** - splitting the bits of each byte across multiple disks
    - With 8 disks, the i-th bit of a byte goes to disk i.
  - **Block-level striping** – stripping the blocks of a file across multiple disks.
    - With *n* disks, block *i* of a file goes to disk (i mod n) + 1.
    - For example if n=4 and i=5, then block 5 goes to disk (5 mod 4) + 1 = 1+1 = 2. The assumption here is that Disk numbering starts from 1 and block numbering starts from 0.
    - Most common

Parity

- **Parity** records whether the number of bits in the byte set to 1 is even (parity = 0) or odd (parity = 1).
  - Parity (10011000) = 1
  - Parity (11011000) = 0
  - Parity calculated by performing an XOR (“eXclusive OR”) operation of the bits in a byte.
- **XOR** ($\oplus$) is a logical operation that is true if and only if its arguments differ.

Error Detection

- Error detection determines if a problem has occurred
- Parity is used to detect single bit errors in memory systems.
  - Each byte in a memory system has a parity bit associated with it.
  - If one of the bits in the byte is damaged (either a 1 becomes a 0, or a 0 becomes a 1), the parity of the byte changes
  - Thus, it does not match the stored parity, and vice a versa.
  - A double-bit-error *might* go undetected however.

Error correction code (ECC)

An **error-correction code (ECC)** not only detects errors, but also corrects it.

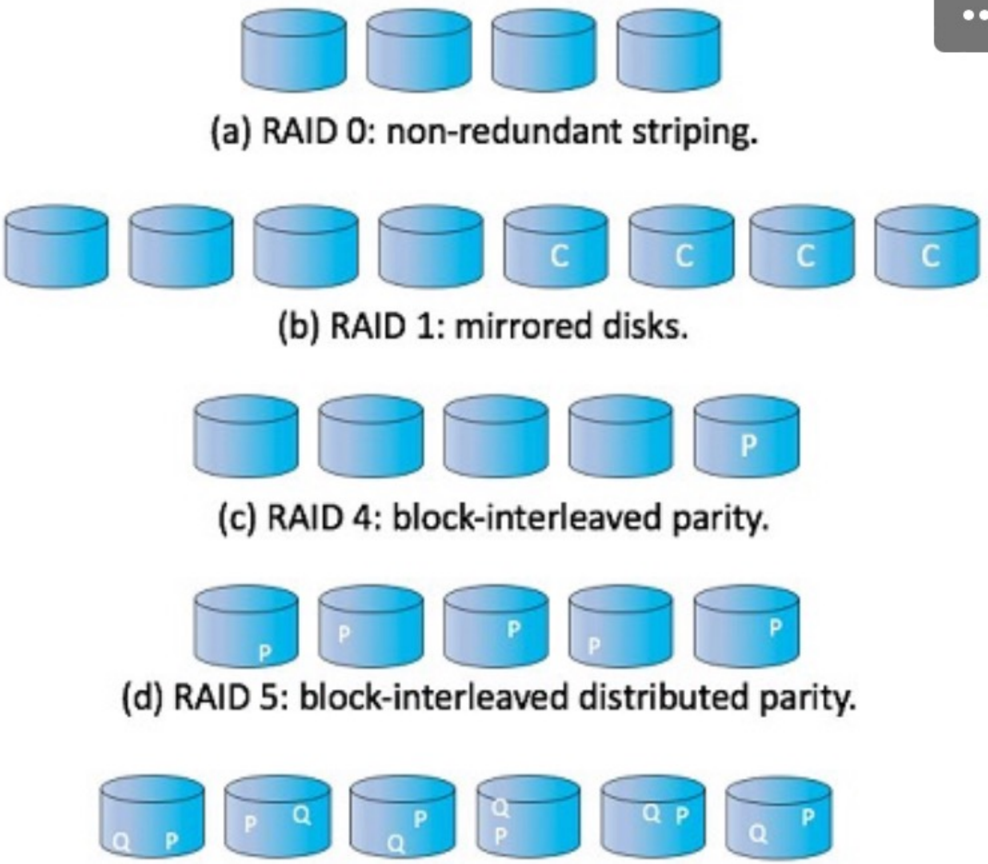
➢ achieved using algorithms and extra amounts of storage.

How does it work?

- When the disk controller writes data on a sector, the ECC is calculated from all the bytes in the data and written on the sector.
- When the sector is read, ECC is recalculated and compared with the stored value.
- If the stored and calculated numbers are different => data corruption.
- If only a few bits of data have been corrupted, ECC can correct the errors. Otherwise, reports data error.

RAID Levels

- In figure on the left
  - C = copy of data
  - P = Parity
  - P, Q = ECC



(a) RAID 0: non-redundant striping.
(b) RAID 1: mirrored disks.
(c) RAID 4: block-interleaved parity.
(d) RAID 5: block-interleaved distributed parity.
(e) RAID 6: P + Q redundancy.

RAID Levels

- **RAID 0**: has *block level striping* with no redundancy.

(a) RAID 0: non-redundant striping.

- **Raid 1**: *has mirroring* only, no striping.

(b) RAID 1: mirrored disks.

RAID 4

- **RAID 4** (block-interleaved parity organization):
  - Uses block-level striping, and in addition keeps *a parity block on a separate additional disk* for corresponding blocks from *N* other disks
  - Therefore, RAID 4, has a dedicated block for parity blocks.

(c) RAID 4: block-interleaved parity.

RAID 5 (block-interleaved distributed parity) (most common):

- Spreads data and parity among all N+1 disks, rather than storing data in *N* disks and parity in one disk.

(d) RAID 5: block-interleaved distributed parity.

- For each block, one of the disks stores the parity and the others store data.
- A parity block cannot store parity for blocks in the same disk
- For example, with an array of five drives, the parity for the nth block is stored in drive (n mod 5) + 1
- By spreading the parity across all the drives in the set, RAID 5 avoids potential overuse of a single parity drive.

RAID 6

- **RAID 6 (P + Q redundancy scheme)** - Like RAID level 5 but stores extra redundant information to guard against multiple disk failures.
- To provide more recovery information error correction codes are used to compute Q.
- In the below RAID 6 example, 2 blocks of redundant data are stored for every 4 blocks of data, as opposed to just one parity block in level 5.
- This enables the system to recover from two drive failures.

(e) RAID 6: P + Q redundancy.

RAID 4, 5 and 6 Analysis

- **Reads: For a single block**
  - A block read accesses ~~only one disk~~ two disk drives.
  - Thus, the data-transfer rate for each block access is slower,
  - However, multiple read accesses can proceed in parallel, leading to a higher overall I/O rate.
- **Reads: For many blocks**
  - The transfer rates for large reads are high, since all the disks can be read in parallel.

RAID 4, 5 and 6 Analysis Cont…

- **Writes: smaller than a block –** Require significantly more time, as the OS needs to do the following:
  - First read the block to which data is to be written, and its corresponding parity block – involves 2 reads (2 disk accesses)
  - Modify the block with new data, and written back. Modify parity and write it back – involves 2 writes (2 disk accesses)
  - This is known as the read-modify-write cycle.
- **Writes: many blocks**
  - Large writes have high transfer rates, since the data and parity can be written in parallel.