CS 5/7330

Query Processing (1) –

Storage & File Structures

Query Processing

- Big question
  - Suppose I sent an SQL query to MySQL like:
    - SELECT id, name

      FROM Student, Advise, Instructor

      WHERE Student.id = Advise.student_id AND

      Advise.instructor_id = Instructor.id AND

      Advise.department = “CS” AND

      Student.gpa >= 3.0
  - This query tell the database what the answer to be
  - But does not tell the database how to get the results
    - The database management system (DBMS) need to figure it out

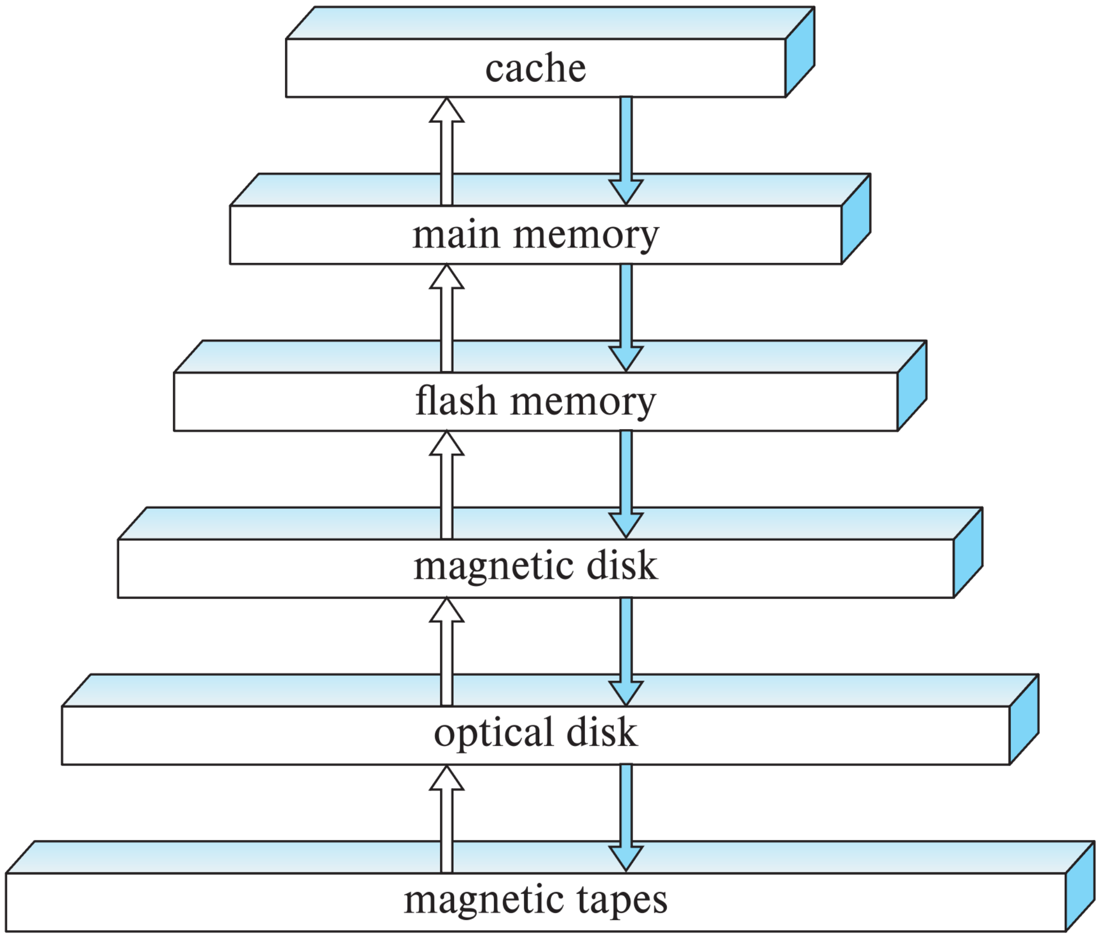
Query Processing

- To figure it out, the DBMS needs to know
  - How the data is organized
  - Where the data are
  - What are the operations involved to retrieve the data
  - The cost of the operations (so that the DBMS can pick the best one)
- We will start with how data are organized

Storage Devices

- Two main types of storage
  - **volatile storage:** loses contents when power is switched off
  - **non-volatile storage**:
    - Contents persist even when power is switched off.
    - Includes secondary and tertiary storage, as well as batter-backed up main-memory.
- Nearly all database applications will require (at least some) non-volatile storage
- Factors affecting choice of storage media include
  - Speed with which data can be accessed
  - Cost per unit of data
  - Reliability

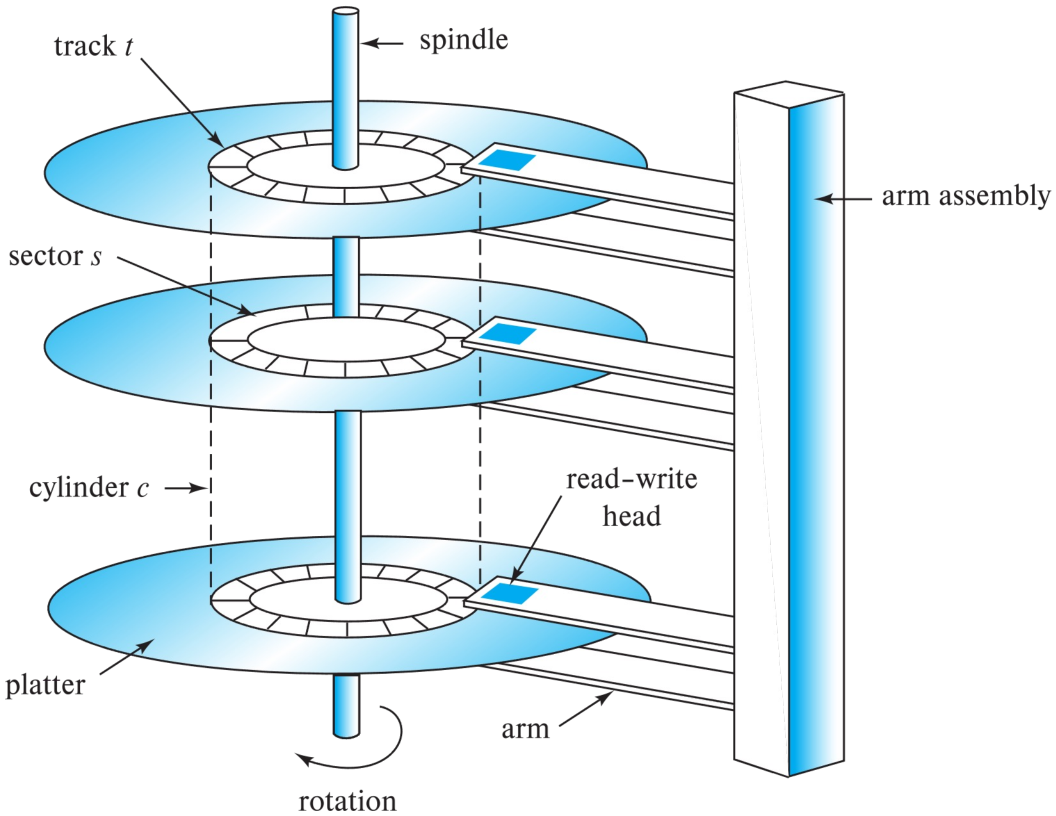
Storage Devices

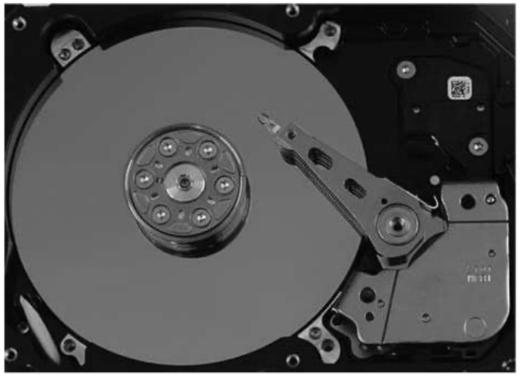


Storage Devices

- **primary storage**: Fastest media but volatile (cache, main memory).
- **secondary storage**: next level in hierarchy, non-volatile, moderately fast access time
  - Also called **on-line storage**
  - E.g., flash memory, magnetic disks
- **tertiary storage**: lowest level in hierarchy, non-volatile, slow access time
  - also called **off-line storage** and used for **archival storage**
  - e.g., magnetic tape, optical storage
  - Magnetic tape
    - Sequential access, 1 to 12 TB capacity
    - A few drives with many tapes
    - Juke boxes with petabytes (1000's of TB) of storage

Magnetic Hard Disk





Magnetic Hard Disk

- **Read-write head**
- Surface of platter divided into circular **tracks**
  - Over 50K-100K tracks per platter on typical hard disks
- Each track is divided into **sectors.**
  - A sector is the smallest unit of data that can be read or written.
  - Sector size typically 512 bytes
  - Typical sectors per track: 500 to 1000 (on inner tracks) to 1000 to 2000 (on outer tracks)
- To read/write a sector
  - disk arm swings to position head on right track
  - platter spins continually; data is read/written as sector passes under head
- Head-disk assemblies
  - multiple disk platters on a single spindle (1 to 5 usually)
  - one head per platter, mounted on a common arm.
- **Cylinder** *i* consists of i^{th} track of all the platters

Magnetic Hard Disk

- **Disk controller** – interfaces between the computer system and the disk drive hardware.
  - accepts high-level commands to read or write a sector
  - initiates actions such as moving the disk arm to the right track and actually reading or writing the data
  - Computes and attaches **checksums** to each sector to verify that data is read back correctly
    - If data is corrupted, with very high probability stored checksum won't match recomputed checksum
  - Ensures successful writing by reading back sector after writing it
  - Performs **remapping of bad sectors**

Magnetic Hard Disk

- **Access time** – the time it takes from when a read or write request is issued to when data transfer begins. Consists of:
  - **Seek time** – time it takes to reposition the arm over the correct track.
    - Average seek time is 1/2 the worst case seek time.
      - Would be 1/3 if all tracks had the same number of sectors, and we ignore the time to start and stop arm movement
    - 4 to 10 milliseconds on typical disks
  - **Rotational latency** – time it takes for the sector to be accessed to appear under the head.
    - 4 to 11 milliseconds on typical disks (5400 to 15000 r.p.m.)
    - Average latency is 1/2 of the above latency.
  - Overall latency is 5 to 20 msec depending on disk model
- **Data-transfer rate** – the rate at which data can be retrieved from or stored to the disk.
  - 25 to 200 MB per second max rate, lower for inner tracks

Magnetic Hard Disk

- **Disk block** is a logical unit for storage allocation and retrieval
  - 4 to 16 kilobytes typically
    - Smaller blocks: more transfers from disk
    - Larger blocks: more space wasted due to partially filled blocks
- **Sequential access pattern**
  - Successive requests are for successive disk blocks
  - Disk seek required only for first block
- **Random access pattern**
  - Successive requests are for blocks that can be anywhere on disk
  - Each access requires a seek
  - Transfer rates are low since a lot of time is wasted in seeks
- **I/O operations per second (IOPS)**
  - Number of random block reads that a disk can support per second
  - 50 to 200 IOPS on current generation magnetic disks

Magnetic Hard Disk

- Example
  - Consider the following numbers
    - Seek time = 7 ms
    - Rotational Latency = 5ms
    - Data transfer rate = 50 MB per second
  - Reading one 4KB block
    - Time = 4K / 50M = 0.078125 ms (1/100 times of seek/rotation)
  - So reading one block = 7 + 5 + 0.078125ms = 12.078125ms
  - Reading 10 consecutive blocks same track = 7 + 5 + 10 * 0.078125 = 12.78125ms
  - Reading 10 blocks on different tracks = 10 * (7 + 5 + 0.078125) = 120.78125ms

Magnetic Hard Disk

- Implication
  - Data that are accessed together should be store "adjacent"
    - Within the same track at the minimum
    - Usually means that each table is to be stored in a file
      - Operating Systems has tools to "defragment" the files
  - Data that is accessed often should be put into memory
    - Buffering

Magnetic Hard Disk

- **Mean time to failure (MTTF)** – the average time the disk is expected to run continuously without any failure.
  - Typically 3 to 5 years
  - Probability of failure of new disks is quite low, corresponding to a "theoretical MTTF" of 500,000 to 1,200,000 hours for a new disk
    - E.g., an MTTF of 1,200,000 hours for a new disk means that given 1000 relatively new disks, on an average one will fail every 1200 hours
  - MTTF decreases as disk ages

Flash Storage

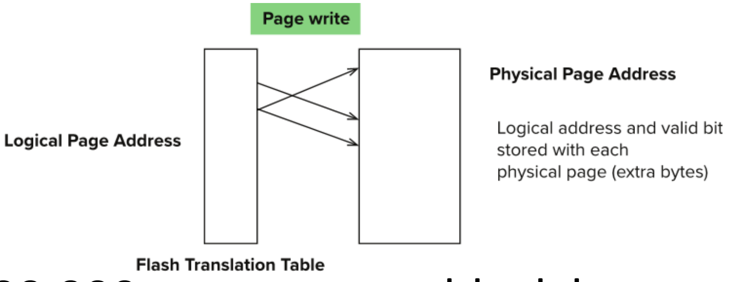
- Non-volatile storage
- Evolved from EPROM/EEPROM
- Each unit is a “gate”
- Storage can be set (to 1) by passing electricity to form a second gate in the structure – that is stable
- Applying electricity in different way can “erase” the bit and allow it to be rewritten again
    - Erase first, than rewrite
- Two types of flash – NAND and NOR (based on the logic used to set the bit)
- NAND bit are cheaper and more scalable as mass storage
    - However, slower but have to be erased in blocks
    - Each time a block need to be erased before

NAND-based storage

- NAND flash
  - used widely for storage, cheaper than NOR flash
  - requires page-at-a-time read (page: 512 bytes to 4 KB)
    - 20 to 100 microseconds for a page read
    - Not much difference between sequential and random read
    - Slower than RAM, but faster than the hard disk
  - Page can only be written once
    - Must be erased to allow rewrite
  - Solid state disks
    - Use standard block-oriented disk interfaces, but store data on multiple flash storage devices internally
    - Transfer rate of up to 500 MB/sec using SATA, and up to 3 GB/sec using NVMe PCIe

NAND-based storage

- Erase happens in units of **erase block**
  - Takes 2 to 5 millisecs
  - Erase block typically 256 KB to 1 MB (128 to 256 pages)
- **Remapping** of logical page addresses to physical page addresses avoids waiting for erase
- **Flash translation table** tracks mapping
  - also stored in a label field of flash page
  - remapping carried out by **flash translation layer**



- After 100,000 to 1,000,000 erases, erase block becomes unreliable and cannot be used
  - **wear leveling**

NAND-based storage

- Random reads/writes per second
  - Typical 4 KB reads: 10,000 reads per second (10,000 IOPS)
  - Typical 4KB writes: 40,000 IOPS
  - SSDs support parallel reads
    - Typical 4KB reads:
      - 100,000 IOPS with 32 requests in parallel (QD-32) on SATA
      - 350,000 IOPS with QD-32 on NVMe PCIe
    - Typical 4KB writes:
      - 100,000 IOPS with QD-32, even higher on some models
- Data transfer rate for sequential reads/writes
  - 400 MB/sec for SATA3, 2 to 3 GB/sec using NVMe PCIe
- **Hybrid disks**: combine small amount of flash cache with larger magnetic disk

RAID

- **RAID: Redundant Arrays of Independent Disks**
- Motivation
  - Suppose you have a hard disk with a reliability of 90%
  - You may apply engineering skills to make a more reliable one, but it tends to be expensive.
  - However, if you take two hard drives with 70% reliability (which will likely be quite a bit cheaper than the first disk) and duplicate the data
  - Then assuming independence
    - Probability of failure = $(1 – 0.7)^2 = 0.09$
    - Achieve same reliability

RAID

- What other advantage can we have with this set up?
  - Parallelism (Two process can read the disk at the same time)
- Does writing necessarily take longer?
  - Not necessarily, since we can write in parallel
  - However, scheduling constraints may affect when one can write
  - Need extra controller hardware to take care of the problem (not that expensive nowadays)
- Any other way that parallelism can help?
  - What if I request a large amount of data?
  - How data is spread around the disks are important

RAID

- The current main drawback
  - Storage efficiency is halved
    - As data is duplicated, you require double the storage
  - How to get around that problem?
    - Parity bits
    - An extra bit that is added to each byte, such that at the end the total number of 1s are even
    - So when we read a byte + the parity bit, we can check whether there is an error with the byte (assume at most one error)

RAID

- The use of multiple disks and make it looks like a single disk to the users
- Advantages
  - **high capacity** and **high speed** by using multiple disks in parallel,
  - **high reliability** by storing data redundantly, so that data can be recovered even if a disk fails

RAID levels

- Various different level proposed
- Differs in
  - Amount of redundancy
  - How data are distributed through various disks
- Basic notion
  - We assume there are N disks storing data
  - We will add K disks for redundancy purposes
    - Notice that N does not have to be equal to K
  - Various methods of organizing data on the disks (e.g. where to put the redundant data)
  - We assume chances of 2 disks failing at the same time is very small

RAID levels

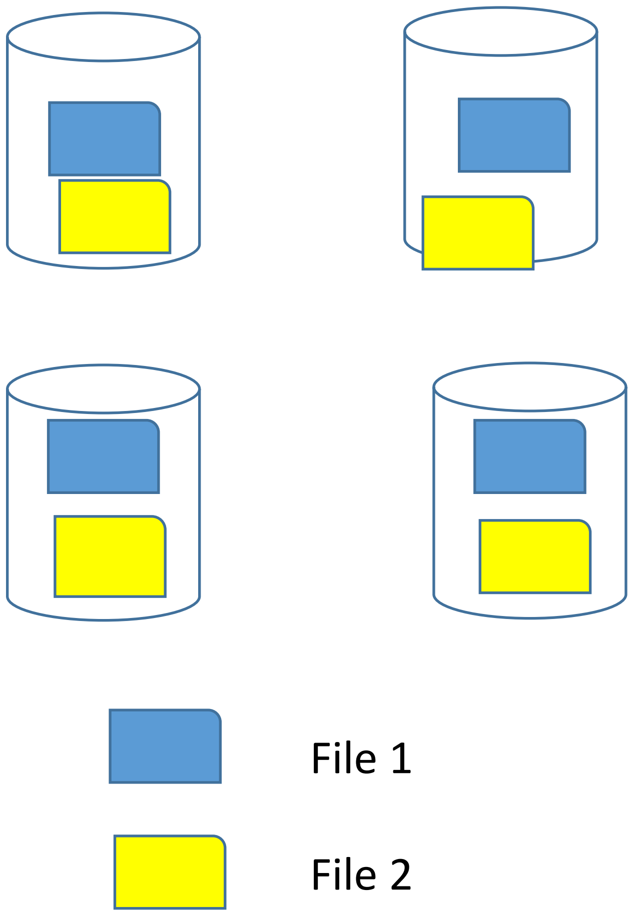
- We can measure effectiveness of various RAID level in the following way:
  - Increased Reliability: how much reliability increased
  - Overhead: How much extra disks are needed (expressed as K/N)
  - I/O performance: How does I/O perform, do parallelism helps
  - Different kind of I/O
    - Short reads/write: reading/writing a few random blocks
    - Long reads/write: reading/writing a (large) file sequentially
    - Read-modify-write: read a block, modify the content and then immediately write it back

RAID-0

- N disk in parallel
- No redundancy
  - No improvement in reliability
- How do parallelism help?
  - Short read/write: Yes, if data are spread across different disks
  - Long read/write: Doesn’t help (since file reside on a single disk
    - Can we do anything about it?

RAID-0

- Block Striping
  - With n disks, block I of a file go to disk (i mod n + 1)
  - Now long reads can be parallized
    - Potential N times performance gain



RAID-1

- N disk of data
- Block-level striping (as in RAID-0)
- Full redundancy / Mirroring
  - Every data disk is duplicated
- Overhead = 100% (N = K)
- How do parallelism help?
  - In addition to RAID-0
  - Now each block appear on two disks.
  - Two requests on the same disk can be served in parallel
    - As a side benefit, tends to shorten wait queues
  - Does not hurt writing much: as writing can be done in parallel
    - Although it is more likely that you have to seek at least one disk (even if in parallel)

RAID-2,3,4

- Problem with RAID-1: 100% Overhead
- Can we save the overhead
- Notice we assume chance of two failures is low
- So if we can store enough information to handle one error, that may be good enough

RAID-2

- Use Hamming code
- Given N bits, one can have an algorithm to add some bits (roughly logarithmic number of bits)
- If there is a single error, the Hamming code can determine which bit is the error bit and correct it automatically
- Use bit-striping (i.e. the bits of a word is striped across disks)
- The hamming code are stored in the extra disk

RAID-3,4

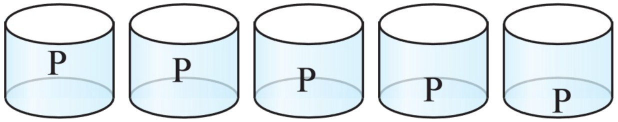
- Hamming code is overkill also
- If there is an error, usually there is some other means (e.g. look at the hardware) to determine where the error is.
- So just need to detect an error
- Use parity bit
  - Given a byte, add either a single 1 or 0 such that at the end the number of 1 bits are even
  - So now if a byte (+ parity) has odd number of 1s, then there is an error

RAID-3,4

- RAID-3: bit-striping
- RAID-4: block-striping, each parity is parity of one block on a disk (not across all disk)
  - Writes are more efficient
- Overhead: 1 / N (since one disk only to store the parity bit)
- Effeciency:
  - Reading is fine
  - Writing can be an issue
    - The disk holding the parity bit is overhead

RAID-5

- To overcome the bottleneck
- Distribute the parity info across all disks

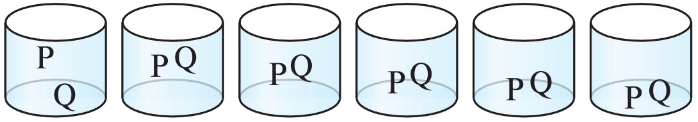


(c) RAID 5: block-interleaved distributed parity

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

RAID-6

- Similar to RAID-5
- But store more error correction blocks to help detect more errors



(d) RAID 6: P + Q redundancy

RAID -- Redundancy

- **Redundancy** – store extra information that can be used to rebuild information lost in a disk failure
- E.g., **Mirroring** (or **shadowing**)
  - Duplicate every disk. Logical disk consists of two physical disks.
  - Every write is carried out on both disks
    - Reads can take place from either disk
  - If one disk in a pair fails, data still available in the other
    - Data loss would occur only if a disk fails, and its mirror disk also fails before the system is repaired
      - Probability of combined event is very small
        - Except for dependent failure modes such as fire or building collapse or electrical power surges
- **Mean time to data loss** depends on mean time to failure, and **mean time to repair**
  - E.g., MTTF of 100,000 hours, mean time to repair of 10 hours gives mean time to data loss of $500*10^6$ hours (or 57,000 years) for a mirrored pair of disks (ignoring dependent failure modes)

RAID -- Redundancy

- **Redundancy** – store extra information that can be used to rebuild information lost in a disk failure
- E.g., **Mirroring** (or **shadowing**)
  - Duplicate every disk. Logical disk consists of two physical disks.
  - Every write is carried out on both disks
    - Reads can take place from either disk
  - If one disk in a pair fails, data still available in the other
    - Data loss would occur only if a disk fails, and its mirror disk also fails before the system is repaired
      - Probability of combined event is very small
        - Except for dependent failure modes such as fire or building collapse or electrical power surges
- **Mean time to data loss** depends on mean time to failure, and **mean time to repair**
  - E.g., MTTF of 100,000 hours, mean time to repair of 10 hours gives mean time to data loss of $500*10^6$ hours (or 57,000 years) for a mirrored pair of disks (ignoring dependent failure modes)

RAID -- Parallelism

- Two main goals of parallelism in a disk system:
  1. Load balance multiple small accesses to increase throughput
  2. Parallelize large accesses to reduce response time.
- Improve transfer rate by striping data across multiple disks.
- **Bit-level striping** – split the bits of each byte across multiple disks
  - In an array of eight disks, write bit i of each byte to disk i.
  - Each access can read data at eight times the rate of a single disk.
  - But seek/access time worse than for a single disk
    - Bit level striping is not used much any more
- **Block-level striping** – with n disks, block i of a file goes to disk $(i \bmod n) + 1$
  - Requests for different blocks can run in parallel if the blocks reside on different disks
  - A request for a long sequence of blocks can utilize all disks in parallel

RAID levels

- Factors in choosing RAID level
  - Monetary cost
  - Performance: Number of I/O operations per second, and bandwidth during normal operation
  - Performance during failure
  - Performance during rebuild of failed disk
    - Including time taken to rebuild failed disk
- RAID 0 is used only when data safety is not important
  - E.g., data can be recovered quickly from other sources

RAID levels

- Level 1 provides much better write performance than level 5
  - Level 5 requires at least 2 block reads and 2 block writes to write a single block, whereas Level 1 only requires 2 block writes
- Level 1 had higher storage cost than level 5
- Level 5 is preferred for applications where writes are sequential and large (many blocks), and need large amounts of data storage
- RAID 1 is preferred for applications with many random/small updates
- Level 6 gives better data protection than RAID 5 since it can tolerate two disk (or disk block) failures
  - Increasing in importance since latent block failures on one disk, coupled with a failure of another disk can result in data loss with RAID 1 and RAID 5.

Hardware Issues

- **Software RAID**: RAID implementations done entirely in software, with no special hardware support
- **Hardware RAID**: RAID implementations with special hardware
    - Use non-volatile RAM to record writes that are being executed
    - Beware: power failure during write can result in corrupted disk
        - E.g., failure after writing one block but before writing the second in a mirrored system
        - Such corrupted data must be detected when power is restored
            - Recovery from corruption is similar to recovery from failed disk
            - NV-RAM helps to efficiently detected potentially corrupted blocks
                - Otherwise all blocks of disk must be read and compared with mirror/parity block

Hardware Issues (Cont.)

- **Latent failures**: data successfully written earlier gets damaged
  - can result in data loss even if only one disk fails
- **Data scrubbing**:
  - continually scan for latent failures, and recover from copy/parity
- **Hot swapping**: replacement of disk while system is running, without power down
  - Supported by some hardware RAID systems,
  - reduces time to recovery, and improves availability greatly
- Many systems maintain spare disks which are kept online, and used as replacements for failed disks immediately on detection of failure
  - Reduces time to recovery greatly
- Many hardware RAID systems ensure that a single point of failure will not stop the functioning of the system by using
  - Redundant power supplies with battery backup
  - Multiple controllers and multiple interconnections to guard against controller/interconnection failures