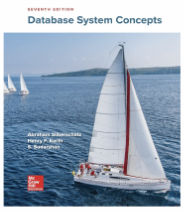
Lesson 4.3: Data Storage

UNIVERSITY OF MARYLAND

UMD DATA605 - Big Data Systems

Lesson 4.3: Data Storage



- **Instructor**: Dr. GP Saggese, gsaggese@umd.edu
- **References**
  - Silberschatz et al. 2020, Chap 12, Physical Storage Systems
  - Silberschatz et al. 2020, Chap 13: Data Storage Structures

SCIENCE ACADEMY

1 / 17

2 / 17: Storage Characteristics

Storage Characteristics

- **Storage media trade-offs**:
  - Speed of access (e.g., 500-3,500 MB/sec)
  - Cost per data unit (e.g., 50 USD/TB)
  - Medium reliability
- **Volatile vs non-volatile storage**
  - *Volatile*: loses contents when power is switched off
  - *Non-volatile*: retains contents even after power is switched off
- **Sequential vs random access**
  - *Sequential*: read the data contiguously
    `SELECT * FROM employee`
  - *Random*: read the data from anywhere at any time
    ```
    SELECT * FROM employee
    WHERE name LIKE '__a__b'
    ```

Commodore 64 cassette player

- Need to know how data is stored in order to optimize access

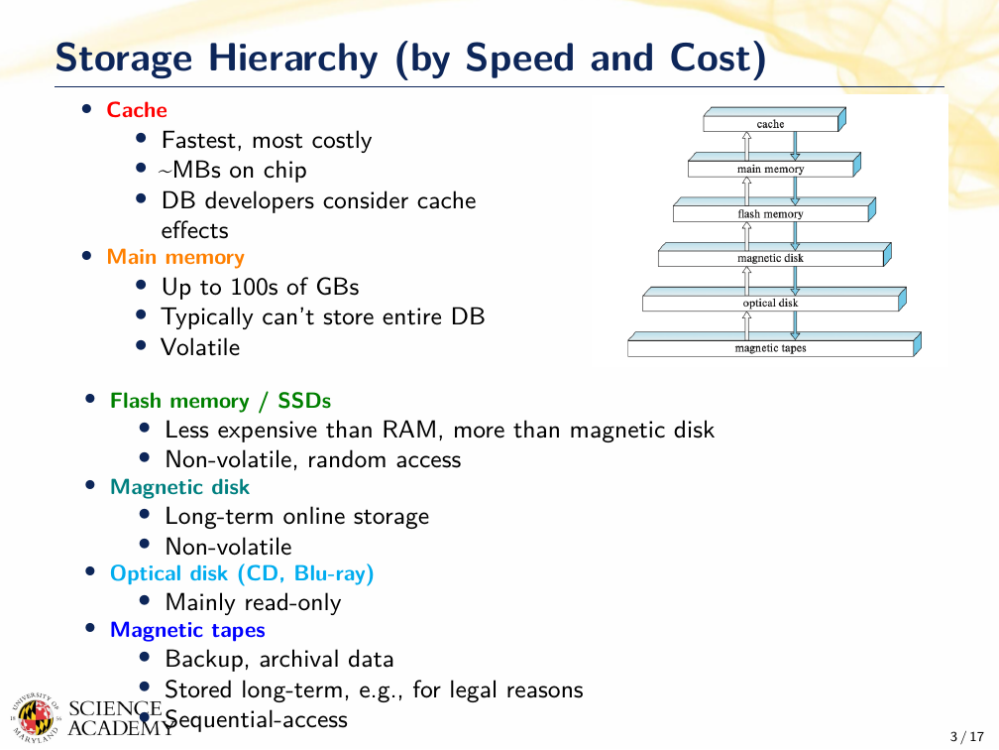
SCIENCE ACADEMY

2 / 17

- **Storage media trade-offs**:
  - When choosing storage media, there are several trade-offs to consider. **Speed of access** refers to how quickly data can be read from or written to the storage. For example, speeds can range from 500 to 3,500 MB per second. Faster access speeds are generally more desirable but can be more expensive.
  - **Cost per data unit** is another factor, often measured in terms of dollars per terabyte (e.g., 50 USD/TB). Lower costs are preferable, but they might come with compromises in speed or reliability.
  - **Medium reliability** refers to how dependable the storage is over time. More reliable storage is less likely to fail, but it might be more costly or slower.
- **Volatile vs non-volatile storage**:
  - *Volatile storage* loses its data when the power is turned off. This includes types like RAM, which are fast but temporary.
  - *Non-volatile storage* retains data even when the power is off, such as hard drives or SSDs. This makes it suitable for long-term data storage.
- **Sequential vs random access**:
  - *Sequential access* involves reading data in a continuous sequence. This is efficient for operations like reading an entire table, as shown in the example `SELECT * FROM employee`.
  - *Random access* allows data to be read from any location at any time, which is useful for queries that need specific data points, like `SELECT * FROM employee WHERE name LIKE '__a__b'`.
- **Need to know how data is stored in order to optimize access**:
  - Understanding the characteristics of storage media and access methods is crucial for optimizing data retrieval and storage efficiency. This knowledge helps in making informed

decisions about which storage solutions to use based on specific needs and constraints.

3 / 17: Storage Hierarchy (by Speed and Cost)

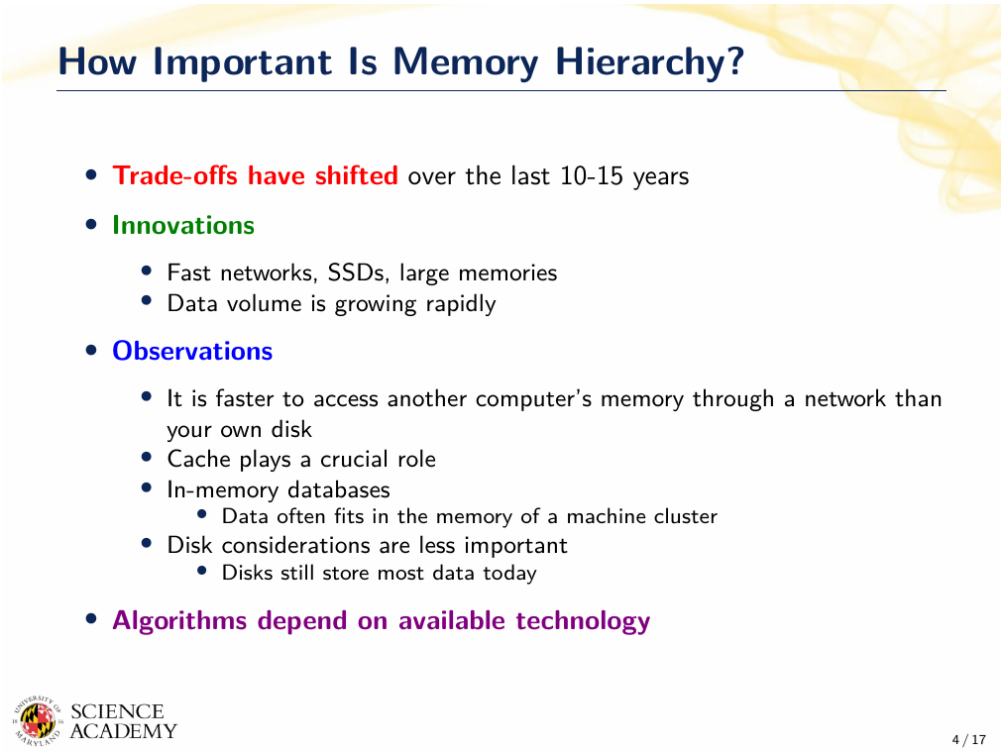


- **Cache**
  - The cache is the fastest type of storage available in a computer system, but it is also the most expensive. It is typically measured in megabytes (MBs) and is located directly on the processor chip. This proximity allows for extremely quick data access, which is crucial for performance. Database developers often need to consider how their applications interact with the cache to optimize speed and efficiency.
- **Main memory**
  - Main memory, or RAM, can store up to hundreds of gigabytes (GBs) of data. However, it is usually not large enough to hold an entire database, especially for large-scale applications. It is also volatile, meaning that it loses its data when the power is turned off, which is a critical consideration for data persistence.
- **Flash memory / SSDs**
  - Flash memory, commonly found in Solid State Drives (SSDs), is less expensive than RAM but more costly than traditional magnetic disks. It is non-volatile, meaning it retains data without power, and offers random access, which allows for faster data retrieval compared to sequential access storage.
- **Magnetic disk**
  - Magnetic disks are used for long-term online storage and are non-volatile, ensuring data is retained without power. They are slower than SSDs but are more cost-effective for storing large amounts of data.
- **Optical disk (CD, Blu-ray)**
  - Optical disks are primarily used for read-only purposes. They are not as commonly used

for active data storage due to their slower access speeds and limited rewrite capabilities.

- **Magnetic tapes**
  - Magnetic tapes are used for backup and archival purposes. They are ideal for storing data long-term, such as for legal compliance, due to their durability and cost-effectiveness. However, they are sequential-access, meaning data retrieval can be slower compared to other storage types.

4 / 17: How Important Is Memory Hierarchy?



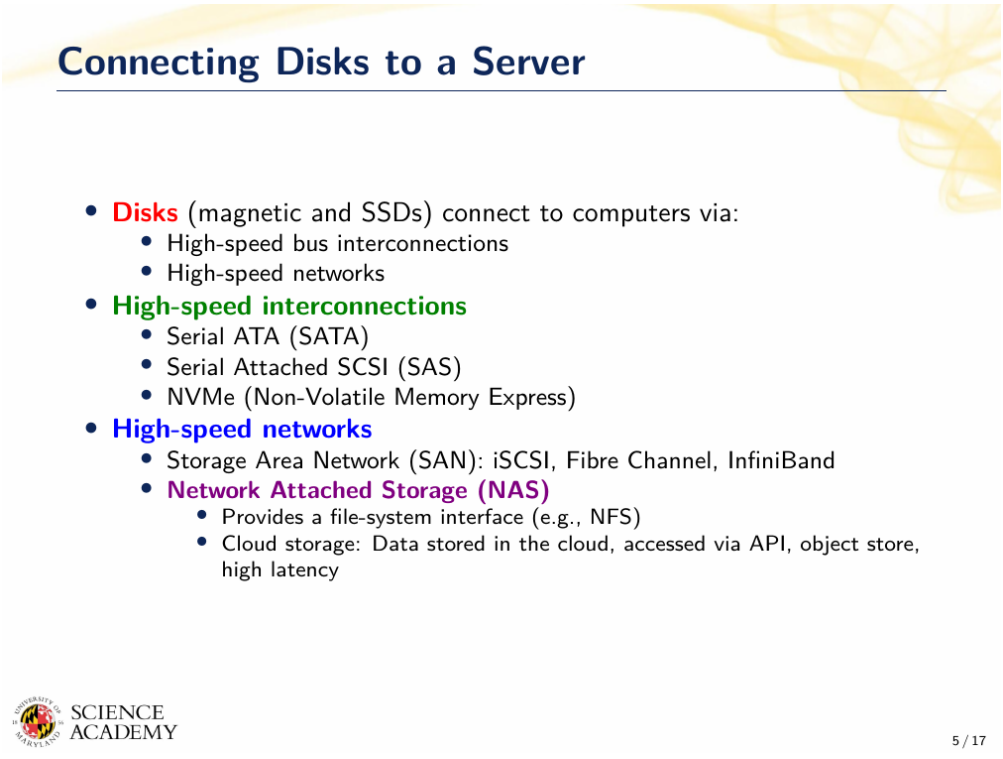
- **Trade-offs have shifted** over the last 10-15 years
  - In the past, the focus was on optimizing for slower, more limited memory resources. However, with technological advancements, the balance between speed, cost, and capacity has changed significantly.
- **Innovations**
  - **Fast networks, SSDs, large memories**: These advancements have transformed how we handle data. Fast networks allow quick data transfer between machines, SSDs provide faster data access compared to traditional hard drives, and larger memory capacities enable more data to be stored and processed in-memory.
  - **Data volume is growing rapidly**: As data generation increases, the ability to efficiently manage and process large datasets becomes crucial.
- **Observations**
  - **It is faster to access another computer's memory through a network than your own disk**: This highlights the speed advantage of networked memory access over traditional disk access, emphasizing the importance of network speed and memory capacity.
  - **Cache plays a crucial role**: Caches help speed up data access by storing frequently

accessed data closer to the processor, reducing the need to access slower memory layers.
  - **In-memory databases**: These databases store data in the main memory rather than on disk, allowing for faster data retrieval and processing. With large memory capacities, data can often fit within a machine cluster's memory.
  - **Disk considerations are less important**: Although disks still store most data, their role in immediate data processing has diminished due to faster alternatives like SSDs and in-memory processing.
- **Algorithms depend on available technology**
  - The design and efficiency of algorithms are influenced by the hardware they run on. As technology evolves, algorithms are adapted to leverage new capabilities, such as faster memory access and larger storage capacities.

Magnetic Disks / SSDs

- This section likely discusses the differences between traditional magnetic disks and modern SSDs, focusing on their impact on data storage and retrieval speeds.

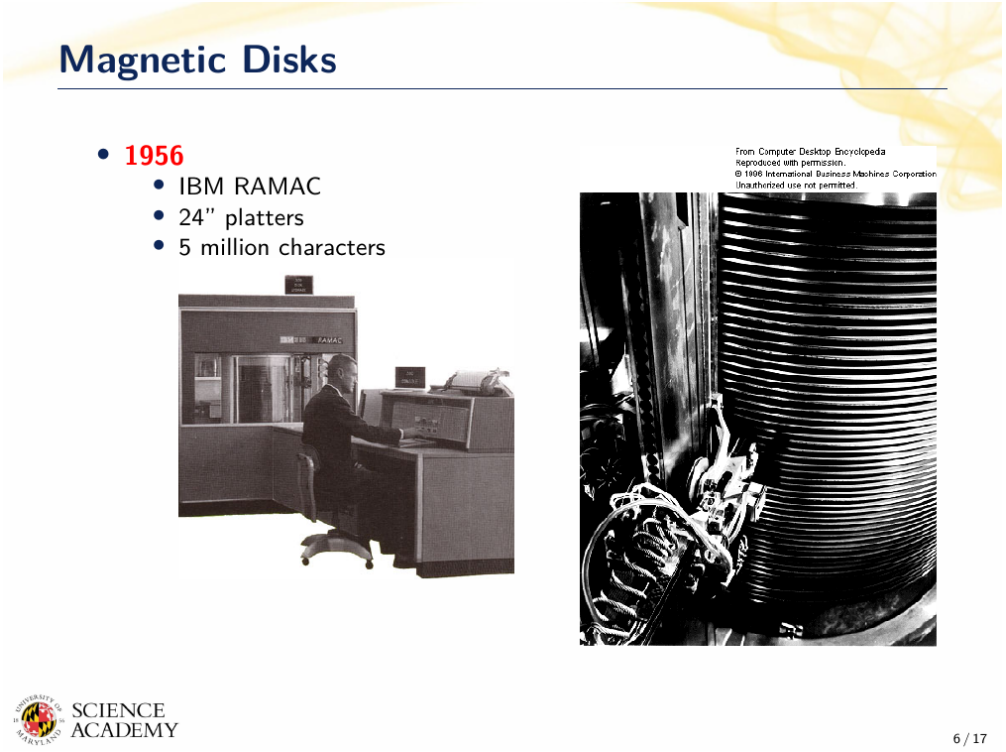
5 / 17: Connecting Disks to a Server



- **Disks** (magnetic and SSDs) connect to computers via:
  - Disks, whether they are traditional magnetic hard drives or modern Solid State Drives (SSDs), need to be connected to a computer to store and retrieve data. This connection can be made through two main methods: high-speed bus interconnections and high-speed networks. These methods ensure that data can be transferred quickly and efficiently between the disk and the computer.
- **High-speed interconnections**
  - **Serial ATA (SATA)**: This is a common interface used to connect hard drives and SSDs

to the motherboard of a computer. It is known for its reliability and is widely used in personal computers.
  - **Serial Attached SCSI (SAS)**: SAS is similar to SATA but is typically used in enterprise environments where higher performance and reliability are required.
  - **NVMe (Non-Volatile Memory Express)**: NVMe is a newer protocol designed specifically for SSDs. It provides faster data transfer speeds by connecting directly to the computer's PCIe bus, making it ideal for high-performance applications.
- **High-speed networks**
  - **Storage Area Network (SAN)**: SANs are specialized networks that provide access to consolidated, block-level data storage. They use protocols like iSCSI, Fibre Channel, and InfiniBand to connect storage devices to servers, allowing for high-speed data transfer and centralized storage management.
  - **Network Attached Storage (NAS)**: NAS devices provide a file-system interface, such as NFS (Network File System), allowing multiple users and devices to access shared storage over a network. NAS is often used for file sharing and backup solutions.
    * *Cloud storage*: This refers to storing data in the cloud, which can be accessed via APIs. Cloud storage is typically an object store, meaning it stores data as objects rather than files or blocks. While it offers scalability and accessibility, it often comes with higher latency compared to local storage solutions.

6 / 17: Magnetic Disks

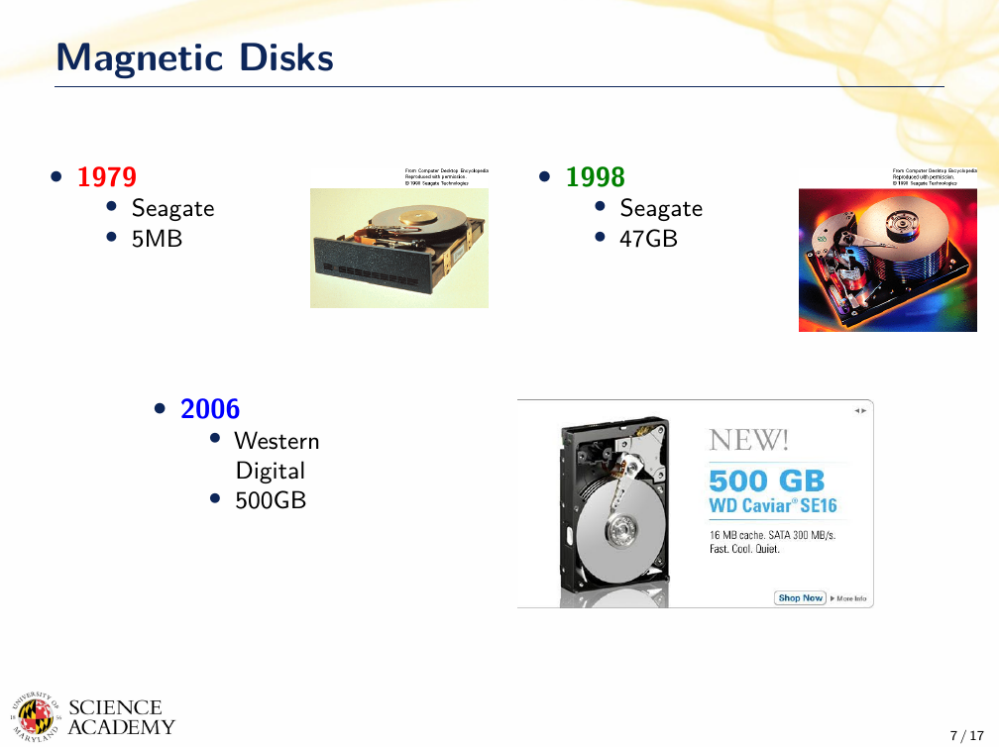


- **1956**
  - **IBM RAMAC**: This was the first computer to use a hard disk drive (HDD) for storage. The IBM RAMAC (Random Access Method of Accounting and Control) was a groundbreaking development in data storage technology. Before this, data was stored on punch

cards or magnetic tapes, which were much slower and less efficient.
  - **24" platters**: The storage medium in the IBM RAMAC consisted of large, 24-inch diameter platters. These platters were coated with a magnetic material that allowed data to be written and read by a magnetic head. The size of these platters highlights how early technology required large physical space to store relatively small amounts of data.
  - **5 million characters**: The storage capacity of the IBM RAMAC was about 5 million characters, which is roughly equivalent to 5 megabytes. While this seems minuscule by today's standards, it was a significant amount of storage at the time and represented a major advancement in the ability to store and retrieve data quickly.

The images on the slide likely depict the IBM RAMAC and its components, providing a visual context for understanding the scale and design of early magnetic disk storage systems.

7 / 17: Magnetic Disks



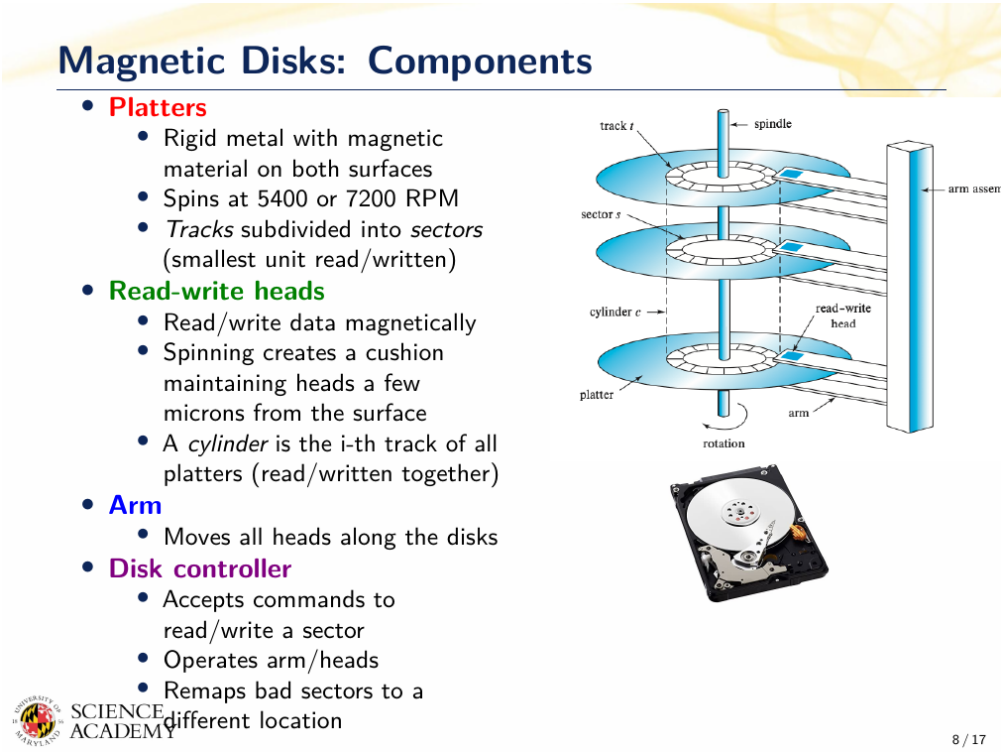
- **1979**
  - **Seagate**: This was the year when Seagate, a major player in the storage industry, introduced one of the first magnetic disks for personal computers.
  - **5MB**: The capacity of this disk was 5 megabytes, which was considered substantial at the time. To put it in perspective, 5MB is roughly equivalent to a single high-quality photo today. This highlights how storage technology has evolved over the years.
- **1998**
  - **Seagate**: Nearly two decades later, Seagate continued to innovate in the field of magnetic storage.
  - **47GB**: By 1998, the capacity of magnetic disks had increased dramatically to 47 gigabytes. This leap in storage capacity reflects the rapid advancements in technology and

the growing demand for more data storage as computers became more integral to daily life.

- **2006**
  - **Western Digital**: Another key player in the storage industry, Western Digital, made significant contributions to the development of magnetic disks.
  - **500GB**: By 2006, the capacity had reached 500 gigabytes. This increase was driven by the need to store more complex data, such as videos and large software applications, as digital technology became more sophisticated and widespread.

These points illustrate the exponential growth in storage capacity over the years, driven by technological advancements and increasing data demands.

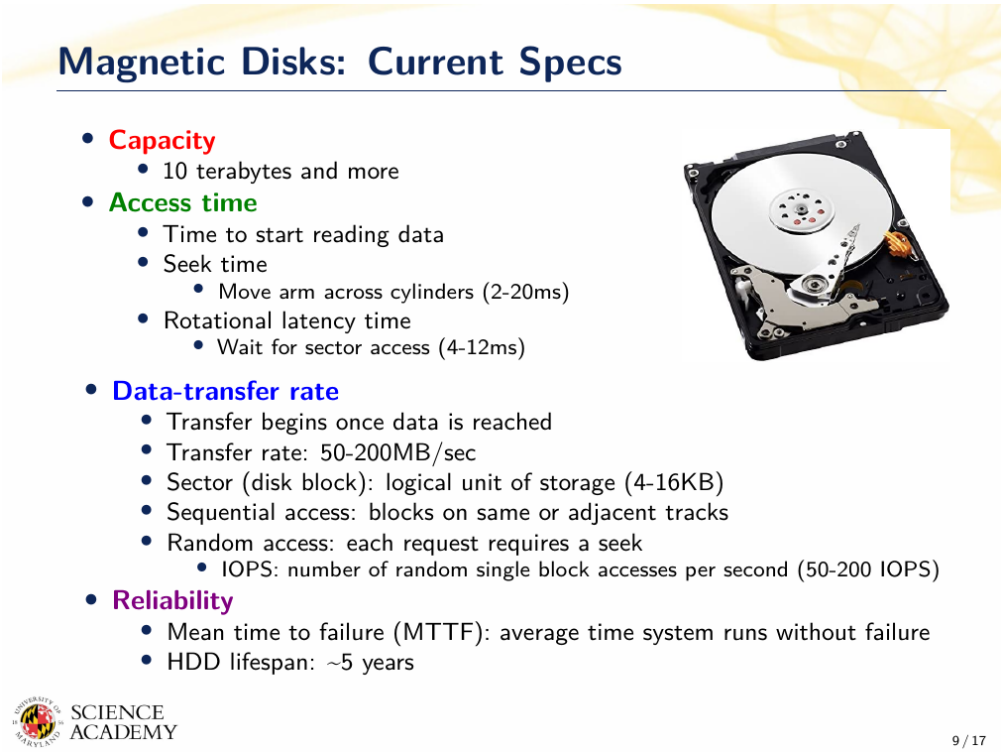
8 / 17: Magnetic Disks: Components



- **Platters**
  - Platters are the core components of a magnetic disk. They are made of a rigid metal material and are coated with a magnetic layer on both sides. This magnetic coating is crucial because it allows data to be stored magnetically.
  - These platters spin at high speeds, typically 5400 or 7200 revolutions per minute (RPM). The speed of the spin affects how quickly data can be read from or written to the disk.
  - The surface of each platter is organized into concentric circles called *tracks*. Each track is further divided into smaller sections known as *sectors*. A sector is the smallest unit of data that can be read or written on the disk.
- **Read-write heads**
  - These are the components responsible for reading data from and writing data to the platters. They do this by detecting and altering the magnetic fields on the platter surfaces.

  - As the platters spin, a cushion of air is created, which keeps the read-write heads just a few microns above the surface of the platters. This is crucial to prevent damage and ensure accurate data reading and writing.
  - A *cylinder* refers to the collection of tracks located at the same position on each platter. All tracks in a cylinder can be accessed simultaneously, which can improve data access speed.
- **Arm**
  - The arm is a mechanical component that moves the read-write heads across the platters. It ensures that the heads can access different tracks on the platters as needed.
- **Disk controller**
  - This is the electronic component that manages the operation of the disk. It receives commands from the computer to read or write data and then controls the movement of the arm and the operation of the read-write heads.
  - The disk controller also handles error management, such as remapping bad sectors. If a sector becomes unreadable, the controller can redirect data to a different, healthy sector, ensuring data integrity.

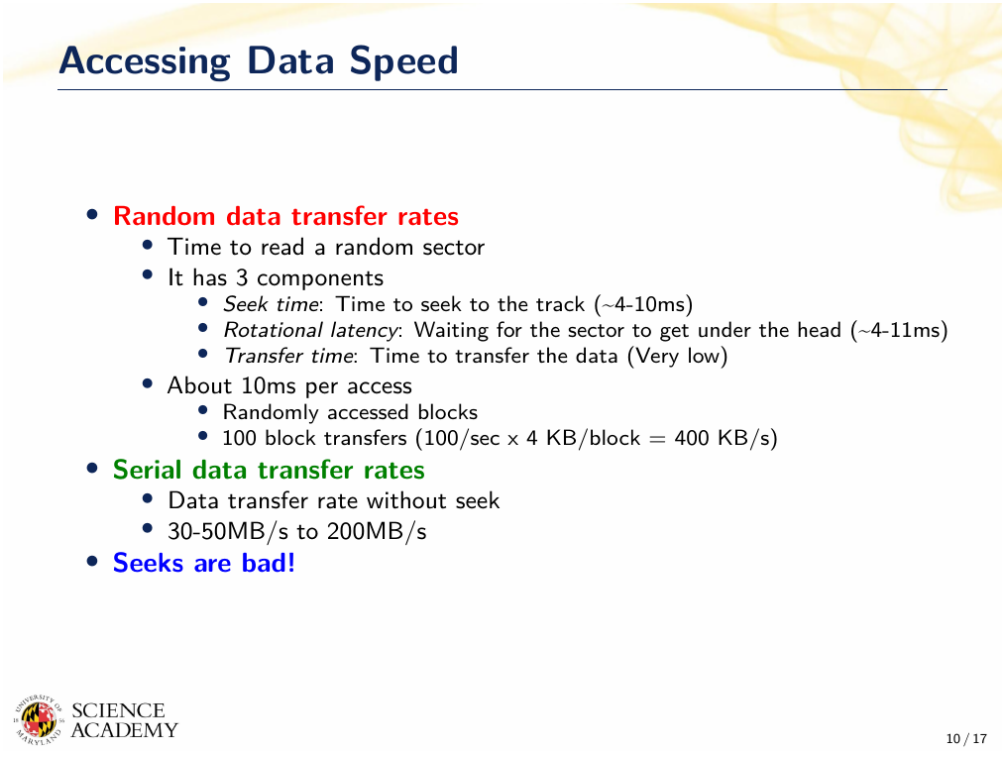
9 / 17: Magnetic Disks: Current Specs



- **Capacity**
  - Modern magnetic disks can store a *huge* amount of data, often 10 terabytes or more. This makes them suitable for applications that require storing large datasets, such as big data analytics and machine learning.
- **Access time**
  - Access time is the duration it takes for a disk to start reading data. It consists of two main components:

    - **Seek time**: This is the time it takes for the disk's read/write arm to move across the disk's cylinders to the correct position. It typically ranges from 2 to 20 milliseconds.
    - **Rotational latency time**: Once the arm is in position, the disk must wait for the correct sector to rotate under the read/write head. This waiting time usually falls between 4 to 12 milliseconds.
- **Data-transfer rate**
  - Once the data is located, the transfer begins. The rate at which data is transferred can vary from 50 to 200 megabytes per second.
  - A sector, or disk block, is the smallest logical unit of storage on a disk, typically ranging from 4 to 16 kilobytes.
  - **Sequential access** involves reading blocks that are on the same or adjacent tracks, which is faster.
  - **Random access** requires moving the read/write head to different locations for each request, which is slower. The performance of random access is often measured in IOPS (Input/Output Operations Per Second), with typical values ranging from 50 to 200 IOPS.
- **Reliability**
  - The reliability of a hard disk drive (HDD) is often measured by the Mean Time to Failure (MTTF), which is the average time the system operates without failure.
  - The typical lifespan of an HDD is around 5 years, after which the risk of failure increases. This is an important consideration for data storage solutions, especially in critical applications.

10 / 17: Accessing Data Speed



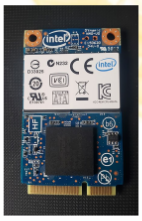
- **Random data transfer rates**
  - When we talk about reading data randomly from a storage device, we're referring to

accessing data that isn't stored in a continuous sequence. This means the device has to jump around to different locations to get the data.
  - **Time to read a random sector**: This is the time it takes to access a specific piece of data on a storage device. It involves three main components:
    * ***Seek time***: This is the time it takes for the read/write head of a hard drive to move to the correct track where the data is stored. It usually takes between 4 to 10 milliseconds.
    * ***Rotational latency***: Once the head is on the right track, it has to wait for the disk to spin around so that the correct sector is under the head. This can take another 4 to 11 milliseconds.
    * ***Transfer time***: This is the actual time it takes to move the data from the disk to the computer. This time is very short compared to the other two components.
  - In total, accessing data randomly can take about 10 milliseconds per access. If you are accessing 100 blocks of data per second, each 4 KB in size, you can transfer about 400 KB per second.
- **Serial data transfer rates**
  - When data is stored in a continuous sequence, it can be read much faster because the device doesn't have to jump around. This is called serial data transfer.
  - Without the need for seeking, data can be transferred at rates ranging from 30-50 MB/s to as high as 200 MB/s.
- **Seeks are bad!**
  - The process of seeking, or moving the read/write head to the correct track, is time-consuming and slows down data access. This is why random access is much slower than serial access. Reducing the need for seeks can significantly improve data transfer speeds.

11 / 17: Solid State Disk (SSD)

Solid State Disk (SSD)

- Mainstream around 2000s
  - Better than HDD for all metrics, more expensive per GB
  - Like non-volatile RAM (NAND and NOR)
- **Capacity**
  - 250-500 GB (vs 1-10 TB for HDD)
- **Access time**
  - Latency for random access is 1,000x smaller than HDD
    - E.g., 20-100 us (vs 10 ms for HDDs)
  - Multiple random requests (e.g., 32) in parallel
  - 10,000 IOPS (vs 50/200 for HDDs)
  - Requires reading an entire "page" of data (typically 4KB)
    - Equivalent to a block in magnetic disks
- **Data-transfer rate**
  - 1 GB/s (vs 200 MB/s for HDD)
  - Typically limited by interface speed
  - Reads and writes ~500 MB/s for SATA and 2-3 GB/s for NVMe
  - Lower power consumption than HDDs
  - Writing to SSD is slower than reading (~2-3x)
    - Requires erasing all pages in the block
- **Reliability**
  - Limit to how many times a flash page can be erased (~1M times)



SCIENCE ACADEMY

11 / 17

- **Solid State Disk (SSD)**
  - *Mainstream around 2000s*: SSDs became popular in the 2000s as they offered significant improvements over traditional Hard Disk Drives (HDDs). They are faster, more reliable, and consume less power, but they are more expensive per gigabyte.
  - *Like non-volatile RAM*: SSDs use NAND and NOR flash memory, which retains data even when the power is off, similar to non-volatile RAM.
- **Capacity**
  - SSDs typically offer capacities ranging from 250 to 500 GB, which is smaller compared to HDDs that can range from 1 to 10 TB. This is a trade-off for the speed and reliability SSDs provide.
- **Access time**
  - SSDs have significantly lower latency for random access, about 1,000 times smaller than HDDs. This means they can access data much faster, with latencies around 20-100 microseconds compared to 10 milliseconds for HDDs.
  - They can handle multiple random requests simultaneously, enhancing performance with up to 10,000 Input/Output Operations Per Second (IOPS), whereas HDDs manage only 50 to 200 IOPS.
  - SSDs read data in "pages" (typically 4KB), similar to blocks in magnetic disks, which is efficient for accessing data quickly.
- **Data-transfer rate**
  - SSDs have a data-transfer rate of about 1 GB/s, significantly higher than the 200 MB/s typical for HDDs. However, this is often limited by the interface speed.
  - For SATA interfaces, read and write speeds are around 500 MB/s, while NVMe interfaces can reach 2-3 GB/s.
  - SSDs consume less power than HDDs, making them more energy-efficient.
  - Writing to SSDs is slower than reading because it requires erasing all pages in a block before writing new data, which can be 2-3 times slower.
- **Reliability**
  - SSDs have a finite number of write/erase cycles, with each flash page capable of being erased approximately 1 million times before it may fail. This is an important consideration for the longevity of SSDs.
- **RAID**
  - The slide hints at RAID, which stands for Redundant Array of Independent Disks, a technology used to improve performance and reliability by combining multiple disk drives into a single unit.

12 / 17: RAID

RAID

- **RAID** = Redundant Array of Independent Disks
- **Problem**
  - Storage capacity is growing exponentially
  - Data-storage needs are growing even faster
  - There is a need for more disks
  - Mean Time To Failure (MTTF) between disk failures is shrinking (e.g., days)
    - A single data copy leads to an unacceptable frequency of data loss

- **Observations**
  - Disks are cheap
  - Failures are costly
  - Use extra disks for reliability
    - Store data redundantly
    - Data survives disk failure
- **Goal**
  - Present a logical view of a large, reliable disk from many unreliable disks
  - Different RAID levels balance reliability and performance

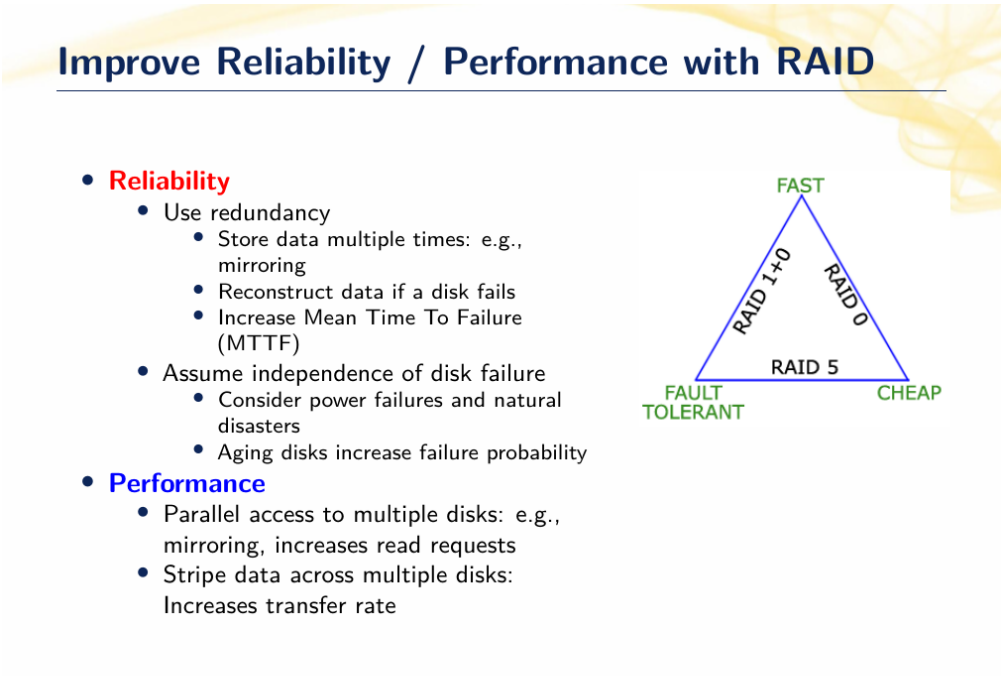
SCIENCE ACADEMY

12 / 17

- **RAID** = Redundant Array of Independent Disks
  - RAID is a technology that combines multiple disk drives into a single unit to improve data reliability and performance. It stands for Redundant Array of Independent Disks, emphasizing the use of multiple disks to store data redundantly.
- **Problem**
  - **Storage capacity is growing exponentially**: As technology advances, the amount of data we generate and need to store is increasing rapidly.
  - **Data-storage needs are growing even faster**: The demand for storing more data is outpacing the growth in storage capacity, creating a need for more efficient storage solutions.
  - **There is a need for more disks**: To meet the growing data demands, more disks are required, which can lead to increased complexity and potential for failure.
  - **Mean Time To Failure (MTTF) between disk failures is shrinking (e.g., days)**: As we use more disks, the likelihood of a disk failing increases, reducing the average time between failures.
    - *A single data copy leads to an unacceptable frequency of data loss*: Relying on a single copy of data is risky because if the disk fails, the data could be lost.
- **Observations**
  - **Disks are cheap**: The cost of individual disks is relatively low, making it feasible to use multiple disks for redundancy.
  - **Failures are costly**: The cost of data loss or downtime due to disk failure can be significant, justifying the investment in redundancy.
  - **Use extra disks for reliability**: By using additional disks, data can be stored redundantly, ensuring that it remains accessible even if one disk fails.

    * *Store data redundantly*: Data is duplicated across multiple disks to prevent loss.
    * *Data survives disk failure*: If one disk fails, the data can still be accessed from another disk.
- **Goal**
  - **Present a logical view of a large, reliable disk from many unreliable disks**: RAID aims to create the illusion of a single, large, and reliable storage system by combining multiple less reliable disks.
  - **Different RAID levels balance reliability and performance**: Various RAID configurations offer different trade-offs between data reliability and system performance, allowing users to choose the best option for their needs.

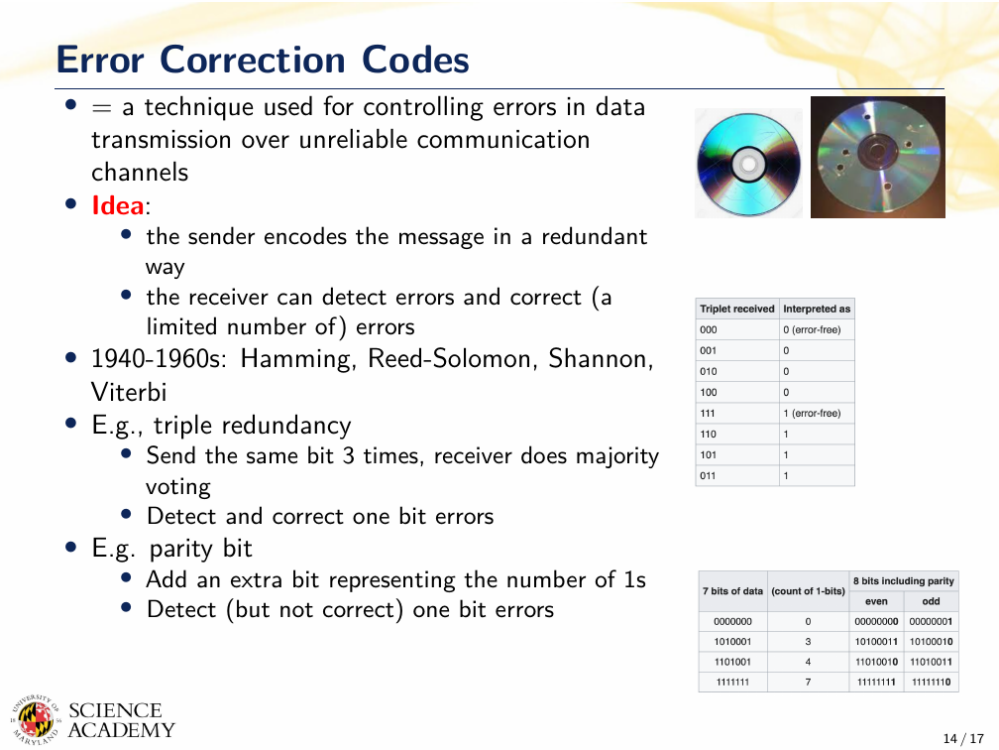
13 / 17: Improve Reliability / Performance with RAID



- **Reliability**
  - *Use redundancy*: This means having extra copies of your data. By storing data in multiple places, like with mirroring, you can protect against data loss if one disk fails. This is like having a backup plan.
    * *Store data multiple times*: Mirroring is a common method where data is copied exactly onto another disk. If one disk fails, the data is still safe on the other disk.
    * *Reconstruct data if a disk fails*: If a disk does fail, the system can rebuild the lost data using the redundant copies.
    * *Increase Mean Time To Failure (MTTF)*: This is a measure of how long you can expect the system to run before a failure happens. Redundancy helps increase this time.
  - *Assume independence of disk failure*: This means that the failure of one disk doesn't necessarily mean others will fail too. However, it's important to consider factors like

power failures or natural disasters that could affect multiple disks at once.
    * *Aging disks increase failure probability*: As disks get older, they are more likely to fail, so it's important to monitor their health.
- **Performance**
  - *Parallel access to multiple disks*: By accessing multiple disks at the same time, systems can handle more read requests. This is especially useful in setups like mirroring, where data is available on multiple disks.
  - *Stripe data across multiple disks*: This technique involves spreading data across several disks. It helps increase the speed at which data can be read or written, improving the overall transfer rate. This is like having multiple lanes on a highway, allowing more cars to travel at once.

14 / 17: Error Correction Codes

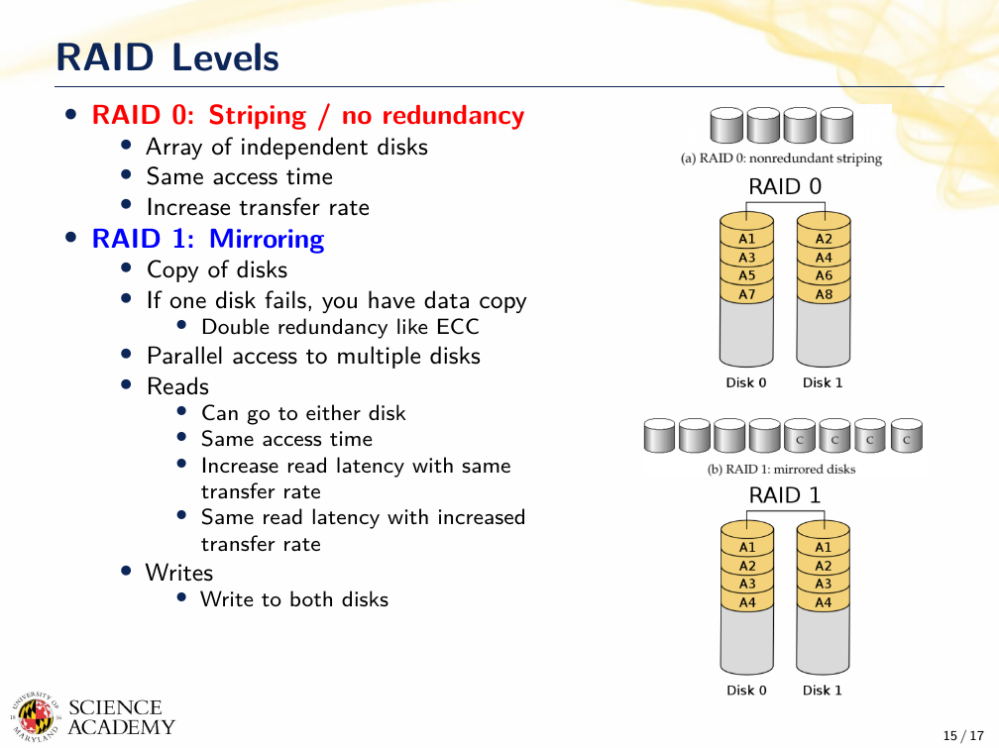


- **Error Correction Codes** are essential for ensuring data integrity during transmission over unreliable channels. These techniques help in identifying and fixing errors that may occur when data is sent from one place to another.
- **Idea**:
  - The sender encodes the message with extra information, known as redundancy, which helps the receiver identify and correct errors. This redundancy allows the receiver to detect errors and, in some cases, correct them without needing the sender to resend the data.
- **Historical Context**:
  - Between the 1940s and 1960s, significant advancements were made by researchers like

Hamming, Reed-Solomon, Shannon, and Viterbi. These pioneers developed foundational techniques that are still in use today.

- **Examples**:
  - *Triple Redundancy*: This method involves sending each bit of data three times. The receiver uses majority voting to determine the correct bit, allowing it to detect and correct single-bit errors.
  - *Parity Bit*: This simpler method adds an extra bit to the data, which indicates whether the number of 1s in the data is even or odd. While it can detect single-bit errors, it cannot correct them.
- The images on the right likely illustrate these concepts visually, showing how redundancy and error detection/correction work in practice.

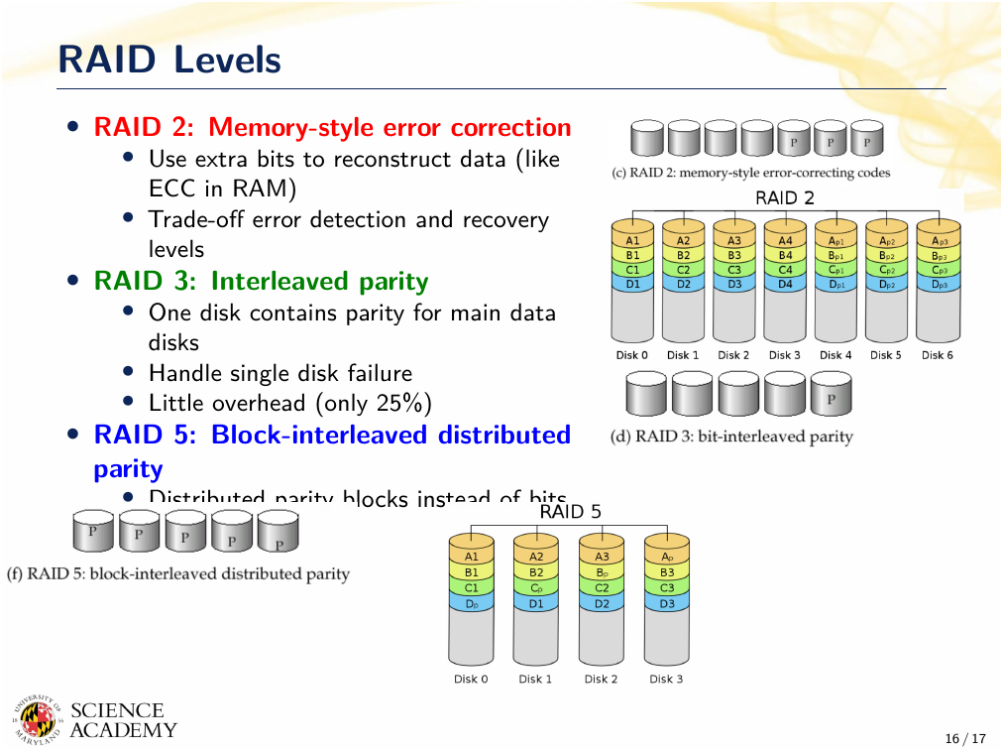
15 / 17: RAID Levels



- **RAID 0: Striping / no redundancy**
  - *Array of independent disks*: RAID 0 involves spreading data across multiple disks without any redundancy. This means that data is divided into blocks and each block is written to a separate disk.
  - *Same access time*: Since data is distributed evenly, each disk can be accessed simultaneously, leading to uniform access times.
  - *Increase transfer rate*: By using multiple disks, RAID 0 can significantly increase the data transfer rate because multiple disks can be read or written to at the same time.
- **RAID 1: Mirroring**
  - *Copy of disks*: RAID 1 duplicates the same data on two or more disks. This means that if one disk fails, the data is still safe on the other disk.

- *If one disk fails, you have data copy*: This redundancy is similar to error-correcting code (ECC) in that it provides a backup in case of failure.
- *Parallel access to multiple disks*: Both disks can be accessed at the same time, which can improve performance.
- *Reads*:
  - *Can go to either disk*: Data can be read from either disk, which can balance the load and improve read performance.
  - *Same access time*: Access time remains consistent because data is available on both disks.
  - *Increase read latency with same transfer rate*: While read latency can improve, the transfer rate remains the same because data is mirrored, not striped.
  - *Same read latency with increased transfer rate*: The redundancy allows for consistent read times while potentially increasing the transfer rate due to parallel reads.
- *Writes*:
  - *Write to both disks*: Every write operation is duplicated on both disks, ensuring data integrity but potentially slowing down write operations compared to RAID 0.

16 / 17: RAID Levels

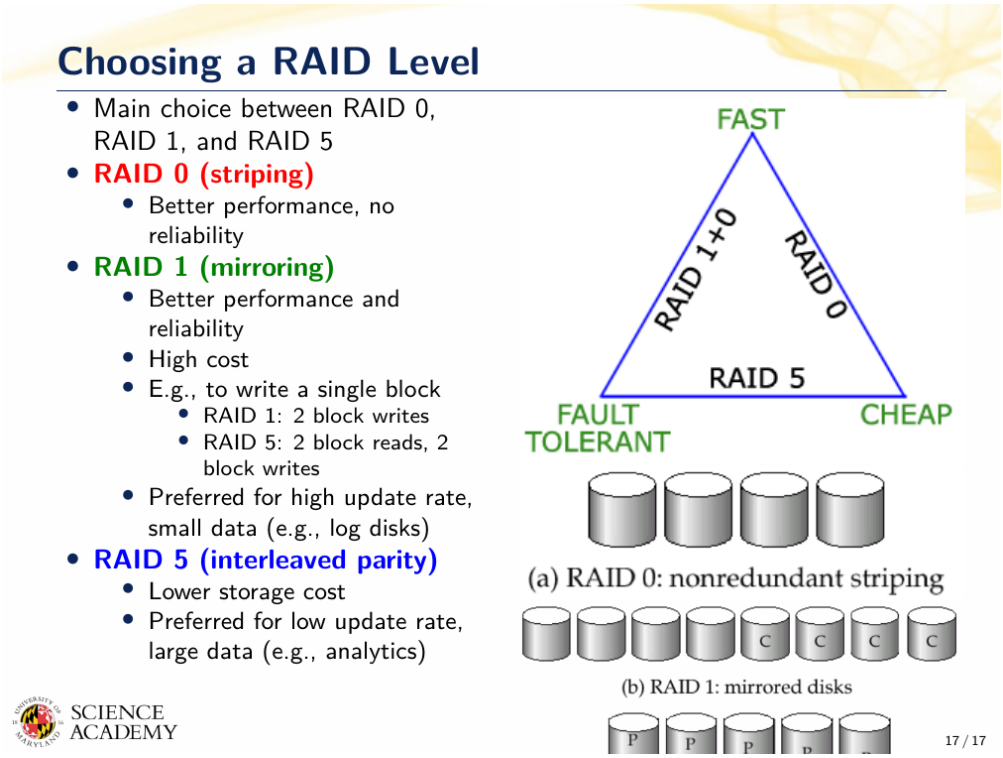


- **RAID 2: Memory-style error correction**
  - RAID 2 uses a method similar to error-correcting code (ECC) in RAM. This means it adds extra bits to the data to help detect and correct errors. This is useful for ensuring data integrity, but it can be complex and costly because it requires synchronized spinning of all disks.
  - The trade-off here is between the level of error detection and recovery you want and the cost and complexity of implementing it. RAID 2 is not commonly used today because

other RAID levels offer better performance and simpler implementations.

- **RAID 3: Interleaved parity**
  - In RAID 3, one disk is dedicated to storing parity information, which is used to recover data if one of the main data disks fails. This setup allows for the recovery of data from a single disk failure.
  - The overhead is relatively low, around 25%, because only one additional disk is needed for parity. However, RAID 3 can be limited by the single parity disk, which can become a bottleneck during data recovery.
- **RAID 5: Block-interleaved distributed parity**
  - Unlike RAID 3, RAID 5 distributes parity blocks across all disks rather than storing them on a single disk. This distribution helps balance the load and improves performance, especially during read operations.
  - RAID 5 is popular because it offers a good balance between performance, storage efficiency, and fault tolerance. It can handle a single disk failure, but rebuilding data after a failure can be time-consuming, especially with large disks.

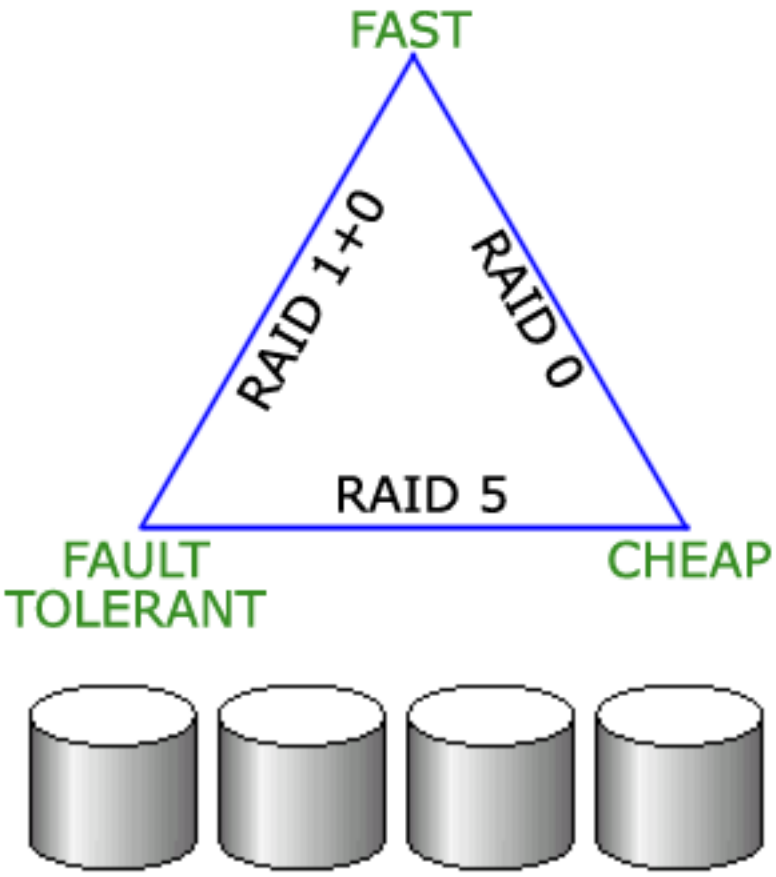
17 / 17: Choosing a RAID Level



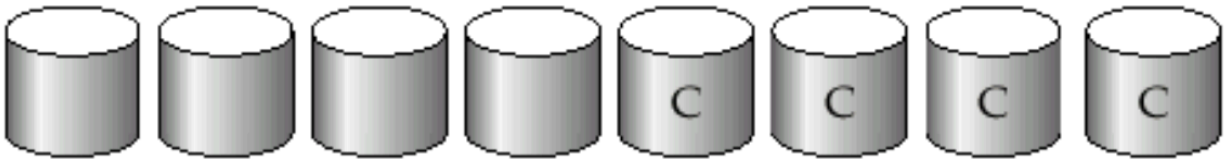
- Choosing a RAID Level :::columns ::::{.column width=50%}
- When deciding on a RAID level, the main options are **RAID 0**, **RAID 1**, and **RAID 5**. Each has its own strengths and weaknesses, and the choice depends on your specific needs for performance, reliability, and cost.
- **RAID 0 (striping)**
  - This setup improves performance by spreading data across multiple disks, allowing for faster read and write speeds. However, it offers no data redundancy, meaning if one disk

fails, all data is lost.

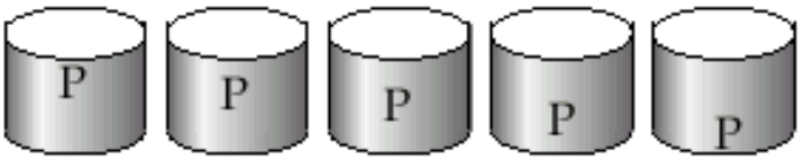
- **RAID 1 (mirroring)**
  - Provides both improved performance and data reliability by duplicating data on two disks. This means if one disk fails, the data is still safe on the other. The downside is the higher cost since you need double the storage capacity. For example, writing a single block requires writing it to both disks. RAID 1 is ideal for systems with high update rates and smaller data sizes, like log disks.
- **RAID 5 (interleaved parity)**
  - Offers a balance between cost and reliability by using parity information to recover data in case of a disk failure. It requires fewer disks than RAID 1 for the same amount of data, making it more cost-effective. RAID 5 is best suited for environments with low update rates and large data sizes, such as data analytics. :::: ::::{.column width=50%}



(a) RAID 0: nonredundant striping



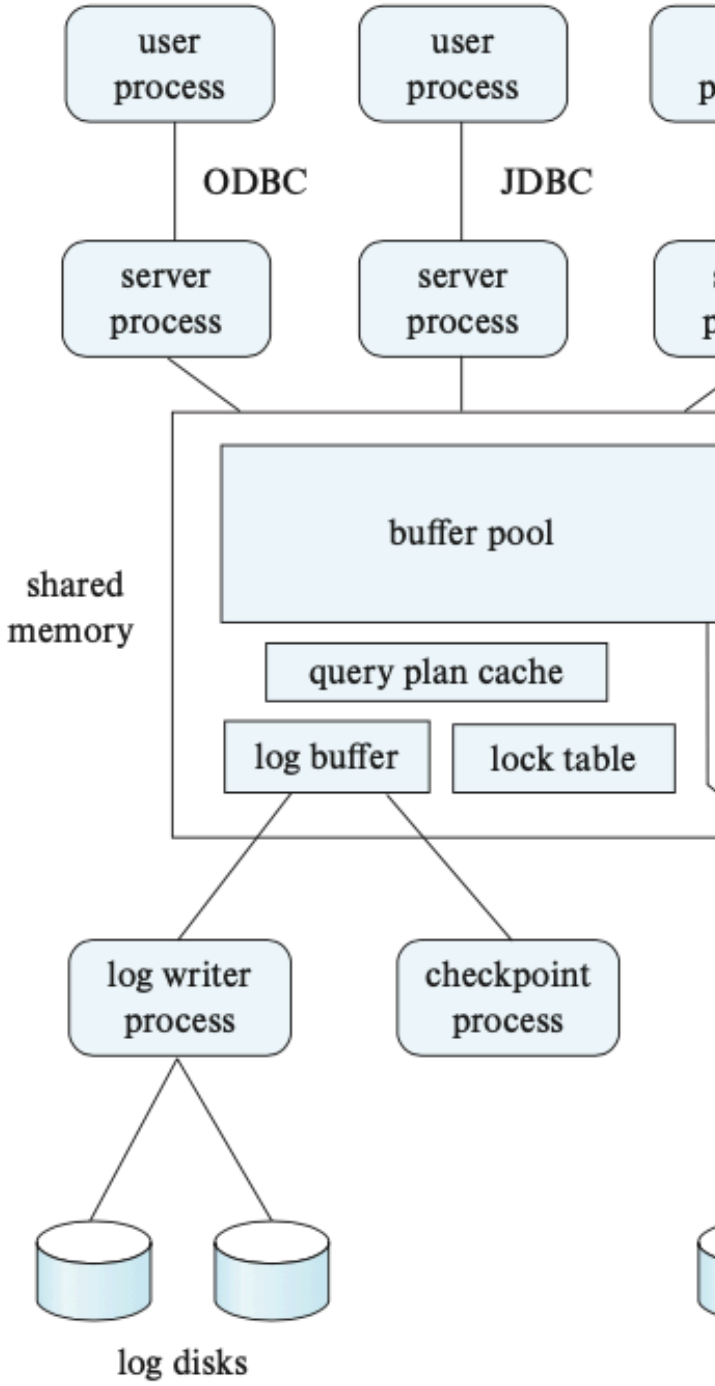
(b) RAID 1: mirrored disks



(f) RAID 5: block-interleaved distributed parity

:::: ::: - DB Internals :::columns ::::{.column width=60%} - **User processes** - These are the actions initiated by users to interact with the database, such as querying or updating data.

- **Server processes**
  - These processes handle the commands from user processes, executing the necessary database operations.
- **Process monitor process**
  - This process oversees the database operations, ensuring everything runs smoothly and recovering from any failures that occur.
- **Lock manager process**
  - Manages access to data by granting and releasing locks, and it also detects deadlocks to prevent system hang-ups.
- **Database writer process**
  - Continuously writes modified data from memory to disk to ensure data persistence.
- **Log writer process**
  - Records changes to the database in a log, which is crucial for data recovery in case of a failure.
- **Checkpoint process**
  - Periodically saves the current state of the database to minimize data loss during a crash.
- **Shared memory**
  - This is a common area where data is stored temporarily for quick access, including the buffer pool, lock table, log buffer, and caches. It uses mutual exclusion locks to protect



data integrity. :::: ::::{.column width=40%}
:::: :::

- DB Internals :::columns ::::{.column width=50%}
- **Query Processing Engine**
  - This component is responsible for executing user queries. It determines the sequence of pages to be accessed in memory and processes the data to produce the desired results.
- **Buffer Manager**
  - Manages the transfer of data pages between disk and memory, optimizing the use of

limited memory resources to ensure efficient data access.

- **Storage hierarchy**
  - Organizes data by mapping tables to files and tuples to disk blocks, facilitating efficient data retrieval and storage management. :::: ::::{.column width=50%}

```
digraph SystemArchitecture {
  graph [rankdir=TB, splines=ortho, nodesep=0.5, ranksep=0.8];
  node [fontname="Helvetica", fontsize=14, shape=box];
  edge [penwidth=2, color=blue, arrowsize=1.2];
  node [style=filled, fillcolor=yellow, width=1.5, height=0.5];
  user_query [label="user\nquery"];
  node [style="bold, rounded", color=blue, fillcolor=white, penwidth=2, width=3.5, height=
  query_engine [label="Query Processing Engine"];
  node [style=filled, fillcolor="cadetblue", width=1.5, height=0.5];
  results [label="results"];
  node [style=filled, fillcolor=yellow, width=1.5, height=0.5];
  page_requests [label="page\nrequests"];
  node [style="bold, rounded", color=blue, fillcolor=white, penwidth=2, width=3.5, height=
  buffer_manager [label="Buffer Manager"];
  node [style=filled, fillcolor="cadetblue", width=1.5, height=0.5];
  pointers [label="pointers\nto pages"];
  node [style=filled, fillcolor=yellow, width=1.5, height=0.5];
  block_requests [label="block\nrequests"];
  node [style="bold, rounded", color=blue, fillcolor=white, penwidth=2, width=3.5, height=
  space_management [label="Space Management on\nPersistent Storage"];
  node [style=filled, fillcolor="cadetblue", width=1.5, height=0.5];
  data [label="data"];
  { rank=same; user_query; results; }
  { rank=same; query_engine; }
  { rank=same; page_requests; pointers; }
  { rank=same; buffer_manager; }
  { rank=same; block_requests; data; }
  { rank=same; space_management; }
  user_query -> query_engine [style=invis];
  results -> query_engine [style=invis];
  page_requests -> buffer_manager [style=invis];
  pointers -> buffer_manager [style=invis];
  block_requests -> space_management [style=invis];
  data -> space_management [style=invis];
  user_query -> query_engine [arrowhead=normal, constraint=false];
  query_engine -> results [arrowhead=normal, constraint=false];
  page_requests -> buffer_manager [arrowhead=normal, constraint=false];
  buffer_manager -> pointers [arrowhead=normal, constraint=false];
  block_requests -> space_management [arrowhead=normal, constraint=false];
  space_management -> data [arrowhead=normal, constraint=false];
  query_engine -> page_requests [style=invis, weight=10];
  buffer_manager -> block_requests [style=invis, weight=10];
}
```

:::: :::