Storage Devices for Database Systems

These slides are mostly taken verbatim, or with minor changes, from those prepared by
Stephen Hegner (http://www.cs.umu.se/ hegner/)
of Umeå University

Bits and Bytes – Some Notation, Terminology, and Conventions

b vs B : The lower-case ending *b* is used to denote bit(s), while the upper-case ending *B* is used to denote byte(s).

K, M, G, T: These are used to identify *kilo*, *mega*, *giga*, and *tera*, respectively.

Decimal vs. binary meaning: Each of K, M, G, T, has two meanings, one decimal and one binary.

- Does 1KB mean 1000 bytes or $2^{10} = 1024$ bytes?
- Does 1MB mean 1000000 bytes or $2^{20} = 1048576$ bytes?

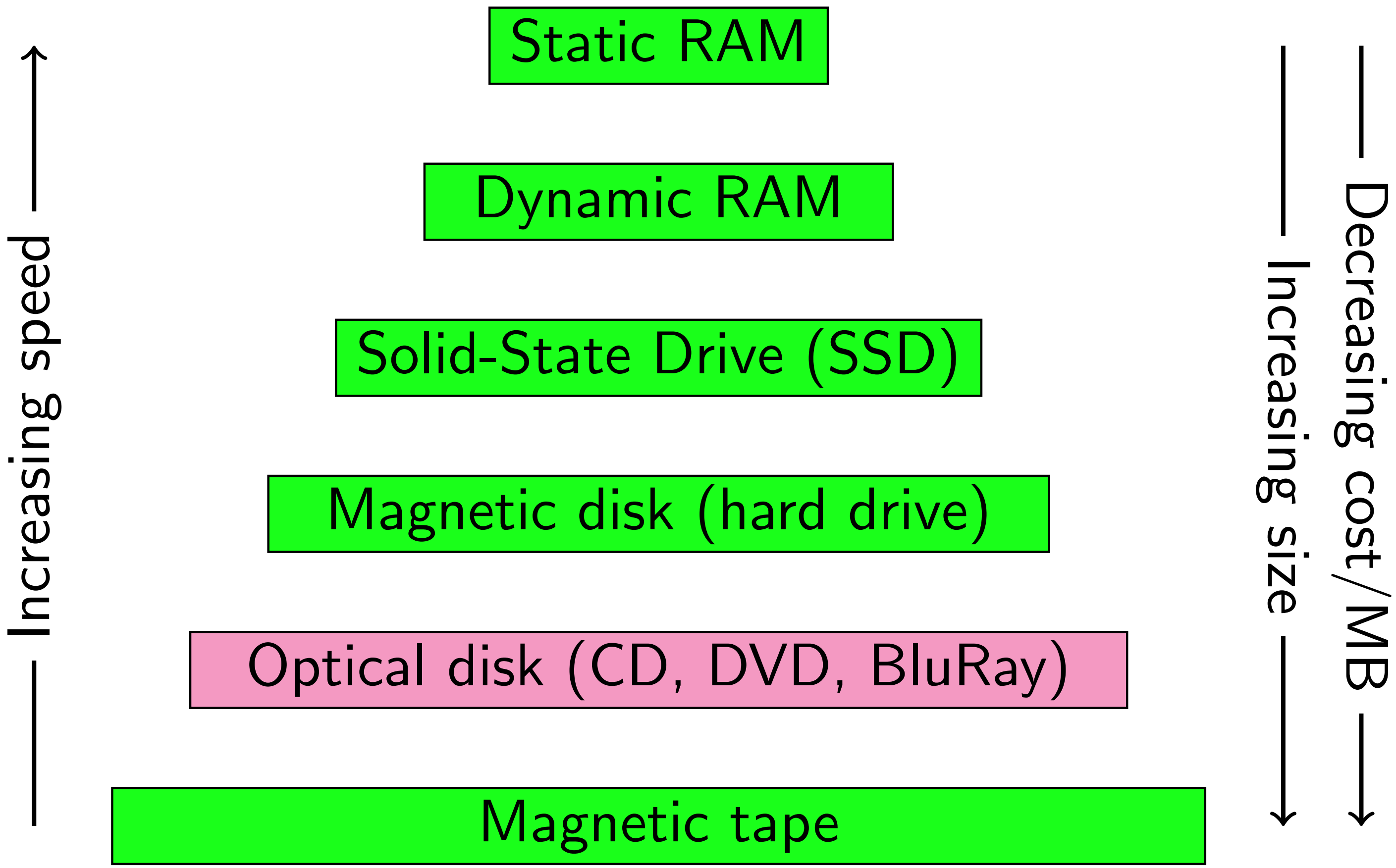
- In common usage, it depends upon context!
- In this course, the numbers will be used only in an approximate sense, so it will not matter much.

Translating bits to bytes: In working with data transfer, there is usually some encoding of byte values, so the approximation of 10 bits (not 8) per byte is often used.

Example: A SATA-2 interface with speed of 3.0Gb/s can transfer 300MB/s.

The Memory Hierarchy

- The full memory hierarchy is shown below.



- Optical disk storage is marked with a special color because it does not respect the size hierarchy.

The Central Part of the Memory Hierarchy for DBMS

- For DBMSs, the two most important parts of the memory hierarchy are identified below.

Static RAM

Dynamic RAM

Solid-State Drive (SSD)

Magnetic disk (hard drive)

Optical disk (CD, DVD, BluRay)

Magnetic tape

- The discussion in these slides will focus exclusively on these two types of memory.

Why It Is Important to Understand Performance of Hard Drives

- The amount of main dynamic RAM (random-access memory) available on even modest systems has increased rapidly in recent years.
- Nevertheless, it is far from true that databases may be moved to RAM.

Volatility: Dynamic RAM is volatile — all is lost in the event of a power failure or system crash.

- Hard-disk storage is permanent.
- Static (nonvolatile) memory is far too expensive to be used in the sizes common in modern systems.
- Hard disks are necessary for nonvolatile storage of databases.

Size: Even though RAM has become inexpensive and plentiful, many databases are terabytes in size, and some petabytes in size, which far outdistances the RAM of even cutting-edge high-end systems.

Bottom line: Hard disks will remain a central component of DBMS hardware for years to come.

Solid-State Drives

- Solid state drives are becoming larger and less expensive, and are increasingly used in laptop and even desktop computers.

Question: Will they replace mechanical hard drives in DBMS usage?

Answer: For the most part, they have not yet.

- They are currently rare in sizes beyond 1/2 terabyte (512GB).
- The cost per gigabyte is still far greater than that of spinning drives.
- They open up a whole set of new technical challenges for DBMSs.
- Access and performance issues differ greatly from both those of dynamic RAM and those of spinning drives.
- More research is required before they can be used optimally in mainstream DBMS.

Bottom line: For several reasons, they are not yet poised to replace spinning hard disks in mainstream DBMS use.

- But stay tuned, technology advances rapidly.

Speed Issues for Hard Disks

Speed issues: (Mechanical) hard disks are much slower than dynamic RAM.

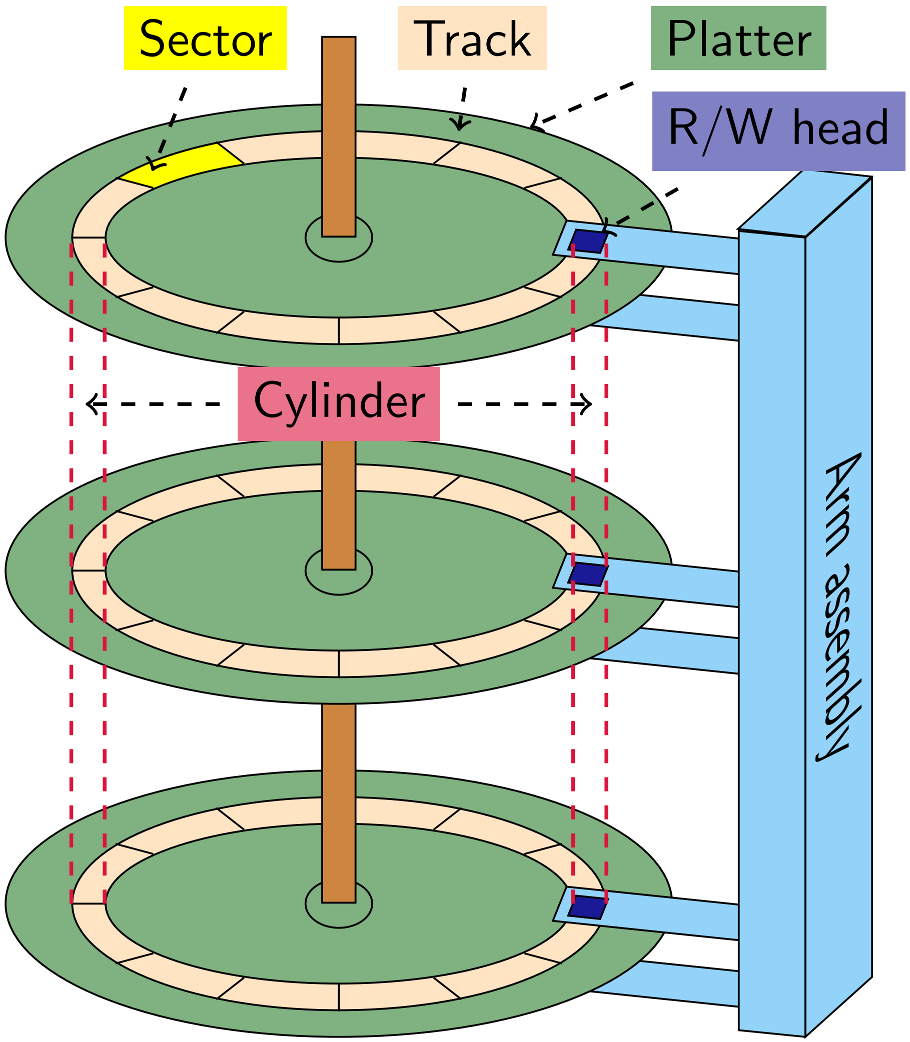
Random access: For random access, RAM is typically 1000-10000 times as fast as a hard disk.

Continuous throughput: For continuous throughput, RAM is typically at least 100 times as fast as a hard disk.

- To understand how to obtain satisfactory performance and reliability under these constraints, it is necessary to understand a bit more about hard-disk storage.

Inside a Hard Drive – the Main Parts

- A hard drive consists of a number of spinning *platters* and an *arm assembly* with one *R/W head* for each surface.
- A *surface* is one side of a platter.
- The data are recorded on a set of concentric *tracks* on each surface.
- The set of all tracks of the same diameter (one for each surface) is a *cylinder*.
- Each track is divided into *sectors*.
- The sector is the smallest amount of data which may be accessed individually at the <u>internal</u> level of the drive.



Typical Physical Parameters for Hard Drives

Platter diameter: 3.5 inches (8.75 cm) for a full-size drive and 2.5 inches (6.25 cm) for a laptop drive.

Speed of rotation:

- 4200-5400 rpm for a laptop drive.
- 5400-7200 rpm for a desktop drive.
- 7200-15000 rpm for high-performance drives.

Number of platters: Rarely more than four.

Sector size:

- 512 and 2048 bytes has been standard for a long time.
- Some newer drives have higher values (*e.g.*, 4096 bytes).

Total storage size:

- Laptop drives up to 1TB.
- Desktop drives up to 4TB.
- High-performance drives are typically much smaller.

Operational Parameters for Hard Drives

- Hard drives are mechanical devices, and their speed is limited by two mechanical parameters.

Seek time: The time required to position the R/W heads over the correct cylinder. Worst-case times:

  - Typically 12ms-15ms for laptop drives.
  - Typically 8ms-9ms for desktop drives.
  - As low as 4ms for very high-performance drives.
  - Reading usually requires a little less time than writing.
  - Average-case times are substantially better.

Rotational latency: The time required for the disk to spin to the correct sector, once the heads are over the correct cylinder.

  - May be computed from the rotational speed; average is for 1/2 revolution.
  - About 7ms at 4200 rpm; 4ms at 7200 rpm; 2ms at 15000 rpm.

- Note that these times are in *milliseconds*, while computer clocks operate at the sub-*nanosecond* level.

Hard-Drive Speed

Internal buffer: Modern hard drives have an internal buffer (also called a *cache*), typically 16MB-64MB in size.

Three speed measurements:

Buffer to Memory: This is the speed of the channel between the drive and the computer.

- SATA-2 has 3.0Gb/s (300MB/s).

Disk to buffer: This is the speed at which the drive can transfer data from the platters to the buffer.

- A little over 100MB/s seems to be a common upper limit.

Random-access time: This is the total time required to fetch one data block (sector) and send it to memory.

- The primary physical factors limiting this parameter are seek time and rotational latency.
- The typical values therefore lie in the <u>milli</u>second range.

Hard-Disk Access and DBMSs

- Although it is sometimes possible to arrange things so that fast transfers (limited by disk-to-buffer or even buffer-to-memory parameters) are possible, it is not possible to optimize for all queries.
- Thus, it is critical to address random-access time in any DBMS configuration.
- An additional, secondary issue is reliability.
- The failure of a single drive should not result in loss of the database.
- In the following slides, some ways to deal with these issues are presented.

RAID

RAID = Redundant Array of Inexpensive Disks
Redundant Array of Independent Disks

Goals: RAID involves one of, or a combination, the following two ideas:

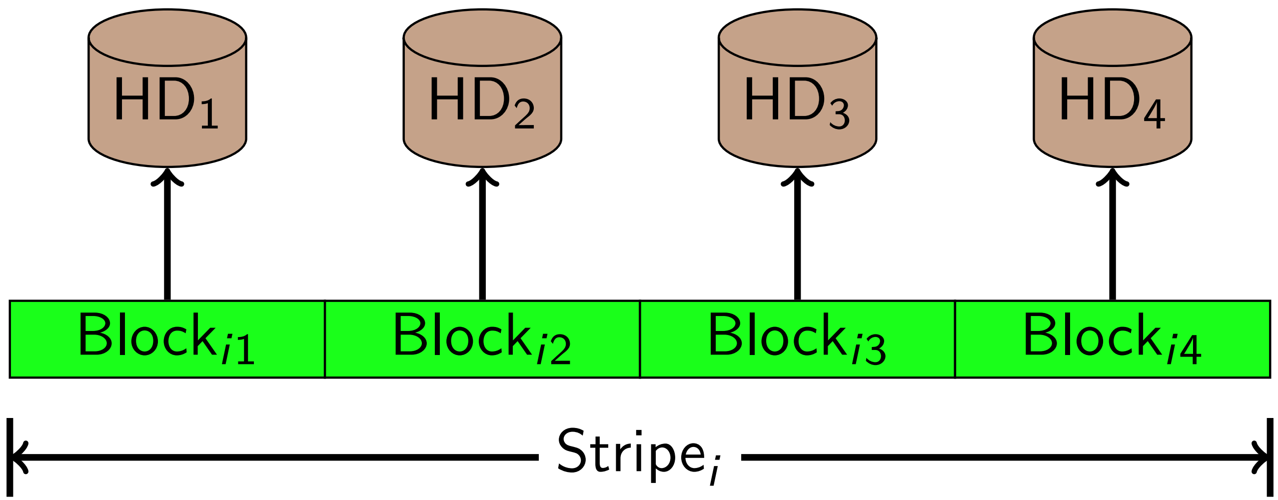
- Distributed Replication of the same data over several drives for redundancy.
- Distribution of the data over several drives, via a technique known as *striping*, for enhanced performance.

Classification terminology: The original classification scheme, which is still in wide use, identifies configuration types by number.

  - Type n RAID, for $0 \leq n \leq 6$.

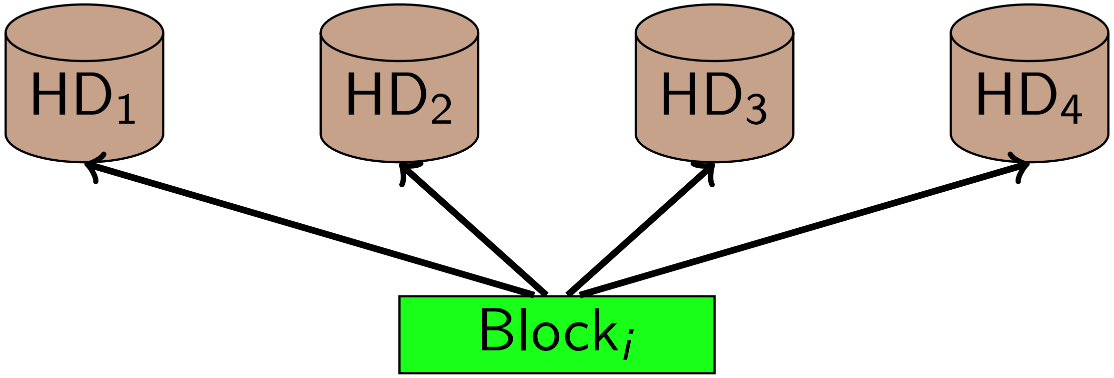
- All except type 0 RAID involve replication for redundancy.
- All except type 1 RAID involve striping.
- Hybrid types, such as 0+1 and 1+0, are also used.

Type 0 RAID: Striping



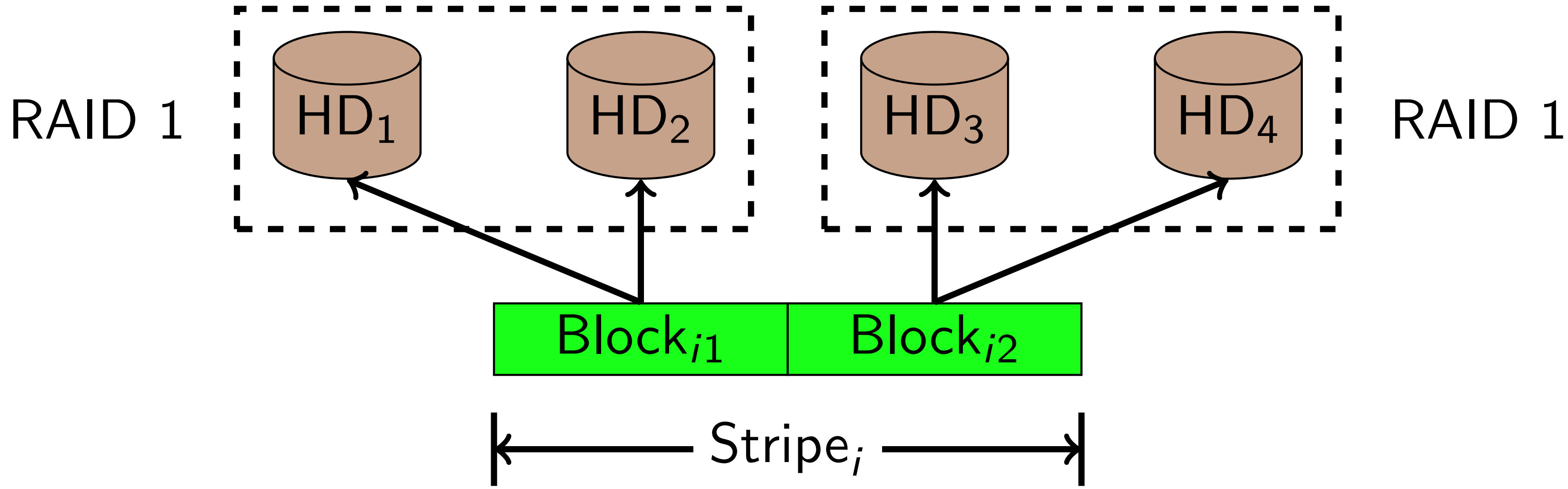
- Type 0 RAID is illustrated above for $n = 4$ drives.
- It involves only striping, there is no replication for redundancy.
- The data are divided into "superblocks" called *stripes*.
- Each stripe is divided into n blocks, with each block of the stripe stored on a distinct drive.
- All drives must be identical (disk geometry).
- There is a theoretical speedup factor of n over a single drive.
- If any drive fails, all data are lost.

Type 1 RAID: Replication



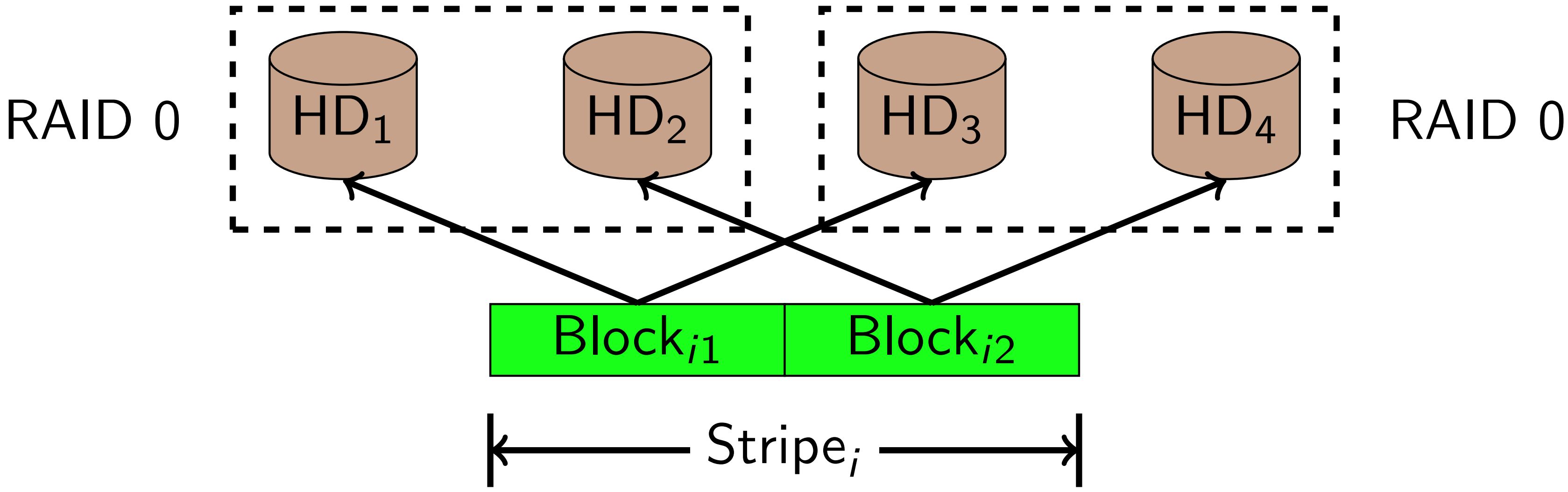
- Type 1 RAID is illustrated above for $n = 4$ drives.
- It involves only replication, there is no striping for performance.
- Each block is written to all drives.
- All drives should the same size, but there is not a strict requirement for identical geometry as in the case of Type 0 RAID.
- There is no speedup; in fact, there may be a modest performance reduction over a single drive.
- As long as one drive remains working, no data are lost.

Type 1+0 RAID: Striping of Replication



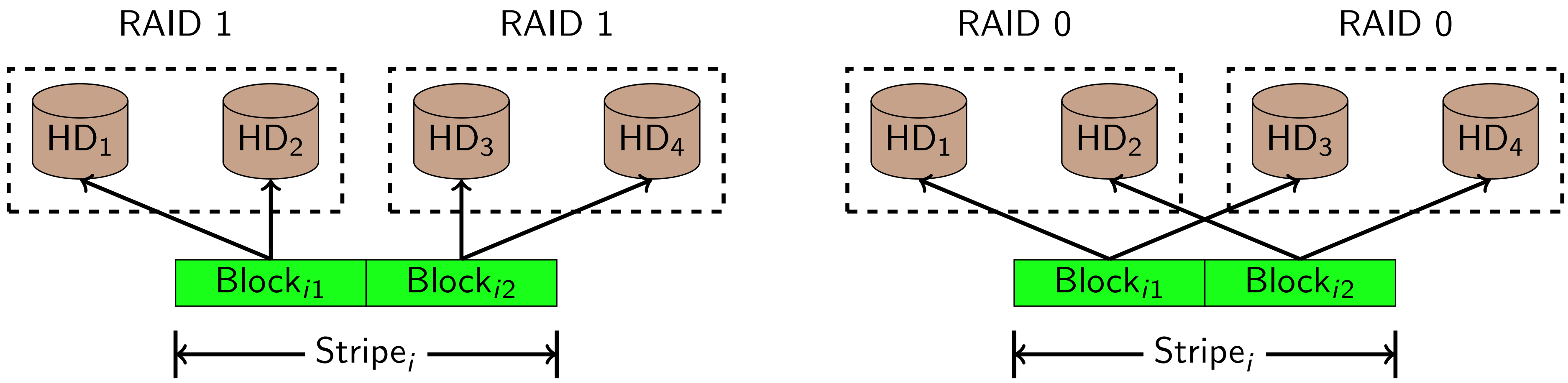
- Type 1+0 RAID (also Type 10) is illustrated above for $n_{\text{Type1}} = 2$ and $n_{\text{Type0}} = 2$ drives.
- Each of the two blocks in a stripe is written to a Type 1 RAID array.
- The minimum number of drives is four.
- As long as one drive in each RAID 1 bank remains operational, no data are lost.
- This solution provides both redundancy and and speedup, and is very widely used in DBMS practice.

Type 0+1 RAID: Replication of Striping



- Type 0+1 RAID (also Type 01) is illustrated above for $n_{\text{Type1}} = 2$ and $n_{\text{Type0}} = 2$ drives.
- It consists of two parallel RAID 0 arrays.
- The minimum number of drives is four.
- As long as one of the RAID 0 banks remains fully operational, no data are lost.
- It presents advantages similar to those of RAID 10.

Comparison of Type 10 and Type 01 RAID



Failure of a single drive: Always tolerated in both configurations.

Failure of two drives: There are six ways to choose two drives out of four.

Type 10: For four of these six combinations, there will be no loss of data.

- Failure only occurs when both drives in a RAID 1 bank fail.

Type 01: For two of these six combinations, there will be no loss of data.

- Failure occurs when one drive in each RAID 0 bank fails.

- Similar results hold for larger numbers of drives.
- From the point of view of tolerating drive failure, Type 10 is preferable to Type 01.

The Argument for Type 1 and Type 10 RAID

Advantages: Type 10 (and Type 1) RAID offers many advantages and so is widely used in practice.

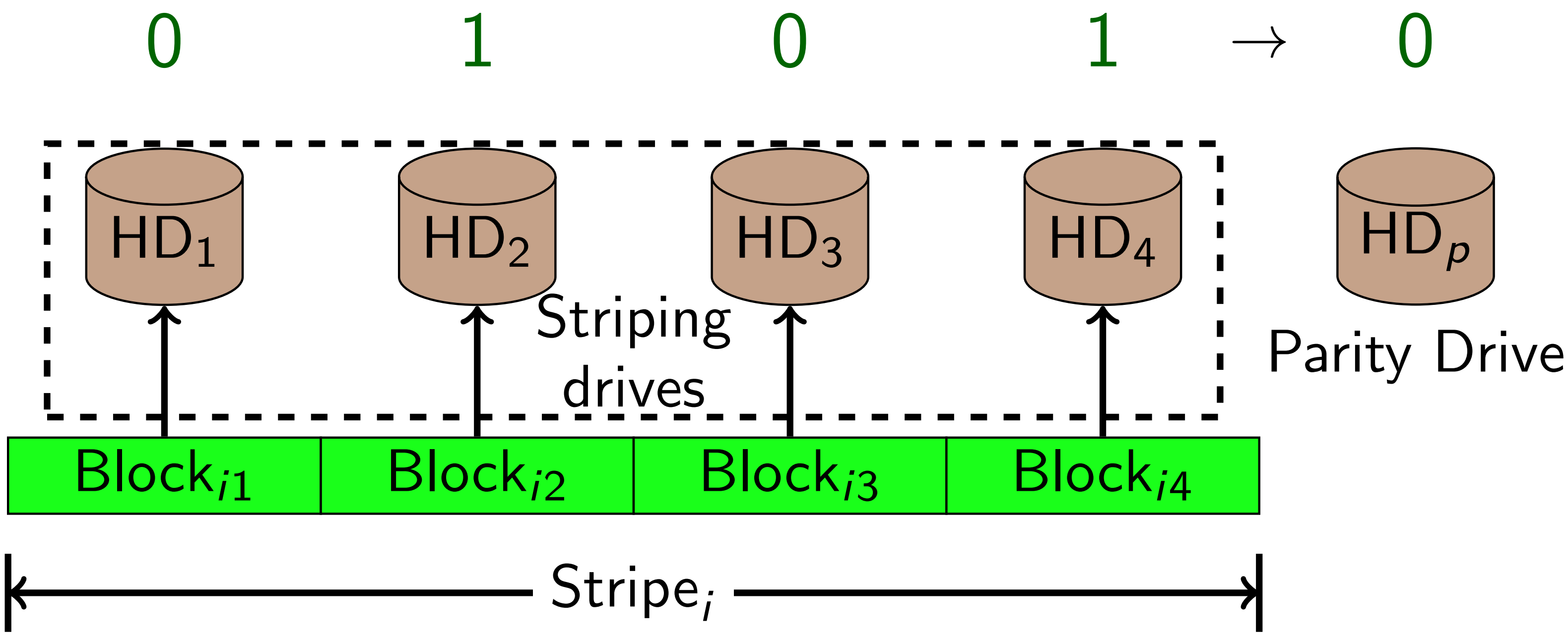
- No loss of performance with a tolerable drive failure.
- With a suitable controller, the possibility to hot swap out a failed drive exists.
- Because of the simple symmetry of replicating drives, hot rebuilding of a new replacement drive does not require excessive computation and may be carried out quickly in the background.

Disadvantages: There is however, an apparent disadvantage.

- Adding the minimal extra replication requires n additional drives, where n is the number of stripe drives.

- For this reason, a number of other solutions have been considered.

Type 3 RAID



- In Type 3 RAID, a *parity drive* is used to record the bitwise parity of the striping drives.
- With the loss of a single striping drive, the parity drive may be used to recover the missing data.
- Thus redundancy is achieved without replication, via one additional drive.

Disadvantages:

- Loss of more than one drive results in non-recoverability.
- The parity drive creates a write bottleneck.

Summary of Types 2, 4, 5, and 6 RAID

Type 2: uses a Hamming code rather than simple parity to achieve redundancy.

- This requires $\lceil \log(n) \rceil$ additional drives, with n the number of striping drives.
- A single drive failure is always correctable.
- Multiple failures may be detectable but not correctable.
- The bottleneck created by the additional drives remains.

Type 4: is similar to Type 3, but it uses block-level rather than byte-level parity checking.

- Offers some performance advantages over Type 2.

Type 5: uses parity checking, but instead of using a separate parity drive, the parity information is spread out over the striping drives.

- The problem of the bottleneck created by the additional drive is avoided, at least to some degree.

Type 6: is similar to Type 5, but uses double parity checking to provide recovery from multiple drive failures.

Choice of RAID Type in DBMS Practice

- Generally, Type 10 (or Type 1 if no performance enhancement is necessary) is generally the best choice.
  - The amount of redundancy may be chosen to meet the needed level of protection against failure.
  - Since redundancy is achieved via drive-level replication:
    - There is no loss of performance with drive failure.
    - The redundancy does not introduce significant performance degradation.
    - Hot swapping is possible with suitable controllers.
  - The main disadvantage of Type 10 RAID is that it uses many drives, which may be a cost issue.
- There are recent claims that Types 5 and 6 RAID do not result in significant performance degradation either, but this is controversial.
  - They are nevertheless more limited in terms of achieving high failure tolerance and hot swapping.
- At this time, RAID 10 (including RAID 1) is the main choice for DBMSs.

Memory Issues in Real DBMSs

Common misconception: Somewhat surprisingly, modern DBMSs are typically not I/O bound in many cases.

Why?: Most disk access is not random, but rather well organized via tuning to:

- keep in memory that which is likely to be needed again;
- bring into memory what is likely to be needed via predictive strategies.

The real bottleneck: Increasingly, the memory-processor speed mismatch is becoming an issue for DBMSs, just as it is for other applications.

- Processor speed is increasing more rapidly than memory speed.

Bottom line: Correct DBMS tuning has become paramount.

- People who know to to tune given DBMS well are in very high demand.

Disk Configuration and Allocation in Real DBMSs

Access method: There are at least two distinct ways to provide disk access for a DBMS:

Raw-disk: The DBMS accesses its own disk(s) via a low-level OS interface.

- \+ Effective, and offered by most commercial DBMSs.
- – Such interfaces are often OS specific, which can make portability across OSs more complex.
- – Can create performance issues when used in conjunction with RAID and other virtual storage strategies.
- – Requires a dedicated drive (only an issue for small systems).

Via a large file or files provided by the OS:

- \+ Simple to implement and portable.
- \+ With large files, physical disk parameters are preserved and so can approximate raw access in that regard.
- – The *double-buffering* problem must be avoided.
  - The data are transferred twice, once between the disk and the OS, and once between the OS and the DBMS.