Storage DB internals

Instructor: Dr. GP Saggese - gsaggese@umd.edu**
TAs: Krishna Pratardan Taduri, kptaduri@umd.edu Prahar Kaushikbhai Modi, pmodi08@umd.edu
with thanks to Prof. Alan Sussman Prof. Amol Deshpande
v1.1

UMD DATA605 - Big Data Systems

Outline

- Storage
  - Physical storage
    - Storage hierarchy
    - Magnetic disks / SSDs
    - RAID
- DB internals Sources: Silberschatz et al. 2020

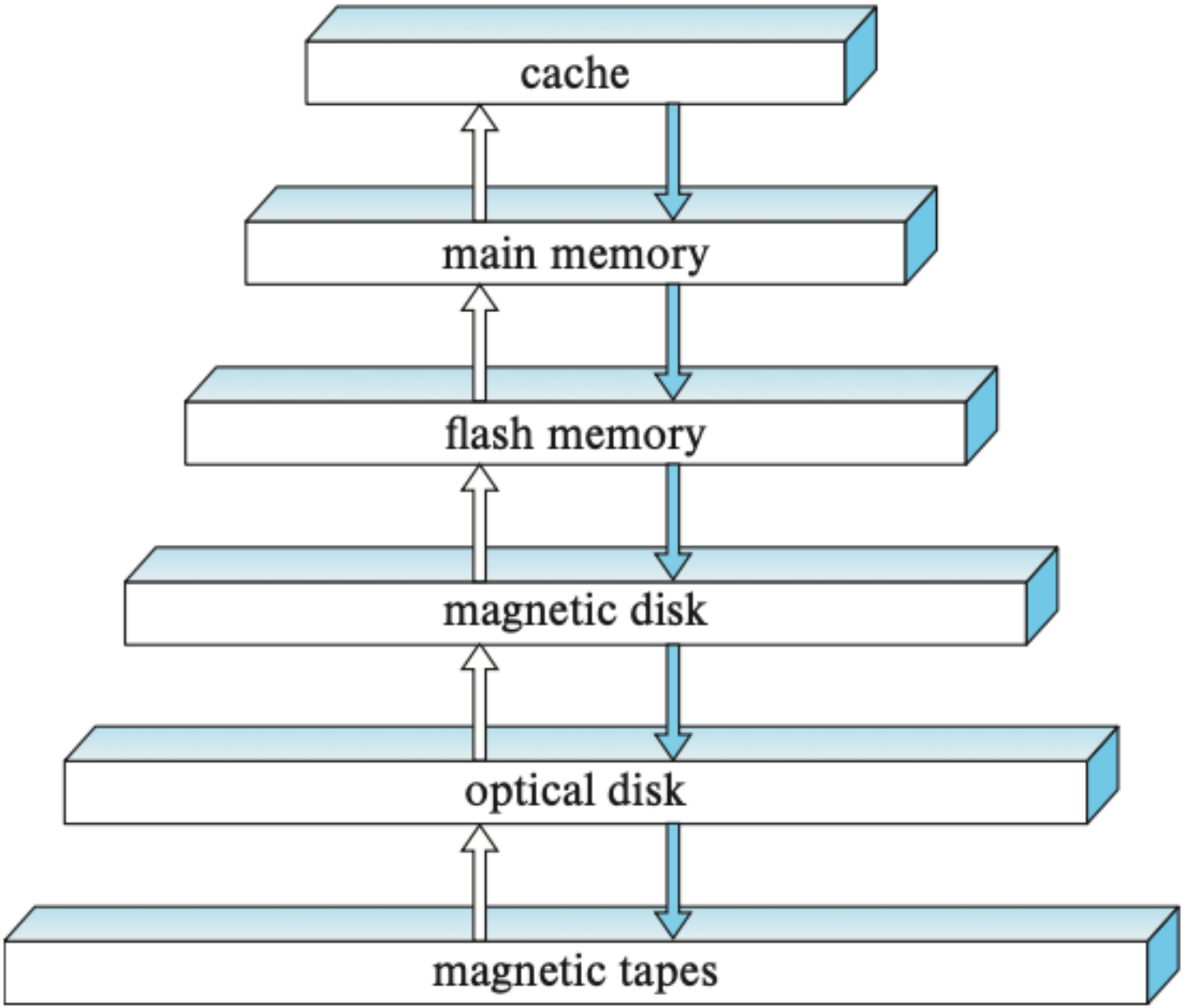
Outline

- Storage
  - Physical storage
    - **Storage hierarchy**
    - Magnetic disks / SSDs
    - RAID
- DB internals Sources: Silberschatz et al. 2020, Chap 12, Physical Storage Systems

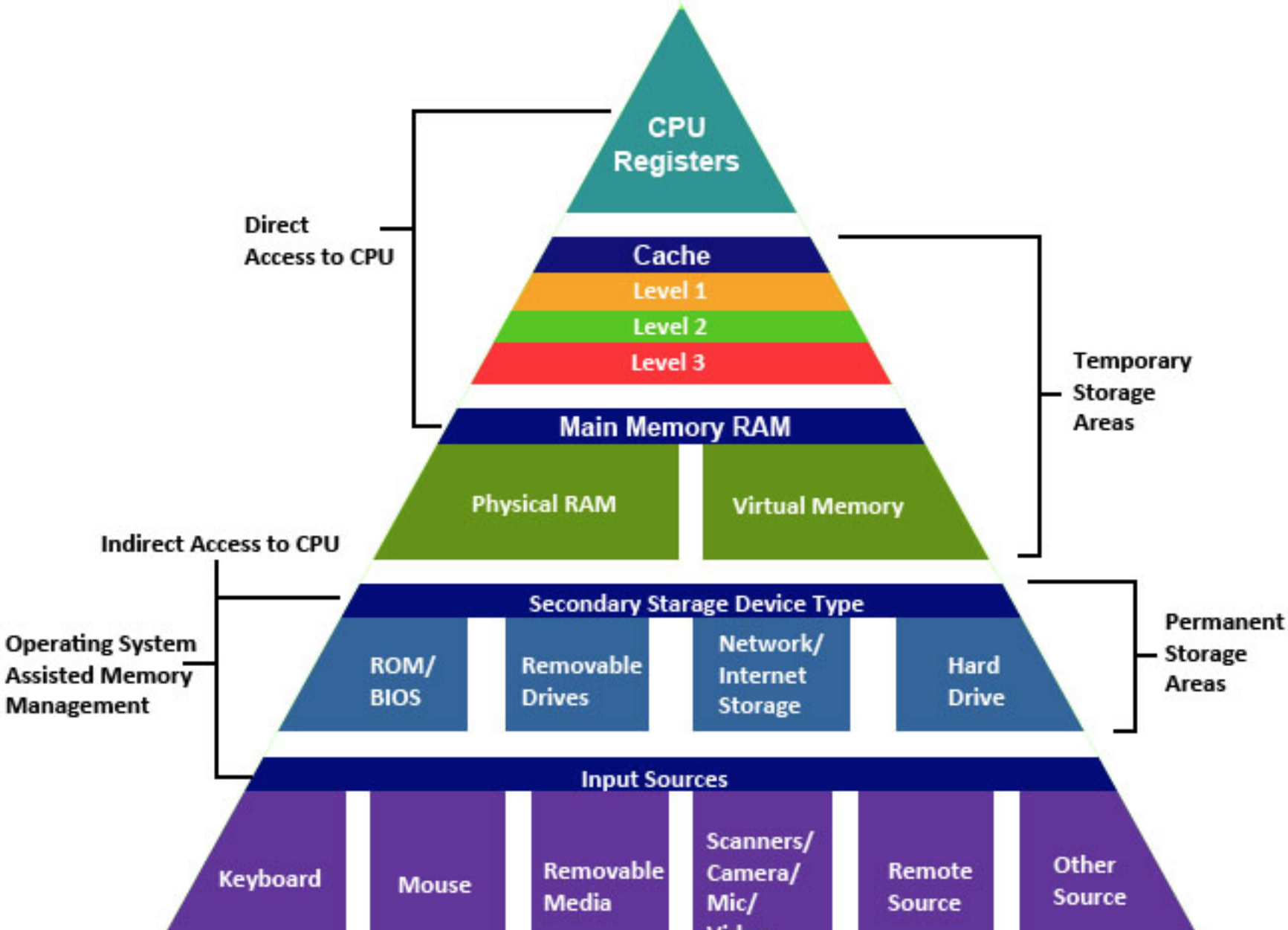
Storage Characteristics

- Storage media presents a trade-off between:
  - speed of access (e.g., 500-3,500MB / sec)
  - cost per unit of data (e.g., 50 USD / TB)
  - medium reliability
- Volatile vs non-volatile storage
  - **Volatile**: loses contents when power switched off
  - **Non-volatile**: can survive failures and system crashes
- 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'**
  - Need to know how data is stored in order to optimize access

Storage Hierarchy



Storage Hierarchy



How Important is Memory Hierarchy?

- Trade-offs shifted drastically over last 10-15 years
- **Innovations:**
  - Fast network, SSDs, and large memories
  - However, the volume of data is growing rapidly
- **Observations**:
  - Faster to access another computer's memory through network than accessing your own disk
  - Cache is playing more and more important role
  - In-memory DBs
    - Enough memory that data often fits in memory of a cluster of machines
  - Disk considerations less important
    - Still disks are where most of the data lives today
- Similar reasoning for algorithms
  - Best algorithm depends on the technology available

Outline

- Storage
  - Physical storage
    - Storage hierarchy
    - **Magnetic disks / SSD**
    - RAID
- DB internals

Connecting disks to a server

- **Disks** (magnetic and SSDs) can be **connected to computer**:
  - Through high-speed bus interconnections; or
  - Through high-speed network
- **Through a high-speed interconnection**
  - Serial ATA (SATA)
  - Serial Attached SCSI (SAS)
  - NVMe (Non-volatile Memory Express)
- `text Through high-speed networks`
  - Storage Area Network (SAN): ISCSI, Fiber Channel, InfiniBand
  - `text Network Attached Storage (NAS)`
    - Provides a file-system interface (e.g., NFS)
    - Cloud storage: Data is stored in the cloud and accessed via an API, Object store, High latency

Magnetic Disks: Components

Magnetic Disks: Current Specs

- **Capacity**
- 10 terabyte and more
- **Access time**
- = Time to start reading data
- Seek time
  - = Move the arm across cylinders (2-20ms)
- Rotational latency time
  - = Wait for sector to be accessed (4-12ms)
- **Data-transfer rate**
  - Once the data is reached the transfer begins
  - Transfer rate = 50-200MB / secs
  - Sector (disk block) = logical unit of storage (4-16KB)
  - Sequential access = when the blocks are on the same or adjacent tracks
  - Random access = each request requires a seek
    - IOPS = number of random single block accesses in a second (50-200 IOPS)
- **Reliability**
  - Mean time to failure (MTTF) = the amount of time that on average the system can run continuously without a failure
  - Lifespan of an HDD is ~5 years

Accessing Data Speed

- **Random data transfer rates**
  - = how long it takes to read a random sector
  - It has 3 components
    - Seek time: Time to seek to the track (Average 4 to 10ms)
    - Rotational latency: Waiting for the sector to get under the head (Average 4 to 11ms)
    - Transfer time: Time to transfer the data (Very low)
  - About 10ms per access
    - So if randomly accessed blocks, can only do 100 block transfers (100 / sec x 4 KB per block = 50 KB/s)
- **Serial data transfer rates**
  - = rate at which data can be transferred (without any seek)
  - 30-50MB/s to up to 200MB/s
- **Seeks are bad!**

Solid State Disk (SSD)

- Mainstream around 2000s
- Like non-volatile RAM (NAND and NOR)
- **Capacity**
  - 250, 500 GBs (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 HDDs)
  - Multiple random requests (e.g., 32) in parallel
  - 10,000 IOPS (vs 50/200 for HDDs)
  - Require to read an entire "page" of data (typically 4KB)
    - Equivalent to a block in magnetic disks
- **Data-transfer rate**
  - 1 GB/s (vs 200 MB/s HDD)
  - Typically limited by the interface speed
  - Reads and writes are ~500MB/s for SATA and 2-3 GB/s for NVMe
  - Lower power consumption than HDDs
  - Writing to SSD is slower than reading (~2-3x)
    - It requires erasing all pages in the block
- **Reliability**
  - There is a limit to how many times a flash page can be erased (~1M times)
  - Better than an HDD from any point of view, but more expensive per GB

Outline

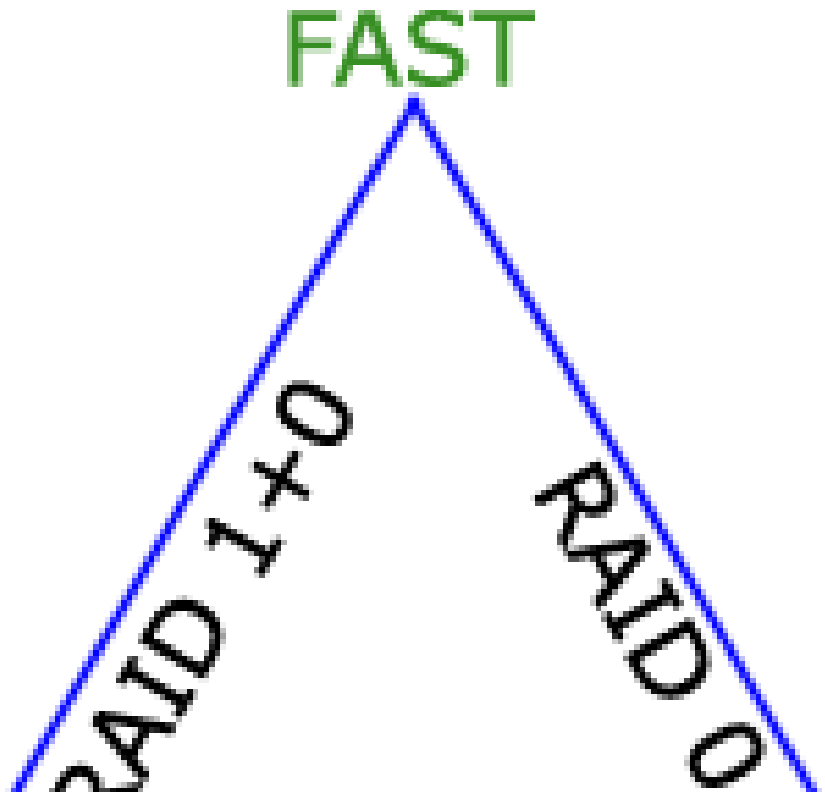
- Storage
  - Physical storage
  - Storage hierarchy
  - Magnetic disks / SSD
  - **RAID**
- DB internals

RAID

- RAID = Redundant Array of Independent Disks
- **Problem**
  - Storage capacity has been growing exponentially
  - Data-storage requirement (e.g., web, DBs, multimedia applications) has been growing even faster
  - You need a lot of disks
  - MTTF between failure of any disk get smaller (e.g., days)
    - If we store a single copy of the data, the frequency of data loss is unacceptable
- **Observations**
  - Disks are very cheap
  - Failures are very costly
  - Use "extra" disks to ensure reliability
    - Store data redundantly
    - If one disk goes down, the data still survives
  - Bonus: allow faster access to data
- **Goal**
  - Expose a logical view of a single large and reliable disk from many unreliable disks
  - Different RAID "levels" (reliability vs performance)

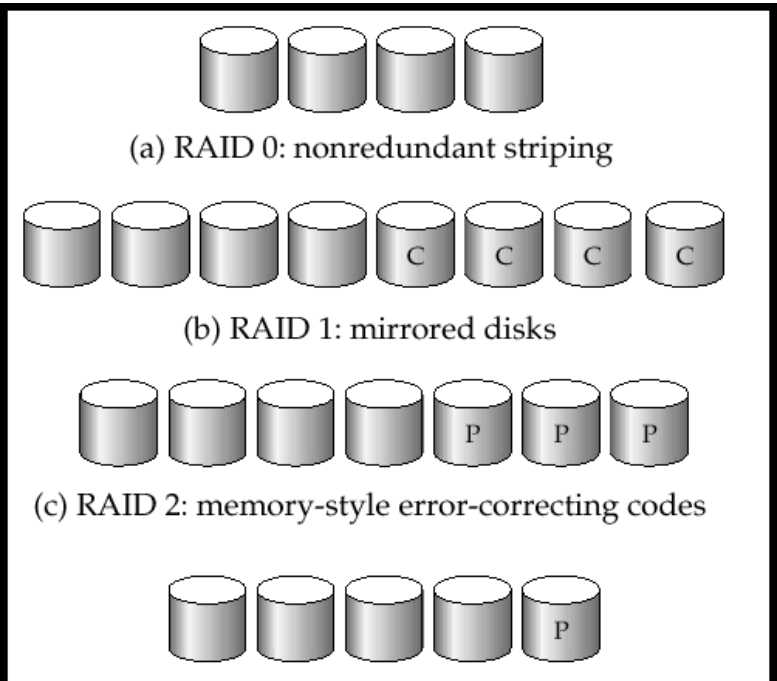
Improve Reliability / Performance with RAID

Reliability - Use redundancy - Store the same data multiple times - E.g., mirroring (aka shadowing) - If a disk fails, the data is not lost but it can be reconstructed - Increased MTTF - Assumption: independence of disk failure - Power failures and natural disasters - As disks age, probability of failure increases together **Performance** - Parallel access to multiple disks - E.g., mirroring - Increase number of read requests - Striping data across multiple disks - Increase transfer rate



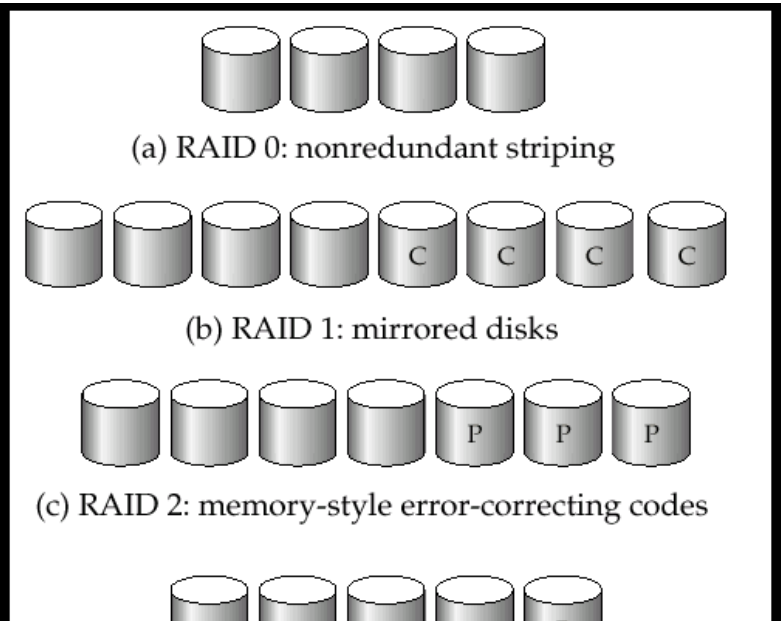
RAID Levels

RAID 0: No redundancy - Array of independent disks - Same access-time - Increased transfer rate **RAID 1: Mirroring** - Make a copy of the disks - If one disk fails, we have a copy - Reads: can go to either disk, so higher data rate possible - Writes: need to write to both disks **RAID 2: Memory-style error correction** - Use extra bits so we can reconstruct - Superseded by RAID 5 **RAID 3: Interleaved parity** - One disk contains "parity" for the main data disks - Can handle a single disk failure - Little overhead (only 25% in the above case) **RAID 5: Block-interleaved distributed parity** - Distributed parity "blocks" instead of bits



Choosing a RAID Level

- Main choice between RAID 1 and RAID 5
- **RAID 1 better write performance**
  - E.g., to write a single block
    - RAID 1: only requires 2 block writes
    - RAID 5: 2 block reads and 2 block writes
  - Preferred for applications with *high update rate* and *small data* (e.g., log disks)
- **RAID 5 lower storage cost**
  - RAID 1: 2x more disks
  - RAID 5 is preferred for applications with *low update rate* and *large amounts of data*

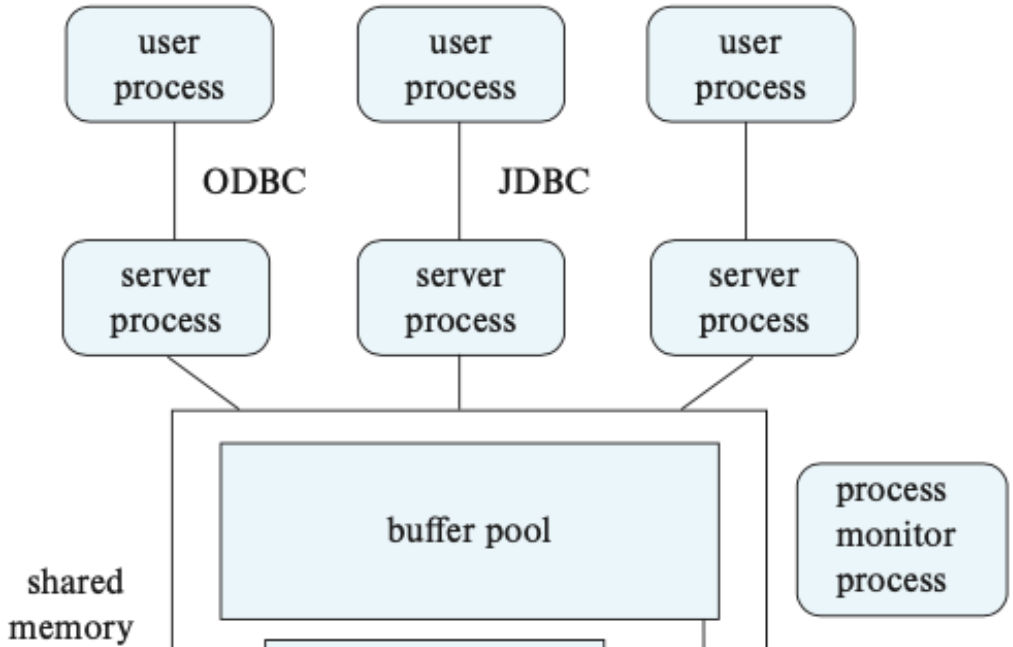


Outline

- Storage
  - Physical storage
- **DB internals** Sources:
- Silberschatz et al. 2020, Chap 13: Data Storage Structures

(Centralized) DB Internals

User processes - Issue commands to the DB Server processes - Receive commands and call into the DB code Process monitor process - Monitor DB processes - Recover processes from failures Lock manager process - Lock grant / release - Deadlock detection Database writer process - Output modified buffer blocks to disk on a continuous basis Log writer process - Output log records to stable storage Checkpoint process - Perform periodic checkpoints Shared memory - Contain all shared data - Buffer pool, Lock table, Log buffer (log records waiting to be saved on stable storage), Caches (e.g., query plans) - Data needs to be projected by mutual exclusion locks



DB Internals

Storage hierarchy - How are tables mapped to files? - How are tuples mapped to disk blocks?

Buffer Manager - Bring pages from disk to memory - Manage the limited memory

Query Processing Engine - Given a user query, decide how to "execute" it - Specify sequence of pages to be brought in memory - Operate upon the tuples to produce results