Operating system:

Table of Contents

[Intro: 3](#_Toc165349531)

[System Call in Windows and Unix 3](#_Toc165349532)

[OS structures 3](#_Toc165349533)

[Linux start-up or booting Process 4](#_Toc165349534)

[Process Management 4](#_Toc165349535)

[ First-come, first-served (FCFS) 5](#_Toc165349536)

[ Shortest Job First (SJF) 5](#_Toc165349537)

[ Round Robin (RR): 5](#_Toc165349538)

[ Priority Scheduling 5](#_Toc165349539)

[ Multilevel queue 5](#_Toc165349540)

[CPU scheduling 5](#_Toc165349541)

[Threads: 6](#_Toc165349542)

[Difference between process and thread: 6](#_Toc165349543)

[Multi-Threading: 6](#_Toc165349544)

[**Benefits of Multithreading:** 6](#_Toc165349545)

[Process Synchronisation: 7](#_Toc165349546)

[Race Condition: 7](#_Toc165349547)

[Deadlock 7](#_Toc165349548)

[File System 8](#_Toc165349549)

[Types: 8](#_Toc165349550)

[File Allocation Methods: 8](#_Toc165349551)

[Free Space management: 8](#_Toc165349552)

[Raids 8](#_Toc165349553)

[Storage Management: 9](#_Toc165349554)

[Disk Partitioning: 9](#_Toc165349555)

[Disk Scheduling: 9](#_Toc165349556)

[Key Terms Associated with Disk Scheduling 9](#_Toc165349557)

[Seek Time 9](#_Toc165349558)

[Rotational Latency 9](#_Toc165349559)

[Transfer Time 9](#_Toc165349560)

[Disk Access Time 9](#_Toc165349561)

[Disk Response Time 9](#_Toc165349562)

[Memory management: 10](#_Toc165349563)

[ Primary Memory 10](#_Toc165349564)

[ Secondary Memory 10](#_Toc165349565)

[Logical Address Space and Physical Address Space 11](#_Toc165349566)

[Static and Dynamic Loading: 11](#_Toc165349567)

[Static and Dynamic Linking: 11](#_Toc165349568)

[Swapping: 11](#_Toc165349569)

[Contiguous Memory Allocation 11](#_Toc165349570)

[Memory Allocation Algorithms: 11](#_Toc165349571)

[Fragmentation: 12](#_Toc165349572)

[Paging in OS: 12](#_Toc165349573)

[Page Fault: 12](#_Toc165349574)

[Virtual memory: 12](#_Toc165349575)

[Thrashing 13](#_Toc165349576)

[Caching: 13](#_Toc165349577)

# Intro:

An operating sys is a s/w program that work as an interface between the h/w and a s/w. Some of the main functions of OS is i/o operations, m/m mgmt., process mgmt, security, file mgmt, device mgmt, networking, provides user interface, virtualization, and etc.

System Call in Windows and Unix: A sys call a way for programs to interact with OS. Whenever a program executes a system call; it transitions from user mode to kernel mode. This transition is typically invoked by a function or interrupting instruction by the programming language.

| **Process** | **Windows** | **Unix** |
| --- | --- | --- |
| Process Control | CreateProcess()  ExitProcess()  WaitForSingleObject() | Fork()  Exit()  Wait() |
| File manipulation | CreateFile()  ReadFile()  WriteFile() | Open()  Read()  Write()  Close() |
| Device Management | SetConsoleMode()  ReadConsole()  WriteConsole() | Ioctl()  Read()  Write() |
| Information Maintenance | GetCurrentProcessID()  SetTimer()  Sleep() | Getpid()  Alarm()  Sleep() |
| Communication | CreatePipe()  CreateFileMapping()  MapViewOfFile() | Pipe()  Shmget()  Mmap() |
| Protection | SetFileSecurity()  InitializeSecurityDescriptor()  SetSecurityDescriptorgroup() | Chmod()  Umask()  Chown() |

OS structures:

Monolithic: MS-DOS

Micro Kernel: This structure designs the operating system by removing all non-essential components from the kernel and implementing them as system and user programs. MAC OS is an example of this.

Hybrid Kernel: Combination of both monolithic and micro kernel.

EXO Kernel: Designed to provide application-level management of hardware resources.

Layered: OS is broken down into number of layers. The bottom layer is the h/w and the topmost layer is an user interface. Unix is an example of this type of OS.

Modular: It involves designing of a modular kernel. The kernel has only a set of core components and other services are added as dynamically loadable modules to the kernel either during runtime or boot time. Solaris OS is an example of this type of OS.

Virtual Machines: When a computer or any other computing device is in a powerless state, its operating system remains stored in secondary storage like a hard disk or SSD. But, when the computer is started, the operating system must be present in the main memory or RAM of the system.

Linux start-up or booting Process (sequence of events performed after the booting of a sys): The procedure of starting a computer by loading the kernel is known as **booting** the system.

Firmware: do the quick check of the h/w called (power-on self test) and look for the boot loader to load the program from bootable device. Traditionally there were BIOS (basic i/o sys) and they are UTFI (Unified extensible firmware interface)

Boot Loader: Runs and determine what kernel to load. LILO (Linux loader) and Grub

Kernel: loads into memory, Start the initialization process.

Initialization: start the background process

Windows:

A diagram of a software development

Description automatically generated

Latest windows server is windows 2022 and latest windows OS is windows 11.

Process Management: A process is **the instance of a computer program that is being executed by one or many threads**. It contains the program code and its activity. A process is an active entity whereas program is a passive entity. Process management refers to the techniques and strategies used by organizations to design, monitor, and control their business processes to achieve their goals efficiently and effectively. A CPU-bound process requires more CPU time or spends more time in the running state. An I/O-bound process requires more I/O time and less CPU time. An I/O-bound process spends more time in the waiting state.

Process of saving the context of a process and loading the context another process is call context switching.

A process that has finished the execution but still has an entry in the process table to report to its parent process is known as a zombie process. A child process always first becomes a zombie before being removed from the process table. The parent process reads the exit status of the child process which reaps off the child process entry from the process table.

The operating system can use different scheduling algorithms to schedule processes. Here are some  commonly used timing algorithms:

* First-come, first-served (FCFS): This is the simplest scheduling algorithm, where the process is executed on a first-come, first-served basis. [FCFS](https://www.geeksforgeeks.org/first-come-first-serve-cpu-scheduling-non-preemptive/) is non-pre-emptive, which means that once a process starts executing, it continues until it is finished or waiting for I/O.
* Shortest Job First (SJF): [SJF](https://www.geeksforgeeks.org/sjf-full-form/) is a proactive scheduling algorithm that selects the process with the shortest burst time. The burst time is the time a process takes to complete its execution. SJF minimizes the average waiting time of processes.
* Round Robin (RR): [Round Robin](https://www.geeksforgeeks.org/round-robin-scheduling-with-different-arrival-times/) is a proactive scheduling algorithm that reserves a fixed amount of time in a round for each process. If a process does not complete its execution within the specified time, it is blocked and added to the end of the queue. RR ensures fair distribution of CPU time to all processes and avoids starvation.
* Priority Scheduling: This scheduling algorithm assigns priority to each process and the process with the highest priority is executed first. Priority can be set based on process type, importance, or resource requirements.
* Multilevel queue: This scheduling algorithm divides the ready queue into several separate queues,  each queue having a different priority. Processes are queued based on their priority, and each queue uses its own scheduling algorithm. This scheduling algorithm is useful in scenarios where different types of processes have different priorities.

Yum (redhat) and Apt-get are two package-manger in unix (Ubunto)

CPU scheduling: It is a process that allows one process to use the CPU while another process is delayed (in standby) due to unavailability of any resources such as I/O etc, thus making full use of the CPU. The purpose of CPU Scheduling is to make the system more efficient, faster, and fairer. Process Scheduling is the process of the process manager handling the removal of an active process from the CPU and selecting another process based on a specific strategy. Process Scheduling is an integral part of multi-programming applications. Such operating systems allow more than one process to be loaded into usable memory at a time and the loaded shared CPU process uses repetition time.

## Threads:

A thread is a single sequence stream within a process. Threads are also called lightweight processes as they possess some of the properties of processes. Each thread belongs to exactly one process. In an operating system that supports multithreading, the process can consist of many threads. Threads run in parallel improving the application performance.

* **User Level Thread** is a type of thread that is not created using system calls. The kernel has no work in the management of user-level threads. User-level threads can be easily implemented by the user. In case when user-level threads are single-handed processes, kernel-level thread manages them.
* **Kernel Level Threads** has its own thread table where it keeps track of the system. The operating System Kernel helps in managing threads. Kernel Threads have somehow longer context switching time. Kernel helps in the management of threads.

## Difference between process and thread:

Here's a comparison of processes and threads in the form of a table:

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Process** | **Thread** |
| Definition | An instance of a running program | A lightweight unit of execution within a process |
| Resource Allocation | Heavyweight: Requires separate memory space and resources | Lightweight: Shares memory space and resources with other threads in the same process |
| Concurrency | Independent entities, run in separate memory spaces | Run within the same memory space, can communicate directly with each other |
| Creation Overhead | Higher overhead due to memory and resource allocation | Lower overhead compared to processes |
| Isolation | Isolated from other processes, cannot access each other's memory directly | Share memory space and resources within the same process |
| Communication | Requires inter-process communication mechanisms (e.g., message passing, shared memory) | Can communicate directly through shared variables or data structures |

## Multi-Threading:

A thread is also known as a lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a [browser](https://www.geeksforgeeks.org/web-browser/), multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. Multithreading is a technique used in operating systems to improve the performance and responsiveness of computer systems. Multithreading allows multiple threads (i.e., lightweight processes) to share the same resources of a single process, such as the CPU, [memory](https://www.geeksforgeeks.org/memory-management-in-operating-system/), and[I/O devices](https://www.geeksforgeeks.org/input-and-output-devices/).

### **Benefits of Multithreading:**

* Multithreading can improve the performance and efficiency of a program by utilizing the available CPU resources more effectively. Executing multiple threads concurrently, it can take advantage of parallelism and reduce overall execution time.
* Multithreading can enhance responsiveness in applications that involve user interaction. By separating time-consuming tasks from the main thread, the user interface can remain responsive and not freeze or become unresponsive.
* Multithreading can enable better resource utilization. For example, in a server application, multiple threads can handle incoming client requests simultaneously, allowing the server to serve more clients concurrently.
* Multithreading can facilitate better code organization and modularity by dividing complex tasks into smaller, manageable units of execution. Each thread can handle a specific part of the task, making the code easier to understand and maintain.

## Process Synchronisation:

It is the coordination of execution of multiple processes in a multi-process system to ensure that they access shared resources in a controlled and predictable manner. It aims to resolve the problem of race conditions and other synchronization issues in a concurrent system.

## Race Condition:

When more than one process is executing the same code or accessing the same memory or any shared variable in that condition there is a possibility that the output or the value of the shared variable is wrong so for that all the processes doing the race to say that my output is correct this condition known as a race condition. Several processes access and process the manipulations over the same data concurrently, and then the outcome depends on the order in which the access takes place.

Deadlock: A deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Deadlock can arise if the following four conditions hold simultaneously (Necessary Conditions)

Mutual Exclusion: Two or more resources are non-shareable (Only one process can use at a time)   
Hold and Wait: A process is holding at least one resource and waiting for resources.   
No Pre-emption: A resource cannot be taken from a process unless the process releases the resource.   
Circular Wait: A set of processes waiting for each other in circular form.

There are several approaches to detecting deadlocks in an operating system:

* **Resource Allocation Graph (RAG):** This method involves constructing a graph that represents the resources and processes, and then analysing the graph to check for cycles.
* **Deadlock Detection Algorithm:** This algorithm periodically checks the resource allocation status and resource requests to identify circular wait conditions.
* **Banker’s Algorithm:** This algorithm uses a mathematical model to determine if a state is safe or unsafe, considering the maximum resource need and available resources.

To eliminate the deadlock, we can simply kill one or more processes. For this, we use two methods:

* **Abort all the Deadlocked Processes**: Aborting all the processes will certainly break the deadlock but at a great expense. The deadlocked processes may have been computed for a long time, and the result of those partial computations must be discarded and there is a probability of recalculating them later.
* **Abort one process at a time until the deadlock is eliminated**: Abort one deadlocked process at a time, until the [deadlock](https://www.geeksforgeeks.org/introduction-of-deadlock-in-operating-system/)cycle is eliminated from the system. Due to this method, there may be considerable overhead, because, after aborting each process, we have to run a[deadlock detection algorithm](https://www.geeksforgeeks.org/deadlock-detection-algorithm-in-operating-system/) to check whether any processes are still deadlocked.

File System: It is a data structure used for storing files on a disk partition. A file is collection of related of info that is recorded on secondary storage. Collections of files is a file directory. The process of placing the file system under a mount called mounting.

## Types:

FAT: used by some operating systems, including older versions of Windows. It uses a table to keep track of the allocation status of each cluster (a fixed-size block of storage) on the disk. The

Ext (Extended file system): Used in linux and unix based os.

NTFS (New Technology file system): Used by windows OS, it supports features such as file and folder permission, compression, and encryption.

HFS (Hierarchal file system) and APFS (Apple file system): File system used by Mac and Apple devices.

## File Allocation Methods:

* Continuous Allocation: A single continuous of block is allocated to a file at the time of file creation. Kinda like array.
* Linked Allocation: Individual block basis and they contain pointer for next block (basically like linked list ds). Blocks need not be continuous.
* Indexed Allocation: In this case, the file allocation table contains a separate one-level index for each file: The index has one entry for each block allocated to the file.

Free Space management: It is a critical aspect of operating systems as it involves managing the available storage space on the hard disk or other secondary storage devices and can be done either by bit table (where a vector containing a bit either 0 or 1; 1 represent block is in use and 0 represents block is free.) or by free block list (each block is assigned a number sequentially and the list of the numbers of all free blocks is maintained in a reserved block of the disk.)

Raids (Redundant arrays of independent disks): a technique that makes use of a combination of multiple disks instead of single disk. Similarly, like a big disk presenting itself as a liner array of blocks. It improves availability, performance, capacity, and reliability. It slied data from disk failure by keeping rumours copies of the data on many disks.

# Storage Management:

## Disk Partitioning:

Different file system types for different partitions, better disk space management, and multi-OS support.

MBR (Master Boot Loader): Based on basic i/o systems, max 4 partitions, and max drive: 2TB.

## Disk Scheduling:

It is a process done by OS to schedule i/o requests it is also k/n as i/o scheduling. Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus, other I/O requests need to wait in the waiting queue and need to be scheduled.

Disk Scheduling Algorithms

* FCFS (First Come First Serve)
* SSTF (Shortest Seek Time First)
* SCAN (Elevator Algorithm)
* C-SCAN (CIrcular SCAN)
* LOOK
* C-LOOK
* RSS
* LIFO (Last-In First-Out)
* N-Step SCAN
* F-SCAN

## Key Terms Associated with Disk Scheduling

Seek Time**:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or written. So, the disk scheduling algorithm that gives a minimum average seek time is better.

Rotational Latency**:** Rotational Latency is the time taken by the desired sector of the disk to rotate into a position so that it can access the read/write heads. So, the disk scheduling algorithm that gives minimum rotational latency is better.

Transfer Time**:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and the number of bytes to be transferred.

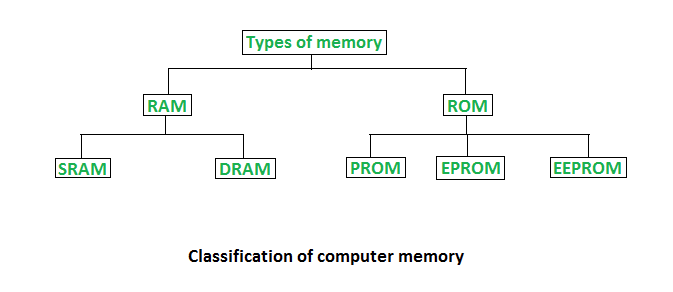
Disk Access Time**:** Seek Time + Rotational Latency + Transfer Time

Disk Response Time: Response Time is the average time spent by a request waiting to perform its I/O operation. The average Response time is the response time of all requests. Variance Response Time is the measure of how individual requests are serviced with respect to average response time. So, the disk scheduling algorithm that gives minimum variance response time is better.

# Memory management:

Memory is an important part of the computer which is responsible for the storage of data and information on a temporary or permanent basis. Memory can be classified into two broad categories:

* Primary Memory: It is a type of Computer Memory which is directly accessed by the Pre-processor. It is basically used to store data on which computer is currently working. It has lesser storage than Secondary Memory. It is basically of two types:
* [Random Access Memory (RAM)](https://www.geeksforgeeks.org/random-access-memory-ram-and-read-only-memory-rom/)
* [Read Only Memory (ROM)](https://www.geeksforgeeks.org/random-access-memory-ram-and-read-only-memory-rom/)



* Secondary Memory: Secondary Memory is a type of Computer Memory which is used to permanently store the data and Information. It has a larger data storage capacity than Primary Memory. Secondary Memory is not directly accessible from CPU. It is basically of four types:
* [HardDisk](https://www.geeksforgeeks.org/difference-between-hard-disk-and-floppy-disk/) and [Compact Disc (CD)](https://www.geeksforgeeks.org/difference-between-cd-and-dvd/)
* [Digital Versatile Disk (DVD)](https://www.geeksforgeeks.org/difference-between-cd-and-dvd/) and [Floppy Disk](https://www.geeksforgeeks.org/difference-between-hard-disk-and-floppy-disk/)

| **Difference** | **Random Access Memory (RAM)** | **Read Only Memory (ROM)** |
| --- | --- | --- |
| Data-Retention | RAM is a volatile memory that could store the data if the power is supplied. | ROM is a non-volatile memory that could retain the data even when the power is turned off. |
| Read/Write | Read and write operations are supported. | Only read operations are supported. |
| Use | Used to store the data that must be currently processed by CPU temporarily. | It is typically used to store firmware or microcode, which is used to initialize and control hardware components of the computer. |
| Speed | It is a high-speed memory. | It is much slower than the RAM. |
| CPU Interaction | CPU can easily access data stored in RAM. | CPU cannot easily access data stored in ROM. |
| Size and Capacity | Large size with higher capacity, concerning ROM. | Small size with less capacity, concerning RAM. |
| Used as/in | [CPU Cache](https://www.geeksforgeeks.org/cache-memory-in-computer-organization/), Primary memory. | Firmware, Micro-controllers. |
| Accessibility | The data stored is easily accessible. | The data stored is not as easily accessible as in the concerning RAM. |
| Cost | RAM is costlier than ROM. | ROM is cheaper than RAM. |
| Chip Size | A RAM chip can store only a few gigabytes (GB) of data. | A ROM chip can store multiple megabytes (MB) of data. |
| Function | Used for the temporary storage of data currently being processed by the CPU. | Used to store firmware, BIOS, and other data that needs to be retained. |

It is crucial for efficient utilization of computer memory. It involves various methods and techniques to manage memory allocation, access, and utilization. Memory management is the process of organizing and controlling computer memory, ensuring that programs and data are appropriately stored, accessed, and utilized. It involves allocating and deallocating memory resources to processes efficiently.

Logical Address Space and Physical Address Space: Logical addresses generated by the CPU are translated into physical addresses by the memory management unit (MMU). Logical address space refers to the address space seen by the CPU, while physical address space refers to the actual physical memory addresses.

Static and Dynamic Loading: Loading a process into memory can be done statically or dynamically. Static loading involves loading the entire program into memory before execution, while dynamic loading loads program segments as needed during execution.

Static and Dynamic Linking: Linking processes involve combining multiple object files into a single executable program. Static linking combines all necessary modules into a single executable, while dynamic linking loads modules as needed during execution.

Swapping: Swapping is the process of moving entire processes temporarily between main memory and secondary storage (e.g., disk) to free up memory space for other processes. It allows for efficient memory utilization and increases the number of processes that can run concurrently.

Contiguous Memory Allocation: Contiguous memory allocation allocates memory in contiguous blocks for each process. It involves dividing memory into fixed-size partitions and allocating each partition to a process.

Memory Allocation Algorithms: Various algorithms are used for allocating memory to processes, such as first fit, best fit, and worst fit. These algorithms aim to minimize fragmentation and maximize memory utilization.

Fragmentation: Fragmentation occurs when memory is allocated inefficiently, resulting in wasted memory space. Internal fragmentation occurs when allocated memory is larger than needed, while external fragmentation occurs when free memory is fragmented and cannot be utilized efficiently.

## Paging in OS:

It is the process of retrieving the processes in the form of pages from secondary memory to main memory is called paging. It is a m/m mgmt scheme that eliminates the need for contiguous allocation of physical memory. In paging, the physical memory is divided into fixed-size blocks called page frames, which are the same size as the pages used by the process. The mapping between logical pages and physical page frames is maintained by the page table, which is used by the memory management unit to translate logical addresses into physical addresses. It involves translating logical addresses to physical addresses using page tables and hardware mechanisms like the Translation Look-aside Buffer (TLB).

## Page Fault:

A page fault occurs when a program attempts to access a page of memory that is not currently in physical memory (RAM). In virtual memory systems, pages are moved between RAM and disk as needed. When a page fault happens, the operating system brings the required page into memory from disk. Page faults can occur due to initial access, demand paging, page replacement, or copy-on-write techniques. The OS handles page faults transparently, updating page tables and resuming program execution after loading the required page into memory.

Virtual memory:

It is a memory management technique used by operating systems to provide an illusion of a larger and contiguous address space than the physical memory (RAM) available on a computer system. It divides the entire address space of a process into fixed-size blocks called pages, similarly, the physical memory is divided into fixed-size blocks called frames. Pages are mapped to frames in physical memory using a page table. Overall, virtual memory allows programs to use more memory than is physically available, provides efficient memory management, and enables multitasking and multitasking on modern computer systems. It is a fundamental feature of modern operating systems, enhancing system performance and stability.

Key features of virtual memory include:

**Memory Address Translation**: Virtual addresses generated by the CPU are dynamically translated into physical addresses using hardware mechanisms like memory management units (MMUs), allowing programs to access memory locations regardless of their physical location.

**Demand Paging:** Only necessary pages of a program are loaded into physical memory, while the rest remain on disk. Pages are loaded into memory on demand as they are accessed by the program, minimizing the amount of physical memory required and enabling efficient use of resources.

**Page Replacement:** When physical memory becomes full, the operating system may need to swap out (evict) pages to make room for new ones. Page replacement algorithms determine which pages to evict based on criteria like least recently used (LRU) or least frequently used (LFU).

**Memory Protection:** Virtual memory systems provide memory protection mechanisms to prevent unauthorized access to memory locations. Each page is associated with access permissions, enforced by the operating system to ensure data integrity and security.

**Virtual to Physical Address Mapping:** The mapping between virtual addresses and physical addresses is maintained by the operating system using page tables. These tables keep track of which pages are currently in physical memory and their corresponding locations.

Thrashing : It occurs when the system spends a significant portion of its time handling page faults while actual processing remains minimal. Causes include a high degree of multiprogramming, insufficient frames, and inefficient page replacement policies. Thrashing severely impacts system performance by increasing the degree of multiprogramming and causing CPU usage to drop as processes wait for pages to be swapped in. When memory is full, swapping in required pages becomes time-consuming, further exacerbating the performance issues.

Caching: It is a technique used in computing to store frequently accessed data temporarily in a faster storage location called a cache. Caches are stored at various levels within a computer system, including hardware, the operating system, the network, and database management systems.

Browser Caching: Browser caching occurs on the user's device, typically in the hard drive or solid-state drive. The exact location varies depending on the operating system and browser being used. For example:

* In Windows, Google Chrome's cache is typically stored in %LocalAppData%\Google\Chrome\User Data\Default\Cache.
* In macOS, Chrome's cache is stored in ~/Library/Caches/Google/Chrome/Default/Cache.

Firefox and other browsers have similar cache directories.

Explanation: When a user visits a website, their web browser downloads and stores certain elements of the web page in its cache. This includes resources such as HTML, CSS, JavaScript files, images, and multimedia content.

Purpose: Browser caching speeds up subsequent visits to the same website or web pages by allowing the browser to retrieve cached resources locally rather than fetching them from the web server. This reduces page load times, improves the overall user experience, and reduces server load and bandwidth usage for website owners.

Hardware Caches: Located within the CPU itself, consisting of small, high-speed memory caches such as L1, L2, and L3 caches.

Operating System Caches: Stored in main memory (RAM) managed by the operating system, including file system caches, page caches, and buffer caches.

Web Caches: Located at different points in the network infrastructure, including client-side caching (in web browsers), proxy server caching, and Content Delivery Network (CDN) caching.

Database Caches: Stored in main memory (RAM) managed by the database management system (DBMS), including query caches and buffer pools.

In summary, caching improves data access times and system performance by storing frequently accessed data closer to the point of use, and caches are stored at various levels within a computer system, including hardware, the operating system, the network, and database management systems. Browser caching, specifically, enhances web browsing performance by storing frequently accessed web page resources locally on the user's device.

Reference:

GeeksforGeeks. (25th Jan 2024). Operating Systems. Retrieved from <https://www.geeksforgeeks.org/operating-systems/?ref=shm>