|  |  |
| --- | --- |
| **[Final project]** | [OPERATING SYSTEMS]  **LIYAN AQEL**  **[21110405]** |

**Table of Contents**

[**Part 1** 2](#_Toc155699005)

[**Part 2** 2](#_Toc155699006)

[**Part 3** 2](#_Toc155699007)

[**Part 4** 2](#_Toc155699008)

[**Part 5** 2](#_Toc155699009)

[**Part 6** 2](#_Toc155699010)

[**Part 7** 2](#_Toc155699011)

[**Part 8** 2](#_Toc155699012)

[**References:** 2](#_Toc155699013)

# **Part 1**

## The definition of Virtual memory

It is a technique used for memory management by operating systems to give an illusion of a larger memory space than the space of the RAM. This helps programs to execute as if they have larger amount of memory space than the actual memory space. In this virtual memory system, programs are given their virtual address spaces which are larger than actual memory spaces. The OS is responsible for managing the mapping between the virtual and the physical addresses. The main goal of this technique is to achieve efficient and reliable utilization of the resources of physical memory.

## The role of Virtual memory in the memory management process of operating systems

* **Isolation and abstraction**

It isolates and abstracts the physical memory, which gives the same address space for every process. This abstraction makes program development simpler, as well as allowing processes to operate independently with no interference with each other. The isolation process stops one process from using the memory of another process, which enhances stability and security of the system.

* **Address space management**

It helps the operating system to efficiently manage the address space of every process by allocating or not allocating memory as the process needs. This ensures that processes can operate with no interference with each process. In addition, it gives more flexible memory allocation by allowing processes to use larger space of memory than the available actual space. This happens by swapping pages in the disk and out of it.

* **Demand paging**

It allows the system to load pages only when the CPU needs them, which helps to conserve the physical memory, and this enhances efficiency of the whole system.

* **Page fault handling**

It is responsible for handling page fault, which occurs when the CPU needs a page which is not in the physical memory. It does this process by bringing the requested page from the hard disk into the RAM.

* **Swap space management**

Its idea is based on swapping between pages to store pages that are not executed actively in hard disk. This is done by OS to ensure that swapping pages from in and out physical memory is done efficiently.

* **Memory protection**

It performs memory protection techniques by preventing illegal access to some areas of the address space. This helps to maintain the security and integrity of the system.

## The enhancement of virtual memory on overall system performance

It improves the flexibility and efficiency of the memory management by several techniques, which leads to enhancing the performance of the overall system. These are some techniques which help virtual memory to enhance system performance:

* **Running larger programs:**

It helps programs which are larger than the physical memory to be executed. This can be achieved by swapping parts of the program in and out of the disk and the RAM. This helps the system to operate applications that need larger space to be executed and can’t be loaded into the RAM.

* **Multitasking support**

It helps simultaneous execution of multiprocessing by given each process its virtual address space. These processes can operate independently, and they don’t interfere with each other due to the ability of the OS to switch between them efficiently. This helps to increase the efficiency of the system.

* **Efficient utilization of resources of the physical memory**

It improves utilization of the resources of physical memory efficiently by keeping the most frequent pages in RAM, whereas the less frequent pages are swapped to the disk. This swapping process helps to increase the efficient utilization of physical memory.

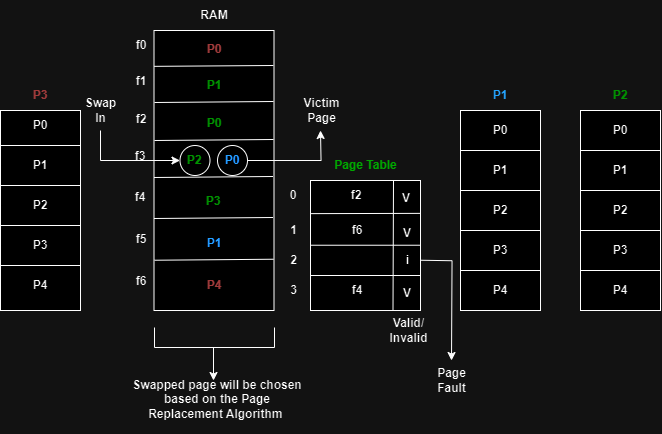
* **Flexibility in memory allocation**

It improves flexibility by helping processes to allocate larger space of virtual memory than the available physical memory. This is based on the assumption that not all the allocated memory while be used at the same time by all processes.

* **Efficient handling of page faults**

It can effectively handle page faults which occurs when the CPU needs a page which is not in the physical memory. In this case, it does this process by bringing the requested page from the hard disk into the RAM. This helps the program to access data that is stored on the hard disk for a particular time without affecting performance.

**For example:**



In the above picture, if proccess2 needs to be executed, it is known that it should be uploaded on the RAM completely. Whereas, when the virtual memory is applied, there is no need to completely upload process 2 into the RAM. Instead, required pages are only uploaded into the RAM and other unrequited pages will remain on the hard disk. This will make the physical address space bigger, and so increases the degree of multi programming, as well as more utilization of the CPU.

When the virtual memory isn’t used, only one process of these three processes can be executed because the size of the RAM = 7 and the size of one process=5.

Whereas, when virtual memory is used, all processes can be executed because it is not necessary to upload all pages of the processes into the RAM.

If a required page needs to be executed such as page 2 from process 2, which doesn’t exist in the RAM. It will be swapped with another less frequent page.

However, if there is no memory space in the RAM, I need a certain technique to swap unrequired page with required page which needs to be executed.

What helps me to know whether the required page is on the RAM or not is having an additional feature which is an added column. If the cells in the column contain the V (one bit), it means that the page exists in the RAM, but if the cells contain the I (one bit), it means that the page doesn’t exist in the RAM, which is called page fault. In this case, the OS will stop the execution of the process and get the control to get the required page from the hard disk.

If there is memory space in the RAM, the OS will upload the page and update the page table.

If there is no memory space in the RAM, the OS will swap between the required page and any less frequency page which is called victim page based on replacement algorithm.

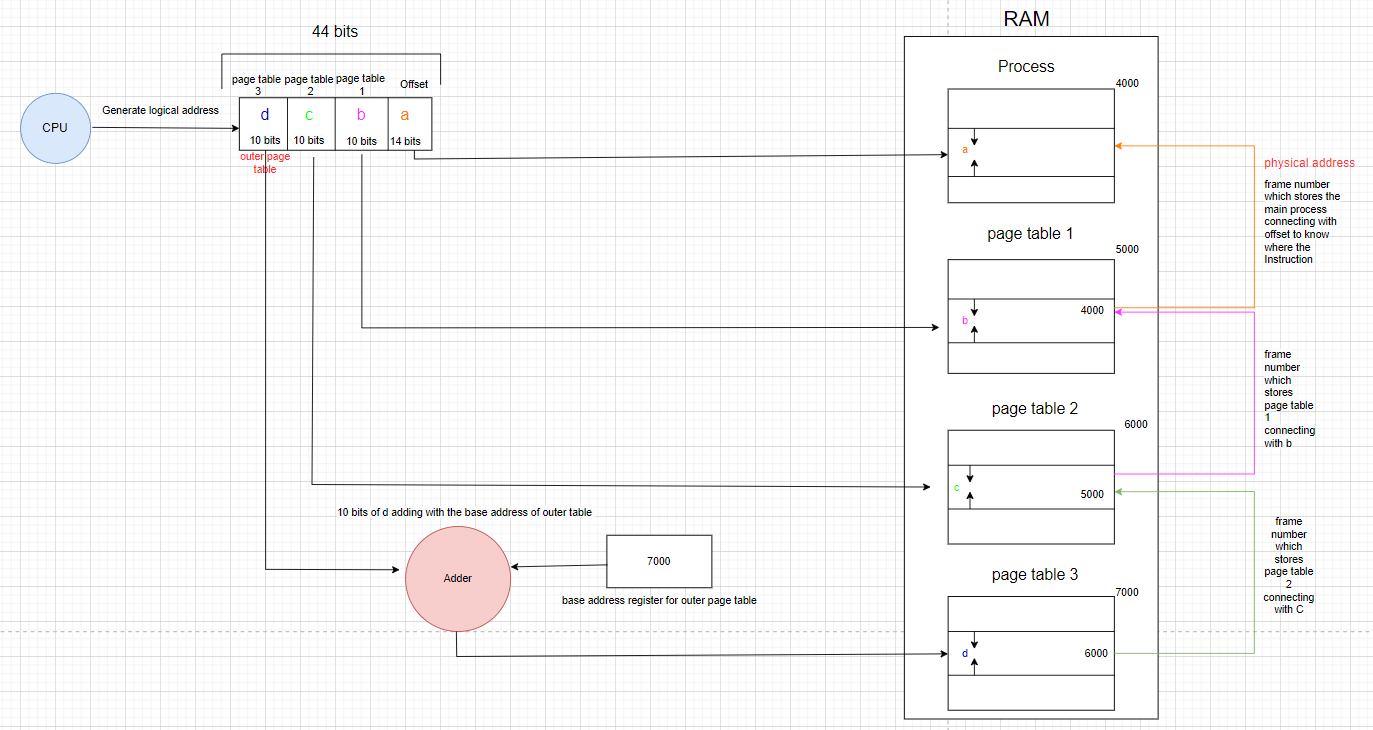
# **Part 2**A piece of paper with writing on it Description automatically generated

A piece of paper with writing on it

Description automatically generated

A paper with writing on it

Description automatically generated



First, the CPU generates a logical address which consists of 44 bits. These bits are divided into 4 parts and each part is responsible for a certain function. The following steps are necessary to allow the CPU to find the physical address which indicates the actual address of the instruction in the memory.

**Note: the process, page table 1, and page table 2 are stored non-contiguously in the RAM but page table 3 (outer page table) is stored contiguously in the RAM**.

The overall process starts from the outer page table because it is stored contiguously in the RAM, so it is stored in one frame, which helps to lead to meaningful location.

The first step of overall process is adding the 10 bits of the d to the base address of outer page table, which is stored in base address register. This will lead me to a certain byte in the frame which stores the outer page table. This byte has the frame number which stores page table 2.

Then, this frame number is connected with the 10 bits of c. This will lead me to certain byte in certain frame which stores page table 2. This byte has the frame number which stores page table 1.

Next, this frame number is connected with the 10 bits of b. This will lead me to certain byte in certain frame which stores page table 1. This byte has the frame number which stores the main process.

After that, this frame number is connected with the 14 bits of a which is called (offset). This will lead me to certain byte in certain frame which stores main process. This byte has the instructions which want to execute.

The connection of offset with the frame number, which stores the process, is called physical address.

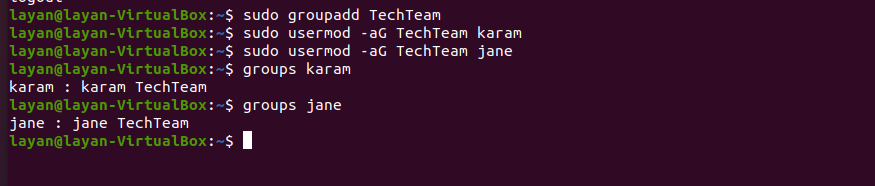
The offset is the order of the instruction in one frame.

In multilevel panging we can’t access the instruction through one page table to find the physical address. We need to use three-page tables to find the physical address in order to access the instructions. This is because the main process, page table 1, and page table 2 are stored non-contiguously, which means they are stored in different frames not one frame.

# **Part 3**

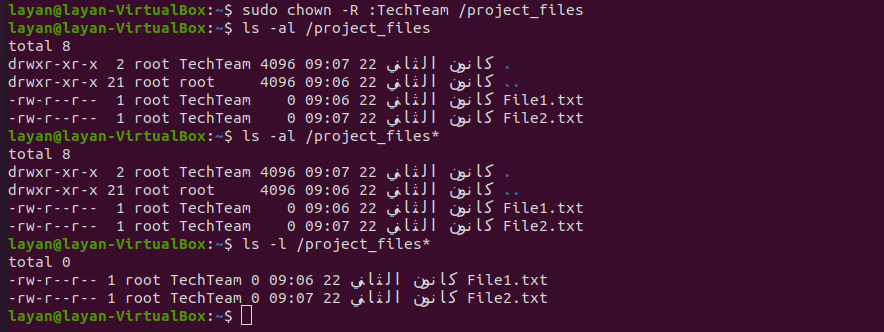
A computer screen with white text

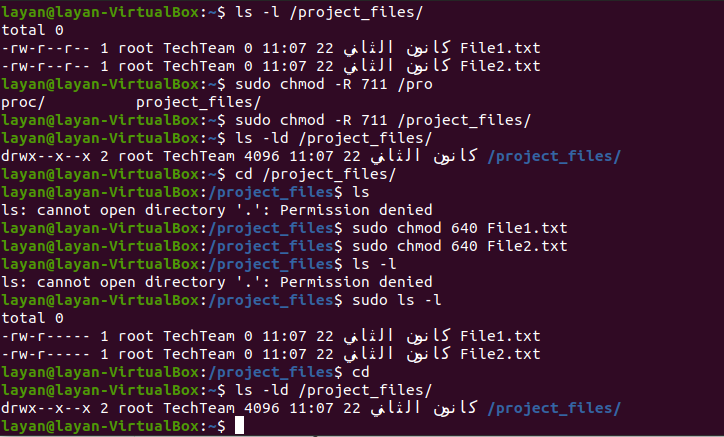
Description automatically generated

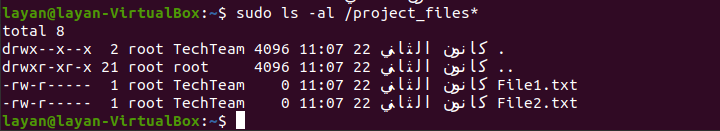


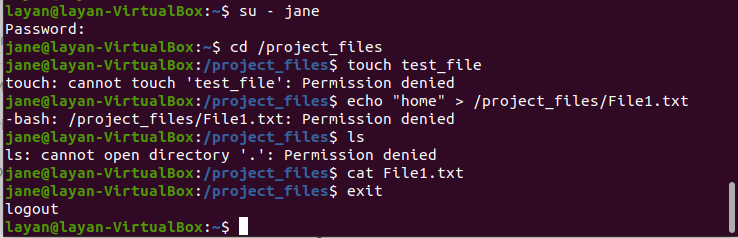
A computer screen shot of a program code

Description automatically generated







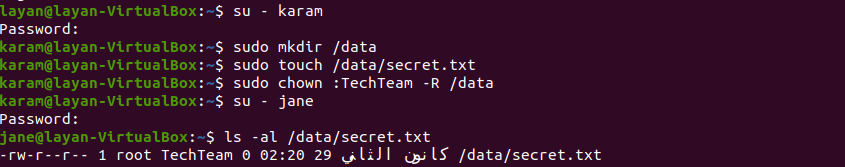


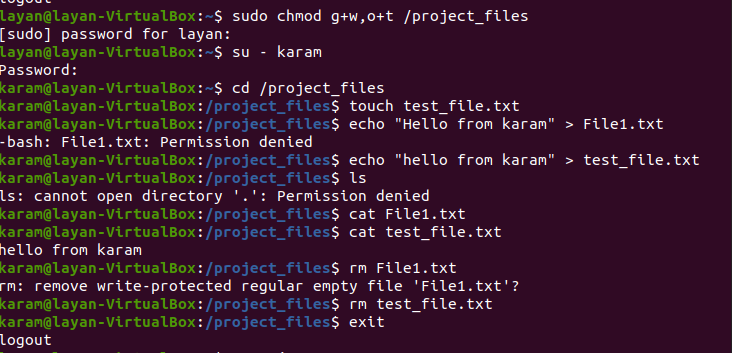
A computer screen shot of a program code

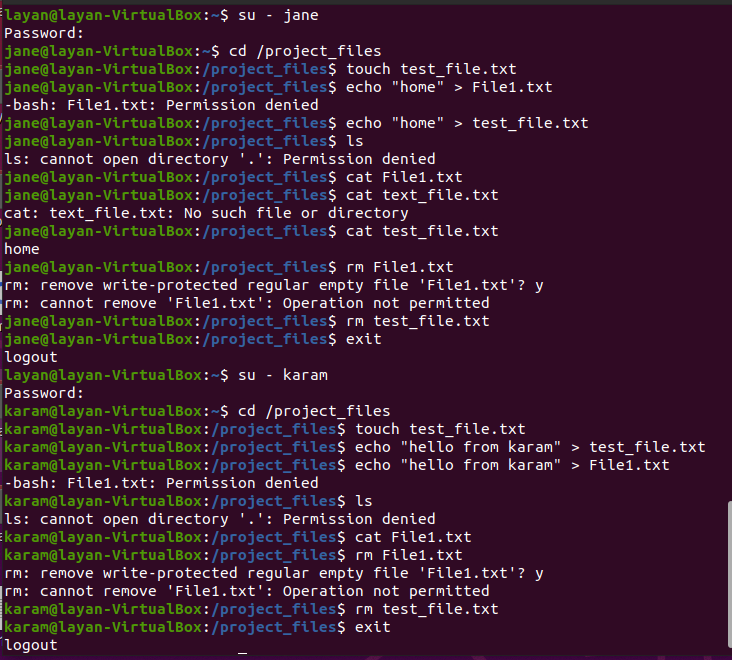
Description automatically generated

A screenshot of a computer

Description automatically generated







A black background with a purple border

Description automatically generated with medium confidence

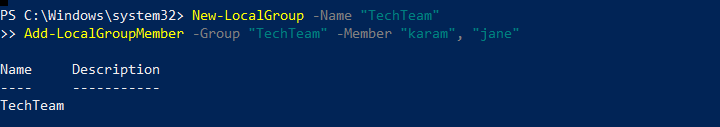
# **Part 4**

1.

A blue screen with text on it

Description automatically generated

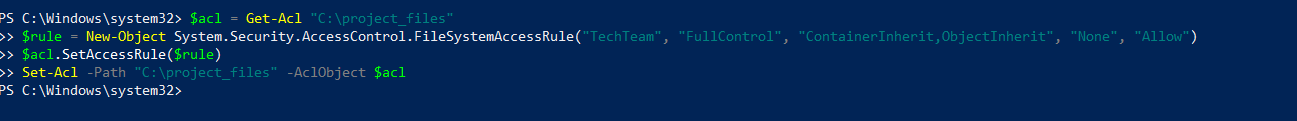
2.



3. A blue screen with white text

Description automatically generated

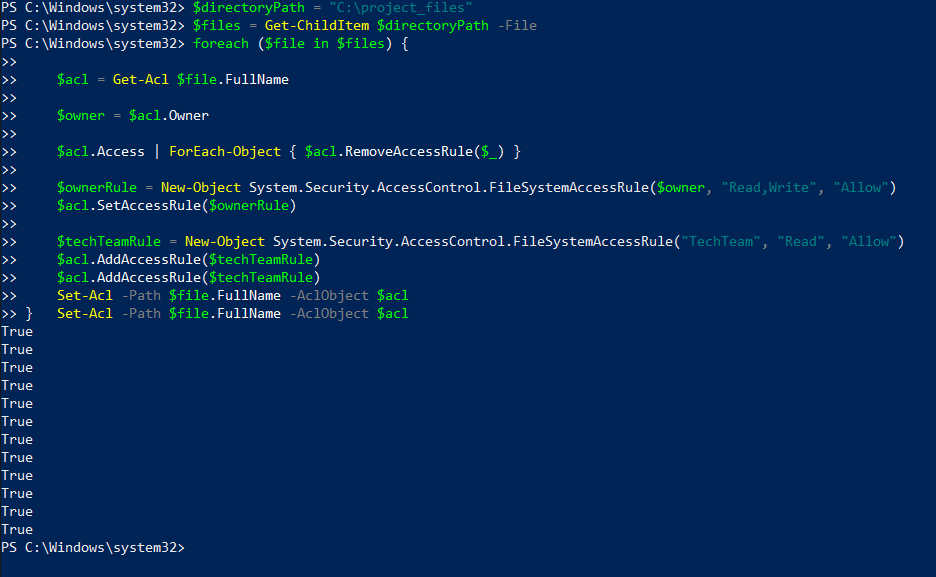
4.



5.

A blue screen with white text

Description automatically generated



# **Part 5**

**The definition of Distributed Operating System:**

It is a specialized group of operating system which operates on many machines and makes them able to work together as one integrated machine. It operates over a number of independent, and physically separate computer nodes. Each node operates as an integrated computing environment and has its own processor, storage, memory, and other hardware. For end-users, it creates balance between cooperation and autonomy and appears as a single, seamless, and unified computer platform. Also, the distributed operating system conceals any essential complexity of the infrastructure distributed.

**The differences from traditional centralized operating systems:**

1. **Utilization and Scalability**

The distributed operating systems have better utilization and scalability of resources because they distribute resources over many machines, while the traditional centralized operating systems have one single operating system which operates on a single machine and manages all resources. **In other words,** the machines in the distributing operating system have their own memories, and CPUs but the system manages the sharing of memories and CPUs across the network, which allows processes and tasks to access memory of other different machines.

1. **Communication and Coordination**

Communication and coordination between distributed machines are very important for processes operated on different machines so that information can be interacted and shared between machines. Whereas this interaction doesn’t exist in traditional centralized operating systems.

1. **Reliability and fault tolerance**

The distributed operating systems have better reliability because if any of the machines fails, the work will not be interrupted because the system is distributed across many machines. While in the traditional centralized operating system, if any failure happens the work will be completely stopped because it depends on one centralized system.

1. **Performance**

The distributed operating system is faster and more efficient in working because it has fast response time, and better utilization of memory and network. **However,** the traditional centralized operating system has performance limitation because it is controlled by one machine, which causes more response time and possible delay.

1. **Concurrency and Parallel**

In the distributed operating system, many processes and tasks can be performed at the same time across multiple machines because this system involves parallel and concurrent processes. While the traditional centralized operating system has sequential way of performing tasks, which means each task is performed in one specific time.

1. **Security challenges**

The distributed operating system has security challenges because it needs more secure communication, confidentiality, as well as the need to ensure data integrity. On the other hand, the traditional centralized operating system has a security challenge due to the fact that if one machine fails, the whole system will breakdown. Also, it is more vulnerable to illegal access and attack because the security of the whole system depends on the security regulations implemented in one location.

**Example of a real-world application that benefits from distributed operating systems:**

**An online banking system** is a good example of a real-world application that benefits from distributed operating systems. In the bank system, a large number of users can access their accounts, and perform different transactions. The distributed operating system can enhance the scalability of the bank system because it can add more servers and resources to cope with the continuous increase in the number of clients. As a result, the bank system can deal with this continuous increase in users and transactions efficiently.

Another benefit of distributed operating system in the banking system is its ability to deal with high availability and risk assessment (fault tolerance). For example, if one server fails, the system can redirect the request to another server, which ensures the reliability of the system, preventing data loss, and interruption of servers.

Moreover, the distributed operating system can deal with a large number of simultaneous transactions(parallel) such as account inquiries, fund transfer, and bill payments. It allows many users to perform different transactions at the same time without conflict. So that, it reduces delay in performing transactions, as well as ensuring efficient utilization of resources. Also, these systems allow geographic distribution of data because they serve customers in many geographic regions.

These systems use load balancing techniques which ensure the best performance and utilization of resources. As a result, this helps to prevent overloading of some servers, and enhances the consistent and responsive experience of users.

# **Part 6**

**Concurrency in operating systems**

It is a feature of the operating system, which enables it to execute many processes or tasks at the same time (simultaneously). The system doesn’t complete one task or process in a sequential manner. Instead, concurrent system executes processes and tasks different concurrently, which makes overlapping between processes or tasks possible through execution. It is very important for encrusting utilizing of resources, enhancing responsive of the system, as well as enhancing performance of the overall system. It is a way of juggling different processes to ensure that executing processes moves forward in a coherent way.

Concurrency has a clear effect on performance in many different ways. It depends on the effectiveness of how it is managed. If it is managed effectively, it can maximize performance of the system though different ways:

* **Enhanced productivity**

Concurrent executions help executing many tasks at the same time, so it increases the whole productivity of the system, mainly in systems which have many processors.

* **Reduce response time:**

It makes overlapping of tasks possible so that it reduces user interaction times due to users experience which makes responses of the system faster, as well as making the overall system more responsive and interactive.

* **Increased utilization of resources:**

Concurrent execution improves utilization of memory, processors, and other resources more effectively because tasks are executed simultaneously while resources are available, and this reduces wasting time.

* **Parallel Execution of Independent Tasks:**

If we have independent tasks which need to be executed and they don’t depend on each other, it is efficient to use concurrency. We use it in order not to waste time waiting for each other because they can be executed at the same time simply and they don’t depend on each other. This will increase the responsiveness, and productivity of the system.

**On the other hand,** if it isn’t managed effectively, it can minimize performance of the system though different ways:

* **Synchronization overhead:**

Using synchronization excessively for executing many processes at the same time will lead to overhead on the resources of the system, mainly when thread of processes spends a large amount of time waiting for being executed. This minimizes the performance of the system.

* **Excessive context switching:**

Excessive Context switching of a large quantity of active processes can lead to overhead on the system and excessive switching between them, as well as reducing overall productivity. Also, it consumes more resources, leading to negative impacts on performance mainly when it occurs very frequently. Also, it consumes more time for saving the PCB of current process and loading the PCB of another process.

* **Race Conditions:**

If the process of accessing shared data simultaneously isn’t managed with efficient synchronization, this leads to unexpected behavior and corruption of data. Also, we need to use synchronization probably to resolve race conditions. However, if it isn’t managed efficiently, this leads to minimizing the benefits of concurrency.

* **Limited Resources:**

When resources of the system are limited, the process of operating large number of concurrent tasks may lead to contention for resources. This leads to more waiting time, as well as minimizing performance, mainly when the system does not have enough efficient techniques or mechanisms for resource allocation.

* **Sophisticated maintenance and debugging:**

Complex interactions between processes in concurrent systems makes maintaining and debugging of the system very challenging. As a result, we need to understand and resolve issues such as race conditions, and synchronization, this causes wasting time, as well as delay in the maintenance or development process.

**In conclusion,** all these problems are related to inefficient management of synchronization mechanisms rather than concurrency itself. As a result, concurrent systems which are well designed take into account these issues and implement strategies to reduce the negative impact on performance. We need profiling, robust testing, and optimization to ensure that concurrency maximizes rather than minimizes performance of the system.

**Example of Concurrency technique:**

Here is an explanation of a technique which applies the concept of concurrency in operating systems that is **multithreading**. This technique includes executing many multiple threads in one single process, as well as sharing the same address space. Every thread shows its own sequence of control which allows various parts of the program to operate separately.

An example of this technique is Web Browser, which uses multiple threads to perform concurrency in many aspects.

**Representing UI:** One thread deals with user interface which shows that the browser represents user interactions such as scrolling down, opening links, clicking buttons whereas other threads are performing tasks in the background.

**Loading pages:** another thread handles downloading and displaying a web page. At the same time the user is interacting with the displayed web page or loading another page.

**Background tasks:** Extra threads deal with background tasks such as checking for updates, updating bookmarks, and managing extensions, as well as interacting with the displayed page at the same time without affecting browsing the pages of the user or the tasks in the background.

**Consequently,** multithreading in the previous example allows for various tasks to execute concurrently in a coherent way, which enhances users’ experience, system responsiveness, as well as enhancing the overall performance of the system. **Most importantly,** mulita threading must be managed effectively in order to ensure data integrity and avoid synchronization issues.

# **Part 7**

**Remote Producer Call (RPC) 🡪**

It is a communication mechanism that is used in distributed operating systems to allow a process on one system to call a procedure from another system to be executed in another remote system, as if it is a local procedure invoked. The programmer doesn’t need to clearly write a code for the remote communication details. It can be performed in different languages, as well as in different frameworks and forms. This mechanism facilitates communication between client and server. The client and server each has a stub (proxy). The stub on the client-side is responsible for packaging parameters and sending a request to the server, while the stub on the server-side is responsible for unpacking the request, calling the procedure, and returning the results.

**Principle of work:**

The client send request to server to calls for a procedure from another system as if it was a local call. This call is done from client-side stub, which is a proxy for the distant procedure.

This proxy packages the parameter of the procedure into a message, and then sends it to the server. The stub deals with communication details and abstracts the complexities from the client. After that, the message is transmitted through the network using TCP/IP which are transport protocols.

The stub on the server side receives the message and unwraps the parameters of the message, then it passes or sends them to the real procedure on the server side, which in term executes the requested procedure by using the given parameters as if they were a local call. **Finally,** the result of this procedure execution will be sent back by the server-side stub to the client, which in turn will receive it and send it back to the client program which called it.

In brief, the main role of the RPC is to abstract the details of remote communication by making clients call procedures on another server as if they were local. It enhances simplicity and modularity in distributed system development.

**Remote Method Invocation (RMI) 🡪**

It is a communication mechanism that is used in distributed operating systems to perform remote communication by enabling java program to call methods on objects from a different java virtual machine, mainly on a remote server. It is specially designed to perform in java, as well as being part of the standard library of java. In other words, objects in one JVM can call methods on objects in another JVM.

**Principle of work:**

On the server side, objects perform a remote interface, so this declares the methods that are invoked remotely. Through the client-side stub, the client calls a method from a remote object, as if it was a local call. The methods parameters are serialized, and then they are packaged into a message by the stub, including the object identifier and the method name.

Using RMI Java protocol, the message is sent through the network. Then, it is received by the server-side stub on the serve side. This stub unwrapped the parameter of the message and calls the actual method, which is on the server object, and then executes it. **Finally,** the outcome of the method execution is serialized and returned to the client via the server-side stub. Then, the client-side stub receives the outcomes, unsterilized it, and send it back to the client program, which called it.

In brief, the main role of the MRI is allowing java objects to call methods on distant objects in various JVM, which provides a new approach of Java to develop distributed systems.

These two mechanisms play an essential role in making the development of distributing operating systems in distributed environments very easy and efficient in many different ways.

* They abstract the complexities of remote communications, which helps developers to concentrate on the logic of the applications instead of the details of the network.
* They make location transparency available, which enables clients to call methods or procedures without recognizing the remote services’ or objects’ physical location.
* They facilitate development of distributed systems by enhancing code maintainability, encouraging modularity, and offering high level abstractions.
* They contain security features, which allow access to remote services. This ensures safe and secure communication between clients and servers.
* RPC are designed to be language neutral, which facilitates communication and development in various programming languages. This enhances interoperability in different environments. Even though RMI is specific to Java, it also enhances interoperability in environments based on Java.
* Both mechanisms improve the development of distributed systems, which helps scalability depending on the growing demand for servers because developers are able to design applications which have distributed components within multiple machines.
* They can incorporate fault tolerance methods such as redundancy, retrying, and error handling, so that they can deal with failure securely, which in turn ensures reliability and development of the system.
* They are adapted to distribute requests of clients among multiple servers, which improves utilization of resources, system efficiency, as well as facilitating development of the system.
* Their implementation processes incorporate features such as effective serialization and data compression to improve bandwidth usage, mainly when the network bandwidth is limited, which facilitates development of the system.
* They include features to generate documentation of available methods and services. Also, they support service discovery, which help clients locate and connect with remote services dynamically, leading to facilitating development of the system.

# **Part 8**

**Infrastructure as a service (IaaS):** it is a service model offered by cloud computing, that offers virtual computing infrastructure and its resources through the Internet. This service model helps users to rent and manage components of virtual infrastructure such as storage, compute, VMs, and networking resources. It offers a service depending on the users demand and it is scalable in the future. When a user uses this service model, he will have control over the applications, operating system, and configurations. Payment in this model is based on the resources used. It provides users with many facilities and features such as managing and deploying applications, which enhances control and flexibility of the overall software.

The functionalities and operations of modern, and distributed OS play an essential role in the management and implementation of IaaS in many different ways:

**Functionalities of IaaS:**

Here's how **modern operating systems** contribute to the functionalities of IaaS:

* Modern operating systems involve features to manage some virtual environment efficiently, so that IaaS depends on virtualization technologies which help to generate virtual examples of computing resources.
* Users of IaaS can provide specific configuration for VMs. Also, the essential modern OS manages and allocates resources such as memory, and CPU for these VMs, as well as storing them.
* This model offers scalable storage solutions. The modern OS deals with access to storage, as well as ensuring the security and integrity of data.
* This model enables users to have control on various network components such as load balancing, firewall, and IP addresses. Modern OS can manage these components effectively by its crucial features such as drivers, and network stack.

Here’s how **distributed operating systems** contribute to the functionalities of IaaS:

* In environments of IaaS, mainly which have a large number of VMs, distributed OS can help this model in load balancing by distributing the incoming traffic of the network among multiple nodes, which prevents overloading on the different servers.
* IaaS has a main feature of scalability and also the distributed OS plays a crucial role in achieving this feature. This happens by helping the dynamic allocation, as well as dynamic deallocation of resources depending on the demand. This ensures scalability of the infrastructure horizontally.
* In IaaS, distributed OS helps for fault tolerance. When a server fails, principles of distributed OS can shift workload to available servers, which ensures maximizing the run time, high availability, as well as increasing reliability.

**Operations of IaaS:**

Here's how **modern operating systems** contribute to the operations of IaaS:

* The modern OS can allocate and deallocate resources effectively by using resource management techniques. This OS helps IaaS in resource management by preventing resource contention and making sure that every VM uses a fair share of resources.
* The modern OS has a key feature which is hypervisors that play essential role in controlling and managing the VMs. Also, they supervise the deletion, creation, and controlling of these VMs. This ensures utilization of resources, and proper isolation.
* The modern OS has security features that IaaS security depends on. These include access control, authentication of user, and encryption, which ensures integrity, and confidentiality of applications and data.

Here's how **distributed operating systems** contribute to the operations of IaaS:

* In IaaS, distributed OS helps for fault tolerance. When a server fails, principles of distributed OS can shift workload to available servers, which ensures maximizing the run time, high availability, as well as increasing reliability.
* IaaS has a main feature of scalability and also the distributed OS plays a crucial role in achieving this feature. This happens by ensuring that the operations can deal with increased workloads by providing the system with more VMs. This ensures scalability of the infrastructure horizontally.
* The distributed OS manages the networks for large IaaS which have multiple data centers. This ensures consistency of data, maintaining communication, and load balancing among distributed infrastructure.

**Platform as a service (PaaS):** it is a service model offered by cloud computing, it provides a platform that has infrastructure, development frameworks, services, and development tools, run time environments which facilitate the process of designing, deploying, testing and expanding applications. It is a scaling and an automated deployment of applications. Also, it enables users to run, develop, deploy, and manage the applications with no need to deal with the difficulties and complexities of the essential infrastructure.

PaaS minimizes complexities of the HW and OS by abstracting much of the significant infrastructure, which allows developers to concentrate on application development. The operations and functionalities of these two OS are essential for implementing PaaS in many different ways.

**Functionalities of PaaS:**

Here's how **modern operating systems** contribute to the functionalities of PaaS:

* PaaS offers development tools, libraries, and frameworks by abstracting significant infrastructure. The modern OS ensures smooth application development by helping the compatibility and integration of these frameworks.
* Its platforms involve database services, and the modern OS deals with the interactions between these databases which ensures data security, and consistency, as well as effective storage operations.
* Its platforms involve middleware components for integrations, communication, and messages. The modern OS gives the needed support for the integration of the middleware to ensure smooth communication between various components of the application.

Here’s how **distributed operating systems** contribute to the functionalities of PaaS:

* Its platforms benefit from the principles of distributed systems for automatic scalability. Both OS give more facilities for the management and development of application instances among nodes or servers. This ensures the efficient utilization of the resources.
* PaaS applications operate among multiple servers or multiple nodes and distributed OS include fault tolerance methods to enhance reliability of the system. If a HW failure occurs in any of the nodes, the distributed OS can distribute the workload among active nodes, which helps the PaaS to continuous operations.
* When PaaS applications interact with databases in distributed environments. The distributed OS applies protocols and algorithms which manage simultaneous access to shared data among multiple nodes. This ensures the consistency and integrity of the data of the application.

**Operations of PaaS:**

Here's how **modern operating systems** contribute to the operations of PaaS:

* PaaS performs abstracting of the significant infrastructure and the modern OS helps this abstraction. The modern OS enhances compatibility, offers a stable environment, and manages resources for the execution of the applications.
* The modern OS has security features that will enhance all security PaaS such as access control, authentication of user, secure execution for the applications.
* PaaS platforms involve storage solutions, and the modern OS can manage storage access, data retrieval and file systems, which ensures the efficient storage and retrieval of data.

Here's how **distributed operating systems** contribute to the operations of PaaS:

* PaaS platforms usually use the principles of distributed OS for expanding the scale of dynamic resource. This includes adding or removing resources according to demand. Also, distributed OS ensures the smoothness and efficiency of the scaling process.
* Its platforms distribute incoming traffic among multiple instances of the application by using load balancing. The principles of the distributed OS prevent loading on certain nodes, as well as ensuring that the load is efficiently balanced.

**Software as a service (SaaS):**

It is a service model offered by a cloud computing, which provides software applications via the internet based on subscriptions. Applications can be accessed by users throw a web browser with no need to install or download locally. Also, the service provider centrally hosts and maintains the software applications. This will allow for handling many tasks such as security, updates, and infrastructure management. It can perform updates and maintenance automatically.

**Functionalities of SaaS:**

Here's how **modern operating systems** contribute to the functionalities of SaaS:

* SaaS provides end-users with completely developed applications. Modern OS play a crucial role in managing system resources, offering a user-friendly interface, as well as providing a seamless user experience.
* SaaS providers deal with maintenance tasks such as security fixes, patches and updates. Modern OS makes it easier to update software to ensure that the software is functioning properly.
* SaaS applications depend on the solutions of data storage. Modern OS can manage storage access, data security and file systems, which enhances the availability and integrity of user data.

Here’s how **distributed operating systems** contribute to the functionalities of SaaS:

* Distributed OS ensures scalability especially in SaaS environments, which extend data servers and multiple systems. These help the management and deployment of SaaS applications, mainly among distributed nodes, which ensures the efficient horizontal scaling of resources to accommodate various workloads.
* Distributed OS helps load balancing, especially for SaaS applications which operate among multiple nodes. They help incoming traffic of the network to be distributed among different servers, which stops overloading over certain nodes. This enhances the utilization of resources, as well as ensuring high availability.
* SaaS applications depend on storage systems and distributed databases. Distributed OS use methods of fault tolerance to deal with hardware failures, as well as enhancing data availability and continuous operation, even in the failures of server or face of node.

**Operations of SaaS:**

Here's how **modern operating systems** contribute to the operations of SaaS:

* Modern OS enhances user experience and user interface in SaaS applications. It can manage system interactions, input/output and graphical elements. This provides end-users with a smooth interface.
* SaaS applications are capable of storing and retrieving data from file systems and underlying databases. Modern OS contribute to effective storing, retrieving and managing data for SaaS providers.
* Modern OS plays a crucial role in offering a secure and efficient execution environment of the application while SaaS providers deal with most of the security issues.

Here's how **distributed operating systems** contribute to the operations of SaaS:

* The principles of distributed OS operate to achieve load balancing in situations when applications of SaaS are distributed among data centers of multiple servers. This makes sure that the application can deal with different workloads and the requests of the user are efficiently distributed.
* Distributed OS contribute to consistency of data among users, which includes managing simultaneous access to the shared data, as well as keeping the integrity of the state of the application.
* Principles of the Distributed OS enhance the high degree of availability of the applications of SaaS. It provides some features such as fault tolerance, redundancy, and load balancing, which make sure that the service is available even if a failure occurs in the server or there is increased demand.

# **References:**

* <https://youtu.be/NYBKXzl5bWU?si=8z3Pf46LR1RU4INy>
* <https://www.techtarget.com/searchstorage/definition/virtual-memory>
* <https://www.geeksforgeeks.org/what-is-a-distributed-operating-system/>
* <https://www.javatpoint.com/distributed-operating-system>
* <https://www.tutorchase.com/notes/ib/computer-science/7-2-1-centralised-vs-distributed-systems>
* <https://www.geeksforgeeks.org/concurrency-in-operating-system/>
* <https://www.techtarget.com/searchapparchitecture/definition/Remote-Procedure-Call-RPC#:~:text=Remote%20Procedure%20Call%20is%20a,systems%20like%20a%20local%20system>.
* <https://www.geeksforgeeks.org/difference-between-rpc-and-rmi/>
* <https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/>