**UNIT – 1**

**What is a Distributed Operating System?**

A Distributed Operating System refers to a model in which applications run on multiple interconnected computers, offering enhanced communication and integration capabilities compared to a network operating system.

In a Distributed OS, multiple CPUs are utilized, but for end-users, it appears as a typical centralized operating system. It enables the sharing of various resources such as CPUs, disks, network interfaces, nodes, and computers across different sites, thereby expanding the available data within the entire system.

Effective communication channels like high-speed buses and telephone lines connect all processors, each equipped with its own local memory and other neighboring processors. This [operating system](https://www.javatpoint.com/os-tutorial) consists of numerous computers, nodes, and sites joined together via [**LAN/WAN**](https://www.javatpoint.com/lan-vs-wan) lines. These are also known as **loosely coupled systems**.

**Examples**: Solaris,OSF/1, Micros, DYNIX, Locus, Mach.

## Types of Distributed Operating System

There are three types of Distributed Operating System.

### **1. Client-Server Systems**

This strongly connected operating system is appropriate for multiprocessors and homogenous multicomputer. It functions as a centralized server, handling and approving all requests originating from client systems.

### **2. Peer-to-Peer Systems**

Peer-to-Peer System is loosely coupled system is implemented in computer network applications, consisting of multiple processors without shared memories or clocks. Each processor possesses its own local memory, and communication between processors occurs through high-speed buses or telephone lines.

### **3. Middleware**

Middleware facilitates interoperability among applications running on different operating systems. By employing these services, applications can exchange data with each other, ensuring distribution transparency.

### 4. Three-Tier

Development is made easier because client data is saved in the intermediate tier rather than the client itself. Online applications are where this kind of architecture is most frequently found.

### 5. N-Tier

N-tier systems are utilized when a server or application has to send requests to other corporate services over a network.

## Applications of Distributed Operating System

The applications of a Distributed OS encompass various domains as below:

* Internet Technology
* Distributed Databases System
* Air Traffic Control System
* Airline Reservation Control Systems
* Peer-to-Peer Networks System
* Telecommunication Networks
* Scientific Computing System
* Cluster Computing
* Grid Computing
* Data Rendering

## Advantages of Distributed Operating System

Below are some Advantages of Distributed Operating System.

* It can increase data availability throughout the system by sharing all resources (CPU, disk, network interface, nodes, computers, and so on) between sites.
* Because all data is replicated across all sites, it reduces the probability of data corruption because users can access data from another operating site in the event that one site fails.
* Data transfer from one site to another is accelerated by it.
* Since it may be accessible from both local and remote sites, it is an open system.
* It facilitates a reduction in the time needed to process data.
* The majority of distributed systems are composed of multiple nodes that work together to provide fault tolerance. Even if one machine malfunctions, the system still functions.

## Disadvantages of Distributed Operating System

Below are some Disadvantages of Distributed Operating System.

* Which tasks need to be completed, when they need to be completed, and where they need to be completed must be determined by the system. The restrictions of a scheduler might result in unpredictable runtimes and unused hardware.
* Since the nodes and connections in DOS need to be secured, it is challenging to establish sufficient security.
* Comparing a DOS-connected database to a single-user system, the latter is easier to maintain and less complex.
* Compared to other systems, the underlying software is incredibly sophisticated and poorly understood.
* One should anticipate more communication latency in systems that are more broadly distributed. Teams and developers are then forced to make trade-offs between latency, consistency, and availability.
* Because they are regarded as being excessively costly, these systems aren’t readily accessible.
* Compiling, analyzing, displaying, and keeping track of hardware utilization metrics for large clusters may be quite challenging.

**System Call vs System Program**

|  |  |  |
| --- | --- | --- |
| **Basis** | **System Call** | **System Program** |
| **Definition** | It is a technique in which a computer system program requests a service from the OS kernel. | It offers an environment for a program to create and run. |
| **Request** | It fulfils the low-level requests of the user program. | It fulfils the high-level request or requirement of the user program. |
| **Programming Languages** | It is usually written in C and C++ programming languages. Assemble-level language is used in system calls where direct hardware access is required. | It is commonly written in high-level programming languages only. |
| **User View** | It defines the interface between the services and the user process provided by the OS. | It defines the user interface (UI) of the OS. |
| **Action** | The user process requests an OS service using a system call. | It transforms the user request into a set of system calls needed to fulfil the requirement. |
| **Classification** | It may be categorized into file manipulation, device manipulation, communication, process control, information maintenance, and protection. | It may be categorized into file management, program loading and execution, programming-language support, status information, file modification, and communication. |