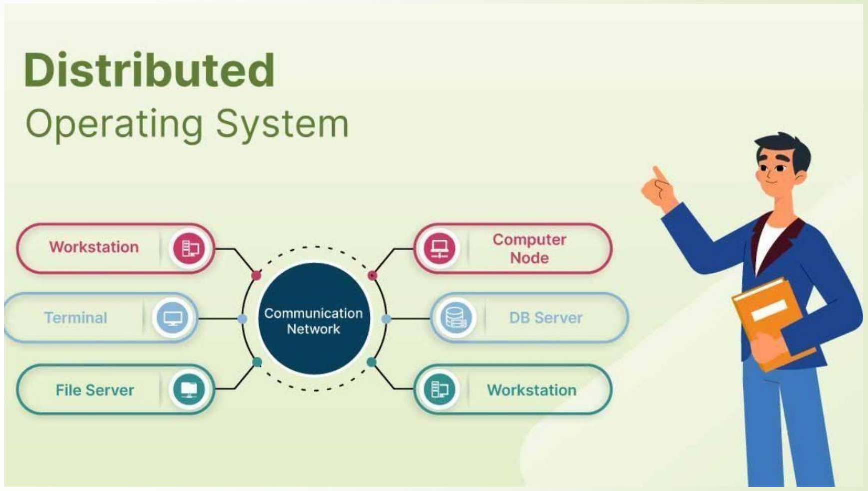
Emerging Trend in Operating System

Introduction to Distributed OS

- **Definition**: A Distributed Operating System (DOS) is a software system that manages a collection of independent computers and makes them appear to users as a single coherent system, often abstracting the complexities of the underlying hardware.



Concept Of Distributed OS

- **Single system image**: Users and applications can access and manipulate resources across the network as if they were local.
- **Reliability and availability**: If one system fails, the other systems can continue to operate.
- **Communication**: Processors communicate over high-speed buses or telephone lines.
- **Control**: Control involves allocating tasks and data to system elements.
- **Scalability**: Distributed systems scale better horizontally and offer fewer points of system-wide failure.
- **Middleware**: Middleware facilitates interoperability, scalability, and fault tolerance.
- **Three-tier design**: Divides a distributed system logically into three layers: presentation, application, and data.

Character Of Distributed OS

- **Transparency**: These systems strive for various forms of transparency, such as *location*, *migration*, *replication*, and *concurrency*, enabling users and applications to access distributed resources like they would local ones.
- **Resource Sharing**: A key function is the ability to share resources—processing power, storage, and network services—across multiple machines, enhancing efficiency and utilization.
- **Scalability**: DOS can easily scale by adding new machines to the system, which can accommodate growing workloads without significant infrastructure changes.
- **Fault Tolerance**: These systems are designed with fault tolerance in mind. If one component fails, the system can redistribute tasks and resources to maintain functionality.
- **Security**: Enhanced security strategies are implemented in distributed systems to ensure authentication, confidentiality, and integrity across different nodes in the network.

Role Of Distributed OS

- **Resource Management:**
- A Distributed OS oversees the allocation of resources (e.g., CPU, memory, storage) across the network of machines, ensuring resources are shared efficiently and fairly.
- **Communication and Synchronization:**
- The DOS provides mechanisms for processes on different nodes to communicate, such as message passing or remote procedure calls (RPC).
- **Fault Tolerance and Reliability:**
- A Distributed OS continuously monitors the health of nodes and processes, detecting failures quickly.
- **Fault Isolation and Recovery:**
- It isolates failures to minimize their impact on the rest of the system. For example, if one node fails, the OS can migrate tasks from that node to others, isolating the failure and allowing the system to continue functioning.
- **User and Process Interaction:**
- A Distributed OS presents a unified interface to users and applications, allowing them to interact with the system as if it were a single, monolithic machine, despite being spread across multiple nodes.

Introduction to Cloud OS

- A cloud operating system enables the management, deployment, and administration of resources within a cloud-based infrastructure. It typically offers a unified interface for data storage, computing resources, and networking facilities, allowing users to efficiently utilize hardware without the limitations of physical servers

Concept Of Cloud OS

- **Lightweight**
- A cloud operating system is a lighter version of a typical mobile or desktop operating system.
- **Access from anywhere**
- A cloud operating system allows users to access data from anywhere and on any device.
- **Pre-installed applications**
- A cloud operating system comes with pre-installed applications that can be accessed through a web browser.
- **Easy upgrades**
- A cloud operating system eliminates the need for software updates. Users can install a new version of the operating system when it's available on the web.
- Some examples of cloud operating systems include: Chrome OS, Netvibes, EyeOS, Joli, and Amoeba.

Cloud OS Architecture: A Multi-Layer Approach

IaaS (Infrastructure as a Service)

Provides access to virtualized computing resources, such as servers, storage, and networking.

PaaS (Platform as a Service)

Offers a platform for developing and deploying applications, providing tools and services for application lifecycle management.

SaaS (Software as a Service)

Delivers software applications over the internet, allowing users to access them through web browsers or mobile devices.

Character Of Cloud OS

- Virtualization: Manages virtual machines (VMs) and containers, abstracting hardware resources for scalable and efficient resource allocation.
- Scalability: Supports dynamic scaling of resources, allowing the system to grow or shrink based on demand.
- Resource management: Handles the allocation, scheduling, and provisioning of cloud resources like compute, storage, and networking.
- Multi-tenancy: Enables multiple users or organizations (tenants) to share resources securely while maintaining isolation.
- Fault Tolerance: Ensures system resilience by distributing workloads across multiple servers and providing redundancy and recovery mechanisms.
- Automation: Supports automation of tasks such as provisioning, deployment,
  - monitoring, and scaling to improve efficiency and reduce manual intervention.

Role Of Cloud OS

- Resource Management: Efficiently allocates and manages computing resources across distributed servers.
- Scalability: Allows easy scaling of resources up or down based on demand, adapting to changing business needs.
- Accessibility: Provides users access to applications and data from anywhere with an internet connection, enhancing mobility.
- Centralized Management: Offers tools for centralized control over monitoring, security, and resource allocation, simplifying IT administration.
- Enhanced Security: Incorporates advanced security measures like encryptionand access controls to protect data and applications.
- Support for Cloud-Native Applications: Facilitates the development and deployment of applications designed specifically for cloud environments.
- Cost Efficiency: Reduces operational costs by minimizing the need for extensive on-premises hardware through a pay-as-you-go model.
- Improved Collaboration: Enhances real-time data sharing and communication, fostering teamwork and productivity among users in different locations.

Introduction to Security Issues in Modern Operating Systems

- Security is a foundational concern in the design and deployment of operating systems, particularly in the context of cloud computing, mobile devices, and distributed systems. Modern OS environments face evolving threats that necessitate robust security measures. This section explores the prevalent security challenges and effective deployment methods ensuring secure system operations.

Key Security Issues: Part 1

1. Vulnerabilities in OS Kernels

Exploits like buffer overflows and privilege escalation are critical risks.

Example: The Spectre and Meltdown vulnerabilities exposed flaws in speculative execution.

2. Malware and Ransomware

Increasingly sophisticated malware targets OS weaknesses.

Example: Wannacry ransomware affected over 200,000 systems worldwide.

Key Security Issues: Part 2

3. Unauthorized Access

Weak or default authentication systems lead to breaches.

Example: Exploits in outdated authentication protocols in IoT devices.

4. Data Breaches

Misconfigured permissions and inadequate encryption lead to leaks.

5. DDoS Attacks

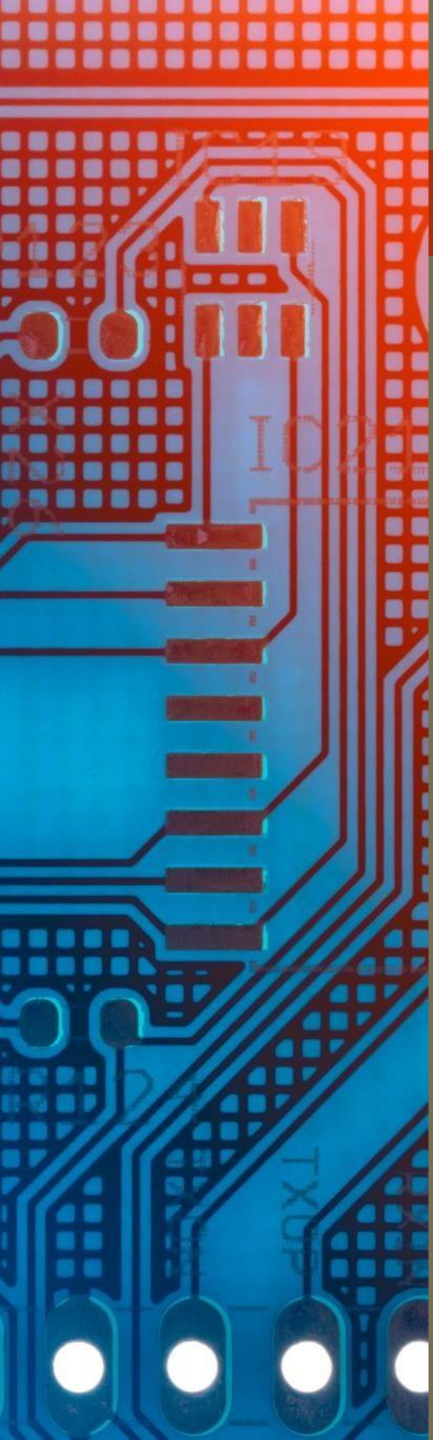
Targeting distributed and cloud-based OS systems, overwhelming resources.

Methods of Deployment for Enhanced Security

- 1 . Secure Boot Processes: Ensures only trusted software loads during startup.
- 2 . Encryption: Implements full-disk encryption for data protection.
- 3 . Sandboxing: Isolates applications to minimize system-wide impact in case of breaches.
- 4 . Regular Updates and Patching: Addresses known vulnerabilities promptly.
- 5 . Zero-Trust Architecture: Ensures every access request is authenticated and verified.
- 6 . Containerization and Virtualization: Separates workloads to contain breaches effectively.

MEMORY WALL & BOTTLENECK

- The memory wall refers to the increasing gap between processor speed and memory bandwidth, where the rate of improvement in processor performance outpaces the rate of improvement in memory performance due to limited I/O and decreasing signal integrity.
- This disparity limits the overall system performance, as the processor spends more time waiting for data from memory, leading to a bottleneck.



Causes of the Memory Wall

1. CPU-Memory Speed Disparity: CPUs process data faster than memory can supply.
2. Memory Bandwidth Limitations: Insufficient data transfer rates for modern multi-core processors.
3. Cache Hierarchy Inefficiencies: Limited cache size leads to frequent misses.
4. Data-Intensive Applications: AI and big data processing demand higher memory throughput.

Impacts of the Memory Wall

1. PERFORMANCE DEGRADATION: SLOWS DOWN CRITICAL PROCESSES.

2. RESOURCE UNDERUTILIZATION: IDLE CPU CYCLES WHILE AWAITING MEMORY FETCH.

3. SCALABILITY CHALLENGES: INCREASED LATENCY IN DISTRIBUTED SYSTEMS.

Solutions to Address Memory Bottlenecks

1. Advanced Caching Techniques: Employing larger, smarter caches to reduce misses.

2. Non-Volatile Memory (NVM): Example: Intel Optane technology for faster access and persistence.

3. Memory Compression: Reducing data size to optimize bandwidth usage.

4. Parallelism and Prefetching: Fetching data in advance and accessing memory in parallel streams.

5. Heterogeneous Memory Systems: Combining DRAM and NVM for efficiency.