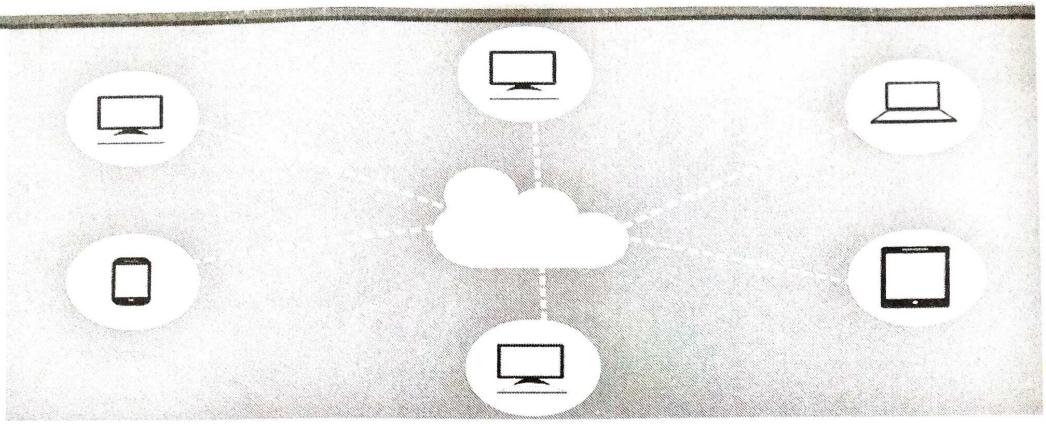
Chapter

1



DISTRIBUTED SYSTEM

1.1 DISTRIBUTED SYSTEM-AN INTRODUCTION

A *distributed system* is a network of multiple independent computers that communicate and coordinate their actions through shared resources to achieve a common goal. Unlike traditional computing, which depends on a single central machine, distributed systems divide the workload among several linked nodes. In this system, every node interacts with other nodes, exchanges resources, and manages tasks collaboratively. These systems are designed to solve problems that may be beyond the capability of a single computer or to improve overall performance and reliability.

A distributed system can consist of any type of machine (such as computers, physical servers, virtual machines, containers, or any other type of node) that can connect to a network, have local memory, and exchange messages.

There are two general ways that distributed systems function:

1. Each machine works toward a common goal and the end-user views results as one cohesive unit.
2. Each machine has its own end-user and the distributed system facilitates sharing resources or communication services.

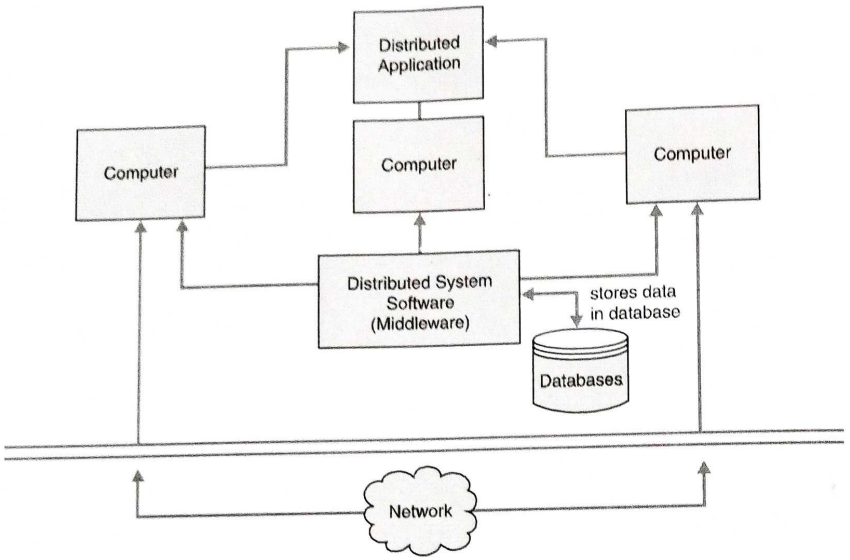


Fig. 1.1 Distributed System

A typical distributed system is shown in **Fig. 1.1**. As we can see that each computer is capable of independent processing and has its own memory and the computers are connected by a communication network. Computers in a distributed system communicate with each other to share information and coordinate their actions. Communication can happen through various channels such as message passing, remote procedure calls (RPC), or shared memory.

- *Distributed System Software (Middlware):* This software layer makes it easier for distributed components to coordinate, communicate, and be managed. Computers can coordinate their actions and share resources like hardware, software, data, etc. It gives developers the tools, libraries, and APIs they need to create distributed applications.
- *Distributed Application:* A software program or group of programs intended to run on several linked computers is referred to as a distributed application in a distributed system. These applications use distributed nodes' cooperative processing power to carry out tasks, share resources, and accomplish shared objectives.
- *Database:* It is used to store the processed data that are processed by each Node/System of the Distributed systems that are connected to the Centralized network.

1.1.1 Characteristics of Distributed System

Key characteristics of distributed System are:

1. **Concurrency:** Multiple components in a distributed system can execute concurrently, enabling parallel processing and efficient resource utilization.
2. **Scalability:** Distributed systems can easily scale by adding or removing resources or nodes to accommodate changing demands, thereby improving performance.
3. **Resource Sharing:** It is the ability to use any Hardware, Software, or Data anywhere in the System.
4. **Fault Tolerance:** They are designed to continue operating even if some components fail, ensuring system reliability.
5. **Transparency:** A distributed system aims to provide users and applications with the illusion of a single, well-integrated computing platform despite the underlying complexity.

1.1.2 Components of a Distributed System

The key Components of a distributed system are:

1. **Nodes/Computers:** These are individual machines or devices connected over a network. Each node is capable of processing, storing, or transmitting data.
2. **Network Communication:** The network infrastructure, whether wired (Ethernet, fiber optics) or wireless (Wi-Fi, cellular), enables data exchange between nodes. Network protocols and technologies govern how information is transmitted and received.
3. **Resource Sharing:** Distributed systems allow sharing resources like data, storage, processing power, and services among nodes.
4. **Middleware:** This is the software layer that facilitates communication, coordination, and management of distributed components. It includes tools, libraries, and APIs for developers to build distributed applications.
5. **Security Infrastructure:** Components such as encryption, authentication, access control, and firewalls protect data and resources, ensuring the system's integrity and confidentiality.

6. **Naming and Directory Services:** Systems rely on services like DNS or directory services for resource discovery and addressing, enabling nodes to locate resources across the network.

1.1.3 Types of Distributed Systems

Distributed systems come in various types, each tailored to specific requirements and application scenarios:

1. **Client-Server Architecture:** A client-server distributed system uses a simple communication method where a client sends input to a separate server and the server returns an output response to the client. This model is prevalent in web applications.
2. **Peer-to-Peer Architecture:** Nodes collaborate without a central server, sharing resources and responsibilities. Peer-to-peer systems are decentralized and often used in file sharing.
3. **Clustered Systems:** Nodes are grouped into clusters, and each cluster operates independently. Clusters enhance fault tolerance and performance.
4. **Cloud Computing:** Services and resources are provided over the internet, allowing users to access computing power and storage on-demand.
5. **Grid Computing:** Utilizes the collective power of networked computers for large-scale computational tasks, often in scientific research or data analysis.
6. **Distributed Databases:** Data is distributed across multiple nodes to enhance performance, scalability, and fault tolerance in database management.
7. **Middleware-Based Systems:** Middleware provides a layer of abstraction, simplifying communication and coordination among distributed components.

1.1.4 Advantages of Distributed System

Distributed systems offer several benefits that contribute to improved performance, reliability, scalability, and flexibility in computing environments. Here are some key benefits:

- **Fault Tolerance:** Distributed systems can replicate data and services across multiple nodes, reducing the impact of failures. If one node fails, others can continue to function, ensuring system availability.

- **Scalability:** Distributed systems can scale by adding more machines or nodes to the network. This allows for better handling of increased workloads and improved performance.
- **Performance Improvement:** Tasks can be divided among multiple nodes, allowing for parallel processing and faster execution of tasks. This is particularly beneficial for computationally intensive applications.
- **Resource Sharing:** Distributed systems enable the efficient use of resources across the network. Tasks can be distributed based on the availability of resources, optimizing overall system performance.
- **Flexibility:** Users and applications may not be aware of the physical location of resources in a distributed system. This provides flexibility in resource allocation and management.
- **Cost-Effectiveness:** Distributed systems allow for the pooling of resources, which can lead to cost savings. Resources can be shared among multiple users or applications, avoiding the need for dedicated resources for each task.
- **Improved Reliability:** By distributing data and services across multiple nodes, the system becomes less susceptible to single points of failure. This improves overall system reliability.
- **Increased Availability:** Distributed systems can balance the load across multiple nodes, ensuring that no single node is overwhelmed with too much work. This helps maintain high availability of services.
- **Data Consistency:** Distributed systems provide mechanisms for replicating data and managing consistency across replicas. This ensures that data remains accurate and up-to-date across the system.
- **Global Reach:** Distributed systems can be designed to span multiple geographical locations. This is essential for providing services to users across different regions and ensuring low-latency access.

1.2 EXAMPLES OF DISTRIBUTED SYSTEM

Distributed systems are prevalent in various domains, spanning from computer networks and cloud computing to large-scale web applications.

Here are a few examples that illustrate the diversity of distributed systems:

- **World Wide Web (WWW):** The *World Wide Web* is a classic example of a distributed system. It involves distributed servers hosting websites, and clients (web browsers) requesting and receiving web pages. Content Delivery Networks (CDNs) further distribute content to edge servers for faster access globally.
- **Web search:** Web search engines like *Google* utilize a distributed system to handle vast amounts of data. Google's search infrastructure involves distributed indexing, crawling, and ranking algorithms across numerous servers. The distributed nature allows for parallel processing, making search results faster and more scalable. Distributed systems in web search efficiently manage the complexities of indexing the entire internet and delivering relevant results to users.
- **Cloud Computing Platforms:** Cloud services like *Amazon Web Services* (AWS), *Microsoft Azure,* and *Google Cloud Platform* (GCP) are distributed systems that provide on-demand computing resources. They involve a network of data centers worldwide, enabling scalable and flexible access to computing, storage, and networking resources.
- **Distributed Databases:** Systems like *Apache Cassandra, Amazon DynamoDB,* and *Google Spanner* are distributed databases. They store and manage data across multiple nodes, offering high availability, fault tolerance, and scalability. Data is partitioned and replicated to ensure efficient and reliable access.
- **Peer-to-Peer (P2P) Networks:** P2P networks, such as *BitTorrent* and *blockchain* networks like Bitcoin, are distributed systems where nodes collaborate without a central server. In BitTorrent, files are distributed among peers, while blockchain networks use a decentralized consensus mechanism to maintain a secure and tamper-resistant ledger.
- **Distributed File Systems:** *Hadoop Distributed File System* (HDFS) and *Google File System* (GFS) are examples of distributed file systems. They enable the storage and retrieval of large amounts of data across multiple nodes, providing fault tolerance and parallel processing capabilities.
- **Social Media Platforms:** Social media platforms like *Facebook* and *Twitter* operate as distributed systems. They handle large user bases, distribute content globally, and utilize distributed databases to store and retrieve user data efficiently.

- **Sensor Networks:** *Internet of Things* (IoT) applications often involves distributed sensor networks. These networks collect data from various sensors, such as temperature or humidity sensors, and transmit the information to centralized servers or other nodes for analysis.
- **Distributed Computing Frameworks:** *Apache Hadoop* and *Apache Spark* are frameworks for distributed data processing. They allow the parallel processing of large datasets across a cluster of machines, facilitating tasks like data analysis, machine learning, and batch processing.
- **Online Shopping Platforms:** E-commerce platforms like *Amazon* and *Alibaba* operate as distributed systems to handle the complexities of online shopping. They manage product catalogs, user accounts, and transactions across distributed servers for scalability and reliability.
- **Telecommunication Networks:** Telecommunication networks, including cellular networks and the Internet, are distributed systems. They involve a vast network of interconnected devices and servers to facilitate communication and data exchange globally.
- **Distributed Real-time Systems:** Distributed real-time systems synchronize and process data simultaneously across multiple interconnected nodes, ensuring timely and reliable response to events. Examples include real-time financial trading systems, industrial automation, and collaborative editing platforms, where instantaneous processing and communication are crucial for system performance and functionality. Airlines use flight control systems, Uber and Lyft use dispatch systems, manufacturing plants use automation control systems, logistics and e-commerce companies use real-time tracking systems.
- **Distributed artificial intelligence:** Distributed artificial intelligence (DAI) involves the collaborative processing of AI tasks across multiple nodes or devices. DAI enables efficient computation, data sharing, and model training in decentralized networks. Examples include federated learning and distributed neural networks, optimizing AI capabilities while addressing scalability, privacy, and resource constraints.

These examples demonstrate the wide-ranging applications of distributed systems, highlighting their importance in enabling scalable, reliable, and efficient computing across diverse domains.

1.3 TRENDS IN DISTRIBUTED SYSTEMS

Distributed systems are undergoing a period of significant change and this can be traced back to a number of following influential trends:

1.3.1 Pervasive networking

Pervasive networking is a concept that involves creating a seamlessly interconnected environment where communication and connectivity are pervasive across a wide array of devices, enabling them to work together seamlessly. This concept extends beyond traditional computer networks to include everyday objects and devices that are embedded with communication capabilities, creating a networked ecosystem that enhances convenience and functionality. Here's a more detailed explanation with an example:

Example: Smart City

Imagine a smart city where pervasive networking is implemented for various applications. In this scenario, a multitude of devices and systems are interconnected to improve the quality of urban life.

(a) Traffic Management:

- **Devices:** Traffic lights, sensors, cameras, and GPS systems embedded in vehicles.
- **Pervasive Networking:** Traffic lights are connected to a central system that monitors real-time traffic conditions through embedded sensors and cameras. Vehicles equipped with GPS systems contribute to the overall traffic data.
- **Benefits:** The central system can dynamically adjust traffic light timings based on the current traffic flow, optimize routes for emergency services, and provide real-time traffic updates to drivers.

(b) Environmental Monitoring:

- **Devices:** Air quality sensors, weather stations, and pollution monitoring equipment.
- **Pervasive Networking:** These devices are interconnected to collect and share data about air quality, weather conditions, and pollution levels in real-time.

- **Benefits:** Authorities can receive instant updates on environmental conditions, enabling them to take prompt actions such as issuing alerts or adjusting city infrastructure to mitigate the impact of adverse conditions.

(c) Smart Grids:

- **Devices:** Smart meters, electricity distribution systems, and renewable energy sources.
- **Pervasive Networking:** Smart meters communicate with the electricity distribution system, providing real-time data on energy consumption. Renewable energy sources are integrated into the grid and contribute data on energy production.
- **Benefits:** The system can dynamically manage energy distribution, optimize usage during peak hours, and seamlessly incorporate renewable energy sources to promote sustainability.

(d) Public Services:

- **Devices:** Public kiosks, waste management sensors, and surveillance cameras.
- **Pervasive Networking:** Public kiosks provide information about city services, waste management sensors signal when bins need emptying, and surveillance cameras enhance security.
- **Benefits:** Citizens can access information easily, waste management becomes more efficient, and the surveillance system can respond rapidly to security incidents.

In this smart city example, pervasive networking enables diverse devices and systems to collaborate, share information, and respond in real-time to changing conditions. This interconnectedness enhances the overall efficiency, sustainability, and quality of life for residents.

1.3.2 Mobile Computing

Mobile computing refers to the use of portable computing devices, such as smartphones, tablets, and laptops, to access and process information while on the move. It involves wireless communication, enabling users to interact with data, applications, and services regardless of their physical location. Mobile computing has become integral to modern lifestyles, providing flexibility, connectivity, and a wide range of applications and services.

Components of Mobile Computing Architecture:

(a) **Mobile Devices:**

- **Hardware:** This includes Smartphone, tablets, wearable, and other portable devices with embedded sensors and communication capabilities.
- **Operating System:** Mobile operating systems like Android, iOS, or others manage device resources and provide a platform for running applications.

(b) **Communication Networks:**

- **Wireless Networks:** Mobile devices connect to cellular networks, Wi-Fi, or other wireless technologies for data communication.
- **Middleware:** Middleware components handle communication protocols, ensuring seamless data exchange between devices and servers.

(c) **Mobile Cloud Computing:**

- **Cloud Services:** Mobile applications often leverage cloud services for data storage, processing, and additional computational resources.
- **Cloud Middleware:** Middleware components in the cloud facilitate communication and data synchronization between the mobile device and cloud services.

(d) **Mobile Application Architecture:**

- **Frontend:** The user interface and user experience components of mobile applications.
- **Backend:** Server-side components that handle data storage, processing, and business logic.
- **Application Logic:** Middleware components that facilitate communication between the frontend and backend.

.3 Edge Computing

Edge computing is a distributed computing paradigm that brings computational abilities closer to the data source or "edge" of the network, reducing latency and roving the efficiency of data processing. This approach involves processing data locally r near the edge devices, rather than relying solely on centralized cloud servers. Edge

computing is particularly beneficial for applications that require real-time processing and low-latency responses.

Example: Smart Manufacturing

In a smart manufacturing environment, edge computing plays a crucial role in optimizing operations. Consider a factory with numerous IoT sensors and connected machinery:

(a) **Sensor Data Processing:**

- **Sensors:** Various sensors collect data on equipment performance, temperature, and other relevant parameters.
- **Local Edge Devices:** On-site edge devices, such as gateway devices or edge servers, process this data in real-time.

(b) **Real-time Decision Making:**

- **Predictive Maintenance:** Edge computing enables the analysis of sensor data locally to predict equipment failures or maintenance needs.
- **Quality Control:** Cameras and sensors on the production line use edge computing to quickly identify and rectify defects.

(c) **Reduced Latency:**

- **Local Processing:** Instead of sending all data to a distant cloud, edge computing processes critical data locally, reducing the latency between data generation and action.
- **Control Systems:** Edge devices execute control commands swiftly, ensuring rapid response times in automated manufacturing processes.

(d) **Bandwidth Optimization:**

- **Data Filtering:** Edge devices filter and aggregate data locally, transmitting only essential information to the central cloud.
- **Efficient Communication:** By sending only relevant data, edge computing optimizes bandwidth usage and reduces the reliance on continuous high-speed connectivity.

(e) **Scalability and Flexibility:**

- **Distributed Architecture:** Edge computing allows for a distributed architecture where additional edge devices can be seamlessly added to accommodate growing computational needs.
- **Adaptability:** The system can adapt to changing manufacturing requirements without significant changes to the overall infrastructure.

In this example, edge computing enhances smart manufacturing processes by providing real-time data analysis, reducing latency, optimizing bandwidth usage, and facilitating scalable and flexible operations, ultimately contributing to improved efficiency, reduced downtime, and enhanced overall productivity in the manufacturing environment.

1.3.4 Cloud Computing

Cloud computing is a paradigm of distributed computing where computing resources, such as processing power, storage, and applications, are delivered as services over the internet. It involves the use of a network of remote servers hosted on the internet to store, manage, and process data rather than a local server or a personal computer. Here's an explanation of how cloud computing exemplifies distributed computing, along with an example:

Key Aspects of Cloud Computing as Distributed Computing:

(a) **Resource Distribution:**

- **Multiple Servers:** Cloud providers maintain data centers with numerous servers distributed across geographical locations.
- **Resource Allocation:** Users access computing resources from this distributed infrastructure, allowing for efficient allocation and utilization.

(b) **Scalability and Flexibility:**

- **Elasticity:** Cloud services offer scalability, allowing users to scale resources up or down based on demand.
- **Flexible Resource Allocation:** Users can provision and de-provision resources dynamically to accommodate changing workloads.

(c) **On-Demand Access:**

- **Resource Provisioning:** Users can access computing resources and services on-demand, without the need for upfront investment in physical infrastructure.
- **Pay-as-You-Go Model:** Users pay for the resources they consume, aligning costs with actual usage.

(d) **Virtualization:**

- **Virtual Machines and Containers:** Cloud providers use virtualization technologies to create virtual instances of servers (virtual machines or containers), allowing multiple workloads to run on a single physical machine.
- **Isolation:** Virtualization enables workload isolation and resource sharing among multiple users.

(e) **Distributed Storage:**

- **Data Replication:** Cloud storage systems replicate data across multiple servers to ensure data availability and fault tolerance.
- **Scalable Storage:** Users can access scalable and distributed storage solutions like Amazon S3 or Google Cloud Storage.

(f) **Managed Services:**

- **Distributed Services:** Cloud providers offer various managed services (e.g., databases, machine learning, analytics) distributed across their infrastructure.
- **Reduced Management Overheads:** Users can leverage these services without the need to manage the underlying infrastructure.

1.3.5 Distributed Multimedia Computing

Distributed multimedia computing involves the processing, storage, and transmission of multimedia data (such as text, images, audio, and video) across multiple networked devices. This approach leverages distributed systems to efficiently handle the complexity and resource requirements of multimedia applications.

Example: Video Streaming Service

Consider a distributed multimedia computing scenario in the context of a video streaming service:

(a) Content Storage and Retrieval:

- **Distributed Servers:** Multimedia content, such as video files, is stored across distributed servers in various geographic locations.
- **Content Delivery Network (CDN):** A CDN caches and distributes multimedia content to strategically located servers, reducing latency and enhancing content retrieval speed.

(b) Parallel Processing for Transcoding:

- **Transcoding Servers:** Distributed servers are responsible for transcoding video files into various formats suitable for different devices and network conditions.
- **Load Balancing:** Load balancing mechanisms distribute transcoding tasks across multiple servers, ensuring efficient processing and faster content delivery.

(c) Real-time Streaming:

- **Edge Servers:** Distributed edge servers, positioned close to end-users, handle real-time streaming requests.
- **Adaptive Bitrate Streaming:** Algorithms on edge servers dynamically adjust the video quality based on the user's network conditions, ensuring a smooth streaming experience.

(d) Interactive Features:

- **Distributed Processing:** Interactive features, such as live chat or user comments, are processed by distributed servers, allowing users to engage with multimedia content in real-time.
- **Feedback and Analytics:** Data on user interactions, feedback, and viewing habits are processed and analyzed across distributed systems for service improvement.

(e) Scalability and Redundancy:

- **Scalable Architecture:** Distributed systems provide scalability to handle varying loads and accommodate increasing numbers of users.
- **Redundancy:** Redundant servers and data storage ensure high availability and fault tolerance, minimizing disruptions in multimedia content delivery.

(f) Collaborative Filtering:

- **Distributed Databases:** User preferences and viewing histories are stored in distributed databases.
- **Collaborative Filtering Algorithms:** These algorithms process data across distributed nodes to recommend personalized content based on user behavior.

In this example, distributed multimedia computing optimizes the delivery of video content by leveraging distributed systems for efficient storage, processing, and real-time streaming. This approach enhances user experience, scalability, and responsiveness in a multimedia-rich environment like a video streaming service.

1.4 FOCUS ON RESOURCE SHARING

Resource sharing in a distributed system refers to the process of allowing multiple nodes or devices within the network to access and utilize shared resources. These resources can include data, files, computing power, storage, services, or devices such as printers or scanners. The primary goal is to efficiently use and manage resources across the distributed environment.

Here's an overview of how resource sharing operates in a distributed system:

1. **Identification of Resources**: The first step involves identifying the resources that can be shared across the network. This includes defining what resources are available, their locations, and access mechanisms.
2. **Resource Discovery**: Nodes in the distributed system need to be aware of the available resources. Discovery mechanisms, such as directories or registries, help nodes locate and access these shared resources.
3. **Access and Communication**: Once identified, shared resources are made accessible to authorized nodes through communication protocols and interfaces. Nodes request access to resources using specific protocols (like HTTP for web services, FTP for file sharing, etc.) and communication channels.
4. **Sharing Mechanisms**: There are various mechanisms to facilitate resource sharing:
   - **Remote Procedure Calls (RPC)**: Nodes can call procedures/functions residing on remote nodes to access resources.

- **Message Passing**: Nodes communicate by sending messages to access or manipulate shared resources.
- **Distributed File Systems**: Allow multiple nodes to access and store files across the network.
- **Databases and Data Replication**: Data can be shared through distributed databases or replicated across nodes to ensure availability and accessibility.

1.4.1 Advantages of Resource Sharing

Advantages of resource sharing in a distributed system include:

- **Efficiency:** Sharing resources allows better utilization of available resources across the system. Instead of each node maintaining its dedicated resources, they can access shared resources when needed, reducing redundancy and optimizing resource allocation.
- **Cost-effectiveness**: Shared resources can lead to cost savings as it eliminates the need for each node to individually procure and maintain its resources. This centralized approach to resource management can be more economical.
- **Scalability**: Distributed systems can easily scale by adding or removing nodes without major changes to the system's architecture. Resource sharing facilitates this scalability by enabling new nodes to access shared resources seamlessly.
- **Improved performance**: When resources are shared efficiently, it can lead to improved system performance. For instance, a distributed database system where multiple nodes share a common database can enhance data access speeds and reduce latency.
- **Collaboration and coordination**: Resource sharing fosters collaboration among nodes in a distributed system. It enables concurrent access to shared data or services, allowing multiple nodes to work together on complex tasks or computations.
- **Reliability and fault tolerance**: Shared resources can be replicated or distributed across multiple nodes, improving system reliability. If one node fails or encounters issues, other nodes can continue to access alternate instances of the resource, ensuring system stability and fault tolerance.

- **Flexibility and adaptability**: Resource sharing enables flexible resource allocation based on changing demands. Nodes can dynamically access and release shared resources based on their requirements, leading to a more adaptable and responsive system.
- **Centralized management**: By sharing resources, system administrators can manage and maintain resources centrally, implementing security protocols, access controls, and updates more efficiently across the distributed environment.

However, resource sharing also poses challenges such as ensuring data consistency, maintaining security and access control, handling concurrent access, and dealing with potential network latency or communication issues. Effective design and implementation strategies are crucial to maximize the advantages while addressing these challenges in a distributed system.

1.5 CHALLENGES OF DISTRIBUTED SYSTEM

While distributed systems offer numerous benefits, they also come with several challenges that need to be addressed for effective design, implementation, and management.

Here are some common challenges associated with distributed systems:

- **Communication Issues:** The network connecting distributed nodes may experience delays, packet loss, or even temporary disconnections. Handling these issues and ensuring reliable communication is a significant challenge.
- **Consistency and Replication:** Maintaining consistency across distributed replicas of data is challenging. Achieving consensus and managing updates in a way that preserves consistency can be complex, especially in the presence of network partitions.
- **Concurrency Control:** Coordinating concurrent access to shared resources requires careful synchronization to prevent conflicts and ensure data integrity. This becomes more challenging as the number of nodes increases.
- **Startup Cost:** Compared to a single system, the implementation cost of a distributed system is significantly higher. The infrastructure used in a distributed system makes it expensive.

- **Fault Tolerance:** Detecting failures promptly and recovering from them without compromising the system's availability and consistency is a non-trivial task. Designing effective fault-tolerant mechanisms is crucial.
- **Security:** Ensuring the security of data during communication and storage in a distributed environment is a significant challenge. Security measures such as encryption and authentication become more complex in a distributed context.
- **Scalability Limits:** While distributed systems can scale horizontally, managing scalability effectively, especially in rapidly changing workloads, requires careful planning. Poorly designed systems may experience bottlenecks or performance degradation.
- **Complexity and Debugging:** Identifying and fixing issues in a distributed system can be challenging due to the complexity of interactions between components. Debugging tools and techniques need to be sophisticated to handle distributed scenarios.
- **Data Partitioning:** Distributing data across multiple nodes requires effective partitioning strategies. Poorly chosen partitioning schemes can lead to imbalanced workloads and increased communication overhead.
- **Consensus Challenges:** Achieving consensus among distributed nodes is critical for maintaining data consistency. Implementing and managing consensus algorithms, such as Paxos or Raft, can be complex.
- **Communication Latency:** Distributed systems often involve communication between nodes, and managing latency becomes crucial. Minimizing communication delays and ensuring responsive interactions pose challenges.
- **Distributed Transactions:** Coordinating transactions across multiple nodes introduces challenges related to atomicity, consistency, isolation, and durability (ACID properties). Implementing distributed transactions requires careful consideration.
- **Node Joining and Leaving:** Handling the dynamic addition or removal of nodes from the distributed system can be challenging. Ensuring smooth transitions without disrupting system operations is crucial.
- **Data Distribution and Locality:** Optimizing data distribution and access patterns becomes complex, especially when dealing with geographically distributed systems. Ensuring data locality for improved performance requires thoughtful design.

Addressing these challenges requires a deep understanding of distributed systems principles, as well as the use of appropriate algorithms, protocols, and technologies. As the demand for scalable and resilient systems continues to grow, overcoming these challenges remains a priority in the field of distributed computing.

1.6 CASE STUDY: THE WORLD WIDE WEB

The *World Wide Web*, abbreviated as *WWW* and commonly known as the *Web*, is a system of interlinked hypertext documents accessed via the internet. World Wide Web is a set of programs, standards and protocols that allows the text, images, animations, sounds, and videos to be stored and accessed and linked together in the form of websites. Basically, WWW is a collection of millions of web pages stored in thousands of computers all over the world. It is a safe house for storing information on the internet. They are very helpful in searching all the information present on the internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them by using hyperlinks.

The components of WWW mainly fall into two categories:

1. Structural Components
2. Semantic Components

Web architecture consists of three *structural* components. They are:

(a) Internet - Internet is a global "network of networks".

(b) Web Server - Web Server refers to hardware (the computer) or the software (the computer application) that is used to deliver web pages to clients.

(c) Web Browser - Web Browser is an application that communicates with a web server. Web browser is actually a web client on your PC that makes the request to the web server.

The communication between web client and web server takes place using the Hypertext Transfer Protocol (HTTP).

Web architecture is a two-tier architecture that works on *client-server* approach. Web Server and Web Client are the two main elements in all internet applications. The Web Client sends requests and the Web Server listens to those requests and responds/ accomplishes the required task. The Web Server and Web Client communicate with each other through a common protocol.

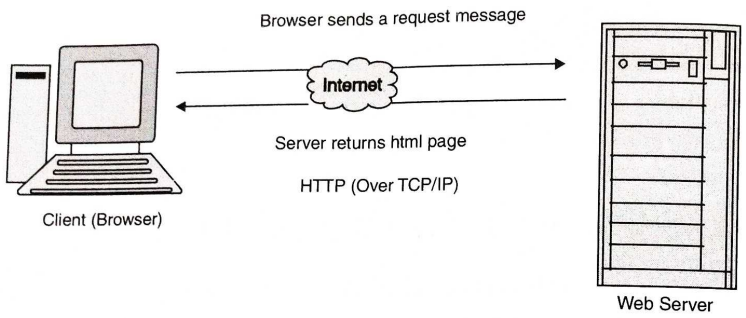


Fig. 1.2 WWW Architecture

The request-response communication between a Web Client and a Web Server illustrated in **Fig. 1.2** is as explained below:

1. A Web client (or browser) sends requests to a Web server. Every retrievable piece of information on the Web is identified by a URL, which includes the hostname, its location, port and the protocol used to get it.
2. For example, in the URL http://www.yahoo.com/docs/index.html, the communication protocol is HTTP; the hostname is www.yahoo.com. The port number takes default number, which is TCP port 80 for HTTP. The path and file name for the resource to be located is "/docs/index.html".
3. The Web server is responsible for document storage, manage and retrieval. The Web server interprets the request message, and returns the document requested (or an error message) back to the requesting client.
4. The Browser interprets and presents the document. The browser is responsible for document presentation.

The language that Web clients and servers use to communicate with each other is called the *Hypertext Transfer Protocol (HTTP).* All Web clients and servers must be able to speak HTTP in order to send and receive hypermedia documents.

Web architecture consists of following *semantic* components. They are:

(a) HTML: On a successful request, a data object is returned from the server to the client. The object is written in the Hypertext Markup Language (HTML) which is a hypertext language with the possibility of containing hyperlinks that the user can follow. It is mainly used to define the contents, structure, and organization of the

web page. HTML employs tags to arrange elements like text, images, links, and forms, enabling browsers to interpret and display information uniformly. HTML supports semantic markup, aiding accessibility and SEO, while integrating multimedia and interactive features. It collaborates with CSS and JavaScript, fostering responsive design and interactivity. As the fundamental language of the internet, HTML shapes the presentation and accessibility of content across the World Wide Web.

(b) HTTP: HTTP (Hypertext Transfer Protocol) is the backbone of data communication on the WWW. It governs how web browsers and servers exchange information. HTTP facilitates requests for web resources, allowing browsers to fetch web pages and assets from servers. It ensures reliable transmission, defining the rules for data formatting, transmission, and response. HTTP's stateless nature enables seamless navigation between web pages and resources. As a foundational protocol, HTTP enables the smooth flow of content across the World Wide Web, ensuring seamless data retrieval and display.

HTTP is also called *request and response* protocol because in the client-server computing model, the communication between the web browser (client) and web server (server) takes place in request and response pairs.

A web browser (client) sends a request for a web page to the web server (server) via HTTP.

The web server searches for the requested web page. If the requested page is found then web server will send it to client with an HTTP response. If the requested web page is not found, then web server will send an *HTTP response: Error 404 Not found.*

The response contains completion status information about the request and may also contain requested content in its message body.

The web browser interprets and presents the document. The web browser is responsible for document presentation.

(c) URL: URL stands for "Uniform Resource Locator". A URL is the location of a specific website or file on the Internet. It is a specific character string that constitutes a reference to a resource on the Internet.

The URL consists of four parts:

- ***Protocol*:** The application-level protocol used by the client and server, e.g.,

HTTP, FTP, and TELNET. It defines a network protocol to be used to access a resource. These strings are short names followed by the three characters '://' (a simple naming convention to denote a protocol definition). Typical URL protocols include- *http://*, *ftp://*, and *mailto://*.

- ***Hostname***: It identifies a computer or other network device such as DNS domain name (e.g., www.yahoo.com) or IP address of the server.
- ***Port***: The TCP port number that the server is listening for incoming requests from the clients.
- ***Path-and-file-name***: The name and location of the requested resource, under the server document base directory.

For example, in the URL http://www.yahoo.com/docs/index.html, the communication protocol is http; the hostname is www.yahoo.com. The port number takes default number, which is TCP port 80 for HTTP. The path and file name for the resource to be located is "/docs/index.html".

QUESTIONS

Short Answer Questions

Q1. Define distributed system.

Ans: A distributed system is a network of autonomous computers that communicate and coordinate their actions to collectively perform tasks. It shares resources across interconnected nodes, enabling parallel processing and decentralized control. This architecture enhances scalability, fault tolerance, and efficiency by distributing workload and services across multiple devices connected via a network.

Q2. What are the applications of distributed system?

Or

What are distributed systems used for?

Ans: The various applications of distributed system are:

- *Cloud computing:* Facilitating resource sharing and on-demand services.
- *Financial systems:* Handling transaction processing and data analysis.

- *Internet of Things (IoT):* Connecting and managing interconnected devices.
- *Content Delivery Networks (CDNs):* Efficiently distributing content.
- *Peer-to-Peer (P2P) networks:* Enabling decentralized file sharing.
- *Distributed databases:* Storing and managing data across multiple nodes.
- *Scientific research (e.g., grid computing):* Collaborative computing for complex tasks.
- *Telecommunications:* Providing reliable and scalable communication infrastructures.

Q3. What are the disadvantages of distributed systems?

Ans: The various disadvantages of distributed system are:

- *Complexity:* Increased intricacy in design, implementation, and maintenance.
- *Communication issues:* Latency, bandwidth constraints affecting performance.
- *Data consistency challenges:* Ensuring synchronized data across nodes.
- *Security risks:* Vulnerability to unauthorized access and breaches.
- *Compatibility issues:* Coordinating diverse hardware and software components.
- *Higher development costs:* Due to increased complexity and maintenance needs.
- *Troubleshooting complexities:* Debugging and problem resolution can be intricate.

Q4. How does a distributed system work?

Ans: A distributed system functions by connecting multiple independent computers through a network, enabling them to communicate and collaborate on tasks. Tasks are divided into smaller parts and allocated to different nodes, which work concurrently. Nodes exchange data through messages, coordinating their actions to achieve a common goal. This architecture allows for resource sharing, fault tolerance, and scalability by distributing workload across interconnected devices.

Q5. What are the types of distributed systems?

Ans: The various types of distributed system:

- *Client-Server Model:* Involves clients requesting services from centralized servers.
- *Peer-to-Peer (P2P):* Nodes act both as clients and servers, sharing resources directly.

- *Cluster Computing:* Groups of interconnected computers working collaboratively as a single system.
- *Grid Computing:* Utilizes resources from multiple administrative domains for complex computations.
- *Cloud Computing:* Provides on-demand resources and services via the internet.
- *Mobile Agent Systems:* Agents move between nodes, executing tasks in a distributed environment.

Q6. Is distributed systems a cloud computing service?

Ans: Distributed systems are not exclusively a cloud computing service, but they form the underlying architecture enabling cloud computing. Cloud computing utilizes distributed systems' principles, like resource sharing, scalability, and fault tolerance, to offer on-demand services over the internet. Distributed systems form the backbone of cloud infrastructure, facilitating flexible and efficient resource allocation across a network of interconnected devices.

Q7. What is Mobile Computing?

Ans: Mobile computing refers to the use of portable computing devices, such as smartphones, tablets, and laptops, to access and process information while on the move. It involves wireless communication, enabling users to interact with data, applications, and services regardless of their physical location. Mobile computing has become integral to modern lifestyles, providing flexibility, connectivity, and a wide range of applications and services.

Q8. What is Pervasive Networking?

Ans: Pervasive networking is a concept that involves creating a seamlessly interconnected environment where communication and connectivity are pervasive across a wide array of devices, enabling them to work together seamlessly. This concept extends beyond traditional computer networks to include everyday objects and devices that are embedded with communication capabilities, creating a networked ecosystem that enhances convenience and functionality.

Q9. What is Edge Computing?

Ans: Edge computing is a distributed computing paradigm that brings computational capabilities closer to the data source or "edge" of the network, reducing latency and

improving the efficiency of data processing. This approach involves processing data locally on or near the edge devices, rather than relying solely on centralized cloud servers. Edge computing is particularly beneficial for applications that require real-time processing and low-latency responses.

Q10. What is Cloud Computing?

Ans: Cloud computing is a paradigm of distributed computing where computing resources, such as processing power, storage, and applications, are delivered as services over the internet. It involves the use of a network of remote servers hosted on the internet to store, manage, and process data rather than a local server or a personal computer.

Q11. What is resource sharing in distributed system?

Ans: Resource sharing in a distributed system refers to the process of allowing multiple nodes or devices within the network to access and utilize shared resources. These resources can include data, files, computing power, storage, services, or devices such as printers or scanners. The primary goal is to efficiently use and manage resources across the distributed environment.

Long Answer Questions

Q1. What is distributed system? Discuss the main characteristics of distributed systems?

Ans: Refer Section **1.1**

Q2. Why we use distributed system? What are the advantages of the distributed system?

Ans: Refer Section **1.1**

Q3. Discuss some examples of a distributed system.

Ans: Refer Section **1.2**

Q4. Explain the latest trends in distribute systems.

Ans: Refer Section **1.3**

Q5. What is resource sharing? Discuss its benefits and challenges.

Ans: Refer Section **1.4**

Q6. Describe the various challenges associated with distributed systems in detail.

Ans: Refer Section **1.5**

EXERCISE

1. Draw and explain the architecture of WWW.
2. List the components of distributed system.
3. Write a short note on:
   (a) Mobile Computing
   (b) Cloud Computing
   (c) Edge Computing

○○○