I'm not a bot

Examples of distributed computing

Distributed Systems have gained widespread recognition due to their capacity for Growth, Dependability, Speed, Versatility, Clarity, Resource-sharing, Global distribution, and more. However, harnessing these benefits requires an environment that supports the execution and development of Distributed Applications. A distributed application is a program running on multiple machines, communicating over a network, comprising separate components executing on different nodes to achieve a unified goal. It utilizes the Client-Server Model. The Distributed Computing Environment (DCE) is a comprehensive set of services and tools for building and operating Distributed Applications. This integrated software platform can be installed as an additional layer on existing Operating Systems, serving as a foundation for developing and running distributed applications. Users can access remote servers' applications and data without needing to know the location of their programs or the desired data. DCE was developed by the Open Software Foundation (OSF) using contributions from member companies now known as The Open Group. Its services include: 1. Remote Procedure Call (RPC): A call made when a program desires to execute a subroutine on another computer. 2. Distributed File System (DFS): Provides transparent access to files, treating them as local. 3. Directory Service: Tracks virtual resources' locations within the distributed system, including files, printers, servers, and other machines. 4. Security Service: Authenticates users and grants access only to authorized individuals and computers on a network of Distributed Systems. 5. Distributed Time Service: Maintains a global clock for synchronizing local clocks and ensuring inter-process communication between system components in a designated order. 6. Thread Service: Implements lightweight processes (threads), facilitating the synchronization of multiple threads within shared memory spaces. The DCE Architecture supports structuring distributed computing systems into cells, comprising three machine types: User, Administrator, and Server. This approach keeps administration manageable by grouping nodes within cells. Managed distributed computing systems operate under a unified authority, with cell boundaries serving as security firewalls. Accessing resources across cells requires special authentication and authorization procedures, differing from intra-cell interactions. The DCE cell administrator role holds the highest privileges, controlling system services remotely and managing all resources within the Distributed Computing Environment. Key components include the Security Server, Cell Directory Server (CDS), and Distributed Time Server, providing a synchronized clock for the entire network. The advantages of this architecture are enhanced security, reduced maintenance costs, scalability, availability, and reduced risks. In distributed systems, multiple computers collaborate to handle specific tasks simultaneously, allowing businesses to scale up as needed without compromising performance or increasing costs. The key to achieving high performance lies in minimizing latency and enhancing response time and throughput. This is achieved by utilizing low-cost commodity hardware, ensuring zero data loss during initial deployments and expansions. At the core of distributed systems is cloud-based software, a complex network of servers accessible over the internet. Components and connectors arrange themselves to facilitate seamless communication, with modules featuring well-defined interfaces that can be replaced or reused. Connectors mediate coordination among components, enabling cooperation. Distributed systems are structured around two primary concepts: software architecture and system architecture. Software architecture focuses on the logical organization of software components and their interactions. The four main architectural styles in software components include layered architecture, object-based architecture, data-centered architecture, and event-based architecture. System architecture, on the other hand, is divided into client-server architecture and peer-to-peer architecture. Layered architecture separates software components into units, allowing for efficient modification of individual layers without affecting the system as a whole. Object-based architecture focuses on loosely coupled objects interacting through an unspecified architecture. Object-based architecture utilizes direct method calls between components through a connector, whereas data-centered architecture relies on a central data repository, and event-based architecture communicates through events. This centralized architecture provides a more stable and secure environment compared to peer-to-peer networks, although it may sacrifice some speed. A key advantage of this system lies in its ability to manage resource usage effectively due to the presence of a security database. On the other hand, peer-to-peer networks operate under a decentralized structure where each node can act as either a client or server, offering flexibility but also introducing single points of failure and scalability issues. In contrast, P2P architecture allows nodes to self-organize and establish connections with one another. For instance, new nodes may choose to register with a centralized lookup server for service provision or broadcast their requests to other nodes within the network. Current P2P networks can be categorized into three types: structured P2P, unstructured P2P, and hybrid P2P, each featuring distinct approaches to neighbor selection and data distribution. Distributed systems consist of primary system controller, system data store, and database as their core components. In non-clustered environments, additional elements such as user interfaces and secondary controllers are also present. A distributed system typically features: A distributed system can be set up on various devices, but every computer must have access to a common datastore. In a distributed system, data is stored in a relational database and shared among multiple users once it's located. This setup can be found in all types of systems and allows for simultaneous use by several users. **Distributed Systems Examples** When processing power is scarce or when systems face unpredictable changes, distributed systems are useful as they help distribute the workload evenly. Distributed systems have numerous applications, from electronic banking to multiplayer online games. Some notable examples of distributed systems include: 1. **Networks** The invention of Ethernet and LAN (local area networks) in the 1970s enabled computers to connect within close proximity. This led to the development of peer-to-peer networks, which are still widely used today in e-mail and internet applications. 2. **Telecommunication Networks** Telephone and cellular networks are examples of peer-to-peer networks that facilitate communication between devices. The integration of Voice over Internet (VoIP) technology has made these systems more complex but effective. 3. **Real-Time Systems** Real-time systems are not limited to specific industries and can be found in sectors like airlines, ride-sharing services, logistics, financial trading, MMOGs, and ecommerce. These systems focus on rapid data processing and transmission to a large number of users who need this information promptly. 4. **Parallel Processors** Parallel computing involves dividing tasks among multiple processors, which creates complex computational processes by combining the outputs of each processor. This method was initially used for running software on multiple threads or processors accessing shared data and memory but has since expanded to include operating systems in parallel processing. 5. **Distributed Database Systems** A distributed database is a system where data is stored across several servers or locations, allowing it to be replicated across various platforms. Distributed databases can be either homogeneous (using the same database management system and data model) or heterogeneous (supporting multiple data models and database management systems). The addition of new nodes and locations makes it easier to manage and scale performance. 6. **Distributed Artificial Intelligence** Distributed artificial intelligence is an approach where AI processes are split across multiple devices, allowing for more efficient processing and decision-making in complex tasks. This method has numerous applications in areas like data analysis, predictive maintenance, and autonomous systems. The concept of distributed systems refers to the utilization of complex learning algorithms and large-scale systems for decision-making purposes, relying on vast computational data points located across various sites. This approach encompasses real-world applications such as video rendering, scientific computing, online booking platforms, cryptocurrency processing, peer-to-peer file sharing, multiplayer gaming, and e-learning software. The primary advantage of distributed systems lies in their capacity to provide scalable and enhanced performance, which is crucial for modern computing infrastructure. They are integral components of wireless networks, cloud computing, and the internet, allowing them to tap into the resources of other devices and processes. As a result, they offer functionalities that would be challenging or impossible to achieve on a single system. Their reliability stems from the collective power of multiple machines, making distributed systems an essential aspect of modern computing. If you are planning to implement such systems within your organization, feel free to share your thoughts on LinkedIn, Twitter, or Facebook.