International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 05 | May - 2021 ISSN: 2582-3930

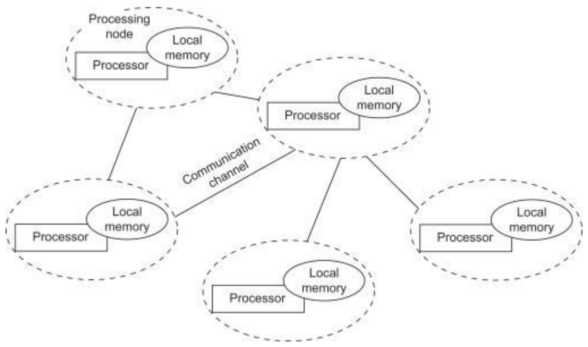
DISTRIBUTED COMPUTING: AN OVERVIEW

Abhishek Jena
Dayananda Sagar College of Engineering

Abstract:

The decline in hardware costs and advances in communication technology have led to an increase in interest in the use of large compatible and distributed computer systems. Distributed computer systems provide the ability to improve performance and resource sharing. In this paper we have made a complete view of the distributed computer. In this paper we have studied the differences between computer-like distribution, computer-distributed terms, distribution of computer-based operations and distributed computer systems, uniformly distributed algorithm models and the benefits of distributed computer and distributed computer scope.
Keywords – Distributed computing, execution time, shared memory, throughput.

Introduction:



Distributed computer refers to two or more computers connected together that share the same computer function. The purpose of computer distribution is to share work between multiple computers. A distributed network is very different in nature in the sense that processing nodes, network topology, network connectivity, and operating system can differ from a different network that is widely distributed around the world. In order for the system to work properly a lot of workload must be distributed between the areas above the network.

Therefore the issue of load balancing became very popular due to the existence of distributed memory systems. The network will have high-speed computer nodes and slow computing nodes. If we do not take into account the processing speed and connection speed (bandwidth), the performance of the entire system will be limited by the slow drive of the network. Thus load balancing techniques measure loads across areas by preventing nodes from working and other nodes can be overcome. In addition, load balancing techniques eliminate the malfunction of any node during operation.

History:

The use of complementary communication systems has the roots of operating system structures studied in the 1960s. The first widely distributed systems were local networks such as Ethernet, founded in the 1970's. The ARPANET, one of the forerunners of the Internet, was introduced in the late 1960's, and the ARPANET email was established in the early 1970's. Email became the most successful use of ARPANET, and is probably the first example of a widely distributed app. In addition to the ARPANET (and its successor, the global Internet), other world-class computer networks including Usenet and FidoNet since the 1980s, both of which have been used to support distributed chat systems.

The study of distributed computing became its branch of computer science in the late 1970s and early 1980s. The first conference in the field, the Symposium on Principles of Distributed Computing (PODC), dates back to 1982, and it’s International Symposium on Distributed Computing (DISC) partner was first held in Ottawa in 1985 as an International Workshop on Distributed Algorithms on Graphs.

The Client/Server model:

The most common way to organize software that will work on distributed applications is to divide tasks into two parts: clients and servers. A client is a program that uses the services that other programs provide. Programs that provide services are called servers. The client requests a service, and the server performs that service. Server operations often require the management of specific resources, where the server synchronizes and controls access to the application, and responds to customer requests with data or status information. Client programs often manage user interactions and often request data or initiate certain data modifications on behalf of the user.

The client is separated from the need to know anything about the resource manager himself. If you change the database you are using, the server may need to be modified, but the client does not need to be modified. Because there are usually fewer copies of the server than the client, and because the servers tend to be in easy-to-update locations (for example, on central machines instead of PCs running on user desks), the upgrade process is also simplified. Also, this method provides additional security. Only servers, not clients, require access to data managed by the resource manager.

Clients can also access various servers, and the servers themselves can act as clients on other servers. Exactly how tasks are distributed on servers is a decision of the application. For example, a single server can provide all the services a client needs, or a client can access multiple servers to make various requests. The app designer must consider factors such as scalability, location, and security. For example, are clients and servers located locally or is the system still widely distributed? Do the servers need to be physically protected? Such design decisions are outside the scope of the introduction.

Some servers are part of an application and are called application servers. Some servers are not part of a specific application. Instead, any app can use them. For example, CICS® Structured File Server (SFS) provides access to record-directed file access to applications.

Architectural patterns for distributed systems:

Common architectural patterns for organizing the architecture of a distributed system:

1. Master-slave architecture:
   Master-slave properties are often used in real-time systems where contact response times are required. There may be different processors related to data acquisition from system environment, data processing and actuator calculations. The 'master' process is usually responsible for integration, coordination and communication and governs the 'slave' processes. 'Slave' processes are dedicated to specific actions, such as data retrieval from multiple lists.
2. Two-tier client-server architecture:
   With the creation of two client clients, the system is used as one logical server and an infinite number of clients using that server.
   A thin client model, in which the presentation layer is performed on the client and in all other layers (data management, application processing and data), is performed on the server. There are very few requests for small clients where all processing is done on a remote server.
   Fat client model, where some or all of the app processing is done to the customer. Data management and data operations are performed on the server.
   However, the distinction between the properties of small and oblivious customers is blurred. JavaScript allows local processing in the browser so that other parts of the 'fat customer' function can be made available without software installation. Mobile applications enable local processing to reduce network needs. Auto-update of apps reduces management problems.
3. Multi-tier client-server architecture:
   The multi-tier server client build, the various layers of the system, namely presentation, data management, application processing, and database, are different processes that can work on different processors. This avoids problems with disability and performance when choosing a two-client model, or system management problems when using an oily client model.
   Usage case: where there is a high volume transaction to be processed by the server.
4. Distributed component architecture:

There is no difference in the structure of distributed elements between customers and servers. Each distributed business is a segment that provides services in some areas and receives services in other areas. Object communication is done through a middleware program. Used when resources from various systems and domains need to be integrated, or as an implementation model for multiple client-service programs Benefits include:

- Allows the app designer to delay decisions about where and how services should be provided.
- It is a very open way to build a system that allows new resources to be added as needed.
- The system is flexible and scalable.
- You may need to reconfigure the system with items that cross the network as needed.

5. Peer-to-peer architecture:

Peer to peer (p2p) systems are distributed systems where calculations can be performed by any network. The complete system is designed to take advantage of computer power and the maintenance of a large number of connected computers.

Most p2p programs have been personal programs but there is a growing business use of this technology. Used when clients exchange information stored locally and the role of the server is to introduce clients to each other. Examples:

- File-sharing programs based on the BitTorrent protocol
- Messaging programs such as Jabber
- Payment systems, e.g. Bitcoin
- Information details, e.g. Freenet is a separate database
- Phone applications, e.g. Viber
- Calculation systems.

P2P structures are used there:

- The system is very computer-friendly and it is possible to split the required processing into a large number of independent computers.
- The system primarily involves the exchange of information between individual computers in the network and there is no need for this information to be stored or handled centrally.

Security issues:

- Security concerns are the main reason why p2p architecture is not widely used.
- Lack of centralized management means that malicious nodes can be set up to deliver spam and malware to other nodes in the network.
- P2P connections need to be carefully set to protect location information and if not done properly, this is exposed to other peers.

Three-tiered client/server architecture:

The standard design of customer / server systems uses three categories:

- User-connected client
- An application server that contains the business concept of an application
- Data resource manager

Advantages of Distributed Computing:

- Reliability, high tolerance error: System crashes on one server do not affect other servers.
- Failures: In computer distribution systems you can add as many machines as needed.
- Flexibility: It makes it easy to install, use and debug new resources.
- Fast Speed Calculator: A distributed computer system can have the computing power of many computers, making it faster than other programs.
- Openness: Being an open system, it can be accessed locally and remotely.

- High Performance: Compared to mid-range computer network collections, they can offer higher performance and better cost efficiency.

Disadvantages of Distributed Computing:

- Troubleshooting: Troubleshooting and diagnostics are extremely difficult due to distribution across multiple servers.
- Limited software support: Low software support is a major problem for distributed computer systems.
- High infrastructure costs: Basic network setup problems, including transfers, overload, and data loss.
- Security issues: The features of open source systems enable data security and risk sharing on distributed computer systems.

Conclusion:

A distributed computer helps improve the performance of large projects by combining the power of multiple machines. It's awesome and allows users to add computers depending on the needs of the growing load. Although distributed computing has its drawbacks, it offers unparalleled consistency, better performance and greater reliability, making it a better solution for businesses facing high workloads and big data.

References:

1. Y. Jadeja and K. Modi, "Cloud computing - concepts, architecture and challenges," 2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET), 2012, pp. 877-880, doi: 10.1109/ICCEET.2012.6203873.
2. Svobodova L. (1985) Client/Server Model of Distributed Processing. In: Heger D., Krüger G., Spaniol O., Zorn W. (eds) Kommunikation in VerteiltenSystemen I. Informatik-Fachberichte, vol 95. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-70285-3_29
3. https://en.wikipedia.org/wiki/Distributed_computing#History
4. Thoke, V. D., &Sangli, V. (2015). THEORY OF DISTRIBUTED COMPUTING AND PARALLEL PROCESSING WITH ITS APPLICATIONS, ADVANTAGES AND DISADVANTAGES.
5. Taylor, I., Shields, M., & Wang, I. (2003, April). Distributed p2p computing within triana: A galaxy visualization test case. In Proceedings International Parallel and Distributed Processing Symposium (pp. 8-pp). IEEE.