VOLUME - 12, ISSUE - 05, MAY - 2023 • PRINT ISSN No. 2277 - 8160 • DOI : 10.36106/gjra

Original Research Paper | Computer Science

DISTRIBUTED OPERATING SYSTEM: A VIRTUAL CONCEPT

Uma Sharma Asstt. Professor of Computer Science Govt. College, Hansi (Hisar) Haryana

ABSTRACT This paper elaborate the virtual concept of distributed operating system, it runs on several machines whose purpose is to provide a useful set of services, generally to make the collection of machines behave more like a single machine. A distributed operating system is one that looks to its users like an ordinary and modern centralized operating system but runs on multiple, independent central processing unit over the networks or a local host system. The key concept here is transparency. The uses of multiple processors should be transparent for the entire user. Another way of expressing the same idea is to say that the user views the system as a "virtual uniprocessor," not as a collection of distinct machines. This is easier said than done.

KEYWORDS : Operating System, Processor, DOS, Replication, Virtual.

INTRODUCTON

The Concept of distributed operating system is consist of large number of independent, networked and separate computers but this whole system is appeared as single system .In distributed operating system with multiple CPUs are connected to each other. To connect these nodes and how they communicate each other various classification scheme are defined. Such classification is called Flynn's classification.

Distributed operating systems have many points in common with centralized ones, but they also have many distinctive features of their own. They often differ in the areas of how processes communicate with each other, how resources are named, how they are managed, how fault tolerance is achieved, and how system services are provided to user processes. A distributed operating system is an operating system that runs on a number of technologies whose function is to make available a useful set of services, generally to make the set of machines act more like a only machine.

A distributed operating system provides the essential services and functionality required of an operating system, adding attributes and particular configurations to allow it to support additional requirements such as increased scale and availability. To a user, a distributed operating system works in a manner similar to a single-node, monolithic operating system. That is, although it consists of multiple computers.

2. FEATURES OF DISTRIBUTED SYSTEMS

- **Economics:** Microprocessor offers a better price and performance than main frames
- **Speed:** A distributed system may have more total computing power than main frame.
- **Inherit distribution:** Some application involves separated machines.
- **Reliability:** If one server machine crashes the system can still survive.
- **Incremental Growth:** Computing power can we added in small increments.

3. TYPES OF DISTRIBUTED OPERATING SYSTEM

3.1. Network Operating Systems:

- Loosely-coupled software on loosely-coupled hardware
- E.g., LAN with file server
- Users are aware that they are using independent hardware, but share a consistent view of the filing system with other network users

3.2. True Distributed Systems:

- Tightly-coupled software on Loosely-coupled hardware
- Give users impression that collection of computers is a single timesharing system - the virtual uni-processor
- Processes are capable of being executed on any computer on the network, and users won't realize

3.3. Multiprocessor Timesharing Systems:

- Tightly-coupled software on tightly-coupled hardware
- Single RUN Queue
- Scheduler must run as a critical section to prevent two CPUs from selecting the same process to run

4. FILE SYSTEM AND FAULT TOLERANCE TECHNIQUE

When connecting two or more distinct systems together, the first issue that must be faced is how to merge the file systems. Three approaches have been tried. The first approach is not to merge them at all. Going this route means that a program on machine A cannot access files on machine B by making system calls. Instead, the user must run a special file transfer program that copies the needed remote files to the local machine, where they can then be accessed normally. These results suggest that reaching a distributed decision is not always possible in common circumstances. Even when it is possible, doing so in unfavorable conditions is very expensive and tricky. Although most distributed systems can be designed to operate in more favorable circumstances than these gloomy theoretical results describe, experience has shown that even pragmatic algorithm design for this environment is difficult.

In the past few years, two approaches to making distributed systems fault tolerant have emerged.

They differ radically in orientation, goals, and attitude toward the theologically sensitive issue of the perfectibility of mankind. One approach is based on redundancy and one is based on the notion of an atomic transaction. Both are described briefly below.

5. ISSUES IN DISTRIBUTED OPERATING SYSTEM

One core problem for distributed operating system designers is concurrency and synchronization. These issues arise in single machine operating systems, but they are easier to solve there. Typical single-machine systems run a single thread of control simultaneously, simplifying many synchronization problems.

The advent of multi-core machines is complicating this issue, but most multi-core machines have relatively few cores, lessoning the problem. Further, they typically have shared access to memory, registers, or other useful physical resources that are directly accessible by all processes that they must synchronize. These shared resources allow use of simple and fast synchronization primitives, such as semaphores. Even modern machines that have multiple processors typically include hardware that makes it easier to synchronize their operations.

Distributed operating systems lack these advantages. Typically, they must control a collection of processors connected by a network, most often a local area network, but occasionally a network with even more difficult characteristics. The access time across this network is orders of magnitude larger than the access time required to reach local main memory and even more orders of magnitude larger than that required to reach information in a local processor cache or register. Further, such networks are not as reliable as a typical bus, so messages are more likely to be lost or corrupted. At best, this unreliability increases the average access time. This imbalance means that running blocking primitives across the network is often infeasible. The performance implications for the individual component systems and the system as a whole do not permit wide spread use of such primitives. Designers must choose between looser synchronization and sluggish performance. The increasing gap between processor and network speeds suggests that this effect will only get worse.

Theoretical results in distributed systems are discouraging. Research on various forms of the Byzantine General problem and other formulations of the problems of reaching decisions in distributed systems has provided surprising results with bad implications for the possibility of providing perfect synchronization of such systems.

6. TRANSPARENCY AND OTHER THINGS

The distributed systems have to be designed carefully, since there are many pitfalls for the unwary. A key issue is transparency i.e. hiding all the distribution from the users and from the application programs. Another issue is flexibility. Other important issues are reliability, performance, scalability, security and failure handling.

6.1. Transparency

The implementation of the distributed system is very complex, as a number of issues have to be considered to achieve its final objective. The complexities should not worry the user of the distributed system from using it i.e., the complexities should be hidden from the user who uses the distributed system. This property of the distributed system is called its transparency. There are different kinds of transparencies that the distributed system has to incorporate. The following are the different transparencies encountered in the distributed systems:

- Access Transparency
- Location Transparency
- Concurrency Transparency
- Replication Transparency
- Failure Transparency
- Migration Transparency
- Performance Transparency
- Scaling Transparency

6.2. Flexibility

Flexibility in a distributed operating system is enhanced through the modular and characteristics of the distributed OS, and by providing a richer set of higher-level services. As the system is very flexible, it is very easy to install, implement and debug new services. Each service is equally accessible to every client remote or local.

6.3. Reliability

Distributed OS can provide the necessary resources and services to achieve high levels of reliability, or the ability to prevent and/or recover from errors. Faults are physical or logical defects that can cause errors in the system. For a system to be reliable, it must somehow overcome the adverse effects of faults. The primary methods for dealing with faults include fault avoidance, fault tolerance, and fault detection and recovery. Fault avoidance covers proactive measures taken to minimize the occurrence of faults.

6.4. Security

Security for information resources in distributed system has three components:-

6.4.1. Confidentiality: protection against disclosure to unauthorized individuals.

6.4.2. Integrity: protection against alteration/corruption

6.4.3. Availability: protection against interference with the means to access the resources.

6.5. Scalability

The concept of scalability refers to the ability of a system to continuously evolve in order to support a growing amount of tasks. A system is described as scalable if it remains effective when there is a significant increase in the number of resources and the number of users.

6.6. Failure Handling

Failures in a distributed system are partial i.e. some components fail while others can function. That is the reason why handling the failures are difficult. Failure handling includes:

6.6.1. Detecting failures: to manage the presence of failures cannot be detected but may be suspected.

6.6.2. Masking failures: hiding failure not guaranteed in the worst case.

6.6.3. Concurrency: Where applications/services process concurrency, it will affect a conflict in operations with one another and produce inconsistence results.

7. ADVANTAGES OF DISTRIBUTED OPERATING SYSTEM

Various advantages of the distributed operating system are as follows:

- Better price and performance as long as everyday hardware is used for the component computers – Better use of existing hardware
- By using the combined processing and storage capacity of many nodes, performance levels can be reached that are out of the scope of centralized machines
- Resources such as processing and storage capacity can be increased incrementally
- The important advantage of distributed computing system is reliability. It is more reliable than a single system. If one machine from system crashes, the rest of the computers remain unaffected and the system can survive as a whole.

8. APPLICATIONS OF DISTRIBUTED OPERATING SYSTEM

The various applications are as follows:

8.1. Telecommunication networks:

- Phone networks and cellular networks.
- PC networks like the web.
- Wireless sensor networks.
- Routing algorithms.

8.2. Network applications:

- Web and p-2-p networks.
- Stupendously multi player web-based games and virtual communities.
- Distributed database management systems and distributed databases.
- Network files systems.

- Distributed info processing systems like banking systems and airline reservation systems.

REFERENCES

1. Nutt, Gary J. (1992). Centralized and Distributed Operating Systems. Prentice Hall. ISBN 978-0-13-122326-4.
2. Go ci ski, Andrzej (1991). Distributed Operating Systems: The Logical Design. Addison-Wesley Pub. Co. ISBN 978-0-201-41704-3.
3. Hansen, Per Brinch, ed. (2001). Classic Operating Systems: From Batch Processing to Distributed Systems. Springer. ISBN 978-0-387-95113-3Birrell, A.D., and Needham, R.M. "A Universal File Server," IEEE Trans. Software Eng., vol. SE-6, pp. 450-453, Sept. 1980.
4. Birrell, A.D. and Nelson, B.J. "Implementing Remote Procedure Calls," ACM Trans. Comp. Syst., vol. 2, pp. 39-59, Feb. 1984.
5. Dalal, Y.K. "Broadcast Protocols in Packet Switched Computer Networks," Ph. D. Thesis, Stanford Univ., 1977.
6. http://www2.cs.siu.edu/~cs401/Textbook/ch6.pdf
7. http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1035&context=compsciwp
8. http://www.cs.helsinki.fi/u/jakangas/Teaching/DistSys/DistSys-08f-1.pdf
9. https:/www.cs.columbia.edu/~smb/classes/s06-4118/l26.pdf
10. Dos concepts and design By Pradeep K. Sinha
11. Dos concepts and design By Andrew S. Tanenbaum