RESEARCH PAPER ON DISTRIBUTED OPERATING SYSTEMS

Mohit Rathi, Manish Lohia

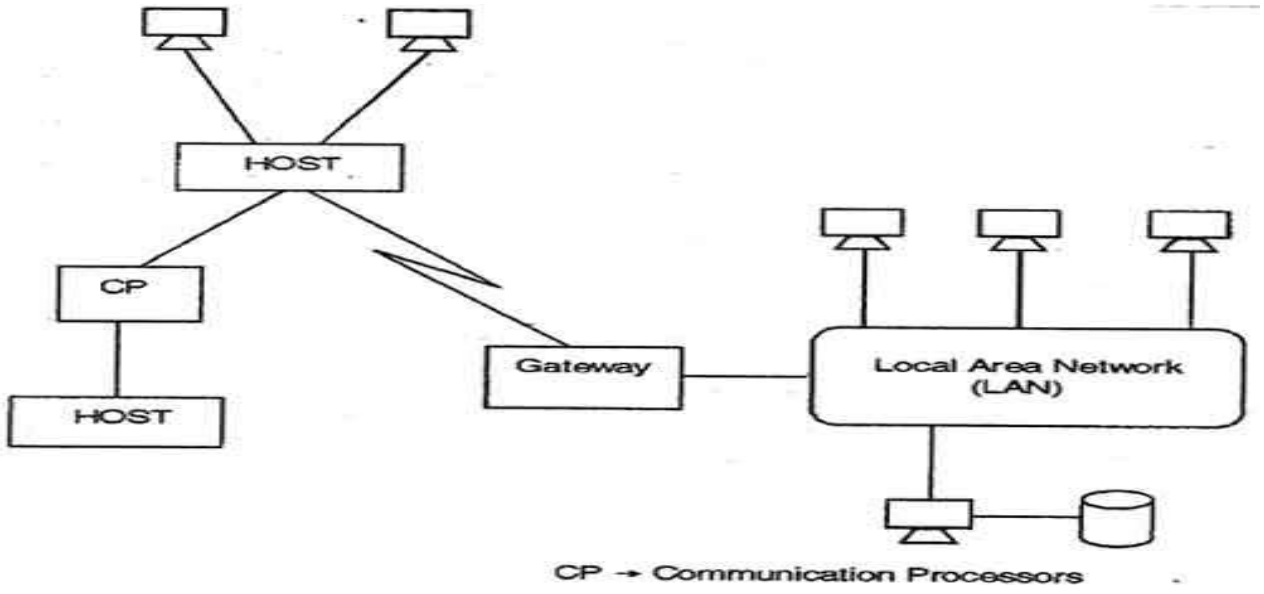
***Abstract*- The paper aims at providing a complete survey of distributed operating systems. This paper also explain in detail various key design issues involved in the building of such system. A distributed operating system is a software over a collection of independent, networked, communicating, and physically separate computational nodes. Each individual node holds a specific software subset of the global aggregate operating system. Each subset is a composite of two distinct service provisioners. The first is a minimal kernel, or microkernel, that directly controls that node's hardware. Second is a higher-level collection of system management components that coordinate the node's individual and collaborative activities. These components abstract microkernel functions and support user applications. The microkernel and the management components collection work together.**

I. INTRODUCTION

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

Separating minimal system-level functionality from additional user-level modular services provides a "separation of mechanism and policy."



A Typical Overview of Distributed Operating System

1.1 HISTORY

Research and experimentation efforts began in earnest in the 1970s and continued through 1990s, with focused interest peaking in the late 1980s. A number of distributed operating systems were introduced during this period; however, very few of these implementations achieved even modest commercial success.

Fundamental and pioneering implementations of primitive distributed operating system component concepts date to the early 1950s.Some of these individual steps were not focused directly on distributed computing, and at the time, many may not have realized their important impact. These pioneering efforts laid important groundwork, and inspired continued research in areas related to distributed computing.

In the mid-1970s, research produced important advances in distributed computing. These breakthroughs provided a solid, stable foundation for efforts that continued through the 1990s.

1.2 EXAMPLES OF DISTRIBUTED OPERATING SYSTEMS

l. IRIX operating system; is the implementation of UNIX System V, Release 3 for Silicon Graphics multiprocessor workstations.

2. AIX operating system for IBM RS/6000 computers.

3. Solaris operating system for SUN multiprocessor workstations.

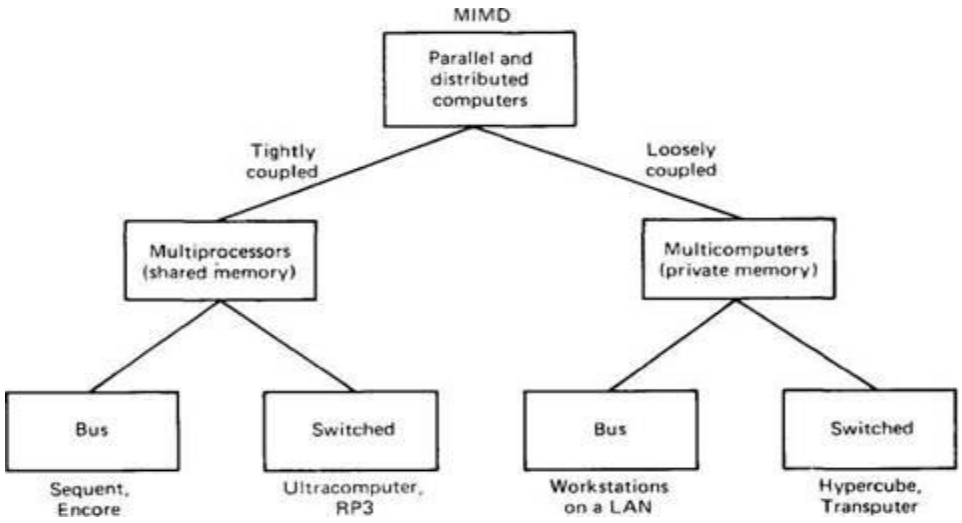
1.3. GOALS

1. There are many different types of distributed computing systems and many challenges to overcome in successfully designing one. The main goal of a distributed computing system is to connect users and resources in a transparent, open, and scalable way.

2. One design goal in building a distributed system is to create a *single system image*; to have a collection of independent computers appear as a single system to the user.

II. ASPECTS OF DISTRIBUTED OPERATING SYSTEM

2.1) Hardware Concepts



Although all Distributed Systems consist of multiple CPUs, there are different ways of interconnecting them and how they communicate

Flynn (1972) identified two essential characteristics to classify multiple CPU computer systems: the number of instruction streams and the number of data streams

1. Uniprocessors SISD

2. Array processors are SIMD - processors cooperate on a single problem

3. MISD - No known computer fits this model

4. Distributed Systems are MIMD

MIMD can be split into two classifications

Tightly-coupled - short delay in communication between computers, high data rate (e.g., Parallel computers working on related computations)(MULTIPROCESSORS)

Loosely-coupled - Large delay in communications, Low data rate (Distributed Systems working on unrelated computations)(MULTICOMPUTERS)

Can be further subclassified as

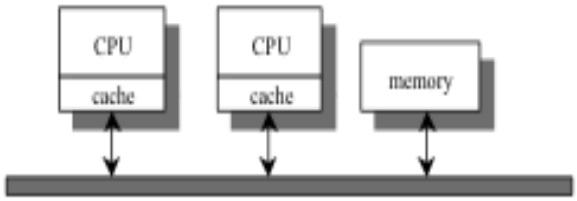
Bus - All machines connected by single medium (e.g., LAN, bus, backplane, cable)

Switched - Single wire from machine to machine, with possibly different wiring patterns (e.g, Internet)

MULTIPROCESSORS: CPUS have shared memory**.**

(A) Bus-based multiprocessors:

In a bus-based system, all CPUs are connected to one system bus. System memory and peripherals are also connected to that bus. If CPU A writes a word to a memory location and CPU B can read that same word back at any time after the write, the memory is *coherent.*



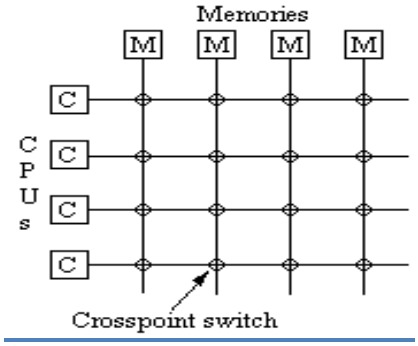
Bus-based interconnect with cache

A bus can get overloaded rather quickly with each CPU accessing the bus for all of its instructions and data. A solution to this is to add cache memory between the CPU and the bus .

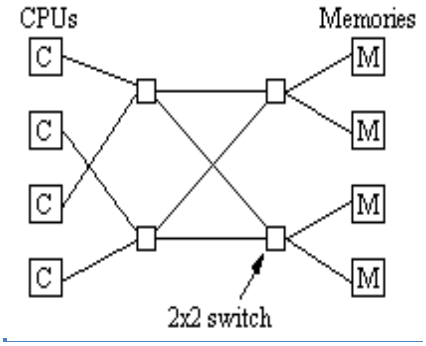
(B) Switched Multiprocessors

- Used for more than 64 CPUs
- Split memory into smaller modules
- Connect all CPUs to each memory module, two common method

Crossbar switch:



Omega network:



MULTICOMPUTERS: CPUs have separate memories.

(A)Bus-based multicomputers:

Bus-based multicomputers are easier to design in that we don't need to contend with issues of shared memory: every CPU simply has its own local memory. The communication network between the two is a bus (for example, an Ethernet local area network).

(B)Switched Multicomputers:

With a switched interconnect, all hosts connect to a network switch. The switch moves traffic only between communicating hosts, allowing other hosts to communicate without seeing their network speeds diminish. The huge benefit of switching is that it gives us a scalable network,.

2.2) SOFTWARE CONCEPTS :

There is no single definition or goal of distributed software but in designing distributed software, we often touch upon the same set of goals and problems. These general goals are transparency and scalability. This covers problems as diverse as:

- A network of redundant web servers

- Thousands of machines participating together in processing your search query..

The next step in building distributed systems is placing tightly-coupled software on loosely-coupled hardware. With this structure we attempt to make a network of machines appear as one single timesharing system, realizing the single system image. Users should not be aware of the fact that the machine is distributed and contains multiple CPUs. If we succeed in this, we will have a true distributed system.

III. TYPES OF DISTRIBUTED OPERATING SYSTEM

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

True Distributed Systems:

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

Multiprocessor Timesharing Systems:

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

IV. ADVANTAGES OVER CENTRALIZED SYSTEMS

1. Speed: When used to implement parallel processing where only goal is to achieve maximum speed on a single problem, distributed systems can achieve very high specd as compared to the centralized ones.

2. Inherent Distribution: Another reason for building a distributed system is that some applications are inherently distributed. Banking, Airline reservation etc. are examples of the applications that are inherently distribute .

3. Reliability: one is higher reliability. By distributing the workload over many machines, a single chip failure will bring down at most one machine, leaving the rest intact. For critical applications, such as control of nuclear reactors or aircraft, using a distributed system to achieve high reliability may be a dominant consideration.

4. Incremental Growth: It may be possible to simply add more processors to the system, thus allowing it to expand gradually as the need arises.

V. DESIGN ISSUES

1.Transparency : At the high levels, transparency means hiding distribution from the users. At the low levels, transparency means hiding the distribution from the programs. There are several forms of transparency:

(A)Location transparency : Users don't care where the resources are located. Migration transparency : Resources may move at will.

(B)Replication transparency : Users cannot tell whether there are multiple copies of the same resource.

(C)Concurrency transparency : Users share resources transparently with each other without interference.

(D)Parallelism transparency : Operations can take place in parallel without the users knowing.

2.Flexibility : It should be easy to develop distributed systems. One popular approach is through the use of a microkernel. A microkernel is a departure from the monolithic operating systems that try to handle all system requests. Instead, it supports only the very basic operations: IPC, some memory management, a small amount of process management, and low-level I/O. All else is performed by user-level servers.

3.Reliability :. Reliability encompasses a few factors: data must not get lost, the system must be secure, and the system must be fault tolerant.

4.Performance :. The communication links may be slow and affect network performance. If we exploit parallelism, it may be on a fine grain (within a procedure, array ops, etc.) or a coarse grain (procedure level, service level).

5.Scalability : We'd like a distributed system to scale indefinitely. This generally won't be possible, but the extent of scalability will always be a consideration.

VI. APPLICATIONS

- **Network applications:**
  - World wide web and peer-to-peer networks
  - Massively multiplayer online games and virtual reality communities
  - Distributed databases and distributed database management systems
  - Distributed information processing systems such as banking systems and airline reservation systems
- **Real-time process control:**
  - Aircraft control systems
  - Industrial control systems
- **Parallel computation:**
  - Scientific computing, including cluster computing and grid computing and various volunteer computing project.

VII. CONCLUSION

Distributed systems consist of independent CPUs that work together to make the absolute system look like a single computer. They have a number of possible selling points, including good price/performance ratios, the capability to match distributed applications well, potentially high consistency and incremental increase as the workload grows. They also have some disadvantages, such as extra composite software, potential communication bottlenecks, and weak security. Nevertheless, there is significant interest worldwide in building and installing them.. Distributed operating systems turn the entire set of hardware and software into a single integrated system, much like a usual timesharing system. Distributed systems have to be designed suspiciously. A key topic is simplicity — hiding all the distribution from the users and still from the application programs. Another issue is flexibility. the design should be made with the scheme of making potential changes easy.

REFERENCES

[1] Dos concepts and design By Pradeep K. Sinha
[2] Dos concepts and design ByAndrew S.Tanenbaum
[3] Distributed Systems: Principles and Paradigms Hardcover, By Andrew S. Tanenbaum (Author), Maarten van Steen (Author)
[4] www.cs.usfca.edu
[5] www.cs.columbia.edu
[6] www.b-u.ac.in/sde_book/distrib_computing.pdf