Journal Of Harmonized Research (JOHR)

Journal Of Harmonized Research in Engineering
1(2), 2013, 95-99

ISSN 2347 – 7393

Review Article

DISTRIBUTED OPERATING SYSTEM

Shweta Garg, Shrishti Vashist, Shruti Aggarwal

CSE Department, Dronacharya College Of Engineering,
Gurgaon, India

Abstract:
A logical model of a distributed operating system has been presented. This model of a distributed operating system contains a set of processes managing resources, connections between these processes, and mappings of events controlling this distributed operating system into processes managing resources. The fundamental types of resources introduced by the architecture of local computer networks, i.e., messages and data structures describing the location of resources in the network, have been defined. Operations on these resources and connections between the processes managing them and processes managing other resources of the distributed operating system have been presented. Addressing processes have been discussed. The model has been constructed in such a way that a synthesis of different simulation tools (models) to study distributed operating systems can be carried out.

Keywords: Ubiquitous- existing or being everywhere, Collaborative- characterised, Mitigates-to lessen in force, Proliferation- the growth of cells by multiplication of parts.
Abbreviation:- DOS- distributed operating system, IPC-inter process communication, OS-operating system

1. Introduction

A distributed operating system (DOS) is software over a collection of independent, networked, communicating, and physically separate computational nodes. Individual nodes each hold a specific software subset of the global aggregate operating system. Each subset is a composite of two distinct service provisioners. The first is a ubiquitous 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. They support the system's goal of integrating

For Correspondence:
yadav21shikhaATgmail.com
Received on: October 2013
Accepted after revision: November 2013
Downloaded from: www.johronline.com

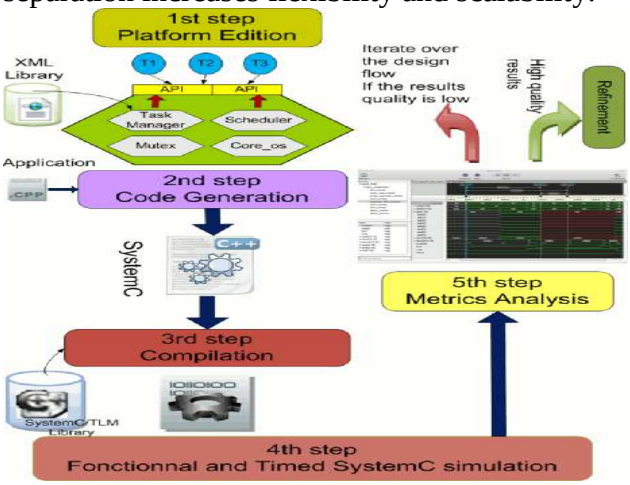
multiple resources and processing functionality into an efficient and stable system .This seamless integration of individual nodes into a global system is referred to as *transparency*, or *single system image*; describing the illusion provided to users of the global system's appearance as a single computational entity.

2. Description:-

Structure of monolithic kernel, microkernel and hybrid kernel-based operating systems

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." Mechanism and policy can be simply interpreted as "how something is done" versus "why something is done," respectively. This separation increases flexibility and scalability.



3. Overview:-

3.1. The kernel

At each locale (typically a node), the kernel provides a minimally complete set of node-level utilities necessary for operating a node's underlying hardware and resources. These mechanisms include allocation, management, and disposition of a node's resources, processes, communication, and input/output management support functions.Within the kernel, the communications sub-system is of foremost importance for a distributed OS.

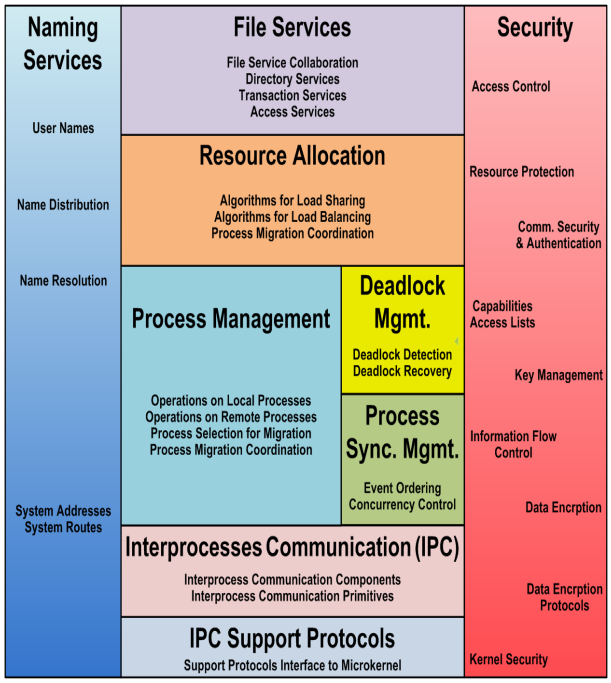
In a distributed OS, the kernel often supports a minimal set of functions, including low-level address space management, thread management, and inter-process communication (IPC). A kernel of this design is referred to as a *microkernel* Its modular nature enhances reliability and security, essential features for a distributed OS. It is common for a kernel to be identically replicated over all nodes in a system and therefore that the nodes in a system use similar hardware. The combination of minimal design and ubiquitous node coverage enhances the global system's extensibility, and the ability to dynamically introduce new nodes or services.

3.2. System management components

System management components are software processes that define the node's *policies*. These components are the part of the OS outside the kernel. These components provide higher-level communication, process and resource management, reliability, performance and security. The components match the functions of a single-entity system, adding the transparency required in a distributed environment.

The distributed nature of the OS requires additional services to support a node's responsibilities to the global system. In addition, the system management components accept the "defensive" responsibilities of reliability, availability, and persistence. These responsibilities can conflict with each other. A consistent approach, balanced perspective, and a deep understanding of the overall system can assist in identifying diminishing returns.

Separation of policy and mechanism mitigates such conflicts.



System management components overview

3.3. Working together as an operating system

The architecture and design of a distributed operating system must realize both individual node and global system goals. Architecture and design must be approached in a manner consistent with separating policy and mechanism. In doing so, a distributed operating system attempts to provide an efficient and reliable distributed computing framework allowing for an absolute minimal user awareness of the underlying command and control efforts.

The multi-level collaboration between a kernel and the system management components and in turn between the distinct nodes in a distributed operating system is the functional challenge of the distributed operating system. This is the point in the system that must maintain a perfect harmony of purpose, and simultaneously maintain a complete disconnect of intent from implementation. This challenge is the distributed operating system's opportunity to produce the foundation and framework for a reliable, efficient, available, robust, extensible, and scalable system. However, this opportunity comes at a very high cost in complexity.

3.4. The price of complexity

In a distributed operating system, the exceptional degree of inherent complexity could easily render the entire system an anathema to any user. As such, the logical price of realizing a distributed operation system must be calculated in terms of overcoming vast amounts of complexity in many areas, and on many levels. This calculation includes the depth, breadth, and range of design investment and architectural planning required in achieving even the most modest implementation.

These design and development considerations are critical and unforgiving. For instance, a deep understanding of a distributed operating system's overall architectural and design detail is required at an exceptionally early point. An exhausting array of design considerations is inherent in the development of a distributed operating system. Each of these design considerations can potentially affect many of the others to a significant degree. This leads to a massive effort in balanced approach, in terms of the individual design considerations, and many of their permutations. As an aid in this effort, most rely on documented experience and research in distributed computing.

4. 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.

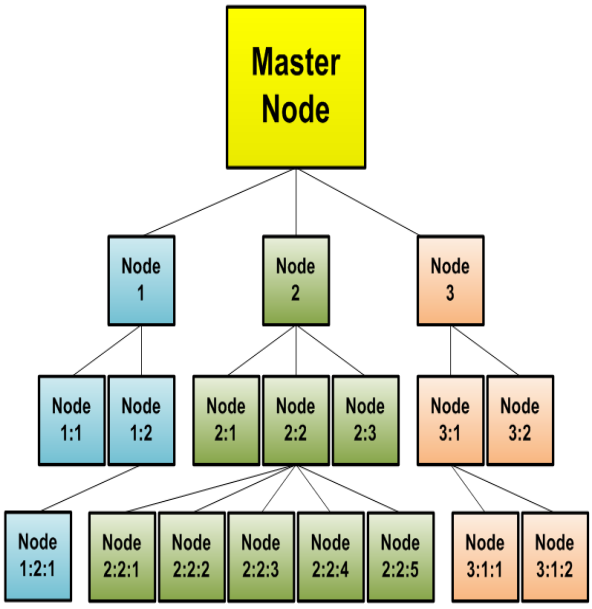
In the mid-1970s, research produced important advances in distributed computing. These breakthroughs provided a solid, stable foundation for efforts that continued through the 1990s.

The accelerating proliferation of multi-processor and multi-core processor systems research led to a resurgence of the distributed OS concept.

5. Distributed computing models:-

5.1. The nature of distribution

A distributed operating system's hardware elements spread across multiple locations within a rack, or around the world. Distributed configurations allow functions to be distributed as well as decentralized. The specific manner of and relative degree in linkage between the elements, or nodes in the systems differentiates the two. The linkages between the two are the lines of communication between the nodes of the system.



5.2. Three basic distributions

To better illustrate this point, examine three system architectures; centralized, decentralized, and distributed. In this examination, consider three structural aspects: organization, connection, and control. Organization describes a system's physical arrangement characteristics. Connection covers the communication pathways among nodes. Control manages the operation of the earlier two considerations.

5.2.1. Organization

A centralized system has one level of structure, where all constituent elements directly depend upon a single control element. A decentralized system is hierarchical. The bottom level unites subsets of a system's entities. These entity subsets in turn combine at higher levels, ultimately culminating at a central master element. A distributed system is a collection of autonomous elements with no concept of levels.

5.2.2. Connection

Centralized systems connect constituents directly to a central master entity in a hub and spoke fashion. A decentralized system (aka network system) incorporates direct and indirect paths between constituent elements and the central entity. Typically this is configured as a hierarchy with only one shortest path between any two elements. Finally, the distributed operating system requires no pattern; direct and indirect connections are possible between any two elements. Consider the 1970s phenomena of "string art" or a spirograph drawing as a fully connected system, and the spider's web or the Interstate Highway System between U.S. cities as examples of a *partially connected system*.

5.2.3. Control

Centralized and decentralized systems have directed flows of connection to and from the central entity, while distributed systems communicate along arbitrary paths. This is the pivotal notion of the third consideration. Control involves allocating tasks and data to system elements balancing efficiency, responsiveness and complexity.

Centralized and decentralized systems offer more control, potentially easing administration by limiting options. Distributed systems are more difficult to explicitly control, but scale better horizontally and offer fewer points of system-wide failure. The associations conform to the needs imposed by its design but not by organizational limitations.

6. Design considerations:-

- Transparency
- Inter-process communication
- Process management
- Resource management
- Reliability
- Availability
- Performance
- Synchronization
- Flexibility

References:-

1. Nutt, Gary J- Centralized and Distributed Operating Systems.
2. Fortier, Paul J. (1986). *Design of Distributed Operating Systems: Concepts and Technology*. Intertext Publications.
3. Chow, Randy; Theodore Johnson (1997). *Distributed Operating Systems and Algorithms*. Addison Wesley.
4. Hansen, Per Brinch, ed. (2001). *Classic Operating Systems: From Batch Processing to Distributed Systems*