IJDCST @ Sep-Oct, Issue- V-2, I-8, SW-18
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Commitment Processing Applications Development in Operating System

Kanduru Nischel[1], Krishna Kanth Tammana[2], Kurapati Yuva Pavan[3]

Abstract: In operating system recently there have been many changes those are services provided by the operating systems like, multiprocessor systems, memory management architectures and software architectures dictate a reevaluation of the virtual memory management have been advanced in many ways. So there is a main problem that is identified by using the Mach virtual memory management systemwhich will only display only the architecture independence and support to distributed system but there are no further features that will improve the functioning of the system. So,we introduce code designedvirtual machinethat will mainly include hardware performance feedback that will be taken continuously and also include adaptive performance features. By this we can mainly increase the performance of system significantly.

Key Words: Virtual memory management, distributed system, multiprocessor systems, adaptive performance.

I. INTRODUCTION

Operating System:

An operating system (OS) is software that manages computer hardware resources and provides common services for computer programs. The operating system is an essential component of the system software in a computer system. Application programs usually require an operating system to function.

Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware,[1][2] although the application code is usually executed directly by the hardware and will frequently make a system call to an OS function or be interrupted by it. Operating systems can be found on almost any device that contains a computer from cellular phones and video game consoles to supercomputers and web servers.

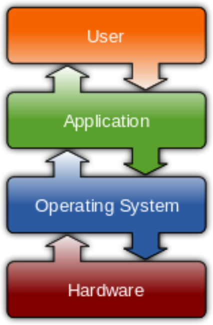

Figure 1: Architecture of OS

Architecture of OS will be explained by the above diagram First the request will be sent to the application software by the user and the application will be carrying it to the operating system and through OS it will be carried to hardware.

II. TYPES OF OPERATING SYSTEMS

Real-time

A real-time operating system is a multitasking operating system that aims at executing real-time applications. Real-time operating systems often use specialized scheduling algorithms so that they can achieve a deterministic nature of behavior. The main objective of real-time operating systems is their quick and predictable response to events. They have an event-driven or time-sharing design and often aspects of both. An event-driven system switches between tasks based on their priorities or external events while time-sharing operating systems switch tasks based on clock interrupts.

Multi-user

A multi-user operating system allows multiple users to access a computer system at the same time. Time-sharing systems and Internet servers can be classified as multi-user systems as they enable multiple-user access to a computer through the sharing of time. Single-user operating systems have only one user but may allow multiple programs to run at the same time.

Multi-tasking vs. single-tasking

A multi-tasking operating system allows more than one program to be running at the same time, from the point of view of human time scales. A single-tasking system has only one running program. Multi-tasking can be of two types: pre-emptive and co-operative. In pre-emptive multitasking, the operating system slices the CPU time and dedicates one slot to each of the programs. Unix-like operating systems such as Solaris and Linux support pre-emptive multitasking, as does AmigaOS.

Distributed

A distributed operating system manages a group of independent computers and makes them appear to be a single computer. The development of networked computers that could be linked and communicate with each other gave rise to distributed computing. Distributed computations are carried out on more than one machine. When computers in a group work in cooperation, they make a distributed system.

Templated

In an o/s, distributed and cloud computing context, templating refers to creating a single virtual machine image as a guest operating system, then saving it as a tool for multiple running virtual machines (Gagne, 2012, p. 716). The technique is used both in virtualization and cloud computing management, and is common in large server warehouses. [3]

Embedded

Embedded operating systems are designed to be used in embedded computer systems. They are designed to operate on small machines like PDAs with less autonomy. They are able to operate with a limited number of resources. They are very compact and extremely efficient by design. Windows CE and Minix 3 are some examples of embedded operating systems.

Virtual Systems:

A virtual machine (VM) is a software-based emulation of a computer. Virtual machines operate based on the computer architecture and functions of a real or hypothetical computer.A virtual machine (VM) is a software implementation of a machine (e.g., a computer) that executes programs like a physical machine. Virtual machines are separated into two major classifications, based on their use and degree of correspondence to any real machine:

- A system virtual machine provides a complete system platform which supports the execution of a complete operating system (OS).These usually emulate an existing architecture, and are built with the purpose of either providing a platform to run programs where the real hardware is not available for use (for example, executing on otherwise obsolete platforms), or of having multiple instances of virtual machines leading to more efficient use of computing resources, both in terms of energy consumption and cost effectiveness (known as hardware virtualization, the key to a cloud computing environment), or both.
- A process virtual machine (also, language virtual machine) is designed to run a single program, which means that it supports a single process. Such virtual machines are usually closely suited to one or more programming languages and built with the purpose of providing program portability and flexibility (amongst other things). An essential characteristic of a virtual machine is that the software running inside is limited to the resources and abstractions provided by the virtual machine—it cannot break out of its virtual environment.

A VM was originally defined by Popek and Goldberg as "an efficient, isolated duplicate of a real machine". Current use includes virtual machines which have no direct correspondence to any real hardware [4].

System virtual machine advantages:

- Multiple OS environments can co-exist on the same computer, in strong isolation from each other
- The virtual machine can provide an instruction set architecture (ISA) that is somewhat different from that of the real machine
- Application provisioning, maintenance, high availability and disaster recovery[3]

The main disadvantages of VMs are:

- A virtual machine is less efficient than a real machine when it accesses the hardware indirectly
- When multiple VMs are concurrently running on the same physical host, each VM may exhibit a varying and unstable performance (speed of execution, and not results), which highly depends on the workload imposed on the system by other VMs, unless proper techniques are used for temporal isolation among virtual machines.

III. RELATED WORK

R. Fitzgerald stated that the integration of virtual memory management and inter-process communication in the Accentnetwork operating system kernel is examined. The design and implementation of the Accent memorymanagement system is discussed and its performance, both on a series of message-orientedbenchmarks and in normal operation, is analyzed in detail.

A. Spector stated that the TABS prototype is an experimental facility that provides operating system-level support for distributed transactions that operate on shared abstract types. The facility is designed to simplify the construction of highly available and reliable distributed applications. This paper describes the TABS system model, the TABS prototype's structure, and certain aspects of its operation. The paper concludes with a discussion of the status of the project and a preliminary evaluation.

R. A. Meyerstated that time-sharing systems are now an important factor in the computer industry. Because they allow a multiplicity of users to have access to a computer by means of a terminal, they encourage increasing numbers of people to utilize computers. Thus, various ways of increasing the capability of a computer in this area are being sought. One such system is the a multi-access system that manages the resources of a computer set up for time-sharing such that each (remote) user appears to have a complete, dedicated computer at his disposal. This concept is known as a virtual machine and allows each user to select the operating system he wishes to run because concurrent operation of several operating systems is possible.

Richard Rashid stated that in the recent times the technologies that are used in memory management for example multiprocessor systems, and software architectures dictate a reevaluation of the virtual memory management support provided by an operating system. So there will the problem posed by this virtual memory system we cannot carry multi-process systems and also the portability issues. So for this Mach virtual memory management is used that will support advanced features.

IV. EXISTING SYSTEM

Mach Virtual Memory:

Mach's virtual memory system has two primary objectives: (a) to be as portable as the UNIX virtual memory system while supporting more functionality (see below), and (b) to support multiprocessing, distributed systems and large address space.

The Virtual Memory User Interface

Its main features are:

1. A consistent virtual memory interface on allmachines supporting Mach: some features, such asshared pages, can be more or less efficiently implementeddepending on the underlying hardware;
2. Full support for multiprocessing: this includesthread support, efficient data sharing mechanismsand a fully parallel implementation of the virtualmemory;
3. Modular paging: there is no dedicated swap area andexternal pagers are allowed to implement filemapping or any application-specific paging policy(such as recoverable virtual memory for transactionmanagement).

Mach uses a global FIFO memory policy but places expelled pages on an inactive list from which they can be reclaimed. This the same policy as VMS

but for the fact that VMS allocated a separate resident set of frames to
each process.

All virtual memory algorithms rely on locks whenever exclusive access to a data structure has to be guaranteed. To prevent deadlocks, all algorithms gain locks using the same ordering. The total size of the machine-dependent part of Mach'svirtual memory implementation is about 16 kilobytes.

Applications:

Mach uses its memory object mapping mechanism to implement standard UNIX I/O semantics while allowing user programs to access directly the mapped file data.

As in Accent, copy-on-write is used to implement efficient message passing: messages can be sent and received without having any data copied until either the sender or the receiver tries to modify the data. It also provides a much faster implementation of the UNIX fork() and eliminates the vfork() kludge. Shared libraries are supported through the mapped file interface.

V. PROPOSED SYSTEM

The Code design Virtual Machine:

CVM models hardware and software behaviorally. In this section we present the underlying semantics and their role in system modeling. The basis for modeling in CVM includes software functions, hardware threads, software threads, and software processes (which may be considered threads with private namespace). These are all schedulable execution entities that advance system state — and are all potentially concurrent entities. We consider a thread the basic unit of modeling state advancement in any system; the development of our CVM focuses on the identification of existing thread types from the hardware and software domains. We then consider novel hierarchical system integration methodologies and system scheduling strategies.

Resource-Based Execution Models:

We define resource-based computation (the R domain) as computation that advances state as a function of system inertia, or F(inertia). Table 1 lists models of state update that closely couple behavior and the underlying physical models that carry out the behavior. When new behavioral threads are added in the resource domain, new resources are added to carry out the behavior. These models of concurrency require, in general, a simulation to correctly advance state specified by the behavior. Simulators typically capture these models of concurrency by controlling the advancement of simulation time and by allowing designers to specify inertial properties of computation.

type	when they occur	design scenarios
C	continuous update	timed “sampling/generating”
D	any value change	async portions of HW
G	global sync	clock edges in HW
WH	hardware wait: change to level	HW synchronization, processor interrupt

Table 1 Resource-based Models of Concurrency

One way of viewing these loops is shown in Figure 1, where each processing element has only one loop assigned to it; it translates inputs to outputs in the same shared memory space by time-interleaving access on an idealized bus. Translation occurs at the inertial processing rate of each processing element. Private memories permit program/data interactions, unlike pure hardware models for which inertial propagation is (ideally) static.

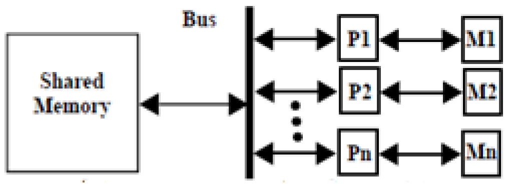

Figure 2: Idealized Resource-based computation model

The structural scheduling implied by a hardware description can thus be thought of as a shared memory that is continuously sampled and updated by the threads. Type C threads are a more general model of resource-based computation than a hardware model, and thus they form the basis of the thread type of the resource modeling domain of the CVM.

Interleaved Execution Models:

We define state-interleaved computation (the I domain) as computation that advances state as a function of system state, alone, or G(F(state)), where the ÒGÓ stands for guarded execution. The threads in Table 2 list models of state update that are designed to be activated by a state-based, functional global scheduling paradigm, independent of resources except for an assumed single implied CPU and shared memory space.

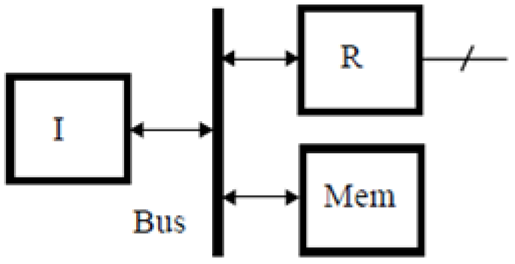

Figure 3: Code Design Virtual Machine Combining Execution Models:

Figure 2 portrays state-interleaved (untimed) and resource-based (timed) models of computation in an idealized architectural view of our CVM. The domains co-execute as peer-based modeling domains as in but with novel scheduling abstractions and hierarchical relationships (described later) allowing the timed and untimed domains to be de-coupled, and yet resolvable to the same scheduling semantic. Each thread in the R domain implies addedresources to support its execution. Resource threads (R) are derived from type C threads, continuously translating state in an unsynchronized fashion that merges inertial modeling with shared state modeling. Each thread in the I domain is derived from type G(F) threads, activated by a state-interleaved scheduler, which resolves to one or more continuous loop resources.

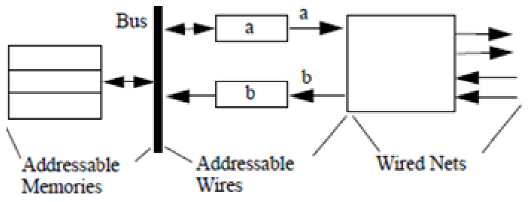

Figure 4:Connecting CVM State Domains

Virtual machines provide a common semantic toward which designers specify systems. CVM has been defined to capture, on a peer basis, the models of thread activation and state advancement necessary to the design of physical systems containing resource based and state-interleaved threads.

VI. RESULTS

In the results we mainly say that the existing system that is Mach virtual machine system will not be having the some properties that will be helping the OS work effectively. So our results show that proposed system that is a co-designed virtual machine will be providing many significant properties that are not provided by the existing system. So, there will be much more improvement for the project.

VII. CONCLUSION

In this paper we conclude that the proposed system will provide many important features that will be helping the OS to work in an efficient manner some of those features are adaptive hardwareperformance features, continuous hardware performancefeedback, and on-the-fly optimizing re-compilation. And these properties are not provided by the existing system and those will be having some problem in portability issues and in multi-processing systems. So as a result we use proposed system to overcome those problems.

VIII. REFERENCES

[1].Stallings (2005). Operating Systems, Internals and Design Principles. Pearson: Prentice Hall.

[2]. Dhotre, I.A. (2009). Operating Systems Technical Publications.

[3]. Silberschatz Galvin Gagne (2012). Operating Systems Concepts. New York: Wiley.

[4].Smith, James; Nair, Ravi (2005). "The Architecture of Virtual Machines". Computer (IEEE Computer Society).

[5].Machine-Independent Virtual Memory Managementfor Paged Uniprocessor and Multiprocessor - Architectures

Richard Rashid, Avadis Tevanian, Jr., Michael Young, David Golub, Robert Baron.

[6].R. Fitzgerald and R. F. Rashid. “The integration of virtual memorymanagement and interprocess communication in Accent,” ACM Trans. Computer.

[7].A. Spector et al., “Support for distributed transactions in the tabsprototype.” in Proc. 4th Symp. Reliability Distributed Software Database System.

[8].R. A. Meyer, L. H. Seawright, “A virtual machinetime-sharing system,” IBM System Journal.