2021 International Conference on Intelligent Computing, Automation and Applications (ICAA) | 978-1-6654-3730-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICAA53760.2021.00135

A Survey of Operating System Microkernel

Buquan Liu[a, *], Chunguang Wu and Hao Guo
Research Institute of Innovation, Kylin Software Company Limited, Tianjin 300450.
[a]liubuquan@kylinos.cn

Abstract—**Microkernel operating system plays an important role in industrial control system, embedded system and real-time system, etc. From many aspects, this paper studies the microkernel technology, such as its architecture, history, characteristics, and scenarios. In view of the deficiencies of microkernel architecture in the future interconnection of everything, this paper expounds the further research direction of microkernel from interoperation, distributed storage, security and so on. Finally, some experiments, based on seL4 microkernel, are discussed, including virtualization, and performance comparison with Linux.**

I. INTRODUCTION

In computer science, the kernel is a crutial component of operating system. It manages all kinds of system resources, such as clock, interrupt, storage, process, device driver, primitive and so on. Applications run on the kernel and access its space through system calls [1].

The kernel of operating system is divided into microkernel and monolithic kernel [2]. Figure 1 is a classic illustration to distinguish the two kernels. The microkernel, abbreviated as μkernel, is designed to be as small as possible. It provides the basic mechanisms of inter-process communication (IPC), virtual memory and scheduling to communicate with underlying hardware and software in user space. However, the monolithic kernel is just like a huge boulder to provide large amount of functions in the kernel space. The minimization of microkernels is called the Jochen Liedtke minimization principle [3].

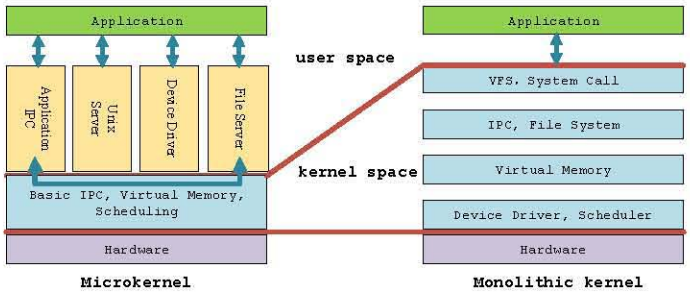

Figure 1. Two kinds of operating system kernels.

In this paper, we will introduce the architecture, history, characteristics and application scenarios of microkernel. Then, we will discuss its shortcomings and put forward the further research directions of microkernel from interconnection, storage and security. Finally, we will introduce several experiments based on seL4 microkernel [4].

II. ARCHITECTURE OF MICROKERNEL

A. Distributed Server Architecture

In the case of the microkernel in Figure 1, according to the official statement, each module in the user space is designed as a server, and the microkernel uses the message to communicate each server. Specifically, the server here may be thought as a server program. The microkernel is equivalent to a client program.

Theoretically, two programs of client and server based on message communication can be located on either the same machine or two machines. Therefore, these servers in the microkernel operating system can be distributed. They may be deployed to different machines in theory although very few operating systems have really implemented in this way.

The kind of distributed architecture has special significance. Both operating systems of Android's Fuchsia and Huawei's Hongmeng claim to have two characteristics: microkernel and distributed. With distributed message communication, operating systems may interconnect with various hardware devices such as mobile phones, pads, watches, PCs, etc. This communication mechanism is also called distributed bus [5].

B. Plug-in Architecture

The server is independent of each other in the microkernel operating system, which is also called plugin [6]. The plugin architecture consists of a core system and several plugin modules. The biggest advantage of this architecture is that it allows third-party developers to follow certain specifications to develop additional plugin applications.

978-1-6654-3730-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICAA53760.2021.00135

As the core system, the microkernel is stable and rarely changed. It only needs the smallest functions to support the applications in user space without involving any other tasks. The operating system has good scalability, and its plugins can be changed with actual requirements. In fact, the essence of plugins is to encapsulate the changing parts of a software system, which achieves the purpose of rapid and flexible reorganization and expansion of the system.

III. History of Microkernel

Since the 1970s, people began to develop microkernel but the term appeared for the first time in 1981. The earliest kernels had always been monolithic, but monolithic operating systems were more like those with microkernel because of simplicity. With the development of computer, new device drivers, protocol stacks, file systems and other low-level systems were constantly emerging. This caused the monolithic kernels to change all the time and brought a serious burden to the maintenance. Therefore, microkernels were applied for a new idea that the new services should be implemented in user space, allowing them to work, start and stop independently as application programs. This not only made these services easier to develop, but also separated the kernels without bringing unexpected side effects. In this way, the microkernels would not be modified frequently so that they were easy to manage and maintain [7].

A. First Generation

Represented by Mach, a microkernel operating system. It attempts to reconstruct the architecture of operating system with monolithic kernel, but it still maintains the original ability of operating system. However, the performance of the first generation microkernel including Mach or IBM workplace is not good. For Mach3, it is about 50% slower than the original monolithic kernel UNIX. In the end, most people tend to think that the reason for low performance is due to the high cost of IPC.

Take the context switching of file operations of reading and writing involved in a user program as an example.

1) The monolithic kernel switches between kernel space and user space twice: user program <--> monolithic kernel (twice for one round trip);

2) Microkernel has 8 times of switching: user program < -- > microkernel < -- > file server <--> microkernel <--> disk device driver.

B. Second Generation

Represented by L3 and L4, the first generation of microkernel has proved the feasibility conceptually. However, due to performance, it has not been widely promoted in the industry, but related academic research has been emerging. In view of the performance disadvantage of the first generation microkernel, Jochen Liedtke designed a second generation microkernel L4 with high performance. Compared with the first generation, the second generation microkernel is no longer a simple reconstruction of the traditional UNIX operating system, but a new design with high performance.

L4 kernel supports three abstract concepts: address space, thread and IPC. It only provides 7 system calls and only 12K byte code. L4 uses many novel technologies to improve performance, such as direct address translation, lazy scheduling, register transfer of short messages, reduction of cache and TLB miss, etc.

C. Third Generation

With seL4 as the representative, the third generation microkernel emphasizes security. SeL4, developed by Australian researchers, is based on L4. SeL4 itself means security L4. It is called through a mechanism similar to object reference, namely capability. Any kernel object such as thread, interrupt, memory, etc. can not be operated directly, which solves the security problem.

D. New Generation

With the development of network technology, the new generation of microkernel emphasizes the interconnection of everything. The representative microkernel operating systems are Android's Fuchsia and Huawei's Hongmeng.

IV. Characteristics of Microkernel

A. Safety, Reliability, Portability and Maintainability

In terms of security, microkernel is generally considered to be more secure than monolithic kernel. The servers above the microkernel are relatively independent, and they may be rebooted after crashing, which usually do not affect the whole operating system. The monolithic kernel are embassed because all the modules are in one kernel, and the collapse of a module is easy to lead to the failure of the whole system. The core problem of monolithic kernel is insecurity, which is a consequence not to comply with the principle of least authority (POLA). Moreover, the biggest hidden danger lies in the expanding of kernel modules. For example, all codes run at the highest level of CPU with ring 0 in Linux kernel. One malware in kernel can destroy key data such as interrupt table and system call table. However, the microkernel is the architecture without dynamic expansion. Process management, file system and network services all run in ring 1 for microkernel operating system.

In addition, as the kernel is small, the operating system with microkernel has better reliability, portability and maintainability than that with monolithic kernel.

B. Scalability

In terms of scalability, microkernel only includes key modules with unchanged functions while other modules of operating system are located above the kernel, so it is easy to expand new modules in user space. However, monolithic kernel contains most modules of operating system and it is not easy to expand to add new modules because they will affect other modules in the kernel.

C. Continuity

In the field of industrial control, some systems run continuously for several months or even years. Continuity refers to long-term work without restart. Of course, if it fails, it should be able to resume state quickly. Compared with monolithic kernel, microkernel has better continuity because of less functions, high security and reliability.

D. Performance

Most modules are in monolithic kernel for the operating system so that these modules only call each other by function. However, most modules are outside the kernel for microkernel operating system so they can not interact with each other directly. In fact, the modules above microkernel interact with each other by context and address space switching. In this way, the performance of microkernel is generally considered worse than that of monolithic kernel.

E. Real-time

Performance and real-time are two different concepts, and there is no necessary connection between them. The contrast of real-time kernel is time-sharing or not real-time kernel. The real-time kernel requires the immediate response of the system, such as the operating system of rocket, UAV and so on. While the time-sharing system does division-time processing for tasks and equally divides the processor time, so it does not have real-time performance [8].

V. SCENARIOS OF MICROKERNEL

In the development of microkernel, there are many microkernel operating systems, such as Mach, Fiasco. OC, OKL4 Microvisor, seL4, NOVA, P4 Pike, Minix, µC/OS-ii, ADEOS, EPOC, EKA1, EROS, Mac OS, Windows NT, Micro Empix, TI-RTOS, WarpOS, embedded Linux, Windows Embedded, PSOS, VxWorks, Zircon, etc.

Theoretically, an operating system can adopt either microkernel or monolithic kernel. For example, the early Windows NT and Mac OS adopted the microkernel, and later moved part of the user space to the kernel space due to performance problems, forming a hybrid kernel. Linux adopts monolithic kernel. However, with the continuous development of computer and operating system, there are practical differences between microkernel and monolithic kernel in different application scenarios.

A. Embedded System

Embedded system has the characteristics of limited resources, strong specificity and high real-time performance. Therefore, the operating system applied to embedded system must meet the requirements of small kernel, real-time multitasking, memory protection and tailorability. Embedded operating system is responsible for the allocation of all hardware and software resources, task scheduling, control and coordination of concurrent activities. It must reflect the characteristics of the system in which it is located and be able to achieve the tasks required by the system by loading and unloading some modules.

This requirement of embedded system is consistent with the development of microkernel. The two basic principles of minimization and high IPC performance are still the driving forces of design and implementation.

In the field of embedded system, many systems do not even have hard disk storage, they often use microkernel operating system.

B. Industrial Computer, PC and Server

For industrial computer, PC and server, operating system needs to complete many complex functions so that monolithic kernel is more suitable. If the emphasis is on real-time, a simple industrial control computer can also equipped with the microkernel operating system, such as VxWorks.

Again, in theory, both microkernel and monolithic kernel operating systems can be used in various scenarios, but the architectures adopted by various operating systems have led to their emphasis in specific application scenarios. For example, microkernel operating system can also run on some PC or server. GNU Hurd is a microkernel operating system that has been developed since 1990 and runs on Mach or L4. The latest version seen from the official website is Debian version developed in 2019. As a microkernel operating system, it can develop and test new Hurd kernel components without restarting the machine. Running its own kernel components will not interfere with other users, so it does not need special system privileges. The mechanism for kernel extensions is designed to be secure: it is impossible to impose changes from one user on another

unless the system administrator or authorized user does so. However, due to the lack of developers and slow progress, it has not yet reached the stage of full maturity and ease of use [9].

C. Mobile and Internet of Things

In the field of mobile and Internet of things, people generally use microkernel operating systems, such as Android and IOS (also called hybrid kernel).

In short, for some special systems, mainly real-time systems, embedded systems, Internet of things, the idea of microkernel is more attractive. The main reason is that these systems usually do not have disks, and the whole system must be placed in EPROM, which is often limited by the storage, and the required services are relatively simple. Other general-purpose systems are more appropriate to use monolithic kernel.

VI. Further Research Directions of Microkernel

Although microkernel has been widely used in many fields, it may be further studied in some directions for better development. Firstly, microkernel may establish interconnection protocols for different CPUs. Secondly, a large amount of front-end data generated by microkernel hardware devices will bring challenges to traditional data storage, and microkernel should also increase support for modern distributed storage. Thirdly, security research should be added in open environment. As shown in Figure 2, the new generation of microkernel architecture can add network interconnection, distributed storage, security gateway and other servers on the basis of the original architecture, so as to better meet the requirements of 5g, big data, mass storage, security.

A. Enhancing Interoperability Based on Network Interconnection Protocol

Google's Fuchsia and Huawei's Hongmeng are the microkernel operating systems that have attracted high attention in recent years [10]. At present, it is mainly used in the embedded field, but it puts forward the idea of extending to the desktop and server in the future.

There are many difficulties in applying a microkernel operating system to desktop and server, or even replacing some existing operating systems in the future. At present, most operating systems of desktop and server adopt monolithic kernel. There had ever been a famous debate about which architecture better between microkernel and monolithic kernel. MINIX and Linux are two famous open source operating systems. MINIX adopts microkernel and Linux adopts monolithic kernel. In 1992, Andrew S. Tanenbaum, the inventor of MINIX, and Linus Benedict Torvalds, the founder of Linux, had a debate on the two architectures, but they did not decide which one was better [11-12].

This paper proposes a more reasonable way to solve the interoperability problem between various operating systems by designing the network interconnection protocol. As shown in Figure 3, various operating systems with monolithic kernel and microkernel communicate with each other through the protocol. The protocol has to solve the heterogeneity for operating systems based on different CPUs such as big endian and little endian. Moreover, it is necessary to solve the problem of heterogeneity between different programming languages.

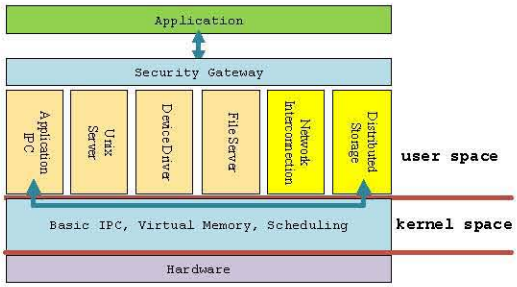

Figure 2. Enhancement of Microkernel.

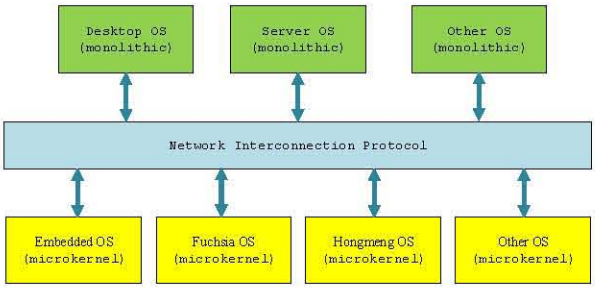

Figure 3. Network Interconnection.

B. Enhancing Microkernel to Support New Generation Storage

In the industrial Internet, the front-end devices using microkernel operating system often have no hard disk, and the data has to be saved to the background storage system or database in C/S mode. If there are many front-end devices, this storage method may lead to many process connections with the back-end server, which makes it difficult to respond to the read-write requests of each front-end device in time. However, the new generation of storage systems such as Ceph and MongoDB can support a large number of concurrent requests, and enhance the driver support so that the front-end devices can directly read and write Ceph and MongoDB, just as to directly read and write the local file system or database.

If the file system in the microkernel operating system is extended to support new generation storage, it can effectively solve the high concurrent read-write requests of many front-end devices, and ensure reliable storage and high availability of big data.

C. Safety Enhancement

Microkernel devices play an important role in industrial Internet, Internet of things, embedded, mobile and military fields. It is really essential to delve into the security of program and data in user space, which is quite different from the traditional research on the security in kernel space.

Program security should limit its access rights and must not be allowed to destroy operating system. Data security refers to the ability to avoid intentional or accidental transmission, leakage, damage and modification without permission.

This paper proposes a design method of adding a security gateway server. The server is located between the user program and the modules of the operating system. It can effectively deal with the illegal operations of external programs to the operating system.

VII. Experiments on seL4 Microkernel

SeL4 is a microkernel that has passed formal verification and the code has been proved to be completely correct. In fact, seL4 is not only a microkernel, but also a hypervisor similar to KVM, which a virtual machine may run on. As shown in Figure 4, seL4 supports running other Linux operating systems with monolithic kernel in a virtual machine [13-14].

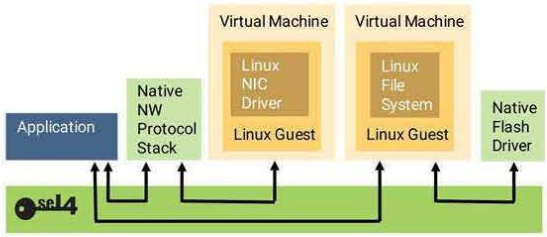

Figure 4. Virtualization technology based on seL4.

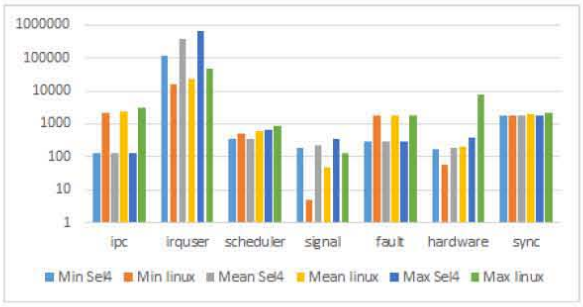

Figure 5. Performance of seL4 and Linux.

A. Experiment 1: Virtualization

The virtualization in Figure 4 is interesting. The experiment is to learn how to run Linux on seL4. In the experiment, an x86 machine with NeoKylin Linux Advanced Server release v7 was selected to compile seL4, make a startup menu and the image of virtual machine. After rebooting the machine, we can observe the output from serial port about the virtual machine running over seL4.

B. Experiment 4: Performance Comparison between seL4 and Linux

SeL4 officially provides sel4bench, a performance test tool. It can directly compile and run to get the test results under the microkernel. We also test it in a Linux release: CentOS 4.18.0-193.19.1.el8_2.x86_sixty-four. Figure 5 shows that the comprehensive performance of seL4 kernel is nearly equal to that of Linux kernel.

VIII. Conclusion

The operating system kernel is divided into monolithic kernel and microkernel. Al most all modules of monolithic kernel reserve in kernel space. Currently, mainstream desktop and server operating systems use this solution. Microkernel only retains minimal modules. This small kernel has the advantages of security, reliability, scalability, portability, maintainability, scalability and continuity. It is widely used in industrial control, embedded, real-time, mobile, Internet of things and other areas. This paper systematically discusses the architecture, history, characteristics and application scenarios of microkernel, and puts forward the idea of building an interconnection protocal to connect various operating systems rather than reinvent an omnipotent one for various areas. Except for interoperability, this paper also expounds the further research directions of microkernel from distributed storage and network security. Finally, some experiments based on seL4 microkernel are discussed.

References

[1] Tanenbaum A S and Bos H 2015 Modern operating systems. Essex: Pearson Education Limited.

[2] Jordan S 2014 Analysis of monolithic and microkernel architecture: towards secure hypervisor design. Proceedings of the 47th Annual Hawaii International Conference on System Sciences, Waikoloa, HI, United States, pp 5008-5017.

[3] Hermann H and Michael R 2012 As time goes by: research on l4-based real-time systems. Advances in Real-Time Systems. Springer Berlin Heidelberg, pp 257-273.

[4] Jiao X 2013 Design and implementation of a microkernel based on L4. Xian: Xidian University.

[5] Huawei. Hongmeng OS: microkernel and distributed. [2021-01-10]. https://ishare.ifeng.com/c/s/7p35GK7m18x.

[6] Talk about microkernel architecture from Hongmeng OS. [2021-01-10]. https://blog.csdn.net/weixin_43258870/article/details/98988429.

[7] Erlingsson U and Kyparlis A. Microkernels . [2021-01-10]. http://www.cs.cornell.edu/info/people/ulfar/ukernel/ukernel.html.

[8] Yan W 2019 Research and implementation of real-time microkernel for industrial control domain. Chengdu: University of Electronic Science and Technology.

[9] GNU HURD. http://www.gnu.org/software/hurd/index.html.

[10] Chokkattu J and Jansen M. Google's Fuchsia OS: Everything you need to know. [2019-01-03] [2021-01-10]. https://www.digitaltrends.com/mobile/google-fuchsia-os-news/.

[11] Open Sources: Voices from the Open Source Revolution. 1st Edition. [2021-01-10]. https://www.oreilly.com/openbook/opensources/book/appa.html.

[12] Wang H M 2012 Research and improvement of a micro-kernel system MINIX3 dissertation. Shanghai: Shanghai Jiaotong University.

[13] Heiser G. The seL4 microkernel - an introduction. White Paper. The seL4 Foundation, Revision 1.2 of 2020-06-10. https://sel4.systems/Foundation/About/.

[14] Wang T C, Zhao C D, Wei X and Gao Y H 2018 Design of health monitoring for partition operating system based on seL4. Computer Engineering and Design, 39(5), pp 1296-1301.