Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9863459, 1 page
https://doi.org/10.1155/2023/9863459

Retraction

Retracted: Design of Embedded Ai Engine Based on the Microkernel Operating System

Wireless Communications and Mobile Computing

Received 10 November 2022; Accepted 10 November 2022; Published 21 January 2023

Wireless Communications and Mobile Computing has retracted the article titled "Design of Embedded Ai Engine Based on the Microkernel Operating System" [1] due to concerns that the peer review process has been compromised.

Following an investigation conducted by the Hindawi Research Integrity team [2], significant concerns were identified with the peer reviewers assigned to this article; the investigation has concluded that the peer review process was compromised. We therefore can no longer trust the peer review process and the article is being retracted with the agreement of the Chief Editor.

References

[1] T. Wang and T. Yu, "Design of Embedded Ai Engine Based on the Microkernel Operating System," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 9304019, 9 pages, 2022.

[2] L. Ferguson, "Advancing Research Integrity Collaboratively and with Vigour," 2022, https://www.hindawi.com/post/advancing-research-integrity-collaboratively-and-vigour/.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 9304019, 9 pages
https://doi.org/10.1155/2022/9304019

Research Article

Design of Embedded Ai Engine Based on the Microkernel Operating System

Tun Wang[1] **and Yu Tian**[2]

[1]*College of Computer Science and Technology, University of Electronic Science and Technology of China, 610000 Chengdu, China*
[2]*Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, 100864 Beijing, China*

Correspondence should be addressed to Tun Wang; tracy.wang@nj3i.com

Received 26 February 2022; Revised 14 March 2022; Accepted 19 March 2022; Published 21 April 2022

Academic Editor: Rashid A Saeed

At present, the application of the embedded microkernel operating system in military and civil fields has begun to take shape, but it has not yet formed a unified method and standard. Due to its high performance, low frequency, and high reliability, dual-core embedded processors are getting the attention of many chip manufacturers. Compatibility has been favored by many telecom equipment manufacturers and embedded high-end application integrators, but the dual-core embedded processor needs a new real-time operating system to support it, so that it can give full play to the high performance of the dual-core. It paves the way for the application of its processor in the embedded field, but the design of embedded AI engine is not transparent to it; so, it needs the support of operating system. The user code is used to be in the operating system processor environment; so, it can be used on dual core processor first. In the real-time operating system that supports dual-core processors, the part that needs to be modified is mainly concentrated in the kernel part; so, the core design is the key point to support dual-core processors. This article is to seize this key point to carry out in-depth research. The difference and influence of hardware architecture between dual-core processor and single-core processor are the primary content of the study. Through the research of the dual-core processor architecture, the general abstraction of the dual-core processor architecture is obtained, which is the starting point of the follow-up research. This paper mainly studies the design of embedded AI engine based on the microkernel operating system, extracts the security requirements of the operating system, designs and implements the operating system from the perspective of formal verification, and considers the verification problem under the background of RTOS development, so as to avoid using too many complex data structures and algorithms in the system design and reduce the difficulty of experimental verification. In this paper, we use the spatiotemporal data model, data sharing security in the cloud environment, symmetric encryption scheme, and Paillier homomorphic encryption method to study the design of embedded AI engine based on the microkernel operating system. According to the idea of microkernel architecture, the kernel is divided into four main modules: task processing, semaphore, message queue, interrupt, and exception processing, and the lock mechanism to prevent reentrancy in software is analyzed separately. Three core function modules, initialization, process grinding, and interrupt processing, are extracted from the microkernel operating system to form the formal verification area of the operating system. At the same time, the system syntax and semantics of related rights are separated, and the main rationalization rules are described. The results show that the core of the microkernel operating system in five states adopts the microkernel architecture in the dual core environment. The microkernel architecture is a compact system kernel with good adjustability. Based on the analysis of the microkernel operating system, the internal structure of each module in the kernel is summarized, and the modules are modified according to the machine characteristics of the dual core processor; it corresponds to adding modules to meet the characteristics of dual core architecture. Kernel design is a systematic theoretical research process. This paper only uncovers the tip of the iceberg of real-time operating system kernel design on dual-core embedded processors. It is necessary to understand the kernel more deeply and master the kernel in the future work and study.

1. Introduction

The real-time operating system refers to the system that can complete system functions and respond in a certain period of time. The factors that determine the accuracy of real-time operation results include not only logic factors but also time factors, soft real-time, and hard real-time. The real-time operating system based on dual-core processor can be obtained in two ways. One is to analyze the mature RTOS running in the environment of single-core processor and make certain modifications in combination with the environmental changes of dual-core processor addition operation. Another method is to simplify the operating system from the multiprocessor environment and apply it to the single-core processor, which is to do subtraction. The safety of its operating system is of great significance in medical, military, nuclear power engineering, UAV, and other fields, As the core foundation of information industry, the reliability of the operating system is directly or indirectly related to national security, economic, and social stability and people's production and life safety. The research of the operating system has always been a hot topic in the industry and academia, especially in the field of security. We must ensure the security of the operating system. At present, the most effective way to control the operating system is the formal method. Most researchers pay attention to the functional correctness and integrity of the operating system and rarely analyze, model, and control the operating system. It is possible to realize more sophisticated artificial intelligence technology in game development. The current mainstream dual-core processor instruction stream architecture mainly includes multi-issue instruction mode (i.e., superscalar instruction set processing mode) and multithreaded instruction processing mode. Among these two instruction streams, the former is widely used in Intel's X86 architecture processors and powerpc series processors. Multithreading technology is widely used in DSP chips and XRL multicore and multithread processors introduced by RMI. The selling point of online games in the 21st century will focus on artificial intelligence engines. At present, the popular rule-based artificial intelligence engine technology at home, and abroad is usually realized by finite state machine or fuzzy state machine, that is, the switch-case model. However, AI engines based on finite state machines have various drawbacks, and players need to be more intelligent. All in all, the AI engine based on finite state machine can no longer meet the needs of players; so, this article proposes a relatively novel design model of artificial intelligence engine-microprocessor-based artificial intelligence engine. It is no longer a rule to regulate the behavior of the game character in a certain state; so, it can create more natural intelligent behavior.

Compared with conventional control system kernel, microkernel reduces the code of hardware encapsulation mechanism to the minimum and runs in privileged mode. In the past, the implementation of microkernel needs a lot of communication, which makes the practicability of micronucleus inferior to that of macrocore. The current design and implementation technology of microkernel can reduce the cost to a certain extent and make the microkernel achieve the ideal performance. With the increase of the complexity of the system software kernel, the reliability of the kernel becomes more and more important. Especially in the embedded real-time systems commonly used in industry, most of these systems are based on the priority of pre scheduling, that is, allowing high priority processes to interrupt the implementation of low priority processes, and with the increase of complexity, security control becomes more and more difficult. An important significance of microkernel design is to improve the scalability of the system. The microkernel can be regarded as a further abstraction of the traditional system; in a sense, it can be called the "engine of the engine." This further abstraction makes the AI engine core only provide a scheduling mechanism, while leaving the implementation strategy to higher-level service programs. When new technologies appear, you only need to add or modify the service program, instead of having to modify the kernel design like the traditional AI engine. In a single-core environment, the interrupt running environment is always preemptive, and the task must ensure data consistency in the running space by turning off interrupts. In a dual-core processor environment, the task's instruction flows, and there are two interrupted instruction streams, which pose a threat to the data consistency in the system. Therefore, the data consistency in the dual-core processor environment should be studied in order to provide a stable dual-core processor-based real-time operating system lay the foundation. The user does not need to care which core the task is running on, so as to achieve the best compatibility between the system and the previous software. Of course, not all user software can meet the scheduling conditions of dual-core processors. The prerequisites for transparent scheduling of user tasks on dual-core processors should be given to lay the foundation for the transition of user software to dual-core processors.

The core of the operating system is the code running in the privileged state of the processor. Only this code can access all hardware resources without restriction. The kernel provides the abstract conditions of the hardware processor and external devices for the upper application to access. With the increase of the size of the kernel, all types of hardware, that is, external devices, also increase, the core code becomes complex and redundant, and microkernel is the opposite. The security of the microkernel operating system is an important foundation of the entire information security. Chen et al. believes that in the industrial security system, the industrial control terminal is at the operating system level and lacks a complete security system to adapt to the security situation under the new situation. In order to effectively solve the above problems, he believes that the research on the operating system of self-control security technology is particularly necessary. In the industrial operating system security, the integrity of the operating system security kernel is an important guarantee. Based on NARIsecOS, he proposed a kernel integrity protection scheme and carried out formal verification of part of the work [1]. Jomaa et al. believes that the memory manager is a key part of the operating system kernel. The kernel usually ensures memory

isolation through hardware called a memory management unit (MMU). However, MMU itself does not provide memory isolation. It is just a tool that the kernel can use to ensure this attribute. He showed how to use proof assistants such as Coq to model the hardware architecture with MMU and the microkernel abstract model that supports preemptive scheduling and memory management [2]. Cheng et al. believes that with the widespread use of multicore processors, operating systems are facing new challenges in scalability. The lock mechanism used by the current monolithic operating system makes some key modules encounter performance bottlenecks caused by lock contention. The contention in Linux process lifecycle management makes it difficult to scale as the number of cores increases in process creation, execution, and termination (CET). Unlike the centralized approach of Linux, he proposed a distributed process management model (DPMM) based on a microkernel for better scalability. Kernel functions are distributed to messaging servers running on different kernels. DPMM splits the shared data including the process control block (PCB), page pool, and buffer cache into these collaborative servers. Actions including PCB creation/delete, page allocation/release, executable loading, and scheduling can be parallelized without locking [3]. Schwahn et al. believes that in many modern operating systems (OS), there is no isolation between different kernel components; that is, the failure of one component will affect the entire kernel. Although microkernel operating systems introduce address space separation for most of the operating system, their improved fault isolation comes at the expense of performance. Despite major improvements in modern microkernels, monolithic operating systems like Linux are still popular in many systems. These methods completely rely on static code analysis to determine which codes to isolate, while ignoring dynamic properties such as call frequency. He suggested using runtime data to enhance static code analysis to better estimate the dynamic properties of common case operations. We evaluate the impact of runtime data on deciding which codes to isolate and the impact of this decision on the performance of such "microkernelized" systems [4].

The kernel provides the abstract contents of the hardware processor and external devices for the upper application to access. With the increase of the size of the kernel, all types of hardware, namely, external devices, increase, and the core code becomes complex and redundant. This kind of kernel is called single kernel or macrokernel, while microkernel is the opposite; microkernel is the smallest software that can realize the mechanism required by the operating system. In order to improve the scalability of the AI streaming engine, the concept of microkernel is introduced. The core idea is to reduce the coupling of components and avoid the mixing of internal algorithms and external implementations of the engine. The hardware architecture of dual-core processor is analyzed, and the hardware abstraction of dual-core processor is obtained by analyzing the hardware architecture of dual-core embedded processor MPC8641D and XRL508. Its independent access memory channel provides hardware for the independence of subsequent instruction streams. Its CACHESNOOPING mechanism provides hardware support for the data consistency of dual-core processor CACHE, which saves the inefficient CACHE operation in software.

2. How to Use the Microkernel Operating System

The development of dual-core embedded processors is getting more and more attention. CPU chip manufacturers such as freescale will launch the MPC8641D chip in the second half of 2006, and Intel's dual-core processor Intel® has been commercialized to the market: Centrino®Duo (note: due to power consumption, it is mainly aimed at the PC, mobile PC, and central office server markets and is not widely used in embedded), RMI has launched XLR series multicore multithreaded processors and successfully applied to Cisco's high-end routers.

2.1. Spatiotemporal Data Model. With the progress of science and technology, human exploration of the world has become more and more in-depth and far away. In the process of this exploration, the scope of information acquisition has gradually expanded from a local area on the ground to the sky, the world, even beyond the earth, and even the whole universe. The development of spatiotemporal data acquisition technologies such as global positioning system (GPS), digital communication technology, and remote sensing technology provides strong support for human exploration [5–7]. The following problem is how to use the collected data to change the world and benefit mankind. Spatiotemporal data refers to the spatial data with time elements and changing with time, which is a way to describe the information of ground elements in the earth's environment [8]. When describing a spatiotemporal process or spatiotemporal object, we need to start with three "*W*" dimensions, which are "what," "where," and "when." These three dimensions are the three elements of spatiotemporal object: what represents the attribute dimension of spatiotemporal object, which is used to describe its own attributes and characteristics, where represents the spatial dimension of spatiotemporal objects, which is used to deal with the geographical location and spatial characteristics of objects, and when is the time dimension of spatiotemporal objects, indicating the time, time period, or time characteristics of spatiotemporal phenomena [9, 10]. A typical spatiotemporal triple model used to describe spatiotemporal objects or phenomena is shown in Figure 1:

2.2. Data Sharing Security in the Cloud Environment

2.2.1. Symmetric Encryption Scheme. Compared with the public key encryption algorithm, the symmetric encryption algorithm can provide encryption and authentication but lack the function of signature, which reduces the scope of use. Symmetric encryption algorithms widely used in computer private network systems include DES and IDEA. AES, advocated by the National Bureau of Standards, is about to replace DES as the new standard.

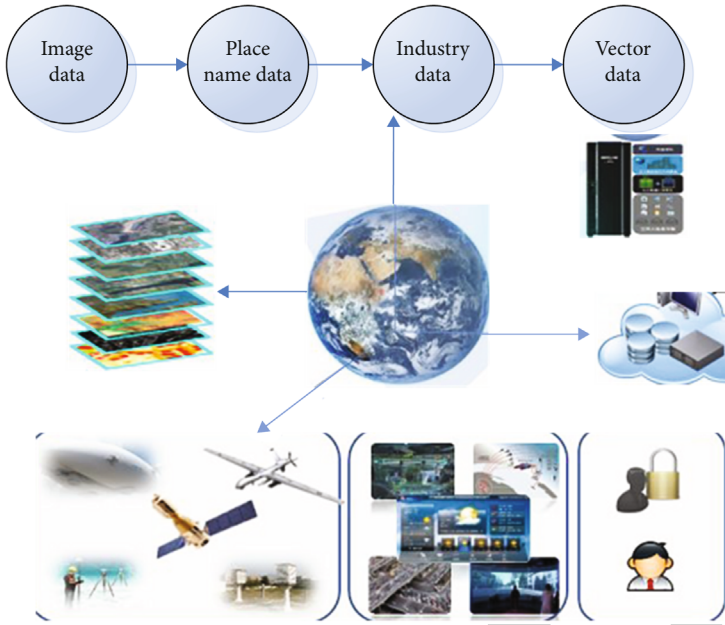

Figure 1: Spatiotemporal data model.

The symmetric encryption algorithm is a traditional encryption algorithm that uses the same key for encryption and decryption. The AES symmetric encryption algorithm includes plaintext, key, AES encryption function, code text and AES decryption function, set plaintext as m, key as N, encryption function as F, ciphertext as B, and decryption function as D:

$$B = F(M, N), \tag{1}$$

$$M = D(F, N). \tag{2}$$

Schnorr proposed a signature scheme, which can be regarded as a variant of the signature scheme. Schnorr Signature Scheme directly integrates hash function into the signature category, and its security is based on the difficulty of solving discrete logarithm. The scheme has the advantages of small amount of computation, fast speed, ability to detect and prevent forgery signature, small communication flow, high security, low cost, good practicability, and other advantages [11]. Therefore, its signature scheme has a wide application prospect:

$$F = h^i, \tag{3}$$

$$d = m + xh(N, M). \tag{4}$$

2.2.2. Presharing Process. Users use an encryption algorithm with verifiable public key to encrypt their services. The key used is the public key of B. the encryption process is abbreviated as follows:

$$H = \text{Engcry}(f, \text{PB}). \tag{5}$$

The signature process of decryption operator f is as follows:

$$s = m + zH(f, R). \tag{6}$$

2.2.3. Share. If you want to share the information with the legitimate users in the authorization set, you can set the number of users as m. A sends the access request of M user decryption operators in the authorization set to the user management center. After verifying the validity of a's identity, the user management center returns the ciphertext and signature of the required user decryption operator [12]. After receiving the M group H, a decrypts h with private key K to get f and verifies its validity with signature:

$$f = \text{Dercy}(C, H), \tag{7}$$

$$f = \text{Ry}^H. \tag{8}$$

In user upload, the data upload process is performed by the data owner. The specific process is reflected in the data sharing process. To complete the data sharing for the new users, the data owner h only needs to send the access request of the user decryption operator to the user management center. After receiving the ciphertext and signature of the new user decryption operator, the process in the data sharing process diagram is updated [13–15]. Upload the updated ciphertext to the cloud, replace the original corresponding ciphertext, and add the new user to the ACL. You need to receive the decryption operator ciphertext and signature of the authorized user returned by the user management center:

$$H = \text{Decry}(v, L). \tag{9}$$

2.3. Ubiquitous Intelligent Cloud Computing. In the rapid

spread of market informatization and digitization, domestic and foreign enterprises have more and more extensive grasp and application of future intelligence, and information manufacturing has become a new mode of enterprise operation and management [16–19]. With the rapid rise and popularization of cloud computing technology, cloud storage, as a basic function of cloud computing, provides data owners with storage space and convenient data access services. According to the storage space, data is stored in the cloud, which is easy to share with other users [20, 21]. But when data owners store data in the cloud, they will consider data security. The ideal solution is to share data safely. In order to reduce the cost of purchasing and maintaining hardware and software equipment, data owners can rent storage space for cloud computing service providers according to their needs. For data security, data is usually encrypted and stored in the cloud storage center; in many practical applications, data owners must share data with specific users [22]. In this process, in addition to the authorized specific users, as a storage function, the cloud and illegal users cannot access the data, as shown in Figure 2.

In this era of knowledge explosion and strong innovation, cloud platform sharing technology has been more and more widely used, and its impact on the traditional supply chain system needs to be analyzed. Since the concept of Internet of things was proposed, the existing research results on the impact of manufacturing capacity sharing on the supply chain have not considered the random volatility of shared manufacturing capacity [23, 24].

2.4. Paillier Homomorphic Encryption. Unlike most real-time embedded systems and most network control systems, the Paillier homomorphic encryption algorithm also takes into account the three elements of computing power, communication efficiency, and control technology and is functionally real-time. It is also highly efficient in performance; it can not only ensure network and physical security but also ensure its reliability. Therefore, the Paillier homomorphic encryption algorithm can not only reasonably schedule and utilize the resources of the AI engine but also perceive the AI engine in real-time on a large-scale and complex system and dynamically monitor the complex information with a wide range of locations.

Based on the analysis of the above mentioned Paillier homomorphic encryption algorithm, it can be decrypted by its private key. Then, we can get the result of adding the corresponding index position of D2D user interest vector, because the decrypted result is the result of adding two vectors, which ensures that the privacy of individual interest of single user is protected [25]. After obtaining the above data, SNS can calculate the cosine similarity score of the interests of users and due:

$$\mathrm{CSS}(i,j) = \frac{|k|}{\sqrt{k} \cdot I}. \tag{10}$$

In social networks, people always interact with other users, such as like others or forward other people's blog posts. Such interaction establishes a social relationship between users. This online interaction is digitized, and a formal expression is given. Model the user's online interaction behavior and establish the relationship between users:

$$\mathrm{RCS}(i,j) = \frac{M+N}{U+I}, \tag{11}$$

$$\mathrm{NDS}(i,j) = \frac{|L-K|}{R} \tag{12}$$

Suppose that the plaintext m to be encrypted is encrypted by the following encryption algorithm,

$$E(f) = h^f r^m \bmod m, \tag{13}$$

$$E(f_1) = h^f r_1{}^m \bmod m, \tag{14}$$

$$E(f_1 + f) = h^f (r_1 + r_2)^m \bmod m. \tag{15}$$

Each algorithm has their advantages and disadvantages. The spatiotemporal data model is at our point in time, for each time period to deal with the geographic location and spatial characteristics of the object, the symmetric encryption scheme has a small amount of calculation, fast speed, and high security.. Cloud computing has a large storage capacity, and it can process more data. The homomorphic encryption algorithm is relatively safe for data. When we integrate these algorithms on this platform, we will combine some of their advantages. Together, this is of great significance to data security.

3. Microkernel Operating System Supports Ar, VR, and Other Related Experiments

3.1. VR Technology. Virtual reality is a kind of virtual environment produced by high-tech computer technology, which brings people visual, auditory, and other sensory feelings. With the development of science and technology, virtual reality technology has been widely used in various fields. It can use computer to simulate real and nonreal three-dimensional scene. Virtual reality technology is the highest level of simulation and virtual reality. In virtual simulation, computer hardware, network technology, computer technology, broadband, and other information technology are needed to enable users to realize real-time human-computer interaction and enter virtual reality. The virtual space created by virtual reality technology can reflect the rich imagination of human beings and pay attention to a new vision in the virtual digital world.

3.2. Microkernel and Its Security. The definition of microkernel is that microkernel does not just mean small, "Modularity is the key, size is butaside effect," and modularity is the key to the microkernel architecture. A well-modular OS is a true microkernel OS. The core of the operating system is the code that runs in the privileged state of the processor. Only this code can access all hardware resources without restriction. The kernel provides the abstract contents of the hardware processor and external devices for the upper application to access. With the increase of the size of the kernel,

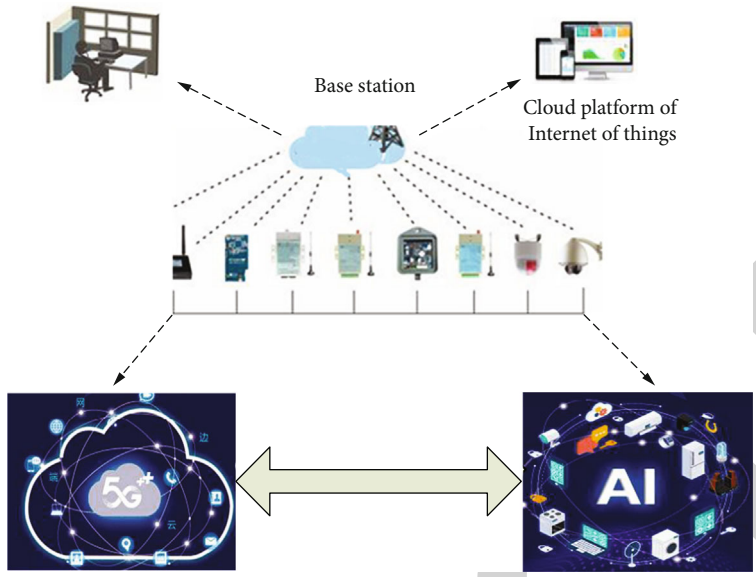

Figure 2: Marlite reservoir.

Table 1: The functions of each program in the system.

Hardware	Program	Kernel
1	Virtual file system	Application
2	Process communication	System services
3	IPC	Device services
4	Virtual memory	Other services
5	Device driver	Process scheduling

all types of hardware, namely, external devices, increase, and the core code becomes complex and redundant. Microkernel is the smallest software that can realize the mechanism required by the operating system, including address management, connection management, and interprocess communication. The functions of each program in the system are shown in Table 1.

3.3. Comparison of Amoeba, Mach, and Chorus. Amoeba provides UNIX emulation through library functions, while Mach provides binary emulation of UNIX, MS-DOS, and other operating systems by capturing the trap of system calls and transferring them to emulation programs in user space. Chorus supports system processes. The core process and user process provide binary emulation of UNIX. Amoeba is based on the object model. It provides the naming and protection mechanism of all objects in the entire system through permissions. The scope of application of permissions in Chorus is much smaller. For Mach, only ports have permissions: Amoeba and Chorus. The thread synchronization is achieved through mutually exclusive semaphores and semaphores, while in Mach, it is through mutually exclusive semaphores and condition variables.

The three systems all run on multiprocessors, but they are different in how to use the processors. The threads of the same process in Amoeba cannot run on different processors, but the processes can run on different processors. Mach passes place. The concept of machine set allows arbitrary allocation of threads, so that threads in the same process are truly parallel: Chorus is similar to Mach, but the user's intervention is less. In Amoeba, the process can have any number of indefinite length segments and can be mapped to the virtual address space. The segment is controlled by permissions and can be read and written by any process with that permission, including remote processes. Amoeba does not support paging on demand; therefore, all segments of the process must be resident in memory at runtime. In Mach, memory is composed of memory objects, which are paged, and pages can be called in and out of memory as needed; so, memory objects do not have to be all in memory. When a page is missing, external pager finds the corresponding page and transfers it into memory. This mechanism supports paging on demand. Pages can be shared between multiple processes in different ways, such as the copy-on-write mechanism of the parent and child processes, but this method is a massively parallel system where the parent and child processes are usually not on the same processor that does not work. (1) Chorus's memory model is very similar to Mach, but Chorus introduces a segment controlled by permission. All three systems support distributed shared memory but the implementation is different. Amoeba supports object-based systems, allowing objects to be shared through the kernel's reliable broadcast mechanism. Mach has a network message server to support page-based systems. Chorus supports page-based systems, and at the same time, there is also a special subsystem COOL running software objects. The current parallel operating system is either a simple runtime system, or a complete UNIX, or a microkernel. The microkernel combines the advantages of the first two systems and will increasingly play a leading role in the design of parallel operating systems in the future.

Three different microkernel systems are analyzed and compared to describe the structure of the microkernel operating system. It is worth noting that although Amoeba,

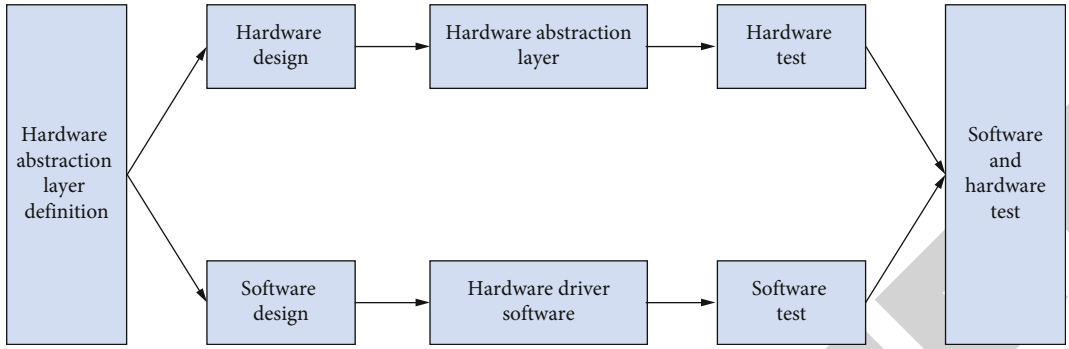

Figure 3: Embedded product development process based on hardware abstraction layer technology.

Table 2: Shared memory identification bit.

Serial number	Memory location	Identification information
1	0 × 00-0 × 03	Length of transferred content
2	0 × 04-0 × 07	Shared memory status
3	0 × 08-0 × 0b	Shared memory ID
4	0 × 0c-0 × 13	Error information of operation
5	0 × 14-0 × 17	Reserved fields

Table 3: The algorithm success rate test.

Largest card	Success rate
2048	100%
4096	100%
8192	100%
16384	95%
32768	33%

Mach, and Chorus were independently developed by institutions in three different countries, they are very similar. This also explains from another side that the microkernel-based operating system in the design of distributed computers and parallel computers will be the future development direction. Of course, the specific structure of the microkernel will follow continue to develop and improve with the development of technology.

3.4. Embedded Product Development. The hardware abstraction layer can be divided into two parts: the implementation layer and the interface layer. The yellow interface is the channel for the operating system to call the hardware abstraction layer, which provides a clear and standardized interface for other operating system modules, In addition to the implementation of the interface, it also includes some other functions. The implementation of different platforms may be completely different. This migration method does not require personnel to understand the specific implementation of the operating system nor does it require software designers to consider the impact of changes in the hardware platform; in this way, designers can focus on their own professional fields and shorten the development cycle, as shown in Figure 3:

4. Experimental Analysis of Embedded Ai Engine Design of the Microkernel Operating System

4.1. Server User Mode Program Dynamic. They constantly monitor whether there is a request in the shared memory bound to themselves, and if there is one, process it accordingly. After creating a subthread, the main thread goes to sleep and listens to SIGINT signal. SIGINT signal is a common signal in the operating system, which is usually sent through ctrl-c of the command line, as shown in Table 2:

The shared memory ID is the most important information to identify the shared memory. It is written when the shared memory is created, and it needs to be checked whether the ID is wrong when the memory is remapped or recycled. Therefore, we use the 4th to 7th bytes of the shared memory to store the length of the request content or result.

The way to test the success rate is relatively simple. Run the Paillier homomorphic encryption algorithm a hundred times, and whenever the game is executed to obtain 2048 cards, the game is still in progress, and the Paillier homomorphic encryption algorithm is considered to have won.

In the test, it is found that because our algorithm has a very high execution winning rate, the probability of obtaining 2048 has reached 100%. Therefore, using 2048 as the game's game victory criterion cannot measure the superiority of the algorithm. So, another 100 trials were performed. This trial will record the maximum card obtained when the algorithm is executed until the end of the game. The algorithm success rate test is shown in Table 3.

5. Conclusion

This paper mainly studies the design of embedded AI engine based on the microkernel operating system, extracts the security requirements of the operating system, designs and implements the operating system from the perspective of formal verification, and considers the verification problem under the background of RTOS development, so as to avoid using too many complex data structures and algorithms in the system design and reduce the difficulty of experimental verification. Secondly, the influence of the dual-core processor environment on the real-time operating system kernel is analyzed. Data consistency is the key content in the kernel

design. The CACHE operation of the dual-core processor on the hardware only ensures the data consistency in the CACHE. In the two independent instruction streams of the real-time processor, the relative independence of the task space and the reentrancy of the kernel operation are needed to ensure the data consistency of the real-time system in the dual-core processor environment. In this paper, we use spatiotemporal data model, data sharing security in cloud environment, symmetric encryption scheme, and Paillier homomorphic encryption method to study the design of embedded AI engine based on the microkernel operating system. Three core function modules, initialization, process grinding, and interrupt processing, are extracted from the microkernel operating system to form the formal verification area of the operating system, the system syntax and semantics of related rights are separated, and the main rationalization rules are described. This article uses spatiotemporal data model, data in cloud environment, shared security, symmetric encryption scheme, and Paillier homomorphic encryption method that have studied the embedded AI engine based on the microkernel operating system, designs and integrates these models and data security to create a highly secure platform, extracts the three core functional modules of initialization, and processes polishing and interrupt processing from the microkernel operating system to form an operating system. The formal verification area separates the system syntax and semantics of related permissions and describes the main rationalization rules. Smarter helps us deal with the current era of data flooding in this new era and provides convenience for us to extract information that we know. Gradually remove the application of many global variables in the kernel code, reduce the degradation of kernel performance caused by the application of spin locks, and improve the reentrancy of the kernel. The interrupt arbitration part should form a specification to lay a foundation for the promotion of future dual-core processors, avoid developing a different interrupt arbitration mechanism for each dual-core processor, and improve the reusability of the core.

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

[1] J. Chen, W. Liu, X. Lv, D. Ji, J. Shi, and B. Li, "Research on microkernel-based power dedicated secure operating system," *Journal of signal processing systems for signal, image, and video technology*, vol. 91, no. 10, pp. 1127–1136, 2019.

[2] N. Jomaa, D. Nowak, G. Grimaud, and S. Hym, "Formal proof of dynamic memory isolation based on MMU," *Science of Computer Programming*, vol. 162, no. SEP.15, pp. 76–92, 2018.

[3] Z. Cheng, R. Zhu, P. Chen, and H. Huang, "A distributed process management model for better scalability on multicore platform," *Chinese Journal of Electronics*, vol. 26, no. 2, pp. 263–270, 2017.

[4] O. Schwahn, S. Winter, N. Coppik, and N. Suri, "How to fillet a penguin: runtime data driven partitioning of Linux code," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 945–958, 2018.

[5] Y. Klimiankou, "Design and implementation of port-mapped IO management subsystem and kernel Interface for true microkernels on IA-32 processors," *Programming and Computer Software*, vol. 45, no. 6, pp. 319–323, 2019.

[6] Y.-H. Hsieh and Y.-T. Lo, "Understanding customer motivation to share information in social commerce," *Journal of Organizational and End User Computing*, vol. 33, no. 6, pp. 1–26, 2021.

[7] X. Wang, J. Liu, O. I. Khalaf, and Z. Liu, "Remote sensing monitoring method based on BDS-based maritime joint positioning model," *CMES-Computer Modeling in Engineering & Sciences*, vol. 127, no. 2, pp. 801–818, 2021.

[8] C. Dragomir, L. Mogosanu, M. Carabas, R. Deaconescu, and N. Tapus, "μQC: a property-based testing framework for L4 microkernels," *International Journal of Critical Computer Based Systems*, vol. 8, no. 1, pp. 1–24, 2018.

[9] R. Debab and W. K. Hidouci, "Boosting the cloud meta-operating system with heterogeneous kernels. A novel approach based on containers and microservices," *Journal of Engineering Science and Technology Review*, vol. 11, no. 1, pp. 103–108, 2018.

[10] C. Kyrkou, C. S. Bouganis, T. Theocharides, and M. M. Polycarpou, "Embedded hardware-efficient real-time classification with Cascade support vector machines," *IEEE Transactions on Neural Networks & Learning Systems*, vol. 27, no. 1, pp. 99–112, 2016.

[11] W. Keegan, "What's the difference between separation kernel hypervisor and microkernel?," *Electronic Design*, vol. 65, no. 4, pp. 36–45, 2017.

[12] A. I. Nakamura, N. Noriyuki, N. Hideki et al., "Image processor and image processing method," *Procedia Cirp*, vol. 29, no. 5, pp. 56–61, 2017.

[13] M. R. Bhatt, H. Mehta, and S. H. Buch, "Automatic die design and fatigue life prediction of forming die using AI technique expert system," *International Journal of Computer Sciences and Engineering*, vol. 6, no. 4, pp. 20–30, 2018.

[14] M. Zaher, A. Shehab, M. Elhoseny, and F. F. Farahat, "Unsupervised model for detecting plagiarism in Internet-based handwritten Arabic documents," *Journal of Organizational and End User Computing*, vol. 32, no. 2, pp. 42–66, 2020.

[15] L. Ogiela, M. R. Ogiela, and H. Ko, "Intelligent data management and security in cloud computing," *Sensors*, vol. 20, no. 12, p. 3458, 2020.

[16] D. Grzonka, A. Jakóbik, J. Kołodziej, and S. Pllana, "Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security," *Future Generation Computer Systems*, vol. 86, no. SEP., pp. 1106–1117, 2018.

[17] V. Age and J. P. Doomernik, "Artificial intelligence potential in power distribution system planning," *Cired Open Access Proceedings Journal*, vol. 2017, no. 1, pp. 2115–2117, 2018.

[18] O. Wisesa, A. Andriansyah, and O. Khalaf, "Prediction analysis for business to business (B2B) sales of telecommunication

services using machine learning techniques," *Majlesi Journal of Electrical Engineering*, vol. 14, no. 4, pp. 145–153, 2020.

[19] O. Wisesa, A. Adriansyah, and O. I. Khalaf, "Prediction analysis sales for corporate services telecommunications company using gradient boost algorithm," in *2nd International Conference on Broadband Communications, Wireless Sensors and Powering. BCWSP 2020*, pp. 101–106, Yogyakarta, Indonesia, 2020.

[20] "Rethinking memory management in modern operating system: horizontal, vertical or random?," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1921–1935, 2016.

[21] O. I. Khalaf and G. M. Abdulsahib, "Optimized dynamic storage of data (ODSD) in IoT based on blockchain for wireless sensor networks," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2858–2873, 2021.

[22] Y. Sela, "Mobile telephone gateway apparatus, communication system, and gateway operating system," *Gondwana Research*, vol. 23, no. 2, pp. 661–665, 2017.

[23] N. Choi, D. Kim, S. J. Lee, and Y. Yi, "A fog operating system for user-oriented IoT services: challenges and research directions," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 44–51, 2017.

[24] A. Siahaan, "Securing short message service using Vernam cipher in android operating system," *International Journal of Mobile Computing & Multimedia Communications*, vol. 3, no. 4, p. 2394, 2016.

[25] H. Hu, Z. Wang, G. Cheng, and J. Wu, "MNOS: a mimic network operating system for software defined networks," *IET Information Security*, vol. 11, no. 6, pp. 345–355, 2017.