

面向新型计算范式的外核操作系统综述

王立新^{1,2}, 陈建森¹, 王子哲^{1,2}, 王俊超¹, 秦晓军²

(1.信息工程大学,河南 郑州 450001; 2.先进计算与智能工程(国家级)重点实验室,江苏 无锡 214083)

摘要:面对云计算、边缘计算和量子计算等新型计算范式对低时延、高隔离、异构适配和可裁剪软件栈的需求,传统宏内核在固定抽象、共享内核路径和通用驱动栈方面逐渐暴露出难以兼顾性能与灵活性的局限。外核操作系统基于资源保护与管理分离原则,为突破这一瓶颈提供新路径。首先,梳理外核的基本思想、概念边界、架构组件和运行机制,从安全隔离、性能路径和生态兼容性3个维度,对外核、宏内核和微内核进行比较。其次,针对云计算和边缘计算场景,剖析外核在低延迟网络和专用加速中的适配价值及面临的开发复杂性挑战。最后,为应对量子计算纳秒级时序控制与噪声隔离的特殊需求,提出一种量子计算机专用外核架构设想,论证其作为下一代特种操作系统的演进潜力,为外核技术的场景化落地及国产专用操作系统研发提供理论参考。

关键词:操作系统; 外核; 宏内核; 微内核; 量子计算

中图分类号:TP316

文献标识码:A

文章编号:1671-0673(2026)04-0466-07

Survey of Exokernel Operating Systems for New Computing Paradigms

WANG Lixin^{1,2}, CHEN Jiansen¹, WANG Zizhe^{1,2}, WANG Junchao¹, QIN Xiaojun²

(1.Information Engineering University, Zhengzhou 450001, China;

2.Laboratory for Advanced Computing and Intelligence Engineering, Wuxi 214083, China)

Abstract: Faced with the demands of new computing paradigms, such as cloud computing, edge computing, and quantum computing, for low latency, high isolation, heterogeneous adaptation, and tailorable software stacks, traditional monolithic kernels are increasingly revealing limitations in balancing performance with flexibility due to their fixed abstractions, shared kernel paths, and generic driver stacks. Exokernel operating systems, based on the principle of separation of resource protection and management, offer a new pathway to break through this bottleneck. Firstly, the basic ideas, conceptual boundaries, architectural components, and operational mechanisms of exokernels are outlined, comparing them with monolithic and microkernels across 3 dimensions: security isolation, performance paths, and ecosystem compatibility. Secondly, focusing on cloud computing and edge computing scenarios, the adaptation value of exokernels in low-latency networking and specialized acceleration, as well as the challenges of development complexity, are analyzed. Finally, to address the specific requirements of quantum computing for nanosecond-level timing control and noise isolation, an architectural design for a quantum computer-specific exokernel is proposed, demonstrating its evolutionary potential as a next-generation specialized operating system and providing a theoretical reference for the scenario-based implementation of exokernel technology and the research and development of domestic specialized operating systems.

Key words: operating system; exokernel; monolithic kernel; microkernel; quantum computing

操作系统内核作为硬件与应用程序之间的桥梁,其架构设计深刻影响计算机系统的性能、安全性、稳定性和灵活性。传统宏内核(monolithic kernel),以Linux^[1-2]为代表,是一类将进程管理、内存管理、文件系统、网络协议栈与设备驱动等主要系统服务集中在内核态统一实现的操作系统内核架构。这种架构虽能获得较高的内部通信效率,却因高度耦合而导致扩展性不足。微内核^[3-4](micro-kernel),如seL4,是一类仅在内核态保留线程调度、地址空间管理和进程间通信等基本机制,而将文件系统、驱动程序和网络服务等尽可能下移到用户态实现的操作系统内核架构。微内核借助模块化设计提升安全性^[5-7],但频繁的用户态/内核态切换引发显著的性能损耗^[8]。相比之下,外核(exokernel)架构基于“资源保护与管理分离”的哲学,将资源的安全隔离与策略制定解耦,为解决传统架构在灵活性与性能之间的权衡困境提供了全新路径^[9-10]。

随着云计算、边缘计算和量子计算等新型计算范式的兴起,传统操作系统在灵活性、性能表现及资源控制粒度方面遭遇严峻挑战^[11-14]。外核架构及其衍生技术路线,以其“保护归内核,管理归应用”的核心理念,展现出应对多样化场景需求的巨大潜力。云计算要求轻量化部署与资源高效复用,边缘计算强调低延迟响应与异构硬件适配,量子计算则需纳秒级时序控制与基于保真度的作业调度。上述3种新型范式在计算模型、资源供给模式、系统架构及应用目标上均与传统集中式或并行计算存在本质差异,其核心在于实现计算资源供给与使用的根本性变革^[15-16]。

新型计算范式的鲜明特征体现为服务导向、按需分配和资源抽象,并在弹性、分布式、异构协同等维度呈现显著差异,且与特定应用场景深度融合。这一系统性变迁,正驱动操作系统架构由宏内核、微内核向更具灵活性、专一化的外核及其衍生形态持续演进^[17]。

1 外核架构基本思想与架构

1995年,麻省理工学院研究人员提出外核架构设计思想:“保护与管理分离”^[9]。传统内核将资源抽象定义与保护机制紧密耦合:既规定资源的逻辑视图,又承担其安全管控职责。外核设计思想则主张,内核的唯一职责是确保资源在硬件层面的安全隔离与合法访问,而资源的管理策略应

交由应用程序自主决定^[18]。这一思想在当代操作系统研究与实践中产生了一定影响,并演化为多种具体形态,以适配新型计算范式的多元需求。

1.1 架构组件剖析

外核的典型架构如图1所示,其核心构件与运行机制可归纳如下。

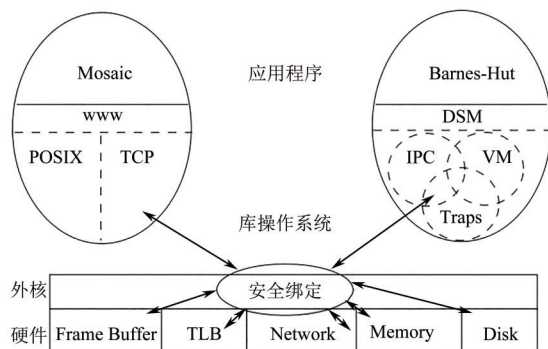


图1 外核架构

1)外核。外核本体为一个极简内核,直接与硬件交互,不提供文件系统、虚拟内存、进程间通信等传统高级抽象,仅负责硬件资源的分配、撤销和保护,确保应用程序之间的隔离。它通过安全绑定、受保护的转移等机制,以极低的开销验证应用程序对资源的访问权限。

2)库操作系统(Library Operating System, LibOS)。为避免各应用重复实现复杂的系统服务,外核架构引入LibOS作为用户空间库,为应用提供传统操作系统抽象^[19]。应用可链接不同LibOS,亦可按需定制乃至重写LibOS,实现资源管理策略的深度优化。LibOS实现了操作系统的高级抽象和服务,如文件系统、网络协议栈、进程和线程模型等。不同的应用程序可以选择或定制不同的LibOS^[20-21]。LibOS作为经典外核架构的核心组件,应被视为一种系统服务的组织范式,并非外核本身。

3)应用程序。它运行在最上层,通过调用LibOS提供的接口来与计算机交互,间接地管理硬件资源。应用程序不再经由固定的系统调用接口被动获取服务,而是直接向内核申请底层硬件资源的控制权,从而依据自身负载特征制定最优管理策略^[9]。

综上,外核架构通过将最小化、可定制的系统软件栈与资源管理权移交给应用层,已成为应对新型计算范式下多样化、专业化需求的关键技术路径^[9-10,13]。

1.2 资源分配与权限控制机制

与宏内核由内核统一管理所有资源不同,外

核的资源分配是去中心化的。应用程序可通过 LibOS 直接向外核请求一块物理内存、一个特定磁盘扇区或一个网络端口的所有权。外核的角色是验证该请求的合法性,并在授权后记录所有权。之后,应用程序可直接操作这些资源,无需再通过内核的抽象层。这种方式有效减少了系统调用的开销和内核介入的次数。

权限控制是外核安全性的关键。外核通过一套严密的机制来跟踪每个资源(如内存页、磁盘

块)的所有权,并确保只有所有者才能访问。当应用程序试图访问一个不属于它的资源时,外核会在硬件级别上阻止这次操作,从而确保系统的整体安全性。

1.3 内核类型对比

在操作系统内核架构的演进中,宏内核、微内核和外核围绕系统服务组织方式与资源抽象程度等核心问题,呈现出不同的设计逻辑与技术特性,可从多维度对比,如表1所示。

表1 内核类型对比

类型	设计哲学	内核大小	抽象层次	灵活性/ 可定制性	理论性能	复杂性 (对于应用开发者)
宏内核	所有服务集成在内核	巨大	高层、固定、强制	低	服务调用快, 但抽象开销固定	低,接口稳定统一
微内核	最小化内核,服务用户化	极小	提供基础通信抽象	中等	IPC 开销大, 可能存在性能瓶颈	中等
外核	保护与管理分离, 抽象用户化	极小	几乎无抽象, 暴露底层硬件	非常高	极高,应用可绕过 抽象层直接优化	高,需要通过 LibOS 管理底层资源

3种内核的差异,本质是“通用性”与“定制化性能”的权衡,而外核“分离资源的保护与管理+用户态定制”的设计思想,也为高性能计算、量子计算等对资源精细化管理要求极高的场景提供了创新思路。

2 面向新型计算范式的外核应用扩展

外核架构的核心价值,在于其能够针对云计算、边缘计算、量子计算等不同新型计算范式的核心需求,通过定制化的技术改进与优化,突破传统内核架构的固有瓶颈,探索外核操作系统在新型计算领域中的应用。

2.1 云计算环境下的外核应用扩展

云计算的核心特征是多用户资源池隔离、按需服务和高吞吐低延迟输入/输出(Input/Output, I/O)。传统宏内核架构在该场景下面临3个核心瓶颈:1)共享内核导致多用户隔离粒度粗糙,安全风险高;2)通用内核抽象带来较大的I/O开销,无法满足数据中心微秒级延迟需求;3)容器与虚拟机镜像体积大,启动速度慢,难以适配毫秒级应用需求。

外核架构凭借极简内核与可定制 LibOS,天然契合云环境的多租户隔离与差异化服务需求^[13]。其最小化的可信计算基(Trusted Computing Base, TCB)显著降低跨租户攻击面。Unikernel 深受外核思想影响,其通过将应用与所需 OS 库编译为单一

轻量镜像,在云原生场景中表现突出。例如,Mirage Unikernel 基于 Xen Hypervisor,可实现毫秒级(≤ 50 ms)启动,适合函数计算与容器快速弹性伸缩场景^[13]。Unikernel 与外核都强调裁剪、专用化和应用导向,但 Unikernel 并不必然遵循“最小保护内核+用户态管理策略”的经典外核结构,因此它更适合被视为受外核/LibOS 思想影响的专用化系统路线。

外核及其衍生技术常作为客户机 OS 运行于 Hypervisor 之上,或以容器镜像形式与 Docker、Kubernetes 等云原生生态集成,实现轻量、高安全的工作负载封装^[22]。

2.2 边缘计算环境下的外核应用扩展

边缘计算将计算能力下沉至网络边缘,靠近数据源,其核心需求是低延迟响应、资源受限环境下的高效运行和高可靠性^[23]。传统宏内核架构在该场景下面临3个核心瓶颈:1)系统镜像体积大,资源占用高,无法适配边缘终端的有限存储和内存资源;2)内核调度抖动大,无法满足工业控制等场景的硬实时需求;3)异构硬件驱动耦合在内核中,扩展与适配难度大,无法适配边缘场景中多样化的异构硬件。

外核架构的轻量性、低开销与可定制性,高度契合边缘设备的资源约束。通过裁剪冗余系统服务,基于外核思想的 LibOS 能够生成极小的系统镜像,高度适配有限的存储空间。此外,其直接控制硬件的潜力,使其能够满足工业控制等对实时性

要求较高的场景需求^[24-28]。

外核架构的通用优势(如极低的服务开销和高灵活性)可以有效支撑边缘设备的高效任务处理。例如,在量子传感器或量子通信设备等前沿领域,外核的直接硬件控制能力可显著减少中间层延迟。

外核衍生的模块化、可定制架构能够为特定边缘硬件与场景构建专用运行时环境,规避通用OS的冗余开销,精准响应边缘计算对轻量、实时、异构与分布式协同的综合需求。

2.3 量子计算环境下的外核应用扩展

量子计算遵循量子力学规律进行信息处理,超导量子计算机是目前量子计算机的主力候选路线之一,其硬件系统由极低温量子芯片与室温经典测控系统构成。这种“量子+经典”的混合架构,对底层经典测控操作系统提出了3个极端需求。1)纳秒级的时序控制精度:要求操作系统提供较低的执行时延,消除不可预测的系统开销;2)极低延迟的硬件访问能力:要求实现对现场可编程门阵列(Field-Programmable Gate Array, FPGA)、模数转换器(Analog-to-Digital Converter, ADC)/数模转换器(Digital-to-Analog Converter, DAC)等测控硬件的极低开销访问;3)极致的噪声与干扰隔离:要求控制进程与系统其他无关进程完全隔离,防止电磁噪声或调度抖动破坏脆弱的量子态^[29-35]。

外核“分离”哲学为此提供可行的解决路径。首先,可为量子控制任务专设LibOS,仅保留必需调度、内存管理与硬件驱动模块,实现高度精简与优化;其次,极简外核仅负责将物理资源(特定内存区、硬件端口)安全分配给量子控制LibOS,之后不再干预,最大限度抑制内核活动引发的时序抖动与噪声;最后,依托安全绑定机制,量子控制应用可获得对FPGA等硬件的直接、受保护访问权限,实现超低延迟控制^[36]。

3 外核架构分析与现实挑战

外核架构的性能优势源于其“将控制权交还应用”的根本设计哲学。通过允许深度定制,它使得操作系统不再是性能瓶颈,而是一个可被精确塑造、以有效适配上层应用和底层硬件特性的基座。

3.1 外核操作系统的主要形态

自经典外核概念提出以来,该架构思想经历了从学术原型到工业级系统的演进,主要出现了

以下两类外核。

1)经典外核(如Xok/Aegis)。Xok/Aegis是麻省理工学院研究人员在20世纪90年代中期开发的经典外核操作系统的原型实现。它是外核架构思想从理论走向实践的第1个完整系统,其中:Aegis是外核操作系统的内核部分;Xok是运行在用户态的软件库LibOS^[37]。特点:严格遵循“保护与管理分离”原则。内核仅负责资源的安全分配与隔离(如物理页帧、磁盘块),完全不提供任何高级抽象(如文件、套接字)。所有管理策略均由用户态的库操作系统(LibOS)实现。优势:提供了极致的灵活性和控制粒度,允许应用完全定制资源管理算法。局限:开发门槛极高,LibOS需从头实现所有系统服务,且早期原型缺乏对现代硬件的原生支持。

2)数据路径外核(Demikernel)。Demikernel是由微软研究院研究人员于2021年提出的现代外核架构代表,专为微秒级数据中心网络场景设计^[19]。它代表了一种面向现代数据中心数据路径的“外核化”演进形态,属于吸收外核思想的数据路径操作系统架构,而不是经典外核的直接复现。特点:面向现代数据中心和高性能网络场景的演进形态。它保留了传统内核处理控制流,但将高频、低延迟的数据流处理移至用户态LibOS,利用内核旁路技术直接访问智能网卡。优势:在兼容现有生态的同时,实现了微秒级甚至纳秒级的I/O延迟,平衡了性能与开发效率。局限:Demikernel仅实现了可移植操作系统接口(Portable Operating System Interface, POSIX)的一个严格子集,主要集中在套接字操作,如socket、bind、connect、send、recv等。它不支持复杂的文件系统操作、进程间通信、信号处理或多线程同步原语。这意味着现有的复杂应用程序无法直接迁移,必须经过大幅度的代码重构才能适配Demikernel接口。

Demikernel体现了外核架构的现代演进方向——“控制面/数据面分离”。它保留了传统内核处理复杂控制流(如连接建立),而将关键数据路径卸载至用户态LibOS。这种混合架构比纯粹的经典外核更具落地可行性,是外核思想向工业界渗透的典型范例。

3.2 理论性能优势的深层机理分析

外核架构之所以能突破传统宏内核的性能瓶颈,并非单纯依靠代码精简,而是源于其从根本上重构了操作系统与应用程序的交互模式。其理论性能优势主要体现在以下3个核心维度。

1)消除上下文切换与特权级转换开销。在传统宏内核中,应用程序发起 I/O 请求需经历“用户态→内核态→用户态”的两次特权级切换。对于高频短耗时的操作(如微秒级网络包处理),这种切换带来的流水线冲刷、页表缓存(Translation Lookaside Buffer, TLB)失效及寄存器保存恢复开销,往往超过了实际业务逻辑的执行时间^[38]。外核架构通过 LibOS 将资源管理逻辑上移至用户态。一旦外核完成初始资源授权,后续的绝大多数操作(如数据包收发、内存读写)均在用户态直接完成,彻底消除了频繁的系统调用开销。理论上,这使得 CPU 周期可高效用于有效计算,而不是浪费在内核调度方面。

2)实现零拷贝与直接内存访问。传统内核为了通用性和安全性,强制引入多层数据抽象。数据在内核空间与用户空间之间往往需要多次拷贝,不仅消耗宝贵的内存带宽,还导致 CPU 缓存命中率下降^[39]。外核允许应用程序直接感知并管理物理资源。LibOS 可直接将网卡 DMA 缓冲区映射到应用地址空间,数据从网络到达至应用处理无需经过内核缓冲区拷贝。

3)高效调度与噪声隔离。通用操作系统中的中断处理、后台守护进程、垃圾回收等操作会引入不可预测的时序抖动,这对于实时控制和低延迟场景是致命的。Demikernel 等现代外核衍生架构在高负载下采用用户态轮询机制替代硬件中断,避免了中断上下文切换带来的延迟不确定性。外核可将特定 CPU 核心、内存大页及硬件队列独占绑定给关键应用。这种硬隔离机制确保了关键任务不受系统中其他组件的干扰,从而提供极低的延迟保障。

3.3 面向新型计算范式的性能评估与分析

外核操作系统在理论上具有显著优势。尽管纯外核系统尚未经过大规模商用测试,但文献[19]对基于外核架构思想衍生的 Demikernel 系统进行了测试。实验使用了 5 台服务器,每台均配备双路 20 核 Xeon Silver 4114 2.2 GHz CPU,通过 Mellanox CX-5 100 Gbps 网卡和 Arista 7060CX 100 Gbps 交换机互联(交换机最小延迟 450 ns)。重复 5 次实验并报告平均值,具体性能测试对比数据如表 2 所示^[19]。Demikernel 继承了外核架构的核心思想,即将操作系统内核的功能拆分为控制路径和数据路径,允许应用程序更直接、更安全地访问硬件资源。主要包含以下优势特征。1)极低延迟:通过

避免内核切换、零拷贝 I/O 和用户态轮询,实现纳秒级 I/O 处理。2)高吞吐量:优化内存访问与 I/O 调度,提升数据处理效率。3)设备无关性:支持多种内核旁路设备,提供统一应用程序编程接口(Application Programming Interface, API),增强系统可移植性。4)适用于现代数据中心:适合微秒级低延迟要求的场景,如高性能计算、实时存储、云服务等。

表 2 性能测试对比

类型	TCP 数据包 处理延迟	UDP 延迟	分布式事务 处理延迟
Demikernel	53 ns	13.9 μs	200 μs
Linux	10 μs 级	24.9 μs	500 μs
提升比例	提升上百倍	提升 1.79 倍	提升 2.5 倍

3.4 现实挑战

尽管性能优势显著,外核及其衍生技术走向大规模应用,主要面临以下挑战。1)开发复杂性激增。外核将资源管理的复杂性转移给了应用程序开发者。虽然 LibOS 可以缓解这一问题,但设计、调试和维护一个高效稳定的 LibOS 极其困难,远超普通应用程序开发的范畴。2)兼容性与标准化问题。在一个外核系统上,可能同时运行着链接不同 LibOS 的应用程序。这些程序如何高效、安全地共享数据和通信,成为一个新的难题。缺乏统一的应用二进制接口(Application Binary Interface, ABI)也阻碍了软件的移植和分发。3)生态系统的惯性。操作系统领域存在显著的马太效应, Linux 等成熟的操作系统已经拥有完整的生态系统,包括海量的应用程序、驱动程序、开发工具和专业人才。任何新的操作系统架构想要与之竞争,都面临着巨大的迁移成本和生态壁垒。

4 面向新型计算范式的系统架构展望

随着新型计算范式的不断演进,对操作系统提出了极致性能、强隔离与场景化适配能力的需求,外核架构的核心设计哲学有可能打破马太效应,迎来发展机遇。本文系统性地探讨了在云计算与边缘计算深度融合的背景下,操作系统技术的未来演进路径,并重点深入剖析了面向量子计算的外核架构设计。

4.1 新型计算场景下的外核架构演进展望

4.1.1 云计算场景的演进展望

在云计算场景中,外核技术将向硬件协同型

模块化方向演进。它将吸收 Unikernel 的按需裁剪理念,但克服其静态链接的缺陷,通过动态 LibOS 容器化技术,实现与 Kubernetes 的原生集成,构建“容器+Unikernel”的统一运行时,兼顾容器的生态兼容性与 Unikernel 的轻量化、高性能优势。同时,外核技术将与硬件深度协同,将外核的资源隔离与安全绑定机制卸载至硬件实现,进一步提升 I/O 性能与隔离能力,构建“软硬件协同”的云原生外核架构。此外,外核技术将通过标准化的 LibOS 组件库,降低开发门槛,解决生态兼容性问题,实现云原生场景的规模化应用。

4.1.2 边缘计算场景的演进展望

在边缘计算与工业物联网场景中,外核技术将朝着功能安全与信息安全融合的硬实时方向演进,成为工业 4.0 时代边缘操作系统的核心架构。未来,外核架构将构建满足工业级功能安全要求的专用外核系统。同时,针对边缘场景的异构硬件碎片化问题,将构建标准化的硬件抽象 LibOS 组件,实现对各类传感器、控制器、AI 加速器的无缝适配。此外,将构建“云一边一端”一体化的外核架构体系,实现从云端数据中心到边缘终端的统一资源调度与安全管控,推动工业互联网的全面发展。

4.2 基于外核架构的量子操作系统

量子计算机处于前沿探索阶段,相应的操作系统尚未成熟,本文主要研究超导量子计算机上的操作系统。除第 2.3 节所述环境对操作系统的 3 个需求外,超导量子计算机系统还需要实现跨层协同优化,即需要编译器、运行时、操作系统乃至硬件协同设计,以优化从高级量子指令到硬件脉冲的整个路径。

针对量子计算场景的需求,本文提出“量子计算机专用外核系统”的架构设想,主要包含以下 4 个方面。1)量子操作系统外核:1 个极小内核,仅负责物理内存、CPU 核心、PCIe 通道等资源在不同量子实验任务间的硬隔离与安全分配。它可能运行在专用的控制处理器上。2)量子操作系统库:用户态库,为量子编译器生成的脉冲程序提供服务。它内置高效调度器、无锁内存管理器,并直接管理 FPGA 测控设备内存,实现从脉冲描述到硬件信号快速切换。3)混合调度模型:采用空间分区+时间分区。将 CPU 核心划分为实时域(独占运行量子控制任务)和服务域(运行监控、网络等非实时任务),两域之间通过严格的内存和总线隔离进

行通信。4)硬件—软件协同设计:操作系统与 FPGA 固件协同,将最关键的时序控制逻辑固化在硬件中,软件只负责高层配置和序列编排,将不确定性降至最低。

这一路径继承了外核“保护与管理分离”的精髓,将资源管理的最高权限交给最了解量子程序需求的“量子运行时”,有望从根本上解决通用操作系统带来的不确定性问题。

5 结束语

尽管纯粹的外核操作系统并未成为通用计算场景的主流,但其“最小化内核抽象,将硬件控制权还给上层应用”的核心思想,深刻影响了过去 30 年操作系统领域的研究与发展,并以高性能 I/O 外核、机密计算 LibOS 等多种形态,在现代新型计算场景中实现了全面复兴。

本文系统梳理了外核技术从经典原型到当代衍生形态的完整演进脉络,明确了各阶段的核心技术成果与学术支撑;深入阐述了外核架构的核心设计思想、基础组件与技术特征,完成了与传统内核架构的多维度对比;针对云计算、边缘计算、量子计算等新型计算范式,构建了外核技术的场景化适配体系,深度剖析了其核心技术改进与优化路径;在此基础上,深入解析了外核架构的理论性能优势,梳理了主流外核系统的分类与特征,明确了性能评估基准的选择逻辑;最后全面展望了外核技术在各类新型计算场景中的演进方向,创新性地完善了“量子计算机专用外核”架构设想。

未来,随着硬件隔离技术的不断发展,以及量子计算、机密计算、边缘计算等对低延迟、高安全、细粒度资源管控要求严苛的场景不断兴起,外核架构的设计哲学将迎来更广阔的发展空间。外核技术不仅将在云计算、边缘计算等场景中以新的形态焕发生机,更有望成为量子计算机等专用领域操作系统的核心架构基础,为下一代操作系统的研发提供全新的技术路径。

参考文献:

- [1] Love R. Linux kernel development [M]. 3rd ed. Upper Saddle River, USA: Addison-Wesley, 2010: 1-10.
- [2] RUSLING D A. The linux kernel [EB/OL]. [2025-12-03]. <https://tldp.org/LDP/tlk/tlk.html>.
- [3] Accetta M, Baron R, Bolosky W, et al. Mach: a new kernel foundation for UNIX development [C]//Proceedings of

- the summer 1986 USENIX conference. Berkeley, USA: USENIX Association, 1986:93-113.
- [4] Black D L, Golub D B, Julin D P, et al. Microkernel operating system architecture and Mach[C]//Proceedings of the USENIX workshop on micro-kernels and other kernel architectures. Berkeley, USA: USENIX Association, 1992: 11-30.
- [5] Klein G, Elphinstone K, Heiser G, et al. seL4: formal verification of an OS kernel[C]//Proceedings of the ACM SIGOPS 22nd symposium on operating systems principles. New York, USA: ACM, 2009: 207-220.
- [6] Herder J N, Bos H, Gras B, et al. MINIX 3: a highly reliable, self-repairing operating system[J]. ACM SIGOPS Operating Systems Review, 2006,40(3):80-89.
- [7] Zhang L P, Zhao Y W, Li J X. A comprehensive specification and verification of the L4 microkernel API[C]//Proceedings of the 30th international conference tools and algorithms for the construction and analysis of systems. Cham, Switzerland: Springer, 2024:217-234.
- [8] Roch B. Monolithic kernel vs. microkernel[DB/OL]. [2025-12-03].<https://api.semanticscholar.org/CorpusID:17918683>.
- [9] Engler D R, Kaashoek M F, O'Toole J Jr. Exokernel: an operating system architecture for application-level resource management[J]. ACM SIGOPS Operating Systems Review, 1995,29(5):251-266.
- [10] Kaashoek M F, Engler D R, Ganger G R, et al. Application performance and flexibility on exokernel systems[C]//Proceedings of the sixteenth ACM symposium on operating systems principles. New York, USA: ACM, 1997: 52-65.
- [11] Blinowski G, Ojdowska A, Przybyłek A. Monolithic vs. microservice architecture: a performance and scalability evaluation[J]. IEEE Access, 2022,10:20357-20374.
- [12] 姚新磊,张铮.轻量级内核 LWK 特性研究[J].信息工程大学学报,2012,13(1):111-114.
- [13] Madhavapeddy A, Mortier R, Rotsos C, et al. Unikernels: library operating systems for the cloud[J]. ACM SIGARCH Computer Architecture News, 2013,41(1): 461-472.
- [14] 郭玉东,张萍.操作系统透明性研究[J].信息工程大学学报,2009,10(4):550-553.
- [15] Su J S, Zhao B K, Dai Y, et al. Technology trends in large-scale high-efficiency network computing[J]. Frontiers of Information Technology & Electronic Engineering, 2022,23(12):1733-1747.
- [16] Lai X B, Guo Y, He M, et al. A UAV-enabled mobile edge computing paradigm for dependent tasks based on a computing power pool[J]. Frontiers of Information Technology & Electronic Engineering, 2025,26(4):623-645.
- [17] Porter D E, Boyd-Wickizer S, Howell J, et al. Rethinking the library OS from the top down[C]//Proceedings of the sixteenth international conference on architectural support for programming languages and operating systems. New York, USA: ACM, 2011:291-304.
- [18] Ganger G R, Engler D R, Kaashoek M F, et al. Fast and flexible application-level networking on exokernel systems[J]. ACM Transactions on Computer Systems, 2002,20(1):49-83.
- [19] Zhang I, Raybuck A, Patel P, et al. The demikernel datapath OS architecture for microsecond-scale datacenter systems[C]//Proceedings of the ACM SIGOPS 28th symposium on operating systems principles. New York, USA: ACM, 2021:195-211.
- [20] Tazaki H, Nakamura R, Sekiya Y. Library operating system with mainline linux network stack[EB/OL]. [2025-12-03].<https://api.semanticscholar.org/CorpusID:67811936>.
- [21] Hur J, Lee B. GRAMINER: fuzz testing gramine LibOS to harden the trusted computing base[C]//Proceedings of the 6th workshop on system software for trusted execution. New York, USA: ACM, 2023:1-3.
- [22] Li Z J, Cheng J G, Chen Q, et al. RunD: a lightweight secure container runtime for high-density deployment and high-concurrency startup in serverless computing[C]//Proceedings of the 2022 USENIX annual technical conference. Berkeley, USA: USENIX Association, 2022:53-68.
- [23] Caprolu M, Di Pietro R, Lombardi F, et al. Edge computing perspectives: architectures, technologies, and open security issues[C]//Proceedings of the 2019 IEEE international conference on edge computing. Piscataway, USA: IEEE, 2019:116-123.
- [24] Goethals T, Sebrechts M, Al-Naday M, et al. A functional and performance benchmark of lightweight virtualization platforms for edge computing[C]//Proceedings of the 2022 IEEE international conference on edge computing and communications. Piscataway, USA: IEEE, 2022: 60-68.
- [25] Chen S C, Xu R J, Sun W Q. An edge computing architecture based on unikernel[C]//Proceedings of the 2022 Australian & New Zealand control conference. Piscataway, USA: IEEE, 2022:188-191.
- [26] Li Y, Wei Z L, Su J S, et al. A multi-agent collaboration scheme for energy-efficient task scheduling in a 3D UAV-MEC space[J]. Frontiers of Information Technology & Electronic Engineering, 2024,25(6):824-839.
- [27] Yang C, Liao F Y, Lan S L. Flexible resource scheduling for software-defined cloud manufacturing with edge computing[J]. Engineering, 2023,22(3):60-70.
- [28] Goethals T, Verkerken M, Sebrechts M, et al. Mixed-runtime pod networking for Kubernetes-based edge computing[J]. IEEE Access, 2025,13:209432-209449.