毕业设计开题报告——王菁芃

ReL4 中基于硬件加速的异步 IPC 设计与实现

一、选题意义

（一）选题背景

当今计算领域对操作系统的性能、安全性和可靠性要求日益严苛，微内核技术因其固有的优势而备受关注。其中，seL4 微内核作为目前最先进的微内核之一，已被广泛应用于安全关键领域。然而，seL4 的同步系统调用和 IPC 会产生大量的特权级切换，且无法充分利用多核的性能。虽然微内核对异步通知有一定的支持，但仍需要内核进行转发，其中的特权级切换开销在某些平台和场景下将造成不可忽视的开销。

为了进一步提高系统性能和效率，ReL4 项目使用 rust 语言重写的 seL4 微内核，基于用户态中断技术改造 seL4 的通知机制，设计了无需陷入内核的异步系统调用和异步 IPC 框架，在提升用户态并发度的同时，减少特权级的切换次数。而 ReL4 项目中，异步 IPC 的引入造成了额外的运行时开销，导致异步 IPC 在低并发的场景下性能显著低于同步 IPC。

（二）研究目的

本研究旨在针对 ReL4 项目中异步进程间通信的性能瓶颈问题，提出并实施了一种创新性的解决方案。该方案通过利用硬件资源，部分替代传统的异步调度器功能，对异步运行时进行硬件层面的加速，以期显著提升异步系统调用的性能表现。

（三）研究意义

本研究旨在提高了 ReL4 项目中异步进程间通信的性能，此研究成果可直接迁移至实际操作系统开发，以增强系统运行效率，降低资源消耗，进而提升用户交互体验。

在安全性及可靠性增强方面，本研究通过降低特权级切换频率，不仅优化了操作系统性能，亦提升了系统的安全性和可靠性。此优化对于安全关键领域，如航空航天、军事、医疗等，具有尤为重要的意义，有助于确保这些领域信息系统的安全稳定。

在技术应用与产业推动层面，本研究为微内核操作系统开发者提供了切实可行的技术路线，推动了微内核技术在多核处理器环境下的广泛应用。同时，本研究对于硬件加速技术的应用具有典范作用，有助于促进相关产业的技术进步和创新。

此外，本研究促进了操作系统与硬件设计、并发编程等领域的深度融合，为跨学科研究提供了新的研究路径和方法论。这对于促进计算机科学与其他工程学科的技术融合，具有重要的学术价值和实践意义。

二、国内外研究现状及发展动态

目前，国内在硬件调度器领域已经有了一些进展。如关沫张晓宇采用 VHDL 语言设计了适用于硬件化的实时操作系统调度器,基于 FPGA 使用组合电路和时序电路完成了系统内核调度器的搭建。

国外学者 Y.Klimiankou 提出了一种提出了 Micro-CLK，一种基于微内核的多服务器操作系统设计，其核心思想是将进程间通信从内核中移除，从而提高效率并简化内核设计。进程间通信的优化成为微内核性能优化的重点。

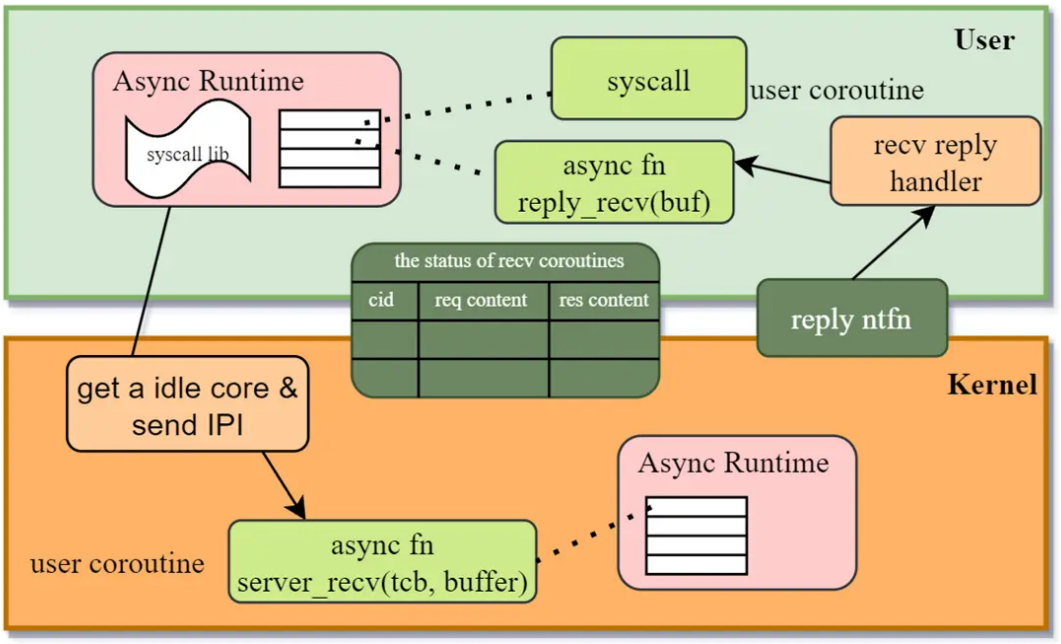
目前的研究工作虽然在不同领域取得了进展，但没有将硬件调度器与异步的进程间通信与异步的系统调用结合的实例。

三、主要的研究思路和研究内容

（一）研究思路

本研究利用 taic 硬件取代 Rel4 系统中异步运行时的部分功能，减少异步系统调用时的调度器开销。

原有的实现中，用户态程序进行异步系统调用后，cpu 陷入内核态。系统在内核态处理时，会找到一个空闲内核，发送一个核间中断并唤醒内核态的处理协程。处理结束后，内核会向用户态程序发送一个用户态中断，在中断处理函数中唤醒回复协程和原来阻塞的发送协程。低并发场景下，频繁的系统调用和用户态中断是导致性能较低的主要原因。



本研究将异步系统调用改为 taic 实现。当用户态程序需要进行异步系统调用时，只需要操作 taic 硬件即可唤醒内核态的协程，解决了用户态中断无法发送给内核的弊端。同时，内核处理结束后，也只需要操作 taic，即可唤醒发送方的回复处理协程。

改进后的处理方法，在一次异步系统调用中，省去了至少一次用户态中断，一次中断处理函数，一次系统调用的开销。

（二）研究内容

本研究将围绕以下几个核心环节展开：

首先，本研究将对 Rel4 内核中的用户态异步运行时机制以及 TAIC 硬件调度器的设计理念与具体实现进行详尽剖析，将深入挖掘其工作原理、性能特点以及接口设计，为后续的优化与改进提供理论基础。

其次，将着手进行硬件适配的异步运行时的体系结构设计与编码，主要包含以下工作：

a. 将内核中异步运行时的队列改为 taic 实现，以适应异步系统调用。

b. 将异步系统调用的库函数实现由系统调用改为读写 taic。

c. 将异步系统调用回复时的用户态中断改为读写 taic，唤醒用户态对应的协程。

接下来，将在 Qemu 模拟器和 FPGA 硬件环境两种不同的平台上进行仿真与测试，以确保我们的异步运行时能够在多种场景下稳定运行。主要测试内容包括：

a. 对改进前后的异步 ipc 和异步系统调用在不同并发度下进行测试。

b. 对改进前后的异步 ipc 和异步系统调用低并发下不同环节的时间消耗进行测试。

最后，我将对仿真与测试的结果进行全面评估与分析，针对发现的问题进行调优。通过这一系列的优化措施，旨在提升异步系统调用的性能，使其更好地适应各种应用场景。

四、研究预期成果与可行性分析

（一）研究预期成果

本研究预期成果包括在 QEMU 模拟器中实现硬件加速的异步系统调用，并在 FPGA 硬件平台上进行实际部署与验证。随后，通过全面的性能验证与分析，量化硬件加速对异步 IPC 性能和异步系统调用的提升效果，特别是针对低并发场景下的性能改进。最终，本研究将形成一篇毕业设计论文，详细记录研究背景、设计思路、实现过程、实验结果及分析。

（二）可行性分析

本研究的可行性主要体现在以下三个方面：技术可行性、理论可行性、资源可行性。通过分析，本方案具备实际实现理论基础与实践条件。

1. 技术可行性

基于现有 TAIC 硬件的技术特性与 Rel4 系统的模块化架构，本方案具备落地的技术基础。

TAIC 硬件通过预置的协程队列管理原语，能够直接替代用户态中断机制，其硬件级操作效率已通过 asyncOS 和 FPGA 原型验证。Rel4 内核采用的用户态-内核态异步运行时分离架构，为 TAIC 驱动的软硬件协同设计提供了标准接口适配空间，核心逻辑无需重构即可实现功能替换。此外，Qemu 模拟器与 FPGA 平台的联合仿真环境已实现对 TAIC 硬件的完整行为模拟，可系统性验证技术方案的稳定性与性能边界。

2. 理论可行性

低并发场景下用户态中断与系统调用产生的上下文切换构成主要性能瓶颈。

TAIC 硬件通过将协程队列管理下沉至硬件层，可消除单次 IPC 中至少两次用户态中断及关联的中断处理函数调用开销。同时，硬件队列的原子性操作符合高并发场景的线性扩展需求。

3. 资源可行性

研究所需的硬件与软件资源已具备充分保障。

TAIC 原型硬件在 FPGA 开发板完成部署，其寄存器映射与驱动接口文档完备，支持用户态与内核态的直接访问。

Rel4 内核的开源代码库具有清晰的模块化结构，异步运行时相关代码（如协程调度器、IPC 通信栈）可局部替换为 TAIC 驱动逻辑，核心系统功能不受影响。

开发周期方面，90%以上的代码修改集中于用户态库函数封装与内核驱动适配层，预计总代码量变动低于 2000 行，开发与调试成本可控。

五、学术创新点

1. **结合硬件加速和异步系统调用机制的设计**

在传统微内核系统中，异步系统调用的性能瓶颈通常表现为频繁的上下文切换和内核干预。本研究创新性地提出了通过硬件加速异步系统调用的方案。

2. **Rust 异步编程模型的应用**

本研究通过 Rust 语言的异步编程特性实现低开销的任务调度和并发处理。Rust 提供的零成本抽象和轻量级的任务调度机制，使得在进行异步任务时不会引入过多的运行时开销。这

一特性使得在微内核中实现高效的异步系统嗲用成为可能。

3. 跨学科技术融合

本研究的创新点还体现在操作系统、硬件加速、并发编程等多个领域的跨学科融合。通过在硬件层面引入加速技术与异步编程模型的结合，不仅推动了操作系统设计的创新，也为硬件加速技术的应用提供了新的思路和方向。

五、拟采取的进度安排

时间	内容
2024 年 11 月 31 日前	确定毕业论文选题或方向，报学院备案。
2025 年 1 月 1 日至 1 月 7 日	学习 Rust 异步编程基础
2025 年 1 月 7 日至 1 月 22 日	学习 ReL4 和 seL4 的相关论文、背景知识和熟悉相关代码。
2025 年 1 月 22 日至 2 月 4 日	学习 TAIC 硬件设计及使用，熟悉相关代码。
2025 年 2 月 26 日至 3 月 12 日	修改 Rel4 内核中的异步运行时，适配 taic
2025 年 3 月 12 日至 3 月 26 日	修改系统调用测试函数，跑通异步系统调用测试
2025 年 3 月 24 日前	完成中期报告及文献翻译
2025 年 3 月 26 日至 4 月 7 日	在 FPGA 中进行性能测试并分析数据
2025 年 4 月 7 日至 4 月 21 日	进行调优，优化
2025 年 4 月 21 日至 5 月 5 日	完成毕业论文，提交软件及相关文档
2025 年 5 月 6 日 5 月底	对毕业论文。相关文档进行修改
2025 年 6 月初	完成本科生毕业设计（论文）答辩

六、参考文献

[1] 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. 2009: 207-220.

[2] Heiser G, Elphinstone K. L4 microkernels: The lessons from 20 years of research and deployment[J]. ACM Transactions on Computer Systems (TOCS), 2016, 34(1): 1-29.

[3] Klimiankou Y. Micro-CLK: returning to the asynchronicity with communication-less microkernel[C]//Proceedings of the 12th ACM SIGOPS Asia-Pacific Workshop on Systems. 2021: 106-114.

[4]关沫,张晓宇.基于 FPGA 的实时操作系统调度器硬件化设计与实现[J].信息技术与网络安全,2019,38(06):83-89.DOI:10.19358/j.issn.2096-5133.2019.06.016.