中期检查报告

一、毕业设计（论文）主要研究内容、进展情况及取得成果

（一）目前进度总体完成情况

- [x] 学习Rust异步编程基础
- [x] 学习ReL4和seL4的相关论文、背景知识和熟悉相关代码。
- [x] 学习TAIC硬件设计及使用，熟悉相关代码。
- [x] 修改Rel4 内核中的异步运行时，适配taic
- [x] 修改系统调用测试函数，跑通异步系统调用测试
- [ ] 在 FPGA 中分析 taic 对rel4异步功能的性能提升（在下阶段中进行）
- [ ] 完成毕业论文，提交软件及相关文档（在下阶段中进行）

（二）主要研究内容综述

1.Rust异步编程的学习，rel4/taic项目复现

在前期准备阶段，我从官方文档、社区博客入手，学习了Rust语言中的异步编程模型，熟悉其底层的 `Future`、`Waker` 等核心机制。重点关注了 `Future` 的生命周期管理、`Pin` 与 `Unpin` trait 的使用规范，以及任务调度中 `Waker` 机制的注册与触发方式。

同时，我分析了ReL4操作系统中的异步任务调度实现。其通过在用户态和内核态引入异步运行时，实现了在seL4系统中实现异步编程。其异步通信模型使用了用户态中断机制，通过系统调用与事件队列相结合，实现了兼容seL4的异步通知机制。其中seL4_Signal/seL4_Send会向目标发送用户态中断以唤醒目标协程。seL4_Wait基于异步运行时和用户态中断实现。seL4_Wait会在该协程中阻塞并等待的用户态中断，当用户态中断到来时由异步运行时调度该协程，当实现非阻塞的等待通知。

随后，我在本地对两个项目的环境进行了构建，对现有的ReL4项目和Taic的项目成果进行了复现。通过对代码的走读，深入理解了原seL4项目中的能力机制与通知机制，部分内核对象以及它们在微内核权限管理与进程间通信中的角色。与此同时，我也系统性地梳理了Rel4中用户态异步运行时的结构设计，包括任务生成、就绪队列管理、唤醒机制以及协程状态维护等核心部分，进一步明确了本项目中需要改造的目标模块与技术要点。

2.兼容TAIC硬件调度器的异步系统调用

在完成Rel4与TAIC硬件项目的基础环境搭建与理解之后，我开始着手对Rel4内核中的异步运行时进行实质性的改造，目标是实现TAIC硬件调度器对异步任务调度流程的接入。Rel4原本的异步运行时主要依赖软件层面维护的就绪队列来管理协程的调度与唤醒，该结构虽然逻辑清晰，但在高并发场景下容易成为性能瓶颈。因此，为了充分利用TAIC硬件提供的异步调度能力，我将异步运行时中的就绪队列（ready queue）与协程唤醒逻辑进行了重构和替换。

首先，为了更方面的在内核中使用TAIC，我将硬件驱动`taic_driver`引入内核并基于原有的接口进行了简单封装。`taic_driver` 在本项目中作为内核与硬件调度器通信职责的重要中间层，任务入队、出队、通道配置、中断控制等功能接口。

其次，在就绪队列部分，我移除了原本由Rust容器结构或实现的就绪队列，改为 `taic_driver` 中抽象的`LocalQueue`对象。通过对该对象读写方法的调用，可以访问TAIC硬件内部维护的调度队列资源。在这一基础上，我将原有的异步运行时中执行器里生成协程的`spawn`方法改为使用`taic_driver`中的`task_enqueue`实现、获取就绪队列的`fetch`方法中改为`task_dequeue`实现。此外对队列的分配，协程序号分配等细节进行适配修改。以此替代原有的任务插入与分发逻辑，使得协程调度动作可以在硬件层实现，从而降低内核开销并提升调度响应速度。此外，我对异步系统调用的注册逻辑进行了修改为相应的硬件操作以适配新的内核协程生成模式。

3.异步系统调用测试

在完成Rel4异步运行时向TAIC硬件调度器的适配改造之后，原有的异步系统调用测试框架已无法完整覆盖或验证当前的新机制。因此，我对测试模块也进行了系统性的修改，确保整个异步系统调用流程能在新架构下正常运行，并具备可观测性与可复现性。

首先，对进程初始化时，通知注册、缓冲区注册、异步系统调用注册等初始化代码进行了改写，使之符合当前异步系统调用逻辑。其次，修改了异步库中的若干辅助函数，如seL4_call中原有的陷入内核的实现改成了在用户态发送硬件信号，实现了无需陷入内核的异步系统调用。此外，我还修改了进行异步系统调用的协程函数，实现了TAIC调度队列中发送方和接收方的注册。

4.阻塞队列的缓存数量限制的解决方案

在实现过程中，遇到了硬件中阻塞队列的缓存数量存在限制的问题。协程的唤醒需要使用一个中断向量，硬件调度器的中断向量数量只有32个，因此实际硬件中的阻塞队列的缓存数量仅为32。

因此，需要采用一种复用的方式解决该问题。使用TAIC的用户态程序需要记录中断向量的使用情况，以分配中断向量给不同的协程。中断向量表中常驻一个0号协程(dispatcher协程)。客户端执行异步IPC的协程在发送通知前需要先申请一个中断向量。如果此时中断向量表未满时，则该协程会拿到一个中断向量，并将该中断向量写入IPCItem中传递给服务端。服务端收到异步IPC后，根据IPCItem中的中断向量唤醒阻塞的客户端协程。如果此时中断向量表已满，则该协程申请不到中断向量，则将IPCItem中的向量置0。服务端收到异步IPC后，如果IPCItem中的中断向量为0，则唤醒0号dispatcher协程。dispatcher协程根据IPCItem中的cid，唤醒对应的协程。

具体实现上，我更改了原有IPC_Item的结构声明，加入了vec字段用于记录发送协程所申请的中断向量。然后，修改了的原IPC_Buffer的部分结构，使之可以进行固定位置的读写，以满足协程绕过分发直接唤醒的需求。最后，我修改了原调用方，内核中的异步系统调用处理函数，dispatcher协程的逻辑。

5.多次注册模式导致的性能问题

原有的模式中，TAIC信号接收方的注册是单次可用的，即每次接收完协程唤醒的信号后，需要重新对队列进行注册。多次注册导致高并发场景下的性能开销有所增大，因此我尝试将单次注册的逻辑改为重复注册，并尝试对比两种方案的优点与可行性。

二、存在的问题和拟解决方案

在当前阶段的研究与开发过程中，尽管已完成ReL4内核中异步运行时与TAIC硬件调度器的初步集成，但在系统运行、测试验证和性能评估方面仍然暴露出若干关键问题，需要在后续工作中予以优化与解决：

1．部分测试用例选取不够恰当，不能很好地反映异步所带来的性能提升

问题描述：

当前所进行的性能测试排除了异步协程阻塞等待中的时间，这样的测试方式不能很好的反应异步带来的吞吐率提升。

拟解决方案：

重新选取测试指标并编写测试用例，在不同并发场景下，测试异步IPC和异步系统调用所用的总时间，以反应吞吐率的提升情况。

2．当前实现仅在QEMU的模拟环境中得以验证，结果可靠性有待验证

问题描述：

由于开发初期以QEMU虚拟机为主要运行环境，便于快速调试与功能验证，目前所有修改与测试均在该虚拟化环境中进行。然而QEMU对硬件行为的模拟存在一定的抽象和延迟，部分行为无法真实还原TAIC硬件在实际运行中的行为。因此，一些基于性能优化的假设和调度逻辑仍缺乏真实环境下的验证。

拟解决方案：

计划在后续阶段将目前的系统部署至实际的FPGA开发板上运行测试，进一步验证系统设计的正确性与可行性。

3．内核中的异步处理协程若阻塞，只能等待时钟中断后被唤醒执行，调用处理的延迟较高

问题描述：

在现有设计中，内核中异步与同步的执行流共同存在。当内核中异步系统调用的处理函数因无处理内容而阻塞时，系统需等待时钟中断来选择核心执行协程。该机制虽然简单可靠，但在延迟敏感型场景下会造成性能瓶颈。

拟解决方案：

三、下一步研究任务与进度安排

在完成前期异步运行时与TAIC硬件调度器的集成、异步系统调用的适配和功能测试之后，下一阶段的工作将重点转向可靠性验证、性能测试及毕业论文的撰写与整理。具体安排如下：

（一） 性能评估与测试验证（4月中旬 - 5月初）

1. 在FPGA平台部署当前系统
   将当前已完成的ReL4 + TAIC集成系统烧录至FPGA开发板，测试其在真实硬件平台上的运行效果，验证系统稳定性和调度的正确性。
2. 丰富测试覆盖面
   覆盖不同负载强度和并发模式下的异步IPC、系统调用，进一步对比异步与同步机制在吞吐量、响应延迟方面的差异。

（二）功能与兼容性完善（4月中旬 - 5月初）

1. 针对异步系统调用处理的延迟较高问题进行改进

（三） 毕业论文撰写与系统整理（5月初 - 5月中旬）

1. 撰写毕业论文初稿
2. 整理代码与文档，将所有项目源码、测试脚本进整理并附带注释，准备提交材料

（四）预留收尾与论文修改时间（5月中旬 - 5月下旬）

1. 根据指导老师的反馈意见进行论文修改
2. 准备毕设答辩相关材料