答辩PPT讲稿

介绍

各位评审老师、同学大家好，我是计算机科学与技术专业的陈伟豪，我毕业设计的题目为《ReL4操作系统中基于用户态中断的异步系统调用设计》。

下面是我的论文检测结果和盲审结果，均符合答辩要求，申请参加评优答辩。

我将从以下几个部分介绍我的毕业设计工作内容。

第一部分：研究背景

首先进入第一部分，研究背景。

由于微内核结构的特殊性，低IPC与系统调用成本至关重要。

此前优化的相关工作主要分为三个部分：提升多核利用率、减少内核路径与减少特权级切换开销。其中，由于前两部分的优化已经趋于极限，减少特权级切换开销成为了优化的主要方向。在这样的现状下，我结合用户态中断提出了改进方案。用户态中断无需陷入内核即可发送信号，避免了特权级切换，具有了更好的通信性能。

所以，本次毕设的内容在于利用用户态中断机制改进后的Notification机制，对系统调用进行异步化改造，减少特权级切换频率，以此减少开销，提升系统性能。

第二部分：ReL4微内核介绍

下面进入第二部分，ReL4微内核介绍。

ReL4是本次毕设的开发平台，是Rust重写seL4支持RISC-V接口的微内核，它的结构如图1所示，从图中可以看出，这些模块提供基本的内核支持。

本人的工作为Async Runtime模块，其内实现了异步系统调用的相关方法，它以基于用户态中断的Notification机制为基础。

第三部分：设计与实现

下面进入第三部分，设计与实现。从此部分开始为本人所做工作！

图2是异步系统调用的框架图，这部分是本设计的核心内容。在图中可以看到，用户态、内核态都配备异步运行时模块。

图中设计内容可以分为三个部分：

首先是用户态异步运行时，它包含一个接收协程和多个提交协程，还有一系列用提交函数。

然后是内核态异步运行时，它包含一个处理协程，还有一系列处理函数。此外，内核协程可以在其他核心上运行，实现并行处理。

最后是用户态与内核态的交互部分，二者的交互通过Rust协程进行，交互内容，如系统调用参数、处理结果，将写入共享内存。用户态协程唤醒内核协程采用系统调用，内核协程唤醒用户协程采用基于用户态中断的

Notification机制。

有了框架，在具体实现中，我们还需解决如下三个问题：

1. 共享内存实现与互斥访问控制
2. 系统调用区分与参数传递
3. 协程实现

共享内存Syscall Buffer的实现如图3所示，其中包含两个维护协程状态的原子布尔变量以及两个IPC Queue分别用于储存请求和响应。IPCQueue的元素为IPCItem，IPC Queue的读写特性如图3上标注所示。由于用户态与内核在同一时间可能并发访问共享内存，因此需要实现互斥访问：原子布尔的写操作具有原子性，而IPC Queue通过互斥锁Mutex实现用户态与内核态的访问控制。

对于系统调用参数传递，如图4所示，Item内的MessageInformation成员用于区分系统调用，而ExtendMessage成员用于传递系统调用参数。

协程的实现如图5所示，两类协程中都包含了处理方法以及必要的数据结构来进行系统调用参数传递、解析，以及进行协程间通信。

第四部分：实验与结果分析

下面进入第四部分，实验与结果分析。

在系统的功能测试实验中，异步化改造的的四类系统调用都可以进行正常的工作，功能性测试实验将在演示环节中展示。

而系统的性能测试选择在不同并发环境下提交一定量的系统调用请求，并比较异步实现与原有同步实现的处理速度和特权级切换开销。由于FPGA开发板更接近真实的硬件性能，因此答辩展示数据为源于FPGA开发板。

首先，从每轮测试耗时的指标上看：横轴为并发度，纵轴为每轮所占时钟周期数，表征处理速度，越低越好。在低并发环境下改进后性能低于改进前，这是因为内核协程负载低，完成处理后自我阻塞，导致用户协程需要不断陷入唤醒内核协程。在高并发环境下改进后性能可以达到同步实现的130%，具有良好的性能优势。

其次，从陷入频率的指标上看：横轴为并发度，纵轴为陷入频率，表征特权级切换开销。改进前的陷入频率恒为1，而改进后的陷入频率远低于同步实现。这体现出本设计的工作达到了减少特权级切换频率的目的。

第五部分：总结与展望

下面进入第五部分，总结与展望。

在本次毕业设计中，我的工作为引入异步化思想，结合了共享内存通信内容丰富与用户态中断机制方式快捷有效的优势，并发、集中处理大量系统调用，最终实现异步系统调用框架。结果表明在高并发环境下异步系统调用具有优于原有实现的性能。

未来的发展中可以通过优化异步系统调用框架性能、增加内核协程数量以及实现调度器动态选择同步与异步接口的方式来获得更好的系统性能。

结尾

以上就是我的毕设工作，感谢各位专家老师，请您批评指正！

答辩问题

Q：毕业设计的创新点?

A：毕设工作的研究创新点在于在系统调用的设计实现中引入了异步的思想，能够并发且集中地在内核态处理大量的系统调用请求，这是以往的同步系统调用所无法实现的，适用于存在大量系统调用需求的应用场景。此外，本文工作将常用的共享内存通信方式与用户态中断机制进行了结合，结合了共享内存通信通信量大且通信内容丰富与用户态中断机制方式快捷有效的优势。

Q：如何使用的用户态中断?

A：首先，在异步系统调用的注册流程中就需要将用户态接收协程生成，并将其注册为用户态中断接收协程，这样用户态中断处理例程就可以在信号到来时唤醒它。而在注册异步系统调用时内核协程也将获得对应的UITTE，可以向用户协程发送用户态中断。

Q：为什么选用共享内存?

A：原有的系统调用采用消息寄存器的方式传递参数，而异步系统调用的内核处理协程可能在其他核心上运行，无法读取对应消息寄存器内容，因此需要考虑其他实现方式。信号量的方式可以传递的参数有限，而共享内存提供的大空间就可以满足这一需求。

Q：为什么选用处理速度和陷入频率指标?

A：处理速度指标可以最直观的体现异步系统调用框架的性能，而陷入频率指标可以反映是否达到了研究初所确定的减少特权级切换频率的研究目的。

演示视频讲稿

下面进行qemu模拟器上的异步系统调用功能演示。首先进行内核与用户态程序的编译。编译完成后运行系统。

此部分为异步系统调用的注册流程。

首先是输出类系统调用的演示，可以看到，PutChar和PutString系统调用都输出了应有的字符和字符串，GetAddress系统调用也获得了正确的物理地址。

Notification机制类系统调用的演示讲解由于时间原因忽略，具体内容见论文。

然后是内存映射类系统调用的正确性。可以看到，我们使用UntypedRetype系统调用生成了一个页表对象与一个页框对象并将其映射到虚拟地址，然后通过将虚拟地址强制转换为类对象指针并调用成员函数的方式进行地址访问，可以看到可以进行正常的读写。而解除映射在进行读写则会触发错误杀死线程。

以上内容证明了异步系统调用框架的正确性。

下面进行qemu模拟器上的异步系统调用性能演示。首先进行内核与用户态程序的编译。编译完成后运行系统。从Qemu的仿真结果上可以看出，异步系统调用的平均每轮时钟周期占比数据约为原有实现的40%，而陷入频率约为原有实现的1%，具有良好的性能。

下面进行FPGA开发板上的异步系统调用性能演示。首先进行内核与用户态程序的编译。编译完成后将程序通过远程连接的方式烧入开发板中运行。从FPGA串口的运行结果上可以看出，异步系统调用的平均每轮时钟周期占

比数据约为原有实现的77%，而陷入频率约为原有实现的1%，异步系统调用框架在FPGA的环境下相较于同步系统调用带来了30%的性能提升，也达到了减少陷入频率的目的，具有良好的性能优势。