sv6 for RV64 SMP - 课程设计期末报告

计53 谭闻德 尹宇峰

选题概述

近年来，人工智能及大数据的众多应用为计算能力提出了新的需求，而处理器核心频率的提高已经遇到困难，现如今，多核心处理器、并行化已成为提高计算系统性能的主要手段。事实上，在2013年，Austin T. Clements等人[1, 2]就提出过一种新的手段，用于优化并行系统软件的性能。此项工作提出了一个名为commuter的工具，借助符号化执行技术[10, 11]以及SMT求解器[12]，用于挖掘系统规范中潜在的并行化优化点；同时，他们用此工具辅助设计实现了一个并行度很好的精简的操作系统，名为sv6。sv6在x86-64指令集架构上实现，考虑到目前新兴的RISC-V指令集架构[3, 4]更开放、灵活性更好、硬件设计更简洁，为此，本课程设计基于上述工作，将上述工作提出的sv6移植到RISC-V 64位多核平台，以期能够推动RISC-V并行架构的发展。此外，我们深入理解了commuter的设计与实现并对其进行总结。最后，本课程设计还基于commuter对unix socket API进行了建模，未来还可以用此模型对实际操作系统的网络子模块进行并行性测试与优化。

由此，本课程设计的主体工作计划分成以下三个部分：

1. **移植** 将sv6操作系统从x86-64移植到RISC-V 64位（以下简称RV64）对称多处理器（以下简称SMP）环境。
2. **分析** 详细阅读[2]的文献和代码，分析和理解commuter工具的设计与实现并对其进行总结，方便后人阅读理解。
3. **建模** 基于commuter工具对unix socket API进行建模。

相关工作

这一小节简要介绍本课程设计会用到的前人的一些已有工作。RISC-V [3, 4]是一个新的指令集架构，它更开放、灵活性更好、硬件设计更简洁。sv6操作系统[1]则是一款基于xv6，并行度很好的精简的类POSIX的研究性质的操作系统。通过符号化执行的方法[10, 11]，可以将一个函数的输入参数指定为符号值，函数的每一个路径可以产生一个先决条件逻辑表达式，通过Z3等SMT求解器[12]进行求解，就可以知道这个路径是否可达，是否会产生特定的情况（大多数情况是断言失败）。Austin T. Clements等人在2013年提出的Commuter [2]能够利用上面提到的符号化执行技术以及Z3求解器，完成对系统调用接口规范模型的分析，为设计良好并行化的接口给出建设性意见。

我们的工作

设计方案

本课程设计分为以下9个步骤分阶段进行：

1. 准备开发环境 — 于第6周完成

把RV64 SMP的工具链，包括编译器GCC、模拟器QEMU等环境准备好。

这是基础性的工作，为进行RV64的开发，必不可少。

2. 学习RV64 — 于第6周完成

学习RV64用户态指令集架构和特权指令集架构。

这也是基础性的工作，为进行RV64的开发，必不可少。当然，若之后遇到问题，可以进一步查阅文档。

3. 动手实践RV64 SMP — 于第6周完成

基于bbl编写几个RV64 SMP的小例子，注重SMP的启动以及管理。这个环节旨在熟悉RV64 SMP的一些关键性问题，例如谁负责启动AP（除了启动处理器之外的处理器核心称为AP），启动AP后的状态如何设置等。

4. 开始移植 — 于第6周开始,第11周完成

开始将sv6移植到RV64单核环境。

这个阶段的重点以及难点都在于将原有的x86-64相关的部分在RV64架构下重新实现，包括虚拟内存管理、中断和异常（包括系统调用）、原子操作指令、寄存器上下文、内核栈设置、外设中断配置、设备驱动程序等。

5. 多核移植 — 于第11周完成

进一步将sv6移植到RV64多核系统。

注：根据实际情况，单核和多核的移植可以分开或者同时进行。

6. 真实硬件测试 — 于第12周完成

从辉总那里借来了RV开发板一套，含12V 2A DC5.5电源一个、装板子的瓦楞纸质包装盒一个、保护用粉色塑料海绵纸一张、RV板一个。在RV开发板上成功运行sv6移植。

注：感谢辉总大力支持！

7. 学习并实践符号化执行方法、Z3求解器 — 于第10周完成

学习符号化执行方法以及Z3求解器的使用。

commuter工具使用到了符号化执行方法以及SMT求解器，为了更好地完成sv6的优化，更好地写出接口规范，需要学习这两项内容。

8. 深入分析和理解commuter的设计与实现 — 于第10周开始，第13周完成

详细阅读[2]的文献和代码，分析和理解commuter工具的运行原理和代码实现并对其进行总结，方便后人阅读理解。

9. 基于commuter对socket建模 — 于第13周完成

利用commuter工具对unix socket API进行建模。

主要工作是接口规范的编写。

小组分工

1. sv6移植RV64 SMP

谭闻德主导，尹宇峰辅助。

2. 深入分析和理解commuter的设计与实现

尹宇峰主导，谭闻德辅助。

3. 基于commuter对socket建模

尹宇峰主导，谭闻德辅助。

具体细节

sv6移植

commuter分析和理解

基于commuter对socket建模

结束语

实验成果

实验收获

在本次课程设计中，我们：

- 熟悉了RV64指令集
- 熟悉了一个操作系统的移植过程
- 学习了符号化执行方法
- 学习了Z3 SMT求解器的使用方法
- 了解了sv6
- 深入了解了commuter
- 了解了socket

未来的工作

致谢

感谢陈渝老师，向勇老师对于我们课程设计的耐心指导；

感谢辉总给予我们硬件的大力支持；

感谢路橙，于志竟成组与我们小组合作与讨论。

参考文献

1. sv6 https://github.com/aclements/sv6
2. Commuter https://pdos.csail.mit.edu/archive/commuter/ https://github.com/aclements/commuter
3. The RISC-V Instruction Set Manual Volume I: User-Level ISA
   https://riscv.org/specifications/
4. The RISC-V Instruction Set Manual Volume II: Privileged Architecture
   https://riscv.org/specifications/privileged-isa/
5. bbl-ucore https://ring00.github.io/bbl-ucore/
6. ucore labs RISC-V 64移植 https://gitee.com/shzhxh/ucore_os_lab/tree/riscv64-priv-1.10
7. Booting a RISC-V Linux Kernel https://www.sifive.com/blog/2017/10/09/all-aboard-part-6-booting-a-risc-v-linux-kernel/

8. Entering and Exiting the Linux Kernel on RISC-V https://www.sifive.com/blog/2017/10/23/all-aboard-part-7-entering-and-exiting-the-linux-kernel-on-risc-v/
9. Paging and the MMU in the RISC-V Linux Kernel https://www.sifive.com/blog/2017/12/11/all-aboard-part-9-paging-and-mmu-in-risc-v-linux-kernel/
10. 符号执行入门 https://zhuanlan.zhihu.com/p/26927127
11. mini-mc https://github.com/xiw/mini-mc
12. Z3 https://github.com/Z3Prover/z3
13. HiFive Unleashed https://www.sifive.com/products/hifive-unleashed/
14. Freedom U540-C000 Manual https://www.sifive.com/documentation/chips/freedom-u540-c000-manual/