RISC-V引导程序环境：应用与规范

洛佳

华中科技大学 网络空间安全学院

物联网安全实验室

2022年4月9日 在线会议

本次演讲……

- 我的名字是洛佳
- 各种各样的社交平台：@luojia65
- 面向产业界的听众
- 着重讲架构无关的实现要点
- 欢迎批评指正！

引导程序环境是什么？

如果引导程序能支持内核运行的话……

引导程序环境是什么?

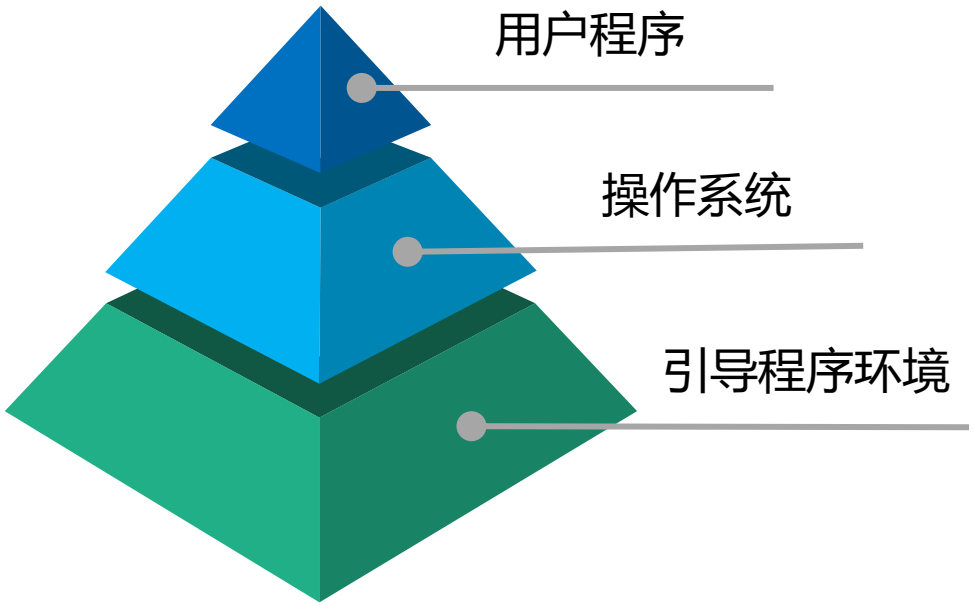
- “摇篮到坟墓”：从上电到内核生命周期
- 保持在后台运行——安全性至关重要!
- 支持系统运行的平台无关接口

RISC-V引导程序环境的发展过程

- 伯克利引导程序[1]
  - 相当于传统架构的引导程序
  - 负责初始化并进入M态或S态
- 支持内核运转的固件接口
  - RISC-V的S态可以使用ecall指令，就好像是系统调用一样
  - 先保持固件运行，再用环境调用支持内核运行
  - 若能够规范化，将显著提高RISC-V内核的可移植性
- 具有SBI接口的RISC-V固件
  - 引导程序环境的概念开始出现
  - 模型：金字塔（对），~~链条（错）~~

对比：不同架构的固件运行框架

RISC-V架构



其它传统架构

SBI接口的常见实现模型

- RustSBI
  - 处理接口规范（函数模块编号等）、常见数据结构，以及提供参考实现
  - 安全性至关重要——完全使用Rust语言编写
- 其它：Coffer、OpenSBI、Diosix等

RISC-V设备固件的组成部分

- SBI接口的生态位：只是一个接口，不能单独启动系统。必须和引导程序部分的代码配合，才能启动复杂的操作系统。
- 将SBI接口与具体的固件软件结合
  - 常见组合：RustSBI + Oreboot、OpenSBI + U-Boot等
  - 静态编译法（通常必须相同编程语言）、二进制拼接法（编程语言无关）
  - 动态库法就有点像UEFI固件，在RISC-V平台上很少见
- 由引导程序代码启动Linux等操作系统，由SBI部分代码支持操作系统稳定运行，两部分相互配合，不可或缺

为什么我们需要内核支持接口？

- 考虑一些常见的平台有关操作，再考虑平台的多样性
  - 启动其它的处理器核
  - NUMA节点？单芯片？多芯片？
  - 如果有一个操作能统一不同平台的多样性……SBI调用如何？
- 节省内核开发的开销
  - 考虑一个问题：这个RISC-V平台如何关机？
  - 我们希望降低操作系统支持各个平台的开发难度
- 模拟一些软件实现的寄存器
  - RISC-V的寄存器并非全由硬件实现，软件CSR寄存器也是RISC-V标准的一部分

应用场景：虚拟机迁移

- 考虑配置不同的RISC-V虚拟机和同一个系统镜像。我们需要这个镜像在两台虚拟机上都能运行
- 细致的驱动暂不考虑时，我们希望至少能完成开关机、多核启动等最简单的功能，来支持内核的正常运行
- 考虑到不同硬件和主板差别巨大，内核全部支持有困难，可以选择支持一个系列的硬件和统一的接口功能，然后其余部分交由引导程序完成。（如果考虑RISC-V专有指令集，情况会更多）
- 引导程序成为虚拟机和宿主系统间的统一媒介
- 此时固件向上提供统一的接口，以便于操作系统迁移工作

RISC-V架构版本更新了？不用怕！

- 固件是支持环境，它可以通过M态陷入不存在的新版指令和寄存器，然后用旧版的模拟它，至少在正确性上是完整的
  - 历史上成功解决K210芯片运行完整类Unix系统的问题
  - 如果需要模拟的指令太多，也就是该换CPU的时候了……
- 屏蔽一些外设，然后取而代之
  - 可以在不改内核代码的前提下修复一部分硬件bug（否则upstream的问题是个灾难！）
  - 反过来想也就是固件需要保证安全性的一点
- 这两条仔细想想其实相当震撼，以前的处理器设计少有考虑

好的，我们收回来……

- 引导程序环境 = 引导程序 + 支持环境
- 降低内核的支持难度
- 提高硬件间兼容性
- 常常被忽视但至关重要的基础软件!

RISC-V SBI接口标准规范

版本v1.0.0正式批准版：2022年3月23日

基础扩展

- 获取版本号和扩展模块检测函数
  - 详见规范正文[2]
- 用途：内核检查SBI实现的版本和硬件的版本
- 实现注意：
  - 不能返回错误！所有的基础扩展必须正确返回
  - 探测扩展模块函数实现时，若返回“此模块存在”，则必须完整实现对应模块的所有函数，这一点SBI规范的正文是有明确规定的。

时钟设置扩展

- 一个函数：时钟设置函数
  - 传入时刻值
  - 当芯片内部时钟到达上次设置时刻值的时候，发生特权态时钟中断
- 内核设计时，配合S态时钟中断陷入，完成上下文切换过程
- 实现SBI固件注意：
  - 使用芯片上拥有的时钟外设，它可能叫做PLIC，也可能是芯片厂家专门设计的外设
  - 应当搭配time CSR寄存器的读模拟操作共同实现！
  - 时刻值永远是64位！
  - 实现的开销不大，Linux内核会使用

核间中断扩展

- 只有一个函数：核间中断发送函数
  - 给选中的核发送一个软中断
- 这是第一个接触需要“位集数据结构”的扩展，它需要在一次SBI调用中选中多个核，来降低调用的次数。
- 实现注意以下事项：
  - SBI标准定义了位集数据结构，了解吗？遮罩位集和起始编号是什么？
    - 在RustSBI中，使用HartMask结构体操作位集数据结构。
  - 一般是用CLINT外设或相似的外设实现的，做一个易失性写操作即可
  - 函数永远成功，即使选中的核不存在

远程栅栏扩展

- 可以给其它的处理器核远程发送栅栏指令
  - 这些栅栏指令包括FENCE.I、SFENCE.VMA、HFENCE.{G,V}VMA。
  - 栅栏指令可以带地址空间、虚拟机编号参数，也可以不带，取决于调用者选择本模块内的哪个函数名。
  - 就像在其它的核执行一个栅栏指令一样。
- 实现注意：
  - 通常可以和核间中断扩展共享代码。如果这样做，将下一步执行的操作保存在某处，当另一个核被M态软件中断打断，查询应当执行的指令，然后执行它，来完成函数的功能。
  - 如果平台不支持或禁用了虚拟化，虚拟化远程栅栏函数返回“未实现”。

核状态管理扩展

- 操作多核系统各个核的启动状态
  - 启动函数、停止函数、获取核状态函数
  - 暂停函数：暂停后通过核间中断扩展来唤醒
- 多核启动重头戏…… [3]
  - RISC-V的核状态管理可以启动相同架构的核，也可以启动不同架构的。在异构多核处理器上这部分扩展至关重要
- 如何实现此扩展？
  - 通常这需要同步机构，但请注意，不是所有的RISC-V核都支持异步A指令集，或者不是所有的地址区间的总线都支持原子读写，这种情况需要自己编写合适的同步数据结构

系统复位扩展

- 重启和关机
  - 在具体的主板上很容易实现
- 实现方法和注意实现
  - 主板上有BMC芯片吗?
  - 如果不用单独的电源芯片，SoC芯片内是否有电源启停机制?
  - 冷重启和热重启
  - 其它问题写传统架构的固件也常见
- 模拟器上复位
  - 内核单元测试常用，虽然实现具体主板的产品不会遇到这类问题
  - 使用常见的测试设备，如sifive-test

性能监视器扩展

- 包含的函数可读取硬件和固件的事件计数器
  - 如果芯片具备这些计数器，请实现性能计数器扩展，以便内核合理地发现和使用芯片硬件提供的微架构计数器功能
- 实现注意事项
  - 固件计数器：与RISC-V引导程序环境固件有关的计数器事项，比如处理的读写异常次数等等
  - 还记得位集数据结构吗？
  - 不要忘了用软件实现cycle、instret和hpmcounterX寄存器！

自定义扩展空间

- 包含三个部分。分别是：实验性扩展空间、供应商定义的扩展空间、固件定义的扩展空间
- 提供较强的开发灵活性
- 如果供应商或客户要求添加专有的固件功能，请添加到这几个扩展空间中！

SBI标准不包括……

- ……标准输入输出
  - 请操作平台设备完成功能!
- ……检测和枚举设备
  - 请搭配其它的规范标准共同实现
  - 如：mconfigptr或UEFI接口等
- ……枚举所有的核
  - RISC-V是容易虚拟化的指令集
- ……提供面向各种设备的抽象
  - 通常设备对内核都是透明的，这样能降低开销
  - 如果这样做SBI标准就太大了，稳定和批准流程遥遥无期

引导程序环境开发选型指北

现在你已经学会加法了，来试试这道微积分题吧！

Oreboot + RustSBI

- Oreboot原来是Coreboot的分支，去掉了C语言
  - 它可以引导LinuxBoot格式的固件，它和另一个仅有busybox的Linux固件结合（可实现维护功能），使用kexec启动真正的Linux内核
  - 支持x86、ARM、PowerPC和RISC-V等指令集架构
- RustSBI是Oreboot支持内核运行的接口实现
- 开发方法
  - 在Oreboot的src/soc下添加芯片，在src/mainboards下添加主板
- 为什么使用Rust语言？
  - 安全，直接使用嵌入式生态，工具链配置容易，软件包成熟

SBI固件实现细节事项

- 异构多核：请合理规划管理核的用途
  - 详见参考文献[3]
- 如何调试SBI固件？
  - 和调试嵌入式设备是一样的

专有指令集和专有设备……

- 专有固件软件设计
  - 因为运行在M态的固件通常不需要跨主机平台，请尽可能充分地利用处理器的资源和外设
  - 若有可能，在处理器有专有指令集的情况下，启用专有指令集。并且，使用支持专有指令集的编译器编译固件，来节省运行时间和占用空间
- RustSBI是一个依赖库
  - RustSBI有意将实现和接口抽象分离出来，便于专有设备实现RustSBI
  - 相比O开头的SBI实现，软件贡献upstream受到制约

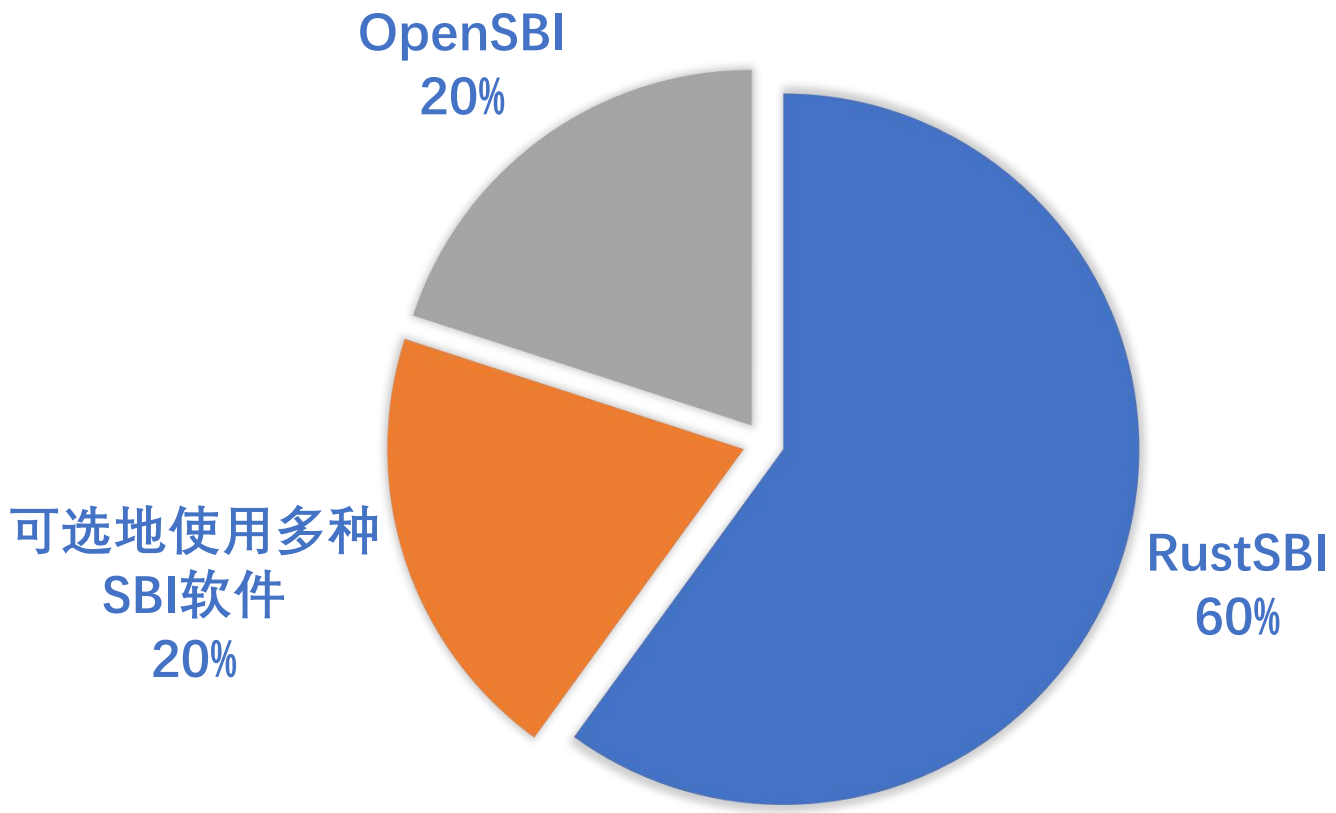
引导程序固件开发模型

- 裸机模型（嵌入式应用模型）
  - 使用嵌入式应用的开发方法制作RISC-V固件
- 实时系统模型
  - 将常见的实时系统作为RISC-V固件，由调度程序处理内核请求
- 嵌入式或完整Linux模型
  - Linux作为引导程序部分，复用Linux系统完整的驱动生态
- 开发难度：裸机模型 < 实时系统模型 < Linux模型
- 运行性能：取决于SBI接口实现（SBI接口和引导程序是独立的）

U-Boot + OpenSBI

- 通过多个阶段启动：U-Boot SPL -> OpenSBI -> U-Boot -> Linux
- 要求使用二进制拼接的方法，将SBI实现和U-Boot共同编译
  - 这部分对OpenSBI项目要有较强的了解
- 较难配置开发环境（通常只能使用和Linux相似的系统）

60%的国家一等奖赛队选择RustSBI



数据来源：2021年全国大学生系统能力竞赛操作系统赛一等奖作品公开仓库（https://github.com/oscomp/2021oscomp-best-kernel-design-impl），根据其中的源码和文档统计

RustSBI是RISC-V SBI的官方标准实现

3.9. SBI Implementation IDs

Table 4. SBI Implementation IDs

Implementation ID	Name
0	Berkeley Boot Loader (BBL)
1	OpenSBI
2	Xvisor
3	KVM
4	RustSBI
5	Diosix

原始链接：https://github.com/riscv-non-isa/riscv-sbi-doc/blob/5b856c26dc6b12d5b15ee4f93651592803ac4c50/riscv-sbi.adoc#39-sbi-implementation-ids

RustSBI与其它SBI实现产品的功能对比

功能	RustSBI	以O开头的其它SBI实现
类Unix内核的运行环境	✓支持	✓支持
SBI 0.2版本IPI、TIMER功能	✓支持	✓支持
提供平台的设备描述	✓支持，通过灵活的serde设备树	✓支持，通过硬编码的设备树
跨平台编译和构建	✓支持，使用xtask框架	✓部分支持，需要配置环境
SBI 0.3版本HSM功能	✓支持，框架已定义	✗不支持
与Rust生态相容性	✓相容性良好	✗较难与Rust生态结合
向后兼容旧特权版本硬件	✓支持，以K210为例	✗不支持
启动管理核	✓支持，如Unmatched	✗不支持，会屏蔽管理核
扩展和定制高级功能	✓支持	✗不支持，较难合并到主分支

*通过代码和文档比较得到。RustSBI文档：https://github.com/rustsbi/rustsbi-hifive-unmatched/wiki。OpenSBI文档：https://github.com/riscv-software-src/opensbi。

欢迎使用RustSBI 0.2.2版本

- 完整支持RISC-V SBI规范标准1.0.0正式批准版
- 两年受科研和产业界用户好评，稳定性值得信赖
- 完全由Rust语言编写
- 可用于裸机主板固件开发以及虚拟化软件开发
- 项目地址：https://github.com/rustsbi/rustsbi

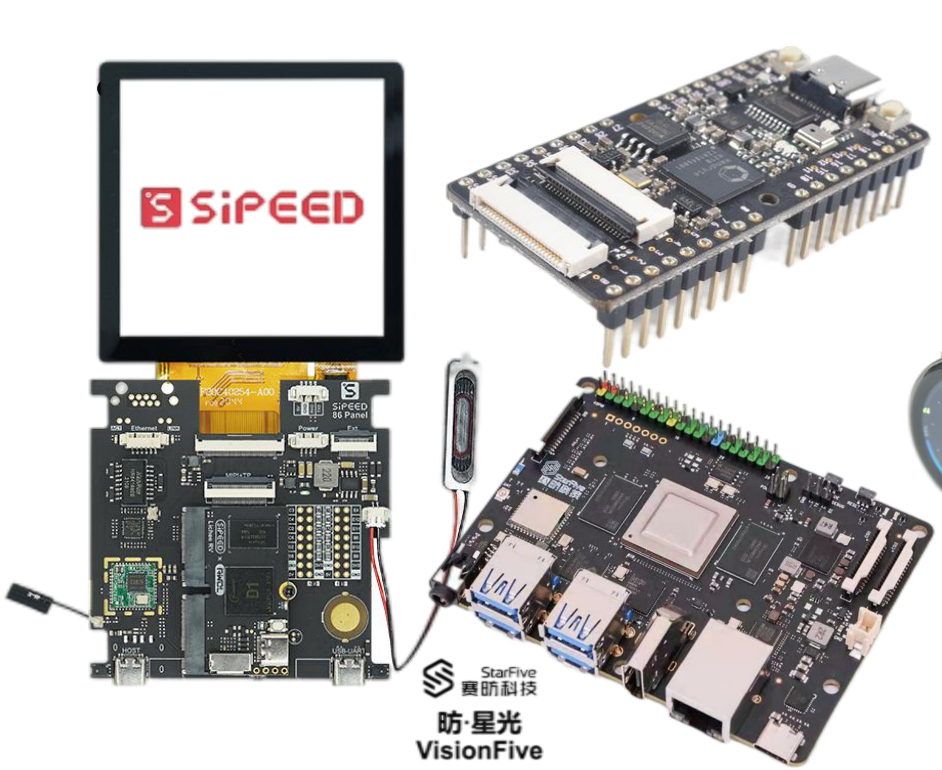
什么时候使用RISC-V SBI?

已经实现和构思中的应用场景……

RISC-V SBI接口的适用范围

- 具备S或VS态的RISC-V处理器
  - 这包括桌面、服务器和移动处理器等等
  - 但通常的嵌入式处理器不在此范围内
- 安装在由以上处理器设计的主板
  - 建议为主板配备闪存，可刷入主板闪存中
  - 单板计算机产品可装入SD卡分区
  - 移动终端可刷入UFS闪存分区

常见使用SBI的RISC-V产品（2022年4月）



单板计算机和智能家居设备

可穿戴设备

个人电脑 服务器

如何为您的单板计算机实现SBI固件?

- 若使用Oreboot路线，可复用Linux驱动，无需单独开发设备驱动。为Oreboot添加主板和SoC芯片支持即可。
- 是否有闪存?
  - 有——固件放置于NAND或NOR闪存中
  - 无——若有SD卡，SD卡分区，芯片从某个区启动，这里就放置引导程序
- 是否有可选的屏幕?
  - 有——检测屏幕是否插入，如果插入，显示一个商标或标志
  - 通常同一个平台固件可通用，即使产品外形不相似
- 其余部分按照SBI标准实现即可

适用于个人电脑和服务器的SBI固件

- 多种可能的处理器和外设
  - 固件安装到主板的闪存固件中，便于更换硬盘
  - 多样的启动方式：SSD启动等。引导程序选型时注意
  - 多种可能的芯片和主板外设：应当拥有外设检测与扫描功能
- 对维护者友好
  - 如点亮EzDebug灯。便于售后合作商维修主板，便于用户升级硬件
- 异构多核
  - RISC-V下桌面处理器研发的趋势之一
  - 注意SBI合理暴露异构处理器，厂家操作系统可合理使用
  - 编译引导程序注意：不是所有的核都支持所有指令集

为您的移动产品实现SBI固件

- 从处理器支持的eMMC或UFS闪存中启动
  - 在启动分区放置引导程序
- 如果系统损坏了……
  - 引导程序具备维护功能！可使用线刷等方法重装操作系统
  - 实现固件时严格按照SBI标准即可，维护功能由后续阶段实现
- 出厂预装引导程序，可定制及包含厂家专有功能
- 安卓10系统支持RISC-V

您的下一款路由器产品可以是RISC-V

- 路由器的外设吞吐速度有时超过处理器的处理速度
  - SBI固件尽量直通外设到内核，以免影响性能
- OpenWrt可以由Oreboot+RustSBI方案启动，Rust语言可保证长期运行的安全和稳定性

快速响应

- 机顶盒、高级嵌入式设备可使用RISC-V引导程序启动
  - 如果用UEFI for RISC-V启动，加载动态库的时间将超过预期
  - 建议选择引导程序方案时注意避免重复初始化硬件
- RISC-V架构下，固件的性能能影响操作系统的运行速度
  - 关键应用中做好优化!
  - 陷入旁路、异构调度算法……

感谢

- Open Source Firmware社区，为Oreboot项目与LinuxBoot项目提供贡献，在审核我的合并请求时提供帮助
- Daniel Maslowski，帮助我测试RustSBI和Oreboot代码，以及在项目发展中提出若干宝贵的意见
- Tuna社区成员，解决了我开发RustSBI时遇到的许多困难
- Rustcc社区与Rust语言庞大的生态群体，许多我们永远不会见面的人给予隐形的帮助，才让所有的事情变得更好，这也许就是开源项目的意义所在
- 所有在我成长路上帮助过我的朋友们和老师们

谢谢！ & 提问时间

RISC-V引导程序环境：应用与规范

洛佳 物联网安全实验室

华中科技大学 网络空间安全学院

引用

- [1] 伯克利引导程序：https://github.com/riscv-software-src/riscv-pk/blob/master/bbl
- [2] SBI接口规范：https://zh.wikisource.org/wiki/Translation:特权层二进制接口规范标准 。原文：https://github.com/riscv-non-isa/riscv-sbi-doc
- [3] 洛佳, RustSBI的多核管理和复位模块, 2021. https://github.com/rustsbi/slides/blob/main/2021/RustSBI的多核管理和复位模块.pdf