RustSBI 0.3.0新版本与应用场景介绍

洛佳

华中科技大学 网络空间安全学院

2022年11月

本次演讲……

- 20分钟时间
- 技术为主。社区发展部分留于文末，本次演讲不涉及
- 性能测试数据
- 欢迎批评指正！如果有问题，请随时提出。
- 项目地址：https://github.com/rustsbi/rustsbi
- 我的联系方式：me@luojia.cc

SBI及其环境如何支持虚拟化系统?

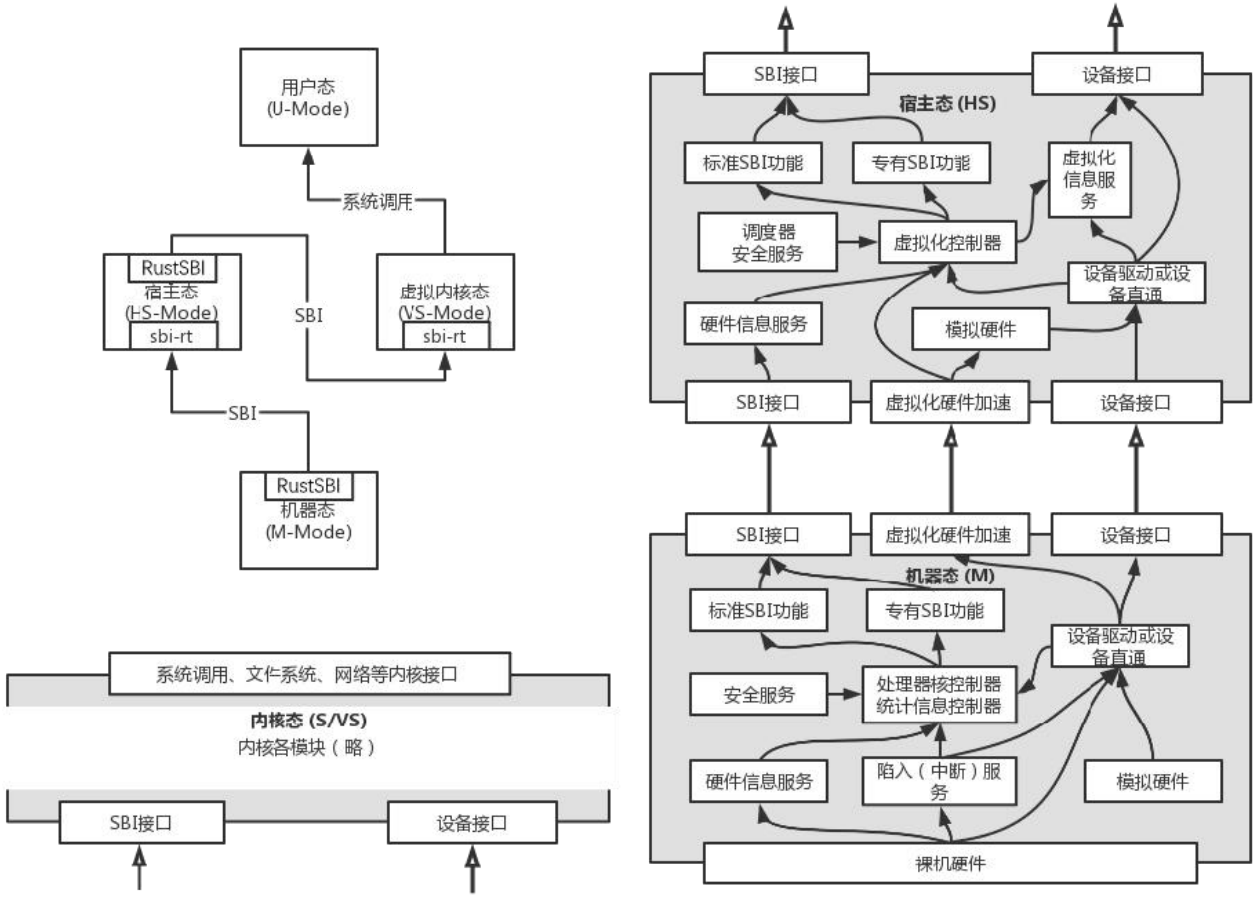


图 1 RISC-V架构的虚拟化模型

- 宿主态（HS-Mode）可用于支持Type-1和Type-2 Hypervisor
- HS-Mode向VS-Mode提供SBI接口，正如M-Mode向S-Mode提供的一样
- 通过SBI接口，可以完成重启、核间中断等操作，就好像内核运行在真实硬件中
- 专有的SBI功能可以也可由HS-Mode实现
- 虚拟的核是HS-Mode提供的，每个核若有S态，均存在SBI服务
- 虚拟的硬件若由HS-Mode模拟，VS-Mode将其看作是真实硬件，此思路也可屏蔽硬件

嵌套虚拟化及模拟器上的SBI

- 嵌套虚拟化情况时，内层虚拟化常常需要软件模拟。开发模拟器时，硬件特性需要软件模拟
- RISC-V嵌套虚拟化的外层可使用H扩展加速
- 实现RustSBI的HSM模块时，需要考虑H扩展的内存刷新和同步操作。刷新软件支持的影子页表
- 实现跨指令集模拟器，若将具备硬件加速的模拟器直接作为固件运行，思想类似于Type-1 Hypervisor（龙芯运行RISC-V等）
- Trap-and-Emulate和影子页表的参考实现：@dramforever的OpenSBI-H项目（https://github.com/dramforever/opensbi-h）

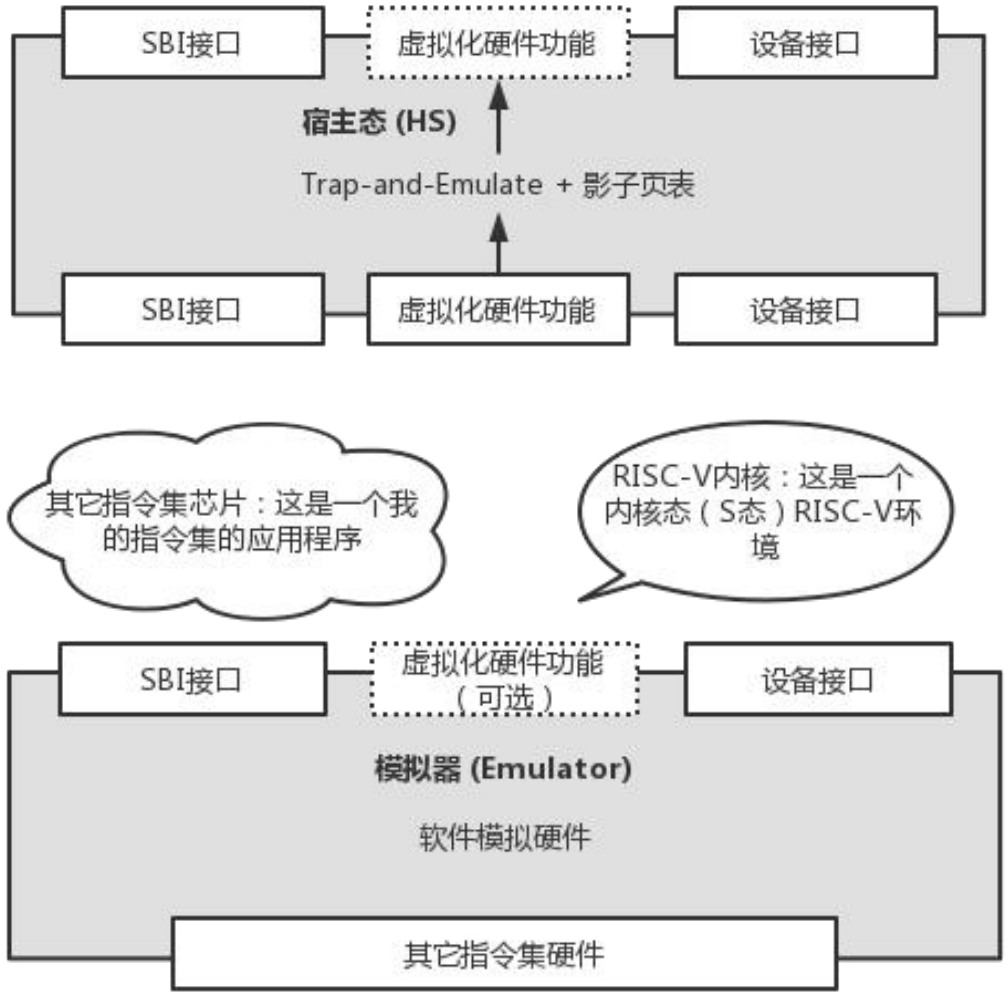


图 2 RISC-V嵌套虚拟化模块和模拟器模块示意

sbi-testing：RustSBI测试框架

- 测试框架在软件开发中常常有意想不到的作用，尤其是需要硬件功能协助的固件，此时测试框架的作用至关重要
- sbi-testing是RustSBI社区提供的测试框架库
- 按照RISC-V SBI规范设计可能的调用流程。以HSM扩展为例，测试启动、停止和暂停、唤醒功能，与HSM核状态探测功能返回的状态比较，测试目标核是否按照扩展的要求变换状态
- 测试集之间有依赖关系，如HSM暂停需要IPI唤醒可用，先测IPI确定IPI能用之后，才能测HSM扩展
- 相比将Linux内核作为SBI测试框架的方法，它的运行时间短，覆盖应用广，能测试Linux并未使用的SBI调用，更适合集成测试
- RFENCE、SRST扩展仍然需要找到更好的测试方法

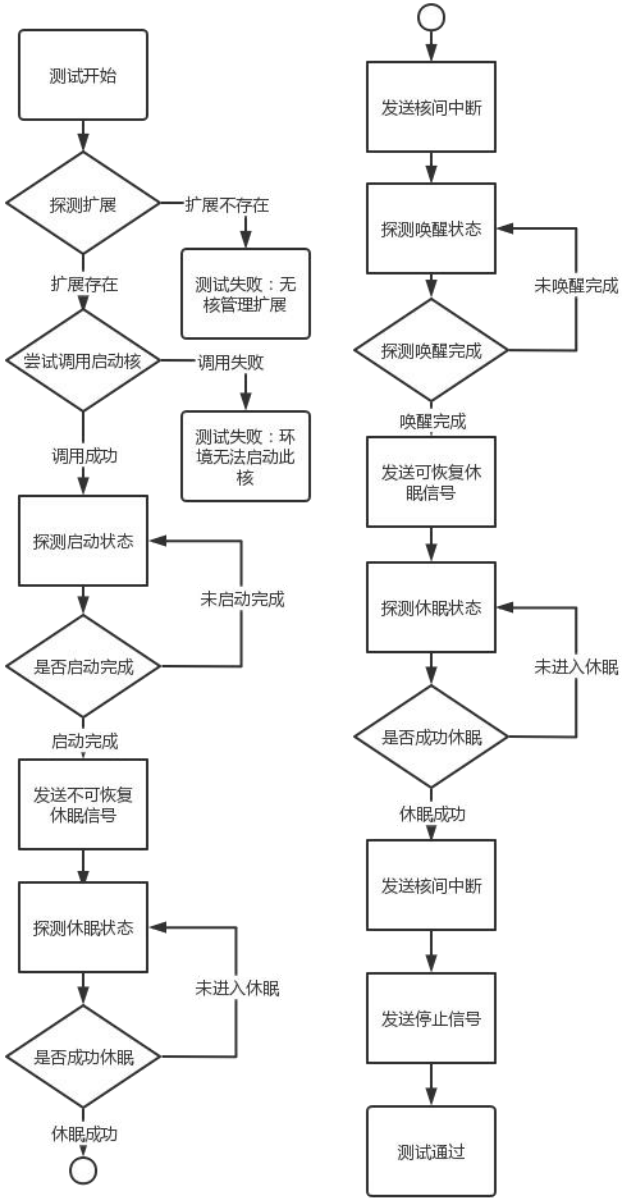


图 3 sbi-testing测试HSM扩展流程

快速陷入通道在SBI实现的应用

- RISC-V并未强制规定陷入栈的内容，它的上下文切换过程可定制，若给予上下文切换更多的信息，它的性能就可得到进一步提升
- 上下文调用时先保存部分寄存器，让高级语言判断是否进入完整流程，或给定需要设置的寄存器数量
- 尽量减少上下文切换对空间局部性的破坏
- 向量化陷入：硬件取向量，分流mtime、msoft等中断过程和异常过程，进一步细化通路，明确上下文保存需求
- 不同等级的上下文保存到不同结构体中，地址存于突发寄存器，快速处理程序可为完整处理程序提供参数
- 项目地址：https://github.com/YdrMaster/fast-trap

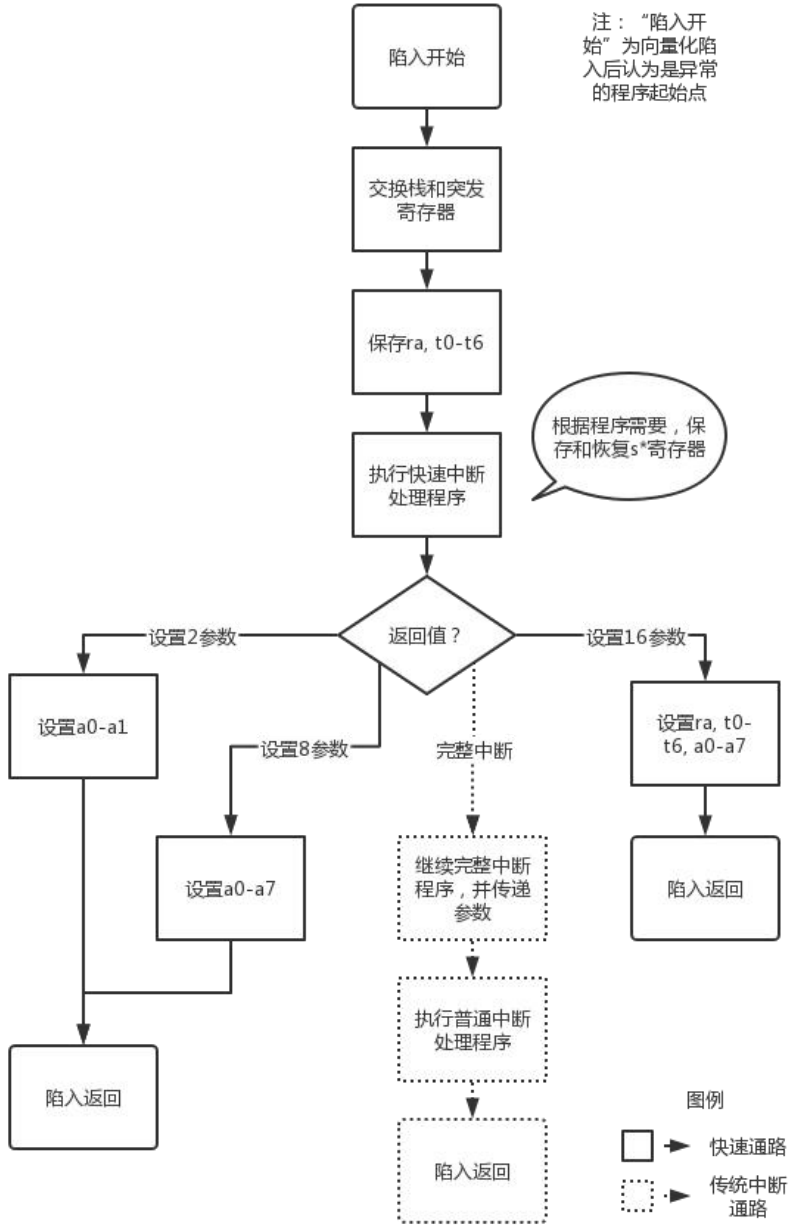


图 4 fast-trap RISC-V快速中断通路

RustSBI仍然要解决哪些问题？

- 兼容已有的引导程序生态
- 帮助开发虚拟化、模拟器软件（valheim、salus等）
- 提高新平台适配的效率
- 解决思路：增加独立包模式之外的第二种开发模式

RustSBI原型设计系统

- RustSBI需要的维护难度更小、容易和C语言生态结合的开发模式
- 软件架构：静态驱动支持、统一的引导路径、设备树支持以及“平台通用包”
  - 驱动开发模式仍然需要定型，可以确定的是尽量接受更多厂家的源码
  - 引导路径的代码可复用，也可定制。可能遇到的局限性是平台模型（内存一致性、异构性等）
  - 设备识别支持通常来说是设备树，可读取设备树准备驱动，或者选择硬编码
  - “平台通用包”打包尽可能多的驱动，牺牲启动时间在单个包获得跨平台特性，对有些用户是有好处的
- 与独立包模式相比，提供一种适配新平台更快速的解决方案。并非专注于“原型设计”
- （正在开发）免费获得固件高级功能
  - 预期中能将Penglai TEE、@dramforever的OpenSBI-H和Raven固件调试器有机结合在一起
  - 适配完毕本平台后，选择高级功能的一种实现，就能免费获得此功能
- 灵感来源：psicasbi（https://github.com/retrhelo/psicasbi）

让RustSBI兼容已有的引导程序生态

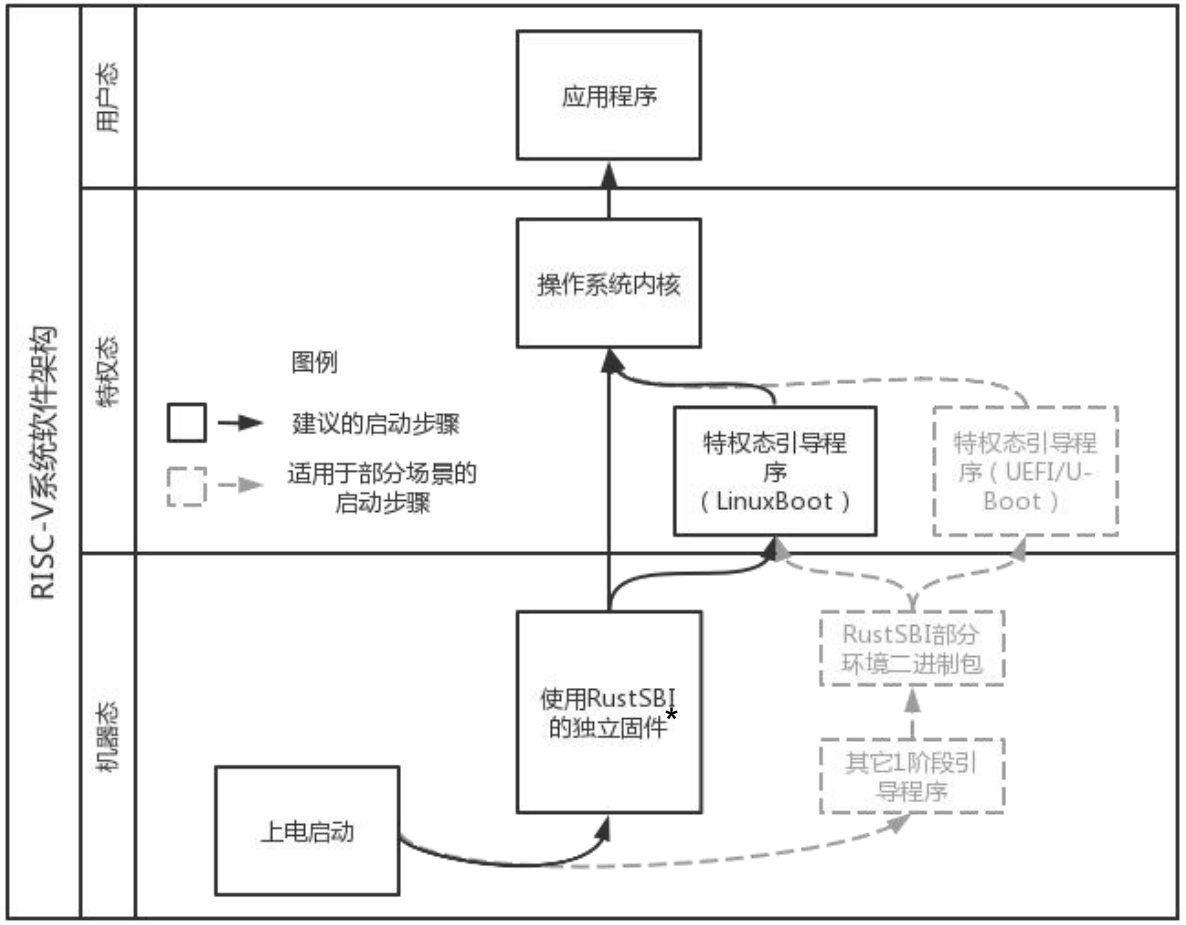


图 5 RISC-V引导系统软件架构（无虚拟化）

*如Oreboot、RustSBI独立支持包等

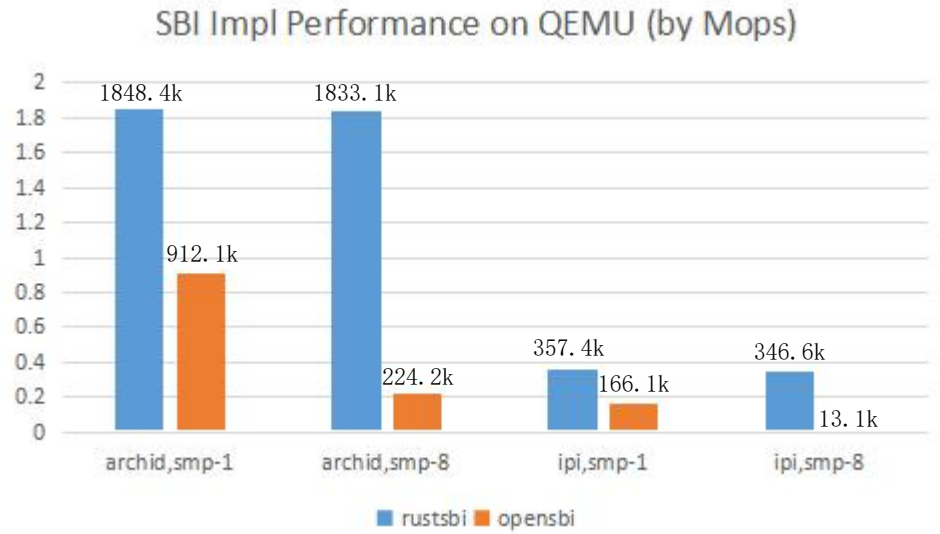
- 已有的程序与标准包括：UEFI、U-Boot和EDK II
- 和相关厂家沟通，认为这些引导程序都属于S态应用，它能够被M态环境启动即可。所以这些标准和软件可以称作是SBI环境的应用之一，而不是竞争者
- SBI需要做的工作是准备好M态环境，然后启动这些S态引导程序
- Unmatched主板的支持工作是一个好的开始，希望有熟悉U-Boot等软件的朋友编写一个简单的例子，能够启动S态U-Boot即可。这部分支持可以合并到原型设计系统里

引导阶段：是什么、为什么、怎么做

- 由二进制拼接得到的引导程序依次启动的步骤
- 内存的成本
  - 举个例子，SRAM立即可用，但比较贵；DDR SDRAM比较便宜，但需要训练
  - 上电后引导程序立即在片内SRAM中运行，片内SRAM不足以运行整个操作系统，或不足以运行整个引导程序。将DDR训练并启动，以支持操作系统的运转
  - 也有不少芯片和这个规律略有差别，但大体上动机是相似的
- 观点：限制引导阶段的数量
  - 除了硬件限制（内存成本、ROM代码等）外，不增加额外的阶段
  - 降低软件设计的难度，降低适配成本

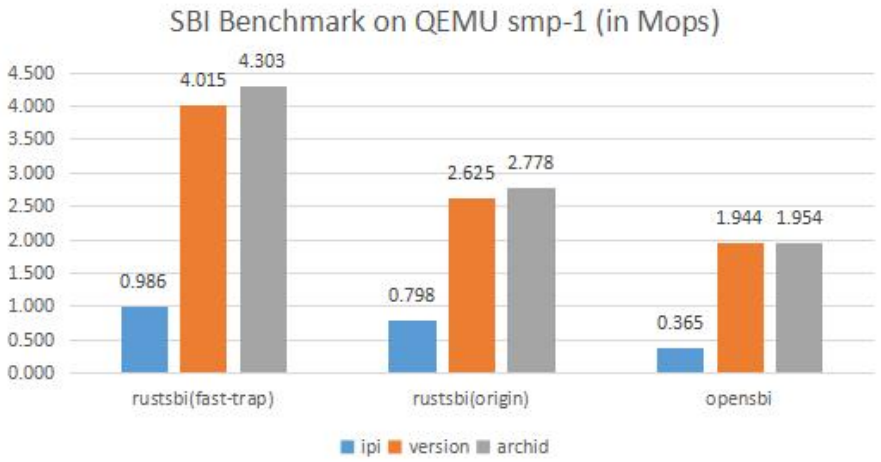
性能测试！

qemu, smp-1, archid, opensbi	1436907	131072	10000000	1.096273041
qemu, smp-1, archid, rustsbi	709103	131072	10000000	0.541002655
qemu, smp-8, archid, opensbi	5846193	131072	10000000	4.46029129
qemu, smp-8, archid, rustsbi	714999	131072	10000000	0.545500946
qemu, smp-1, ipi, opensbi	7886913	131072	10000000	6.017237091
qemu, smp-1, ipi, rustsbi	3667553	131072	10000000	2.79812088
qemu, smp-8, ipi, opensbi	100294809	131072	10000000	76.51886673
qemu, smp-8, ipi, rustsbi	3782170	131072	10000000	2.885566711



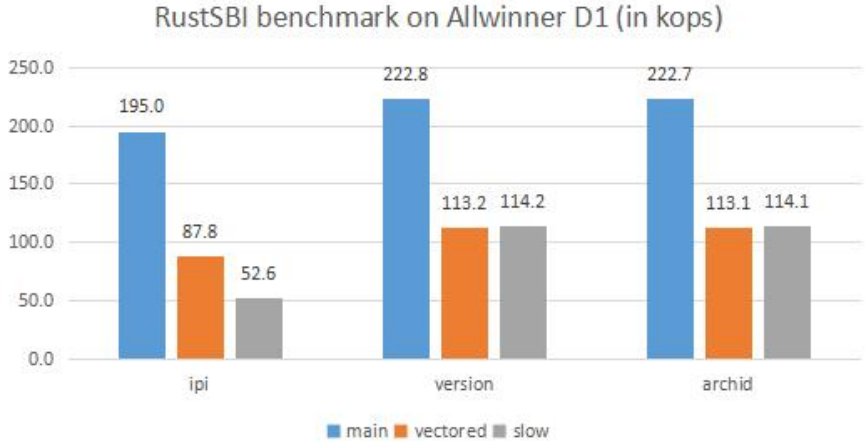
在某物理硬件 + WSL2 的 qemu 上，两个分支的性能对比以及作为对比的 opensbi 数据如下：

SBI	spec_version	marchid	ipi	latency
rustsbi (fast-trap)	2324058	2490860	10141260	75.0%
rustsbi	3600115	3809638	12529395	100.0%
opensbi	5117689	5143339	27404345	188.9%



Allwinner D1上测试性能数据见下表：

branch	spec_version	marchid	ipi	latency
main	107700360	107756601	123088023	48.5%
vectored	212014619	212212320	273411079	100.0%
slow	210191584	210286668	456087262	125.6%



- 在相同环境下测试RustSBI和OpenSBI的性能
- RustSBI 0.3.0，rustc 1.65.0，--release（-Os）
- OpenSBI 1.0.0，gcc 11.2.0，-Os
- QEMU环境#1：QEMU 7.1.0, Windows 11 22H2, Intel i7-11700@2.5GHz
- D1环境：Allwinner Nezha, T-Head C906@240MHz

测试2、3的数据来源：https://github.com/YdrMaster/fast-trap#%E6%80%A7%E8%83%BD%E6%B5%8B%E8%AF%95

性能测试分析

- 在模拟器上，rustsbi的延迟约比opensbi低47%。在支持多核的环境中，rustsbi无论核数性能波动不大，但随着核数增加，opensbi的性能显著下降。
- 应用fast-trap后，rustsbi的综合延迟能在真实硬件上进一步降低61%，在模拟器上进一步降低33%。如果使用常见的优化vectored trap而不用fast-trap，在真实硬件上在不同测例中表现不同，综合延迟只能降低约21%。
- rustsbi对opensbi的性能提升可能来自rust语言的零抽象开销、泛型和代数数据类型（尤其是never type），rustsbi的实例模型以及独立包支持更高的可定制性。
- rustsbi若使用fast-trap，将通过快速中断通路显著提升性能，这无论在模拟器上还是真实硬件中都有效。比较模拟器和具备缓存的真实硬件，我们认为真实硬件性能受空间局部性的影响更大，fast-trap的通路能尽可能减小上下文切换对空间局部性的破坏。
- rustsbi的代码对核的数量不敏感，对此部分数据的具体解释有待进一步分析。

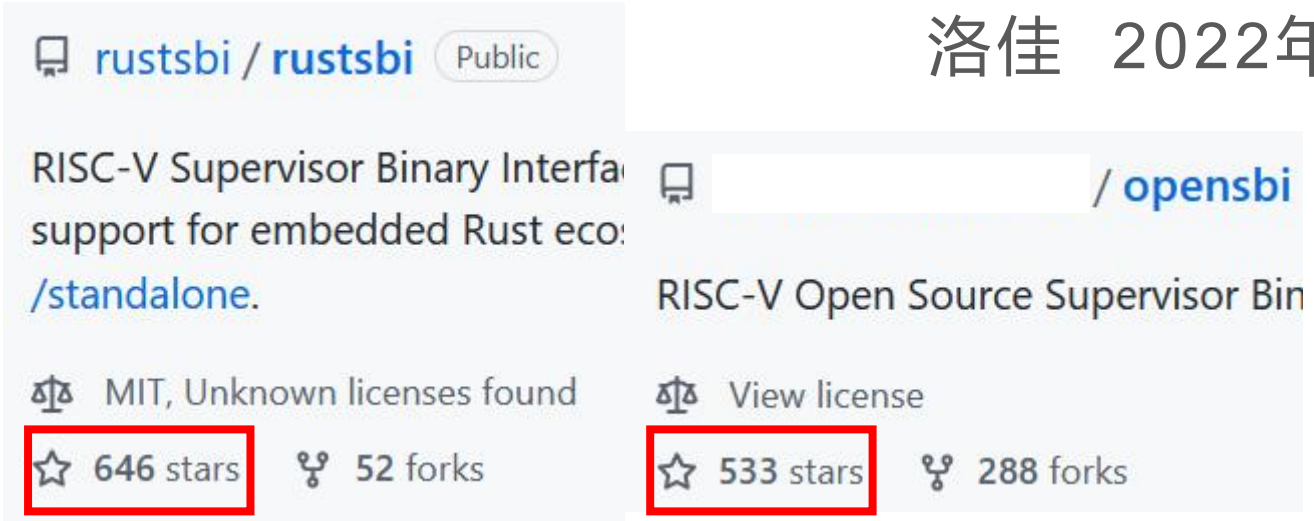
致谢

- 感谢Rust语言为我们提供一种新的认识嵌入式生态的机会
- 感谢Rustcc嵌入式社区、TUNA Embedded社区和RustSBI谈天说地群友在关键问题上的答疑解惑和积极交流，社区良好的氛围对生态发展有积极作用
- 感谢@YdrMaster、@duskmoon314、@OrangeCMS和更多直接参与到RustSBI开发的贡献者贡献代码和性能测试数据，社区无数的贡献者完善生态和找到bugs，@双倍多多冰、@dramforever、@大佬鼠的小粉丝和更多熟悉RISC-V的朋友答疑解惑
- 感谢Sipeed公司的硬件设备支持，以及更多公司的交流和沟通机会

谢谢各位

RustSBI 0.3.0 新版本发布与应用场景介绍

洛佳　2022年11月



社区同步更新（RustSBI）

- RustSBI 0.3.0发布（@luojia65，@YdrMaster，@duskmoon314，@OrangeCMS）
  - 基于实例的SBI实现、机器环境无关的SBI实现，用于开发虚拟化环境，可用stable Rust编译
  - MachineInfo结构体可用于开发非裸机环境，如跨指令集模拟器
  - 构造器语法
  - 对每个trait的引用实现这个trait，对never type（Infallible）实现所有trait
  - 添加了丰富的文档，帮助虚拟化、模拟器和裸机环境的开发工作
  - 所有弃用的遗留扩展被归纳到legacy feature中，实现了LEGACY_CLEAR_IPI，可探测legacy扩展
  - 去除了对alloc包的依赖，RustSBI没有堆也可以运行了

社区同步更新（独立包支持，1/2）

- RustSBI-QEMU独立包项目（@YdrMaster，@duskmoon314）
  - sbi运行时独立为额外的sbi-rt包
  - 性能测试：中断发起到响应、base扩展查询。编写专门的性能测试内核
  - 改用标准的RustSBI实例实现法，legacy扩展不经过RustSBI而是手动编写
  - 增加了github actions，自动单元测试，以及版本兼容测试
  - 使用os-xtask-utils重新编写xtask模块
  - 使用fast-trap增加上下文切换速度
  - 独立clint模块到aclint包，独立sifive-test到sifive-test-device包
  - HSM扩展改用原子变量，以及小修复pmp外设源码
- rustsbi-k210独立包项目（@luojia65）
  - 使用HSM扩展启动多核
  - 使用了稳定的asm_sym语法，更新到riscv 0.9.0

社区同步更新（独立包支持，2/2）

- rustsbi-d1独立包项目（@YdrMaster）
  - 重构项目到spl、see等模块，增加了性能测试内核和功能测试内核
  - 可以启动ddr控制器，成功初始化，并且把内核放入ddr中运行
  - 补充clint和plic外设
  - xtask：支持flash命令，可格式化flash，选择烧写后是否重启
  - 支持fast-trap项目以显著提升调用速度
  - 代理rdtime伪指令
- rustsbi-k510独立包项目（@Gstalker）
  - 已经实现了Andes PLIC_SW、串口驱动以及加密bootloader，能生成SD卡镜像
- rustsbi-bl808独立包项目
  - 论证阶段，正在考虑放在独立包中支持还是放在原型系统中支持

社区同步更新（生态圈包）

- sbi-testing测试套件 v0.0.1（@YdrMaster），便于验证SBI环境实现的正确性
  - 可以测试BASE、TIME、sPI和HSM扩展
- sbi-rt运行时支持包 v0.0.2（@YdrMaster，@luojia65），为特权软件提供SBI运行时库
  - 已支持SBI v1.0.0所有章节的函数和结构体，并满足#![deny(missing_docs)]
- sbi-spec定义与常数库 v0.0.4（@YdrMaster，@luojia65），避免记忆复杂的常数
  - 拥有一个与Rust Result API相同风格的SbiRet结构体
- fast-trap快速陷入处理（@YdrMaster），高效率的陷入处理流程
- dtb-walker设备树遍历器 v0.2.0-alpha.3（@YdrMaster），有助于解析设备树格式
  - 内置标准属性解析（如compatible等），零开销抽象、stable Rust，友好的遍历体验
- os-xtask-utils编译工具支持包 v0.0.0（@YdrMaster），良好支持各类独立包的编译体验
  - 可以用在操作系统和引导程序开发上

社区同步更新（周边项目）

- Oreboot引导程序项目（@OrangeCMS，https://github.com/oreboot/oreboot）
  - 有一次大的重构，确定了设备模型
  - 可以直接启动M态软件，此时没有提供SBI接口，可以作为异构系统的补充
  - 更新d1-pac依赖到0.0.30版本
  - 通过两个阶段引导支持了三款D1主板，包括DDR启动，能够引导LinuxBoot环境
  - 重新支持risc-v QEMU模拟器
- Salus微型虚拟化软件（https://github.com/rivosinc/salus）
  - 已修改项目文件的格式，正在论证如何将RustSBI和专有属性结合起来
- valheim模拟器（@imkiva，https://github.com/imkiva/valheim）
  - 可以启动rustsbi-qemu，目标是在一生一芯中和nemu竞争
- larva龙芯到RISC-V硬件加速模拟器（@xen0n，https://github.com/xen0n/larva）
  - 目前正在等待rust工具链更新

社区同步更新（RISC-V芯片支持）

- bl808-pac博流BL808芯片官方Rust支持包 v0.0.0（@YafeiJin）
- xuantie玄铁处理器支持包 v0.0.5（@luojia65）
- sifive-core SiFive/StarFive处理器核支持包 v0.1.0（@luojia65）
- d1-pac全志D1支持包 v0.0.30（@duskmoon314）
- hpm6750-pac支持包 v0.0.0（@luojia65）

虚拟化软件和模拟器软件

- 可信虚拟化
  - Hypervisor作为TEE的实现途径，如Salus项目
  - RustSBI有助于理清SBI适配的思路，降低开发负担，可用于Type-1和Type-2 Hypervisor的开发工作
- 硬件加速的跨指令集模拟器（LoongArch）
  - 若在龙芯上直接引导RISC-V架构的操作系统，仍然需要SBI接口实现
  - RustSBI可以编译到其它指令集架构的编译目标，辅助开发直接提供S态环境的支持程序
- 模拟器项目
  - 直接在M态上运行固件，或者参与S态模拟器的开发
  - 便于结合专有特性