

兼容三大主流架构的轻量化 Hypervisor 设计

- 兼容三大主流架构的轻量化 Hypervisor 设计

摘要

需求背景

ArceOS简介

架构设计

项目整体结构

具体架构

开发进度

项目运行方式

环境要求

RISC-V Linux

RISC-V rCore-Tutorial-v3

操作演示

感想

未来方向

致谢

摘要

本项目编写了一个轻量化的 hypervisor，兼容 x86，ARM-v8，RISC-V 三大主流架构，以达到运行市面上大多数开源操作系统的目的。

需求背景

hypervisor 在日常生活中有很多的应用，例如在电脑上运行虚拟机等等，目前较为著名的 hypervisor 是 Linux/KVM 以及 windows/Hyper-V，但是这两者分别依赖于 Linux 和 Windows 操作系统。都不属于轻量化的 Hypervisor。不能满足低功耗设备的要求，例如车载设备。

本项目致力于编写一个轻量化的 hypervisor，兼容 x86，ARM-v8，RISC-V，以达到运行市面上大多数开源操作系统，例如 Linux 等等的目的。

ArceOS简介

ARCEOS 是贾越凯学长编写的组件化操作系统，这个操作系统将一个常规的操作系统分为 app，lib，module 以及 crate 四层。通过分层的条件编译达到仅仅编译需要使用的操作系统组件来轻量化操作系统的目的。其中每一个 app 代表一个 unikerne1。

我们在齐呈祥学长开发的 crate 层组件 hypercraft（支持 RISC-V 的虚拟化）以及 app 层的 unikerne1: hv 的基础上，对 hypercraft 进行了架构的拆分以及加工，将 hypercraft 中内存虚拟化的部分剥离出来形成了 crate 层的组件 guest_page_table 以及轻量化的 hypercraft。并在 hypercraft 中加入了 x86 和 aarch 架构相关的代码。

架构设计

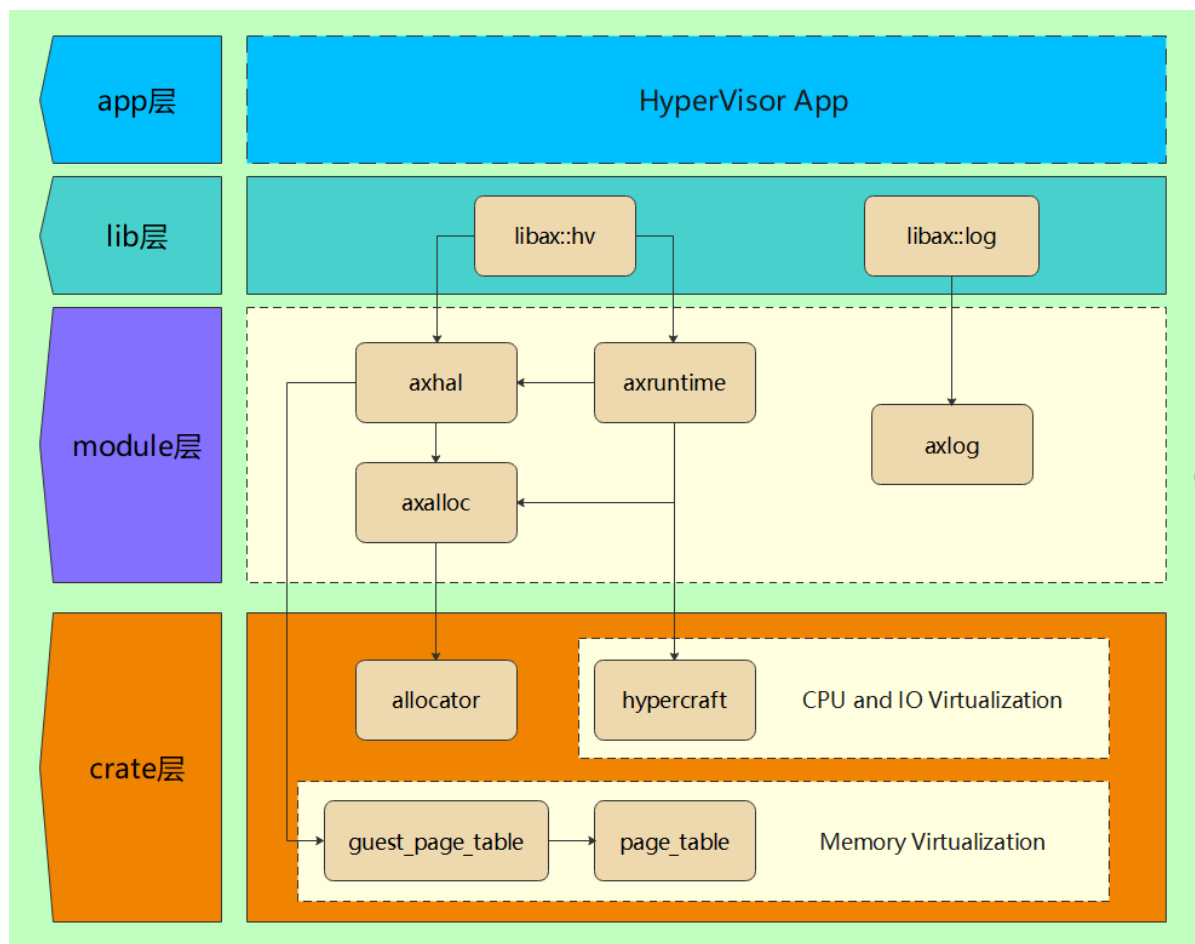
项目整体结构

以下是一个简单的项目树，表示这个项目的大致目录结构。

```
├─guest （Guest OS的集合，目前我们在开发过程中通常是用以下4个操作系统进行测试）
|   ├─hcMinikernel （本人自己写的minikernel，是一个非常简单的batchOS，仅支持RISC-V）
|   ├─linux （linux，目前有ARM-v8和RISC-V两种架构）
|   ├─nimbos （贾越凯学长写的教学性质的操作系统，支持x86，ARM-v8，RISC-V）
|   └─rCore-Tutorial-v3 （清华大学操作系统公开课使用的教学性质的操作系统，仅支持RISC-V）
├─hypervisor （我们写的Hypervisor）
|   ├─apps
|   |   └─hv （我们写的hypervisor的入口）
|   |       └─src
|   ├─crates
|   |   ├─guest_page_table （内存虚拟化模块）
|   |   |   └─src
|   |   └─arch （存放各大架构的页表配置）
|   ├─hypercraft （CPU及IO虚拟化模块）
|   |   └─src
|   |       └─arch
|   |           ├─aarch （存放ARM-v8架构相关代码）
|   |           ├─riscv （存放RISC-V架构相关代码）
|   |           └─x86 （存放x86_64架构相关代码）
|   └─modules
|       ├─axhal （实现了内存虚拟化模块向上提供的接口GuestMemoryInterface）
|       └─axruntime （实现了CPU虚拟化模块）
└─ulib
    └─libax （app层使用的lib，主要是起到一个承上启下的作用）
```

请注意，尽管 ArceOS 的代码结构极为复杂，通过条件编译后，除了少数用于打印错误信息的库，例如 `axlog` 等。其余不再上述项目树中的代码都不会被编译。

具体架构



上图是我们对于轻量化 Hypervisor 的架构设计，我们的 Hypervisor App 仅仅依赖 `libax::hv` 和 `libax::log` 两个 lib 层的模块。

其中 `libax::log` 是用于打印信息的模块，依赖于 module 层的组件 `axlog`。

`libax::hv` 是我们开发的 Hypervisor 库，这个库依赖于 `axruntime` 和 `axhal` 两个 module 层的组件其中 `axhal` 是编码硬件信息的 module 层组件，`axruntime` 是处理运行时异常的 module 层组件。我们在 `axhal` 中实现了用户页表以及 Hypervisor 内核页表。在 `axruntime` 中实现了页表分配和运行时异常处理，包含 `Trap` 以及用户机退出等等。

module 层还依赖于另一个组件 `axalloc`，这是一个内存分配算法相关的组件。依赖于 crate 层组件 `allocator`。

`guest_page_table` 和 `hypercraft` 是我们开发完善的 crate 层组件，这两个组件中定义了 module 层组件需要的所有的 trait

开发进度

截至目前，RISC-V 架构下的工作已经完全开发完成，能够运行 Linux 和支持 GPU 设备的 `rCore-Tutorial-v3`，下文中也给出了测试样例。

x86 和 ARM 架构下我们完成了 CPU 和内存的虚拟化。目前还不足以支持 Linux 的运行。

项目运行方式

环境要求

本仓库要求在 QEMU-7.0.0 下运行，同时请配置 rust 工具链为 rust-nightly-2023.5.23 以及更新的版本，否则可能无法编译 RISC-V H Extension

目前仓库中支持在 Hypervisor 上运行 linux 和 rCore-Tutorial-v3 两种操作系统，编译运行方式分别如下：

RISC-V Linux

必要文件 rootfs.img，这个文件有 1G 大，不能够直接在 gitlab 上传递，所以请在百度网盘上下载下来后放到项目的 guest/linux 目录下。

[链接](#)

提取码：b8p5

```
# linux
cd guest
# 这一步需要保证您是ubuntu系统，并且系统上装好了dtc (Device Tree Compiler)
make linux
cd ../hypervisor
# 运行这一步前，请确保您已经下载了rootfs.img并将其解压至guest/linux下
make A=apps/hv ARCH=riscv64 SMP=1 HV=y GRAPHIC=n run
```

RISC-V rCore-Tutorial-v3

```
# rCore-Tutorial-v3
cd guest
# 这一步需要保证您是ubuntu系统，并且系统上装好了dtc (Device Tree Compiler)
make linux
cd ../hypervisor
# 运行这一步前，请确保您已经下载了rootfs.img并将其解压至guest/linux下
make A=apps/hv ARCH=riscv64 SMP=1 HV=y GRAPHIC=n run
```

操作演示

主题: 陈岳的快速会议

日期: 2023-06-07 15:50:07

录制文件: <https://meeting.tencent.com/v2/cloud-record/share?id=522814cd-f6b4-49cc-b56d-e03788387c0f&from=3>

访问密码: ayAU

感想

本项目从今年的3月份开始，原本是我的大作业项目，最初的计划仅仅是设计一个 RISC-V 版本的 Hypervisor，后来在调研过多个 Hypervisor 项目以及和许多学长、老师们交流之后，决定在 arceos 上写一个兼容 x86，arm，riscv 的 Hypervisor。随着项目的不断推进，我也获得了不少的收获，学会了许多虚拟化相关的知识。

未来方向

正如前文所说，我们只完成了 RISC-V 的大部分代码，后续我们会完善 x86 以及 ARM 的 Hypervisor 设计与实现。最终希望能够在三大架构上分别跑通 Linux 操作系统。

致谢

感谢陈渝老师、向勇老师和任炬老师的指导！

感谢校外导师提供的帮助和硬件支持！

感谢贾越凯、齐呈祥、丁韶峰以及其他学长的帮助！