



北京航空航天大学  
BEIHANG UNIVERSITY

# 基于Rust的嵌入式虚拟机监视器 及热更新技术

报告人：王雷





# 目录

CONTENTS

一

项目背景

二

基于AArch64的  
Rust虚拟机监视器设计与实现

三

资源隔离策略与实时虚拟化

四

嵌入式虚拟机监视器热更新技术

五

实验评估

六

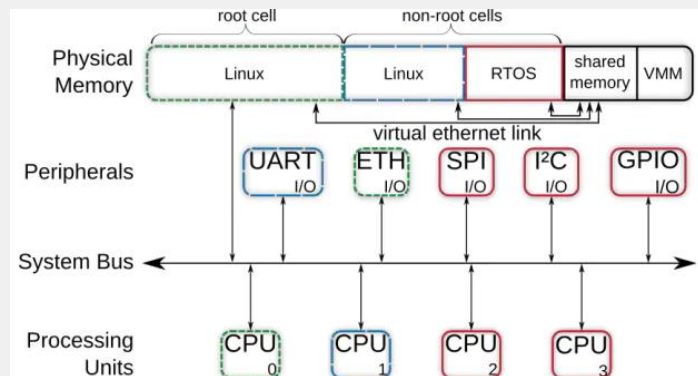
下一步工作





P A R T O N E

## 项目背景



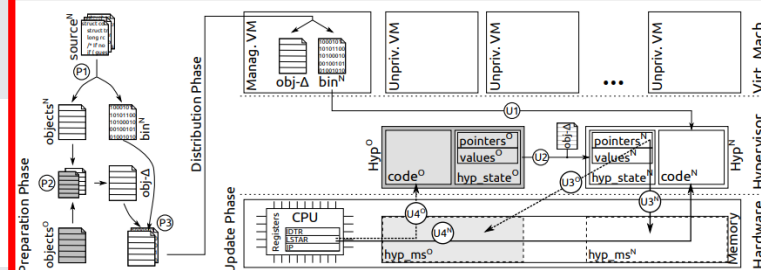
## 嵌入式虚拟化

- 提高资源利用率
- 保障任务间的隔离性
- 实时虚拟化



## Rust编程语言

- 内存安全与线程安全
- 零成本抽象
- 活跃的开源社区



## 监视器热更新

- 替换虚拟机监视器镜像
- 保障虚拟机上任务的有效执行



## 面向服务的虚拟化方案

- KVM、Xen
- 提高资源利用率，在桌面、服务器和云计算领域取得巨大成功
- 提供虚拟机迁移、动态升级等热更新方案
- 难以保障任务间的隔离性与实时性
- 使用传统C语言编程实现，难以保障内存安全



## 基于Rust的虚拟化方案

- StatoVirt、Capsule
- 充分利用Rust语言特性，提供理想的代码安全
- 能够为用户提供基本的虚拟化服务，实现了虚拟机迁移的热更新方案
- 采用Type II设计模式，难以保障任务间的隔离性



## 嵌入式虚拟化方案

- Xvisor、ACRN、Jailhouse
- 采用Type I设计模式，能够保障任务间的隔离性与实时性
- 缺少热更新技术的支持
- 使用传统C语言编程实现，难以保障内存安全





## 研究目标

利用高级程序语言Rust设计实现了一个面向嵌入式场景的Type I虚拟机监视器“Rust-Shyper”。其目标在于充分利用Rust语言安全特性的同时，提供灵活便捷的虚拟化服务以及热更新机制，保障虚拟机的**隔离性、实时性、可靠性**等多种需求。

## 研究内容

- Rust语言虚拟机监视器：使用Rust语言完整实现一个Type I虚拟机监视器，包括异常处理、中断控制、核心调度、模拟设备等12个功能模块。
- 模拟设备与隔离性：为提高资源利用率，实现多种模拟设备和差异化的资源分配策略，在保障隔离性的同时满足不同虚拟机的需求。
- 实时虚拟化技术：为针对性的保障部分虚拟机的实时需求，实现资源直通以及实时虚拟化技术。
- 虚拟机监视器热更新技术：为了保障虚拟化服务的可靠性，实现了虚拟机迁移和监视器动态升级两类监视器热更新机制，提供更高效率的更新方式。



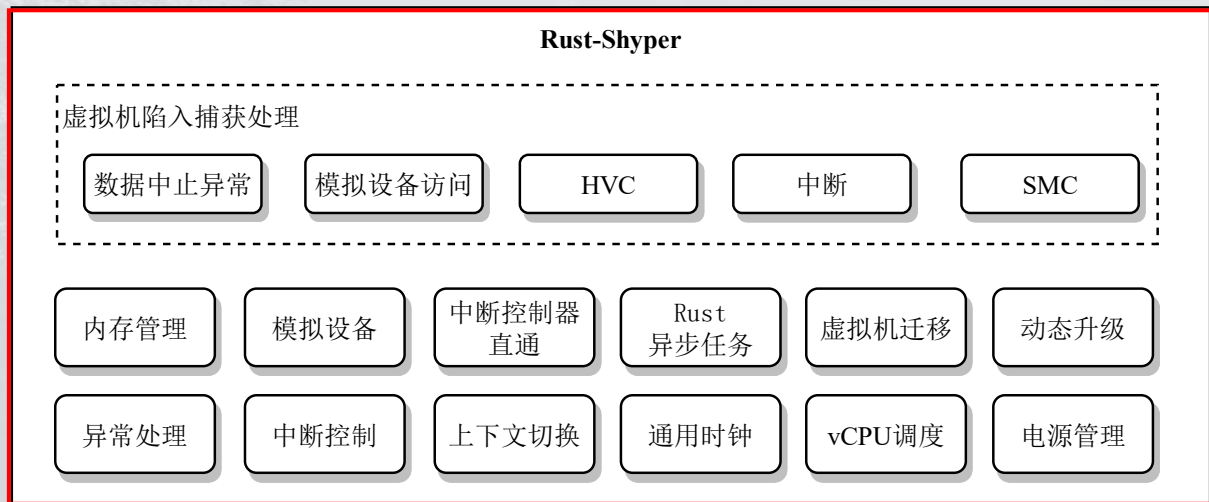


P A R T T W O

---

# 基于AArch64的 Rust虚拟机监视器设计与实现





## Rust-Shyper特点

- 基于Rust语言的Type I虚拟机监视器，具备更好的隔离性和更轻量的代码实现。能够支持Linux、裸应用等系统软件的运行
- 灵活、动态创建并配置虚拟机，能够为实时操作系统创建实时虚拟机，保障嵌入式场景的实时需求
- 具备资源隔离属性，保障不同虚拟机间的隔离性
- 实现多种模拟设备，提供基本的磁盘、网络、串口等设备的虚拟化实现，提高资源利用率
- 提供传统虚拟机迁移机制以及本地热更新机制，保障虚拟机监视器正常运行过程中的可靠性



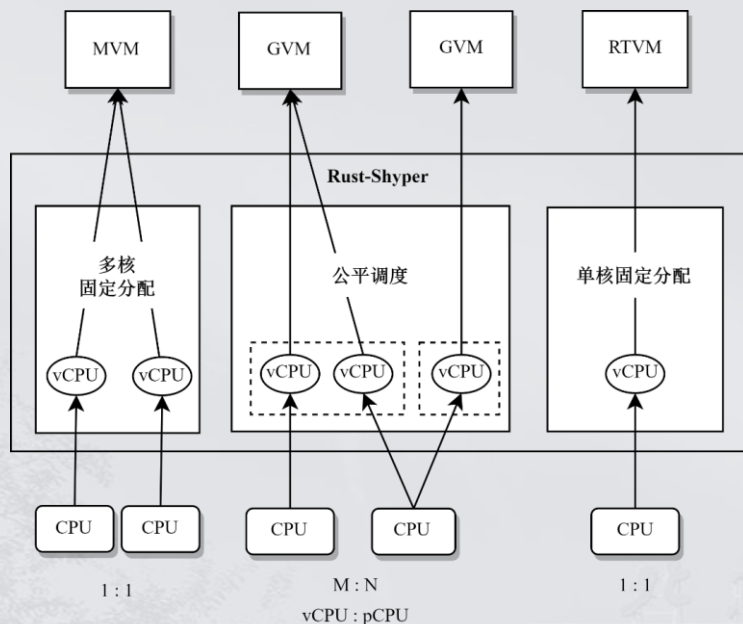


P A R T                      T H R E E

---

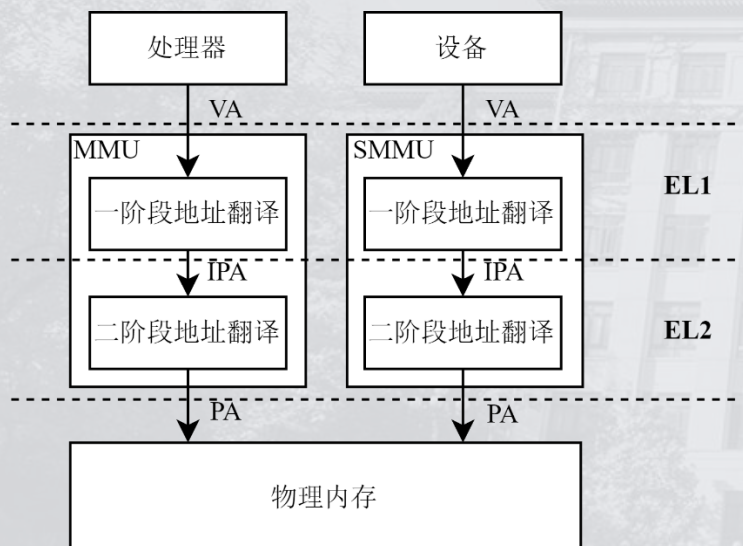
## 资源隔离策略 与实时虚拟化



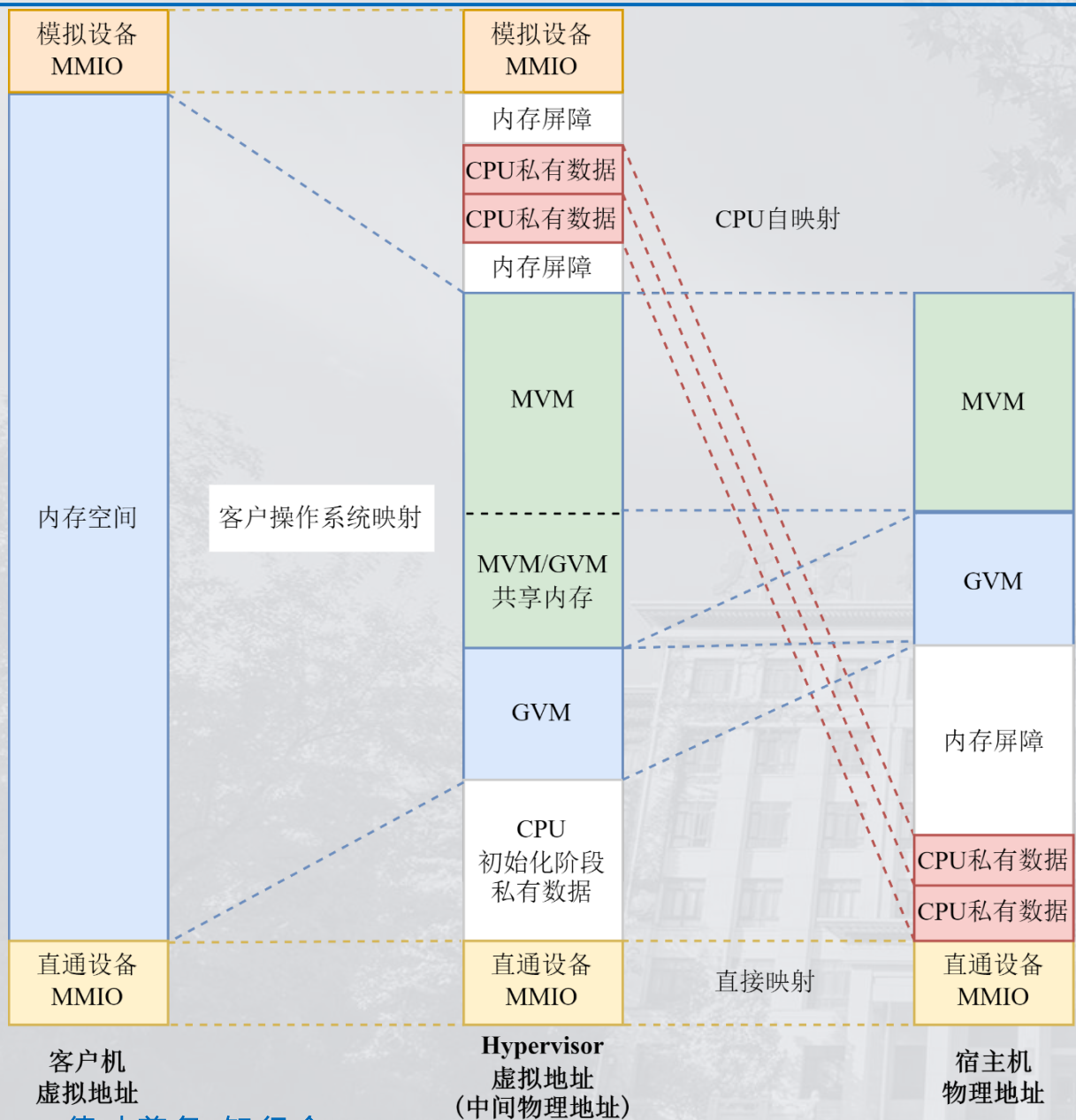


## 资源隔离策略

- 核心隔离需要保障虚拟机的vCPU运行状态不应受到其他虚拟机的篡改
- 中断隔离需要保障虚拟机能够正确接收外部设备或软件生成的中断信号，同时不会收到不属于该虚拟机的非法中断信号
- 设备隔离需要保障虚拟机的设备状态无法被其他虚拟机访问或修改
- 访存隔离需要保障虚拟机无法访问不属于自己的内存地址空间







## 二阶段地址翻译

- 虚拟机的虚拟地址 (GVA, Guest Virtual Address) 经过第一轮地址翻译后得到中间物理地址 (IPA, Intermediated Physical Address), 一阶段地址翻译页表由虚拟机提供
- IPA经由第二轮地址翻译后得到真实物理地址, 二阶段地址翻译页表由虚拟机监视器提供
- 通过对第二阶段地址翻译页表的配置, Rust-Shyper可以在保障隔离性的同时, 实现虚拟机间共享内存的效果

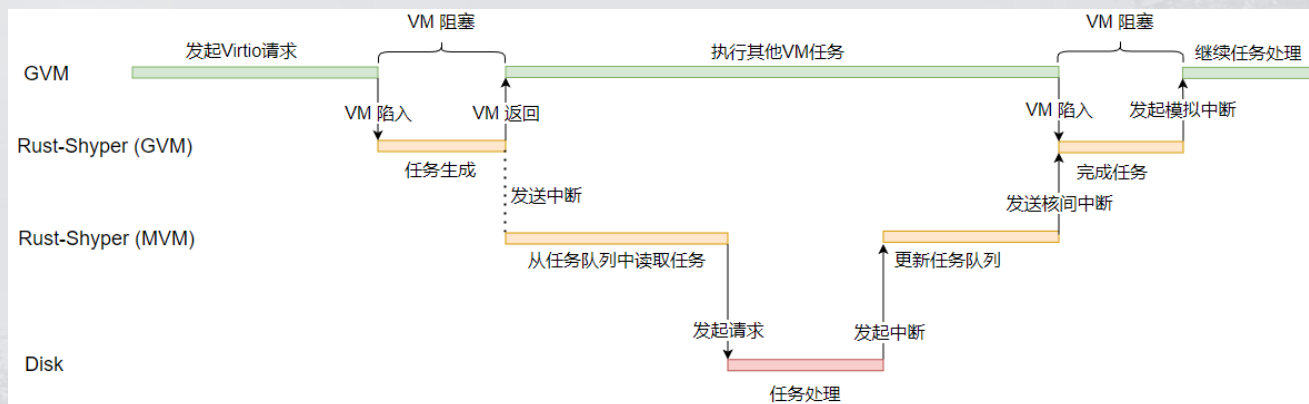




设备名称	设备虚拟化方法	虚拟机
中断控制器	直通	RTVM
	全模拟	MVM/GVM
串口	直通	MVM/RTVM
	Virtio	GVM
磁盘	直通	MVM
	Virtio	MVM/GVM
	中介传递	GVM/RTVM
网卡	直通	MVM
	Virtio	GVM/RTVM

Rust-Shyper所支持的设备虚拟化方法及设备类型





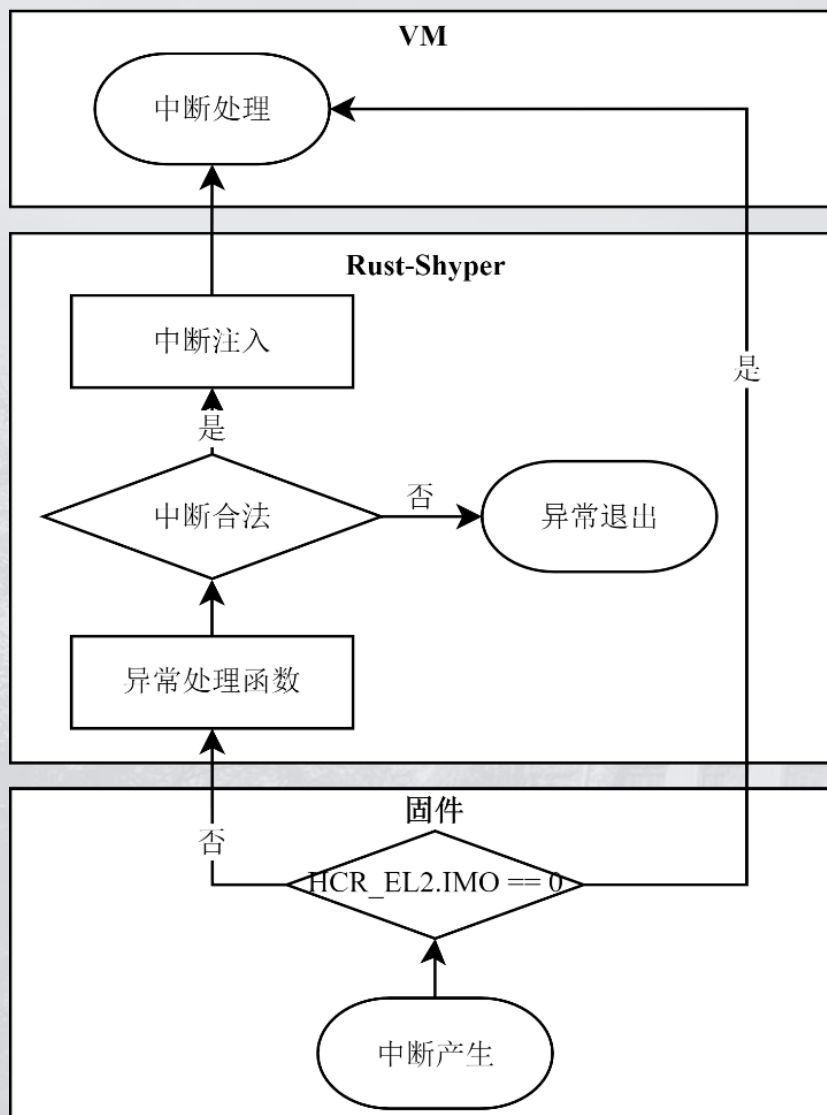
## 中介传递设备与Rust异步任务

- GVM发起Virtio磁盘请求陷入Rust-Shyper
- Rust-Shyper截获请求并将任务添加至异步任务队列，依照队列状态选择是否向MVM发送核间中断
- MVM读取异步任务并调度执行
- 任务完成后向GVM发送核间中断，将Virtio磁盘对应的中断注入至GVM中

```

1 function ADD_ASYNC_TASK(VM, ASYNC_FUNC)
2     VMID ← VMid();
3     Future ← ASYNC_FUNC();
4     Async_Task ← 依照VMID, Future构建异步任务结构体;
5     增加异步任务Async_Task至异步任务队列ASYNC_TASK_LIST中;
6     if 当前核心为MVM所在核心 then
7         调度执行异步任务;
8     end
9 end
    
```





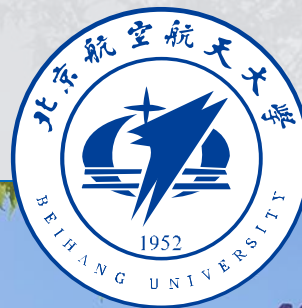
## 中断虚拟化

- 设置HCR\_EL2.IMO
- 直通GICV, 模拟GICD
- 拦截虚拟机GICD请求, 通过中断位图判断中断是否合法
- Rust-Shyper完成中断处理或中断注入

## 中断控制器直通策略

- 设置HCR\_EL2.IMO
- 直通GICC、GICD
- 虚拟机GICD请求不被Rust-Shyper拦截, 由虚拟机直接完成中断处理





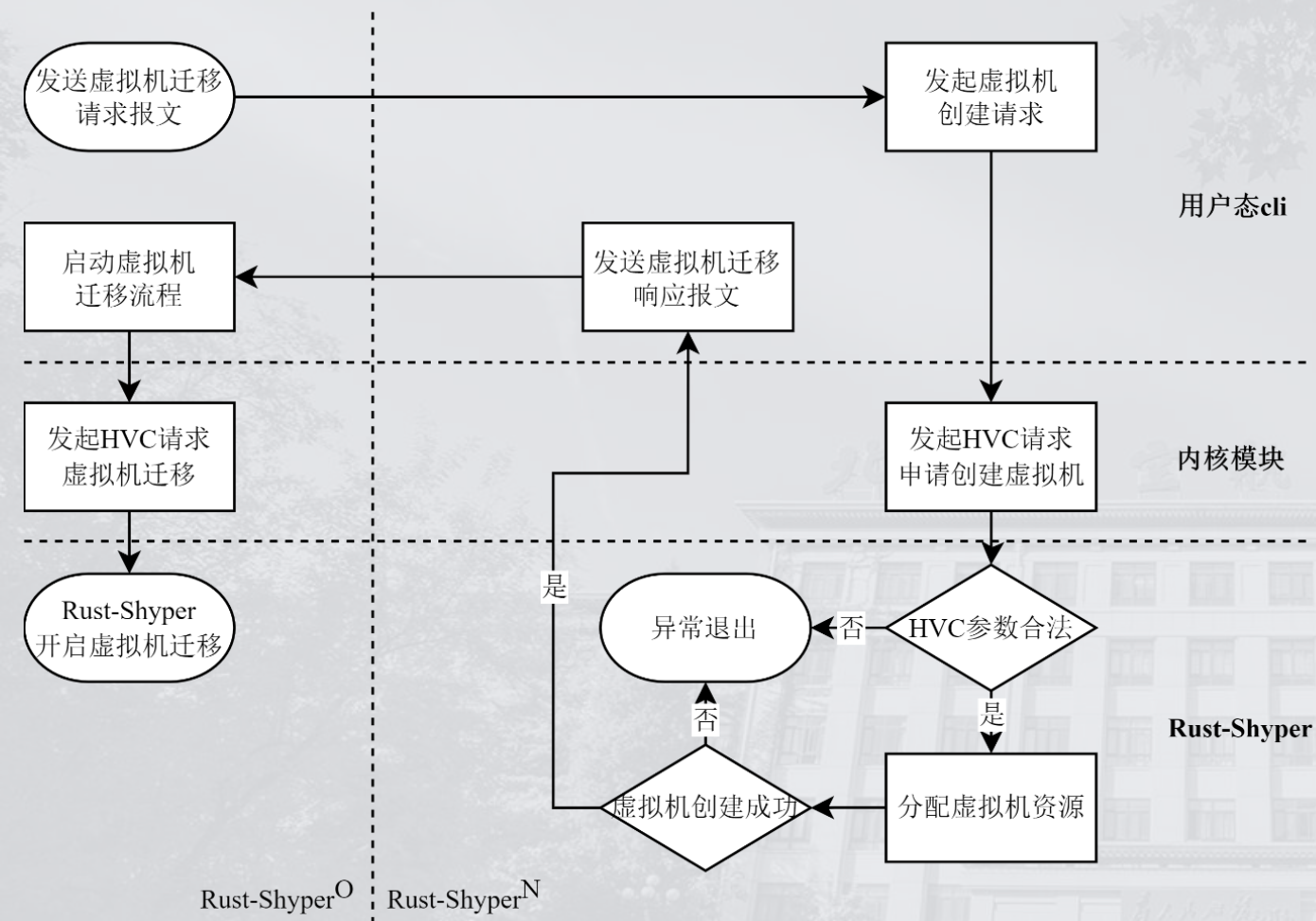
P A R T F O U R

# 嵌入式虚拟机监视器 热更新技术



## 虚拟机迁移技术

- Rust-Shyper虚拟机迁移由MVM通过HVC特权指令发起并执行
- 虚拟机迁移依赖Linux用户态网络服务进行数据传输
- 虚拟机迁移过程中，Rust-Shyper持续记录虚拟机产生的脏页信息，并修改虚拟机页表的读写权限
- 迁移的最后阶段，Rust-Shyper将剩余脏页与虚拟机状态集发送至接收端







状态数据分类	状态数据信息
虚拟机上下文	虚拟机寄存器组
	通用寄存器组
	栈寄存器
vCPU上下文	异常返回寄存器
	中断数量
	中断状态列表
中断控制器	GICV控制寄存器
	GICV优先级掩码寄存器
	模拟VirtioMMIO网卡
模拟设备	模拟VirtioMMIO串口
	vGICD结构体数据
	虚拟机核心数量
模拟中断控制器	核心私有vGIC数据列表

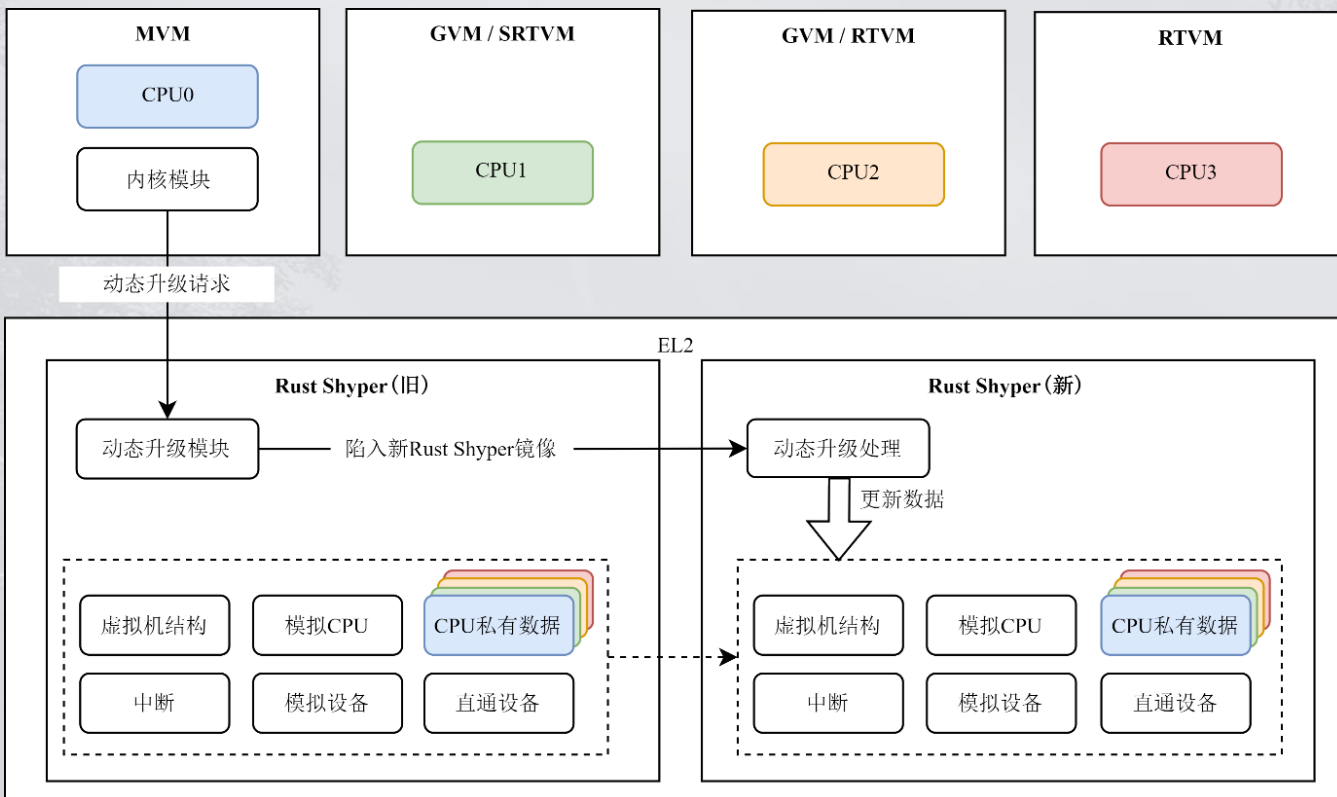
## 虚拟机迁移技术

- Rust-Shyper虚拟机迁移由MVM通过HVC特权指令发起并执行
- 虚拟机迁移依赖Linux用户态网络服务进行数据传输
- 虚拟机迁移过程中，Rust-Shyper持续记录虚拟机产生的脏页信息，并修改虚拟机页表的读写权限
- 迁移的最后阶段，Rust-Shyper将剩余脏页与虚拟机状态集发送至接收端



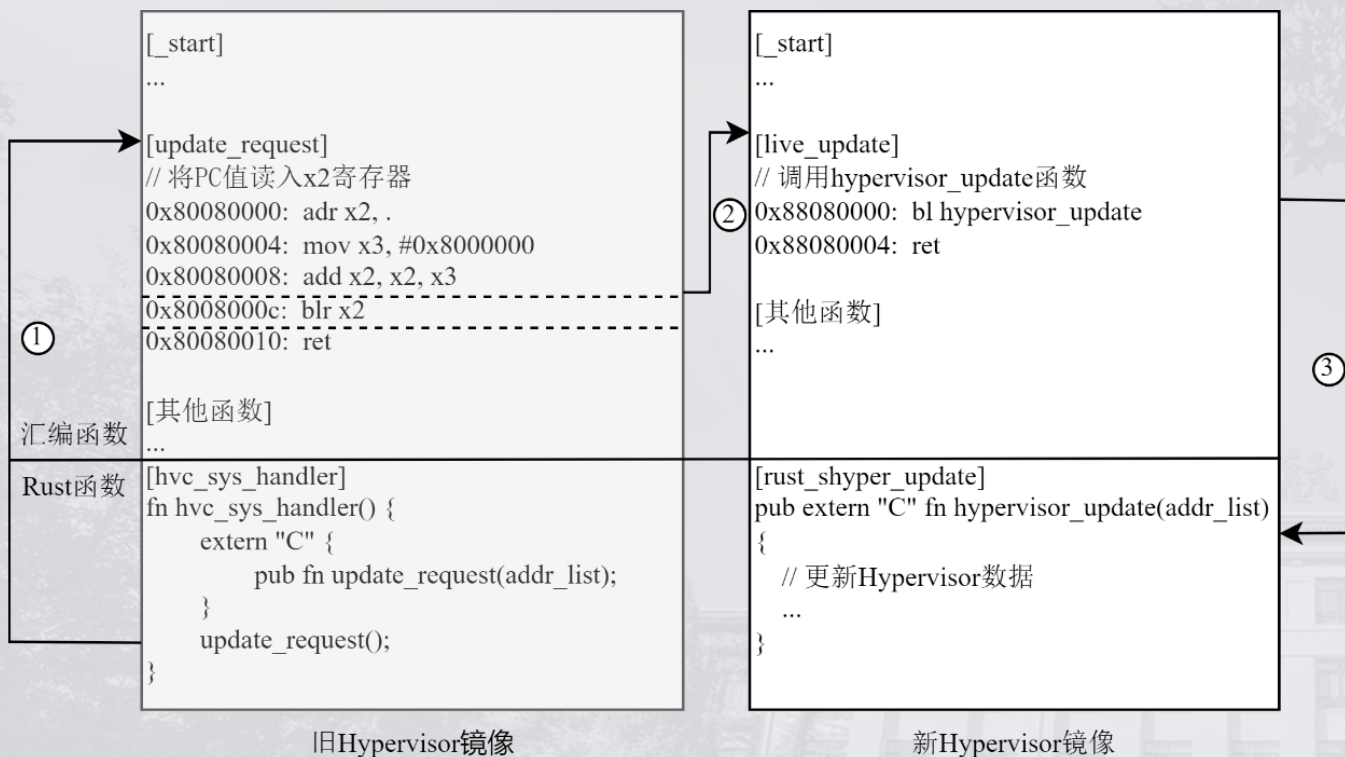
## 监视器动态升级技术

- 镜像拷贝阶段，MVM通过HVC发起动态升级请求，将新的Rust-Shyper镜像存入申请得到的EL2堆空间上
- 镜像切换阶段，CPU0将数据地址打包后，通过调用汇编函数live\_update跳转至新镜像的对应入口
- 监视器预升级阶段，CPU0预先陷入新虚拟机监视器完成部分数据结构初始化
- 数据更新阶段，CPU0调用live update 处理函数存储虚拟机监视器的数据；其他核心收到核间中断后需要将CPU结构体中的数据进行迁移





## 镜像切换



- 作为动态升级的发起者，MVM通过HVC触发hvc\_sys\_handler处理函数
- Rust-Shyper主动调用update\_request汇编处理函数，并计算新镜像处理函数的入口地址
- 跳转至新镜像live\_update函数
- 调用Rust实现的hypervisor\_update函数完成新镜像数据更新动作



```
1 function RUST_SHYPER_UPDATE(Address_List, Pre_Alloc)
2   if Pre_Alloc指明为预升级 then
3     heap_init();
4     mem_heap_region_init();
5     vm_config_table_update(Address_List.vm_config);
6     vm_list_alloc(Address_List.vm_list);
7     vcpu_list_alloc(Address_List.vcpu_list);
8     cpu_if_alloc(Address_List.cpu_if);
9     ipi_handler_list_update(Address_List.handler);
10    arch_timer_update(Address_List.timer);
11    interrupt_update(Address_List.interrupt);
12    emu_dev_list_update(Address_List.emu_dev);
13    gic_lrs_num_update(Address_List.gic_lrs);
14    return;
15  end
16  if 当前核心为MVM所在核心 then
17    vm_list_update(Address_List.vm_list);
18    vcpu_update(Address_List.vcpu_list);
19    current_cpu_update(Address_List.cpu);
20    vm_region_update(Address_List.vm_region);
21    heap_region_update(Address_List.heap_region);
22    vm_if_list_update(Address_List.vm_if);
23    mediated_blk_list_update(Address_List.med_blk);
24    shared_mem_list_update(Address_List.share_mem);
25    cpu_if_update(Address_List.cpu_if);
26    async_task_update(Address_List.cpu);
27  else
28    current_cpu_update(Address_List.cpu);
29  end
30  fresh_hyper();
31 end
```

// 为预升级动作  
// Rust堆初始化  
// 内存堆空间初始化  
// 更新虚拟机配置  
// 创建虚拟机列表  
// 创建vCPU列表  
// 创建CPU接口  
// 更新核间中断处理函数  
// 更新时钟状态  
// 更新中断信息  
// 更新模拟设备  
// 更新GIC相关全局变量

// 为数据更新动作  
// 更新虚拟机列表  
// 更新vCPU列表  
// 更新CPU私有数据  
// 更新虚拟机内存信息  
// 更新Rust-Shyper堆空间信息  
// 更新虚拟机接口信息  
// 更新中介传递磁盘设备  
// 更新共享内存信息  
// 更新CPU接口信息  
// 更新异步任务信息

// 更新CPU私有数据

数据名称	变量类型	数据功能
DEF_VM_CONFIG_TABLE	VmConfigTable	虚拟机配置列表
VM_LIST	Vec<VM>	虚拟机列表
VCPU_LIST	Vec<Vcpu>	vCPU 列表
CPU_IF	Vec<CpuIf>	CPU 接口
IPI_HANDLER_LIST	Vec<IpiHandler>	核间中断处理函数列表
TIMER_FREQ	usize	时钟频率
TIMER_SLICE	usize	时间片长度
INTERRUPT_HYPER_BITMAP	BitMap<BitAlloc256>	虚拟机监视器中断位图
INTERRUPT_GLB_BITMAP	BitMap<BitAlloc256>	全局中断位图
INTERRUPT_HANDLER	BTreemap<usize, InterruptHandler>	中断处理函数列表
EMU_DEV_LIST	Vec<EmuDevEntry>	模拟设备列表
GIC_LRS_NUM	usize	GICH.LR 寄存器数量

## Rust-Shyper预升级数据集



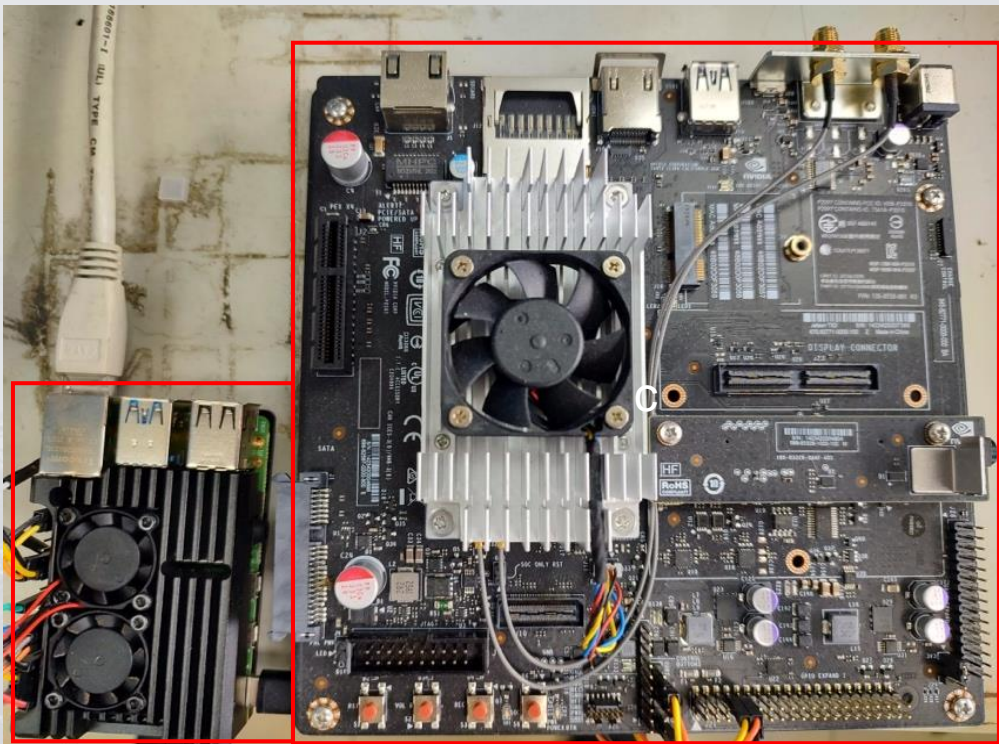


P A R T F I V E

---

## 实验评估





## NVIDIA Jetson TX2硬件信息

处理器	2 * Denver 64-bit处理器 / 4 * Cortex A57处理器
内存	8GB L128 bit DDR4内存
磁盘	32GB eMMC 5.1
网卡	10/100/1000 BAST-T 以太网
I/O插槽	PCI-E x4、SATA、GPIO

## 测试软件信息

虚拟机监视器	Rust-Shyper / KVM / Jailhouse / SyberX
模拟设备	Rust-Shyper模拟设备 / QEMU
管理虚拟机	Linux for Tegra
客户虚拟机	Linux 4.9.140
实时虚拟机	PREEMPT_RT Linux
BIOS	U-BOOT
测试集	Lmbench / UnixBench / MiBench / IOZone / Cyclicttest
扫描规则	ISO-17961 / MISRA_2004 / C_CPP缺陷检测项 / Rust-clippy

	MVM	GVM0	GVM1	RTVM
CPU	1	1	1	1
内存	1.5GB	2GB	2GB	2GB
中断控制器	vGIC	vGIC	vGIC	GIC
串口	NS16550/Virtio	Virtio	Virtio	NS16550
磁盘	100G(SSD)	100G(SSD)	100G(NFS)	32G(eMMC)
网卡	直通/Virtio	Virtio	Virtio	N/A





## SyberX ISO-17961扫描结果

漏洞名称	严重等级	数量
CERT-5.14 空指针解引用	严重——高	1
CERT-5.44 使用无效的格式字符串	重要——中	60
CERT-5.06 带有不正确参数的函数调用	次要——中	18
CERT-5.10 整型和指针之间的相互转换	次要——中	29

## SyberX MISRA\_2004扫描结果

漏洞名称	严重等级	数量
MISRA_01.02不能有对未定义行为或未指定行为的依赖性	次要——中	3

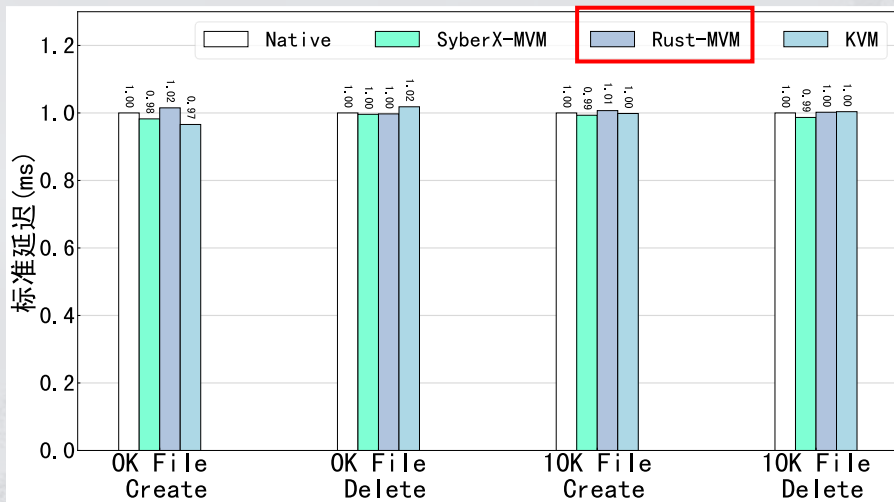
## Rust-Shyper Rust-clippy扫描结果

漏洞名称	严重等级	数量
对于非可变参数的函数具有可变类型的返回	次要——中	1
相似的if代码块	次要——中	1
判断语句恒为真	次要——中	1

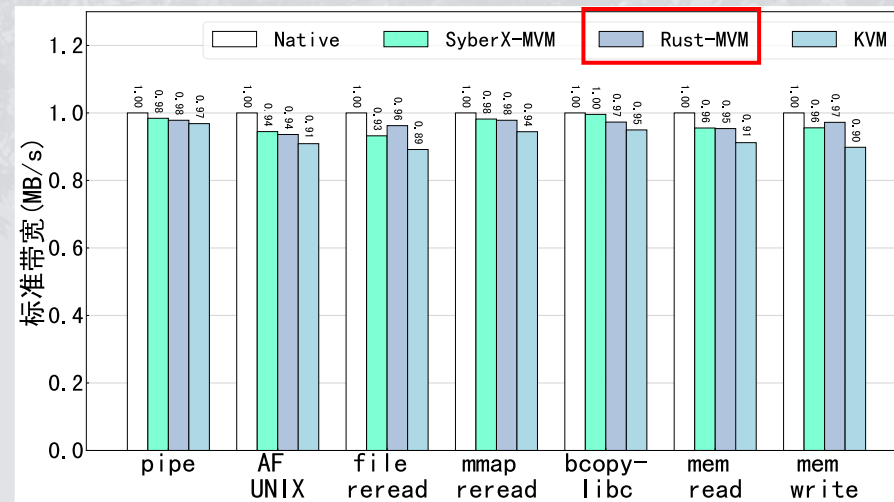
## SyberX C\_CPP缺陷项扫描结果

漏洞名称	严重等级	数量
S_30_05 被赋值为NULL的指针发生空指针解引用	致命——高	1
S_26_02 参数数量不匹配	严重——高	3
S_26_04 参数类型不兼容	严重——高	57
S_30_07 可疑的空指针解引用	严重——高	2
S_30_06 指针解引用前未判空	重要——中	133
S_02_06 不同位数变量进行位运算	次要——中	17
S_02_12 函数通过值传递传占空间过大的参数	次要——中	1
S_02_21 无效的比较大小	次要——中	2
S_32_02 隐式类型转换可能丢失精度	次要——中	13
S_36_02 用户函数返回值未测试	次要——中	2

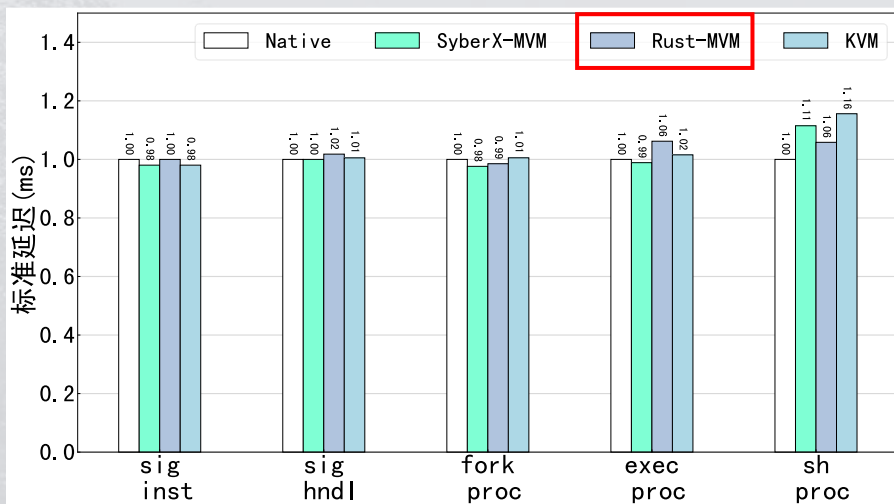




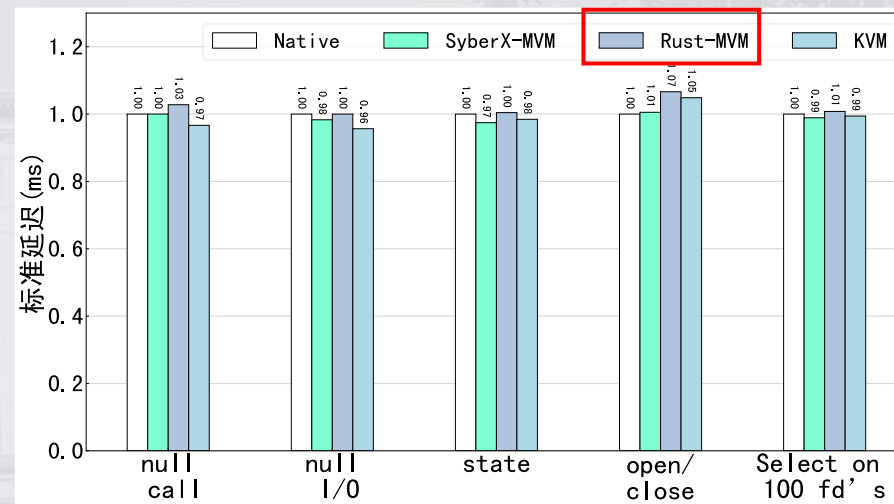
(a) 文件操作延迟



(b) 本地通信带宽



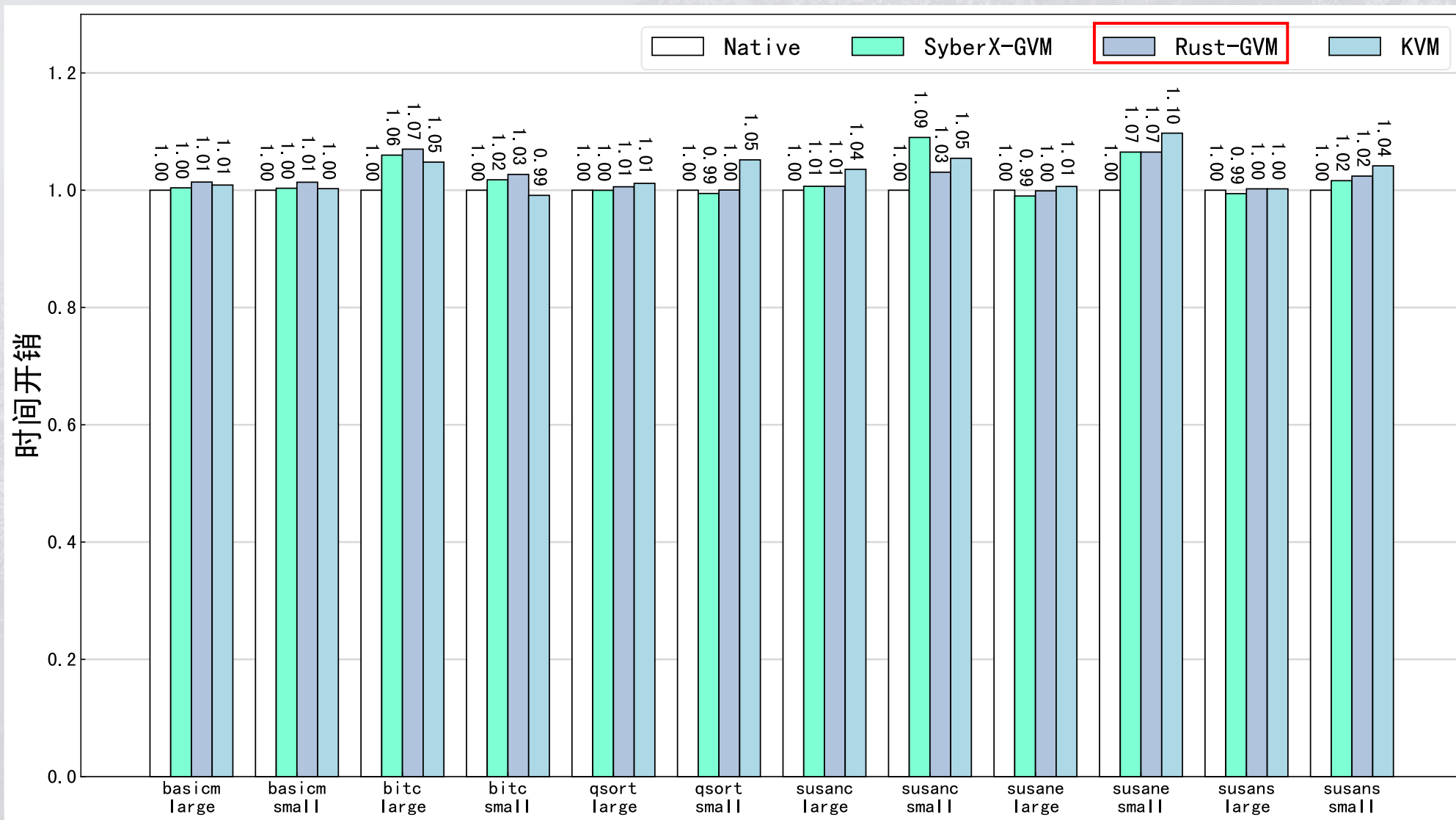
(c) 进程创建延迟



(d) 系统调用延迟

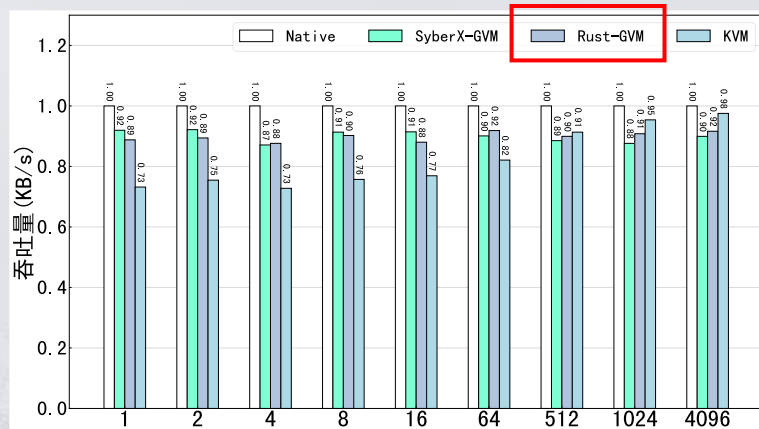
LmBench测试中，Rust-Shyper虚拟机的操作系统基本性能达到原生操作系统的95%，与SyberX、KVM相近



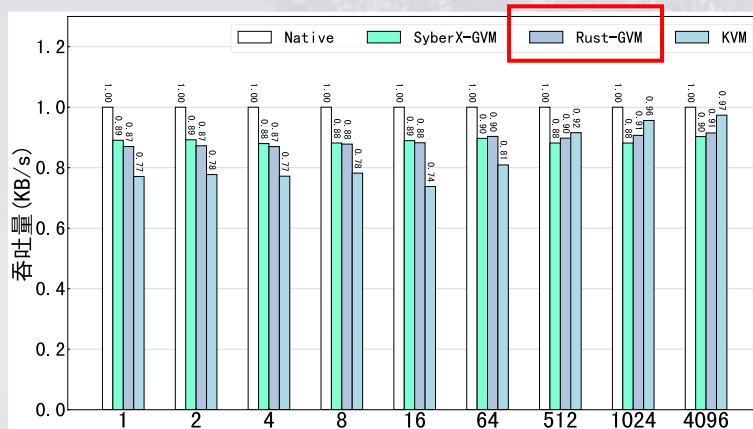


Rust-Shyper虚拟机处理器与内存性能达到原生操作系统的93%，  
与SyberX、KVM基本持平





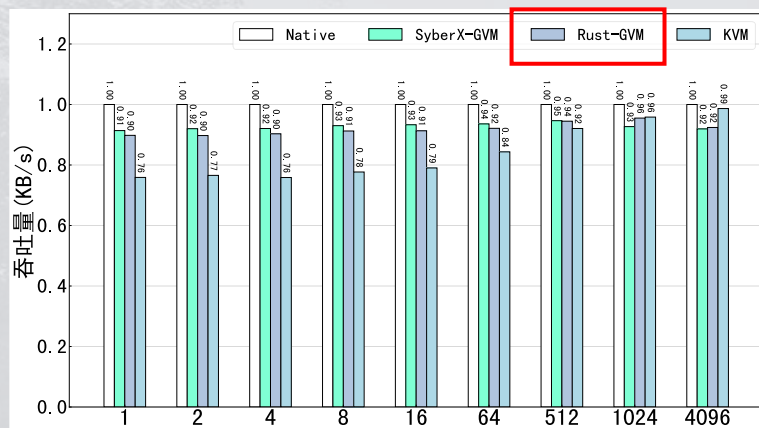
(a) 磁盘随机读吞吐量



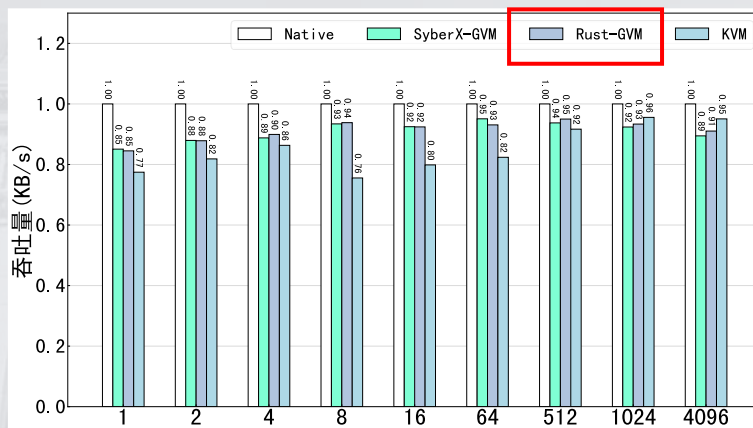
(b) 磁盘读吞吐量



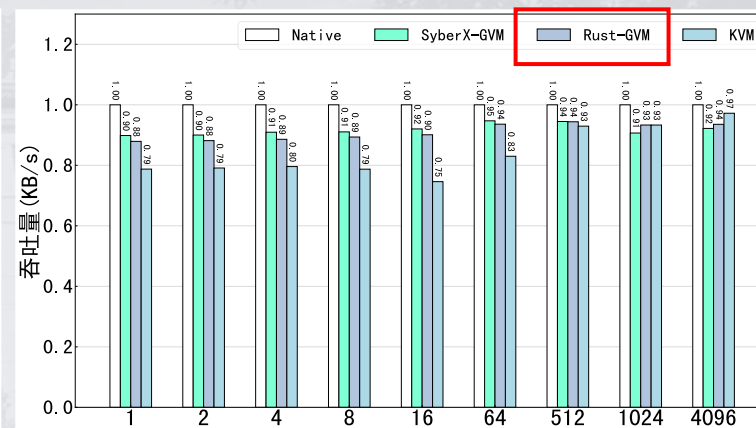
(c) 磁盘重复读吞吐量



(d) 磁盘随机写吞吐量



(e) 磁盘写吞吐量



(f) 磁盘重复写吞吐量

IOZone磁盘吞吐测试，横轴为磁盘单次读写的大小，单位KB

Rust-Shyper虚拟机的I/O性能达到原生操作系统的90%





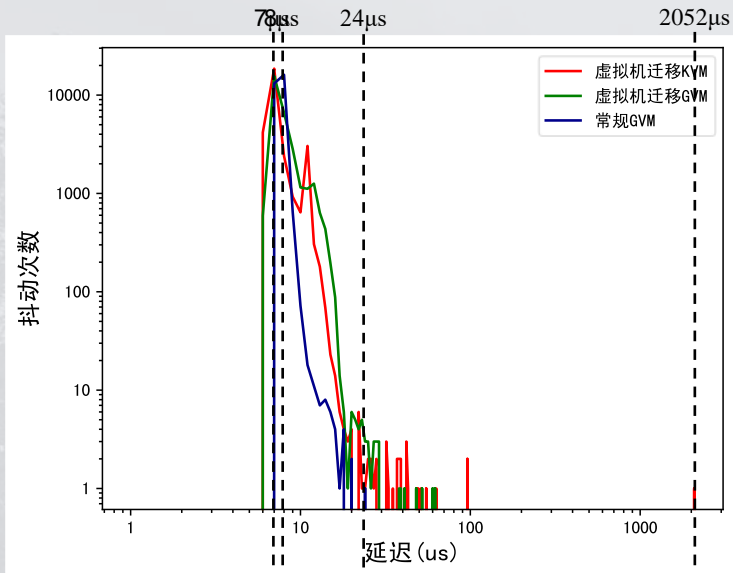
Cyclictest测试结果

测试条目	Min	Avg	Max
Native-L4T	5	12	369
Rust-Shyper-MVM	4	15	456
KVM-Guest	8	15	1022
Rust-Shyper-GVM	3	13	364
Jailhouse-NoneRootCell	5	9	26
Rust-Shyper-RTVM	2	5	21

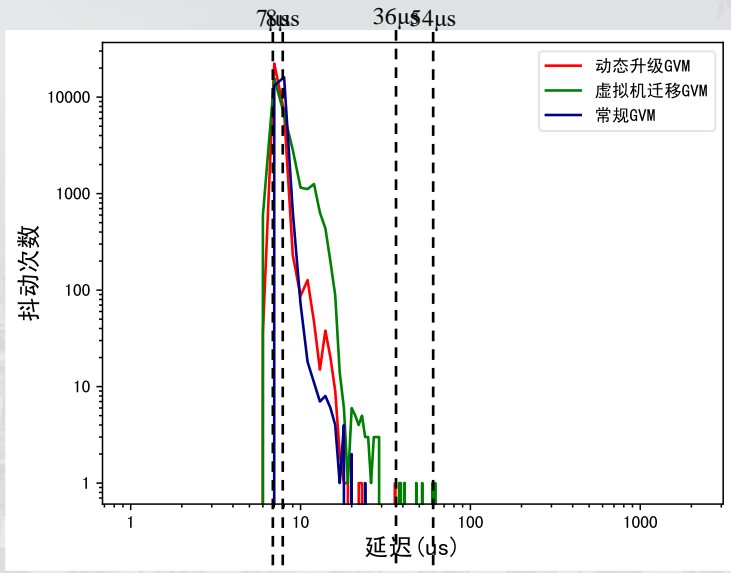
Cyclictest任务抖动测试，Rust-Shyper RTVM虚拟机的延迟抖动低于Jailhouse NoneRootCell，能够满足嵌入式实时需求



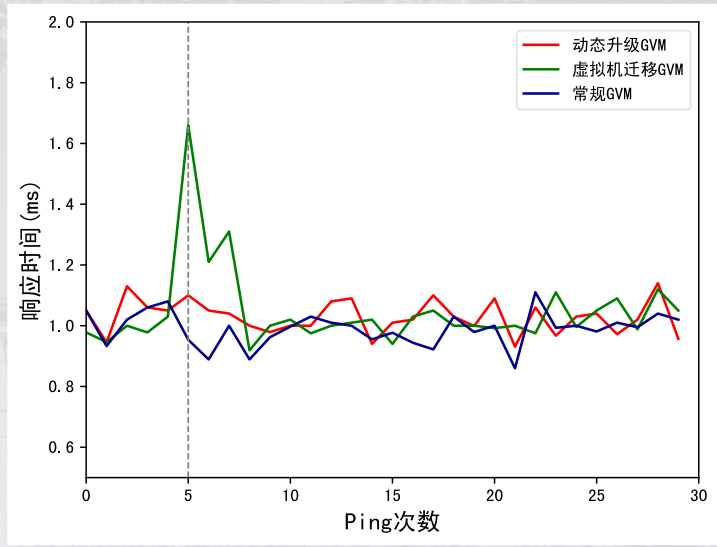
虚拟机监视器	Cyclictest最大延迟抖动	停机延迟
KVM Pre-Copy	2052	1937
Rust-Shyper虚拟机迁移	54	798



(a) Rust-Shyper、KVM虚拟机迁移延迟抖动



(b) Rust-Shyper动态升级、虚拟机迁移延迟抖动



(c) Rust-Shyper动态升级、虚拟机迁移网络波动

通过Cyclictest测试热更新过程中虚拟机的延迟抖动，Rust-Shyper GVM在监视器动态升级场景下抖动仅**36微秒**，低于Rust-Shyper虚拟机迁移**54微秒**以及KVM虚拟机迁移的**2052微秒**





➤ [https://gitee.com/openeuler/rust\\_shyper](https://gitee.com/openeuler/rust_shyper)

gitee.com/openeuler/rust\_shyper

开源软件 企业版 高校版 私有云 博客 我的

搜开源

**Rust-Shyper**

- 介绍
- 目前支持的硬件平台
- 如何编译
- 如何启动客户虚拟机...
- 参考文献
  - 参与贡献
  - 特技

A Reliable Embedded Hypervisor Supporting VM Migration and Hypervisor Live-Update

[English Version README click here](#)

### 介绍

Rust-Shyper 是一个使用高级语言Rust编写的面向嵌入式场景的Type-1型虚拟机监视器 (Hypervisor)。其设计目标在于提高资源利用率的同时, 同时保障虚拟机实时性、隔离性与可靠性的需求。为达成上述目的, 首先Rust-Shyper选用Rust作为编程语言, 利用语言本身的安全特性提升代码质量, 从语言层面保障系统软件的可靠性。其次, 为了保障虚拟机的隔离性需求, Rust-Shyper针对CPU、中断、设备、内存等公共资源实现了有效的隔离策略, 保证了同一资源被不同虚拟机共享的同时, 虚拟机无法越界访问不属于当前虚拟机的资源。另外, 为了保障虚拟机实时性需求, Rust-Shyper实现了中断部分直通机制以及中介传递设备模型, 有效缩减虚拟化对实时性能的影响。最后, 为了进一步保障监视器可靠性, 本文实现了虚拟机迁移 (VM migration) 以及监视器动态升级 (Hypervisor Live-update) 两种热更新机制修复虚拟机监视器可能存在的代码漏洞。

### 目前支持的硬件平台

下表是目前Rust-Shyper已经支持 (或正在开发中) 的硬件平台:

aarch64

- ☒ NVIDIA Jetson TX2
- ☒ Raspberry Pi 4 Model B
- ☐ QEMU (still work in progress)

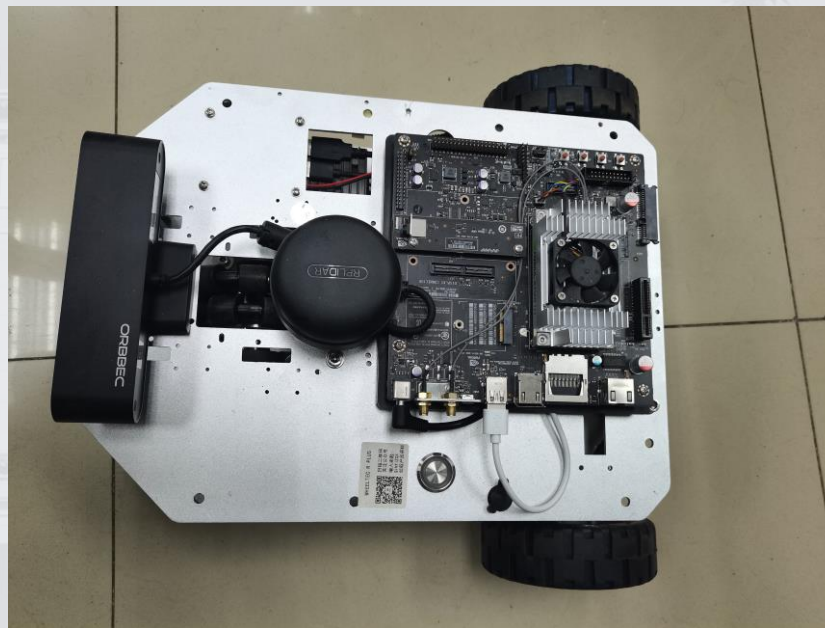
### 如何编译

只需要使用 make 工具即可

```
make <platform>
```



- 移植到QEMU，便于安装和调试
- 针对实时性和隔离性进一步评测，完成针对性优化，提高RTVM的确定性
- 开发基于无人车的应用演示
- 支持ROS，开发基于机器人的应用演示（与牛建伟老师合作）
- 移植至更多体系结构，基于RISC-V利用软硬件结合的方式进一步优化虚拟化效果







北京航空航天大学  
BEIHANG UNIVERSITY

**THANKS**

**请各位专家批评指正**