

# Rockchip平台PCIe设备虚拟化开发指南

---

文件标识: RK-KF-YF-170

发布版本: V1.0.0

日期: 2021-03-30

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

## 免责声明

本文档按“现状”提供, 瑞芯微电子股份有限公司(“本公司”, 下同)不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因, 本文档将可能在未经任何通知的情况下, 不定期进行更新或修改。

## 商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标, 归本公司所有。

本文档可能提及的其他所有注册商标或商标, 由其各自所有者所有。

版权所有 © 2021 瑞芯微电子股份有限公司

超越合理使用范畴, 非经本公司书面许可, 任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部, 并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址: 福建省福州市铜盘路软件园A区18号

网址: [www.rock-chips.com](http://www.rock-chips.com)

客户服务电话: +86-4007-700-590

客户服务传真: +86-591-83951833

客户服务邮箱: [fae@rock-chips.com](mailto:fae@rock-chips.com)

## 前言

## 概述

该方案基于QEMU + ARM KVM + PCIE + SMMU + VFIO来实现外设的虚拟化，主要目的是实现在虚拟机中直接访问PCIe外设，依据本指南可以实现在虚拟机中挂载PCIe NVME SSD以及在虚拟机中实现PCIe网卡的连接。

## 产品版本

芯片名称	内核版本
带SMMU的Rockchip芯片	4.19

## 读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

## 修订记录

日期	版本	作者	修改说明
2021-03-30	V1.0.0	薛小明	初始版本

## 目录

### Rockchip平台PCIe设备虚拟化开发指南

1. 环境搭建
  - 1.1 Guest OS内核配置
  - 1.2 Guest OS ramdisk准备
  - 1.3 HOST OS内核配置
  - 1.4 QEMU准备
  - 1.5 Host OS ramdisk准备
2. DTS 配置
  - 2.1 SMMU配置
  - 2.2 PCIe补充配置
3. 运行
4. 遗留问题

# 1. 环境搭建

## 1.1 Guest OS内核配置

使能如下配置。

```
CONFIG_ARM64_VA_BITS_39=y
CONFIG_ARM64_VA_BITS=39
```

关闭如下配置。

```
CONFIG_HUGETLBFS
CONFIG_HUGETLB_PAGE
```

使用以下命令生成guest os内核固件。

```
make ARCH=arm64
```

固件位置。

```
arch/arm64/boot/Image
```

## 1.2 Guest OS ramdisk准备

包含busybox工具，lspci，ifconfig工具。

## 1.3 HOST OS内核配置

在rockchip\_defconfig基础上再使能如下配置。

```
CONFIG_KVM
CONFIG_VFIO
CONFIG_VFIO_PCI
CONFIG_IOMMU_SUPPORT
CONFIG_ARM_SMMU_V3
CONFIG_VFIO_IOMMU_TYPE1
CONFIG_BLK_DEV_NVME
CONFIG_R8169
CONFIG_PCIE_DW
CONFIG_PCIE_DW_HOST
CONFIG_PCIE_DW_ROCKCHIP
```

## 1.4 QEMU准备

使用Rockchip的buildroot工程编译qemu，可执行文件名称qemu-system-aarch64。

## 1.5 Host OS ramdisk准备

使用Rockchip的buildroot工程生成ramdisk，包含qemu-system-aarch64，guest os内核固件，guest os ramdisk固件。

Rockchip的buildroot工程配置以及编译方法参考buildroot使用文档，本指南提供预先编译好的ramdisk目录，包含需要的所有固件，

在ubuntu环境下，使用abootimg-pack-initrd，会在ramdisk同级目录生成initrd.img固件，使用abootimg-unpack-initrd解压initrd.img。

## 2. DTS 配置

### 2.1 SMMU配置

```
compatible = "arm,smmu-v3";
```

必须配置，用于匹配ARM SMMU-v3驱动。

以下寄存器以及中断配置以RK3588为例。

```
reg = <0x0 0xfc900000 0x0 0x100000>;
```

ARM SMMU-v3基地址以及大小。

```
interrupts = <GIC_SPI 369 IRQ_TYPE_EDGE_RISING>,
             <GIC_SPI 371 IRQ_TYPE_EDGE_RISING>,
             <GIC_SPI 374 IRQ_TYPE_EDGE_RISING>,
             <GIC_SPI 367 IRQ_TYPE_EDGE_RISING>;
interrupt-names = "eventq", "gerror", "priq", "cmdq-sync";
```

详见Documentation/devicetree/bindings/iommu/arm,smmu-v3.txt。

```
#iommu-cells = <1>;
```

必须为1，用于描述设备的stream ID，详见Documentation/devicetree/bindings/iommu/arm,smmu-v3.txt。

### 2.2 PCIe补充配置

```
iommu-map = <0x0 &smmu 0x0 0x10000>;
```

为PCIe外设创建SMMU页表。

### 3. 运行

#### 1. 卸载宿主机PCIe设备驱动

使用lspci得到设备地址，假设设备地址为0002:21:00.0，执行以下命令卸载。

```
echo 0002:21:00.0 > /sys/bus/pci/devices/0002\:21\:00.0/driver/unbind
```

#### 2. 将PCIe设备重新挂载到VFIO-PCI驱动

使用lspci得到设备vendor id和device id，以Realtek 8169PCIe网卡为例，vendor id和device id分别为10ec, 0000，使用以下命令挂载。

```
echo 10ec 0000 > /sys/bus/pci/drivers/vfio-pci/new_id
```

#### 3. 运行qemu

```
qemu-system-aarch64 -nographic -monitor none -m 128 -machine  
virt,highmem=off -serial stdio -cpu host -enable-kvm -device vfio-  
pci,host=0002:21:00.0 -kernel /Image -initrd /ramdisk_ok.img
```

以下qemu参数说明。

```
qemu-system-aarch64
```

Rockchip buildroot编译出来的qemu二进制文件，放在host os根文件系统的bin/目录下。

```
-m 128
```

为虚拟机创建128M的内存空间。

```
highmem=off
```

使用低32位地址空间。

```
-device vfio-pci,host=0002:21:00.0
```

使用PCIe设备地址作为参数创建vfio设备，用于虚拟机访问。

```
-kernel /Image -initrd /ramdisk_ok.img
```

分别是guset os内核固件，guest os根文件系统固件，这里将这两个固件放在host os根文件系统的根目录。

#### 4. 待虚拟机启动完成，使用以下命令配置PCIe网卡

```
ifconfig eth0 up  
  
ifconfig eth0 192.168.31.120 netmask 255.255.255.0  
  
route add default gw 192.168.31.1
```

地址以及网关根据实际情况配置。

#### 5. 实用以下命令测试网络连接

```
ping 8.8.8.8
```

正常情况下，能看到ping信息。

## 4. 遗留问题

---

在虚拟机里面PCIe设备默认被设置为coherent属性，所以申请的dma buffer被设置成cacheable，从而导致数据一致性的问题，需要在虚拟机PCIe设备probe时候，手动设置为non-coherent，如下，才能规避数据一致性问题。

```
dev->archdata.dma_coherent = false;
```