sophon-rpc 使用指南

发布 3.3.0

SOPHGO

2024 年 06 月 06 日

目录

1 声明 **1**

2 sophon-rpc 安装 **3**
2.1 Ubuntu/Debian 系统安装 3
2.2 Centos 系统安装 4
2.3 其他操作系统安装 4
2.4 关闭开机启动 A53 功能 5

3 sophon-rpc 测试用例 **6**
3.1 设置 sophon-rpc 日志等级 6
3.2 获取 sophon-rpc 运行日志 6
3.3 同步 A53 时间 7
3.4 sophon-rpc 加载库执行函数 7

4 mix mode 介绍及使用 **8**
4.1 mix mode 介绍 8
4.2 启动 mix mode 10
4.3 系统定制 12

CHAPTER 1

声明

法律声明

版权所有 © 算能 2022. 保留一切权利。
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。

注意

您购买的产品、服务或特性等应受算能商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，算能对本文档内容不做任何明示或默示的声明或保证。由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

技术支持

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

邮编 100094

网址 https://www.sophgo.com/

邮箱 sales@sophgo.com

电话 +86-10-57590723 +86-10-57590724

SDK 发布记录

版本	发布日期	说明
V0.2.4	2022.10.17	第一次发布。
V0.2.5	2022.11.21	bug fix。
V0.2.6	2022.12.13	支持加载库去重。

Copyright © SOPHGO

CHAPTER 2

sophon-rpc 安装

首先需要确认本地 libc 版本是否大于等于 2.23，sophon-libsophon 软件版本是否大于等于 0.4.2，spi_flash 版本是否大于等于 2.7。只有在安装了 sophon 的 libsophon 软件库后，才能正常安装 sophon-rpc 软件包。安装前请检查/lib/modules/$(uname -r)/kernel/drivers/pci/目录，如果存在 fip.bin、ramboot.itb 和 pre_start_a53.sh，请将这三个文件删除，再进行 sophon-rpc 的安装。

安装 sophon-rpc 之前需要执行 cat /proc/bmsophon/card*/bmsophon*/bmcpu_status，确认返回的结果为 idle 或 running，则可以继续安装 sophon-rpc。若返回结果为 fault，则需要重启机器恢复板卡的 a53 核状态，再安装 sophon-rpc。

2.1 Ubuntu/Debian 系统安装

当前 sophon-rpc 在 Ubuntu/Debian 系统中以 deb 包的形式发布，不同架构的操作系统，安装包名字的 arch 字段不同，使用

```
uname -m
```

命令查看当前机器的硬件架构，通常 x86_64 的机器对应的安装包名的 arch 字段为 amd64，arm64 机器对应的安装包名的 arch 字段为 arm64。用户获取该 deb 包确认依赖满足后，使用以下命令安装 sophon-rpc 软件包。

```
sudo dpkg -i sophon-rpc_3.3.0_$arch.deb
```

卸载时使用

```
sudo apt remove sophon-rpc
```

命令可以卸载 sophon-rpc 组件。

2.2 Centos 系统安装

当前 sophon-rpc 在 Centos 系统中以 rpm 包的形式发布，不同架构的操作系统，安装包名字的 arch 字段不同，使用

```
uname -m
```

命令查看当前机器的硬件架构，通常 x86_64 的机器对应的安装包名的 arch 字段为 amd64，arm64 机器对应的安装包名的 arch 字段为 arm64。用户获取该 rpm 包确认依赖满足后，使用以下命令安装 sophon-rpc 软件包。

```
sudo  rpm -ivh sophon-rpc-3.3.0-1.$arch.rpm
```

卸载方式：

```
sudo rpm -e sophon-rpc
```

2.3 其他操作系统安装

非 Ubuntu/Debian 系统中，需要使用 sophon-rpc_3.3.0.tar.gz 文件来安装 sophon-rpc。安装步骤如下：

```
tar -xvf sophon-rpc_3.3.0.tar.gz
sudo cp -r sophon-rpc_3.3.0/* /
sudo chmod 766 /opt/sophon/sophon-rpc-3.3.0/firmware/fip.bin
sudo chmod 766 /opt/sophon/sophon-rpc-3.3.0/firmware/ramboot_rootfs.itb
sudo chmod 766 /opt/sophon/sophon-rpc-3.3.0/firmware/ramdisk_glibc_mix.itb
sudo cp /opt/sophon/sophon-rpc-3.3.0/firmware/* /opt/sophon/libsophon-current/data
sudo cp /opt/sophon/sophon-rpc-3.3.0/service/99-sophon-rpc-udev.rules /etc/udev/rules.d/
sudo /opt/sophon/libsophon-current/bin/test_pre_start_cpu
```

卸载步骤如下：

```
sudo /opt/sophon/libsophon-current/bin/test_reset_all_cpu
sudo rm -f /etc/udev/rules.d/99-sophon-rpc-udev.rules
sudo rm -f /opt/sophon/libsophon-current/data/fip.bin
sudo rm -f /opt/sophon/libsophon-current/data/ramboot_rootfs.itb
sudo rm -f /opt/sophon/libsophon-current/data/ramdisk_glibc_mix.itb
sudo rm -f /opt/sophon/libsophon-current/data/spi_flash_bm1684.bin
sudo rm -f /opt/sophon/libsophon-current/data/spi_flash_bm1684x.bin
sudo rm -rf /opt/sophon/sophon-rpc-3.3.0
```

Copyright © SOPHGO

2.4 关闭开机启动 A53 功能

当前版本安装 sophon-rpc 软件后，会启动加速卡上的 A53 处理单元，并修改 udev 规则，用于开机自动启动 A53。如果用户不希望开机自动启动 A53，则需要卸载该安装包。

Copyright © SOPHGO

CHAPTER 3

sophon-rpc 测试用例

正常安装完 sophon-rpc 软件包后，A53 就会启动，无需再手动运行使能 A53 的命令。下面介绍当前支持的测试用例。正常安装完 libsophon 包并使能了环境变量后，可以直接运行下面的命令。

3.1 设置 sophon-rpc 日志等级

参数 1 为设备 id，参数 2 为支持的 log 等级，当前可以取 0，1，2，3，4，5，分别表示 DEBUG，INFO，WARN，ERROR，FATAL 和 OFF，参数 3 为是否输出到串口。

```
test_set_log 0 1 1
```

3.2 获取 sophon-rpc 运行日志

参数 1 为设备 id，参数 2 为保存日志文件的完整路径。

```
test_get_log 0 1.txt
```

3.3 同步 A53 时间

参数 1 为设备 id。

```
test_sync_time 0
```

3.4 sophon-rpc 加载库执行函数

参数 1 为设备 id，参数 2 为需要在 A53 上运行的用户算法库文件，参数 3 为用户算法库中的函数名称，参数 4 为参数 3 对应函数需要的参数，参数 5 为并行测试的线程数量。

```
test_lib_cpu 0 /opt/sophon/libsophon-current/lib/sophon-rpc/user_algo.so test helloworld 2
```

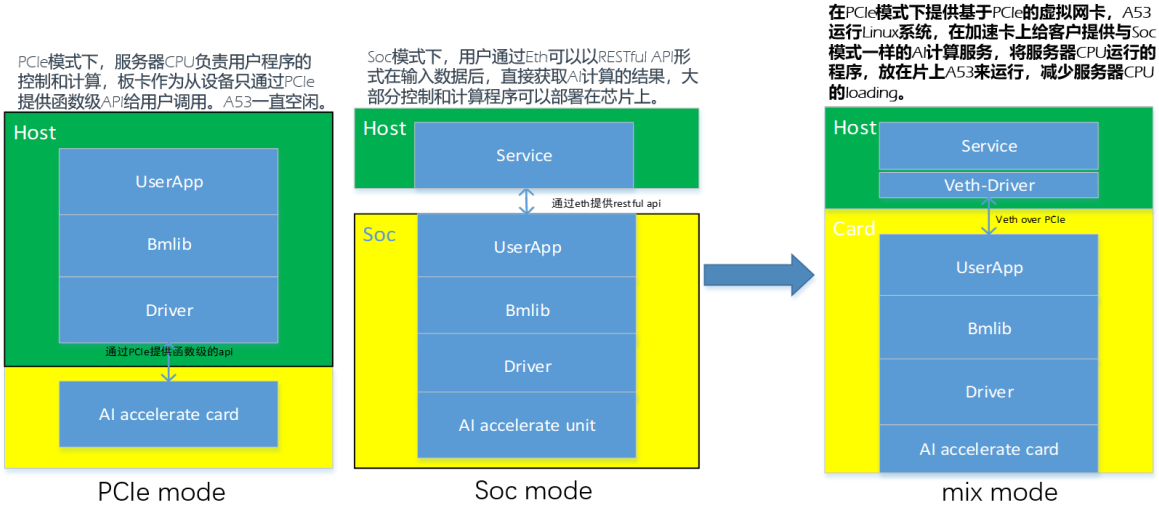
Copyright © SOPHGO

CHAPTER 4

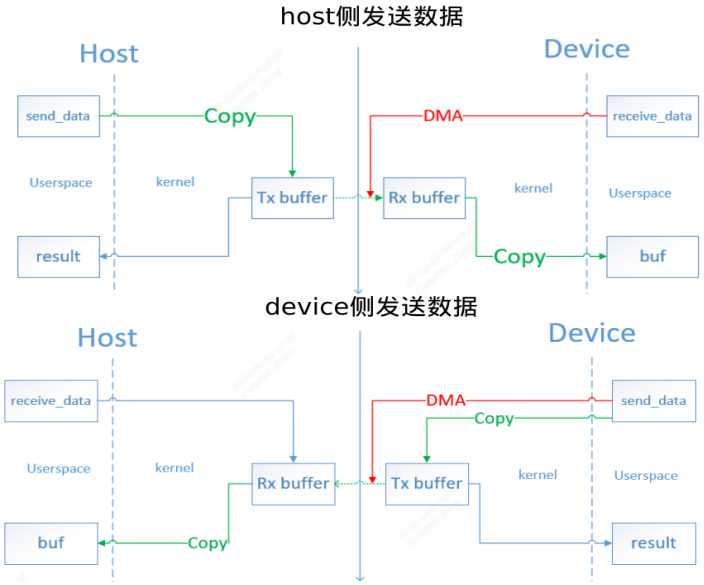
mix mode 介绍及使用

4.1 mix mode 介绍

在当前国产化 CPU 成本较高的场景下，在使用 AI 加速卡产品时，希望能将业务尽可能的部署在我们的产品上，降低其服务器上 CPU 的 loading。mix mode 模式可以满足这种需要，在此模式下，可以在 pcie 模式环境的基础上降低 cpu loading。



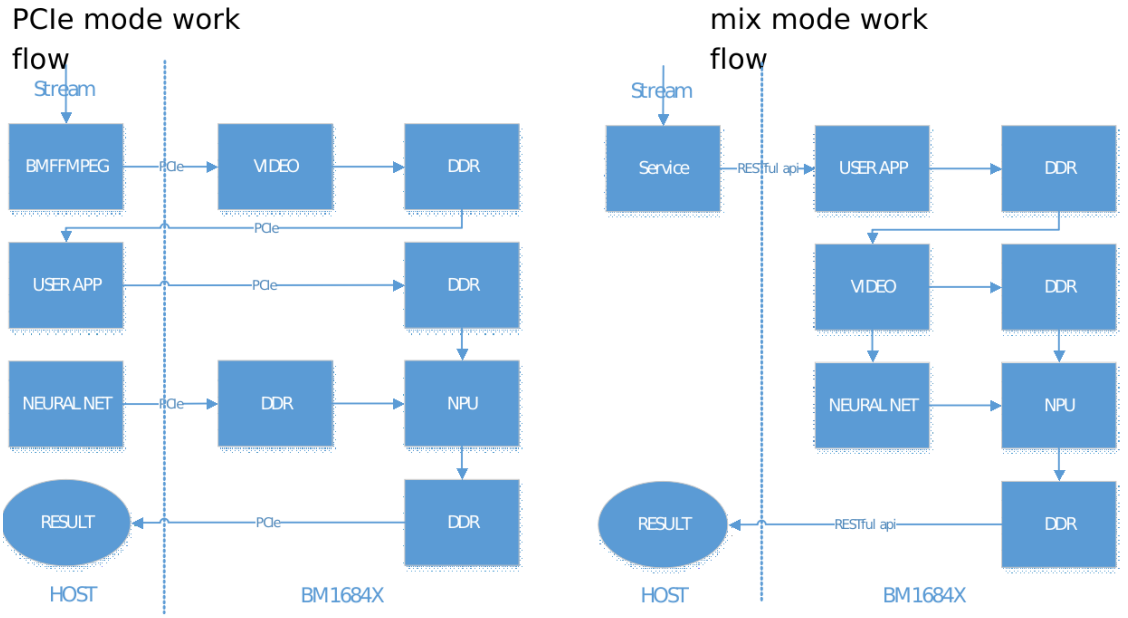
mix mode 模式下，数据传输流程如下图所示：



Host 侧发送数据时，先将数据拷贝到 tx buffer，然后通过中断触发 device 侧 a53，配置 cdma 进行数据搬运。

Device 侧发送数据时，将数据拷贝到 tx buffer，然后配置 cdma 进行数据搬运，完成后发送中断，触发 host 侧 a53 接收数据。

mix mode 模式下，业务运行位置如下图所示：



由于 Host 侧 CPU 只负责通过网络进行数据的拷贝，不需要进行业务的控制和计算，所以能够降低 Host 侧业务处理的 CPU loading。

Copyright © SOPHGO

4.2 启动 mix mode

当前版本仅支持在 1684x 板卡上启动 mix mode。启动 mix mode 后，板卡类似于 soc 模式，可通过 ssh 的方式登录板卡，并运行 soc 模式的程序。

启动 mix mode 前，需要参考之前 sophon-rpc 的卸载方式，卸载 sophon-rpc，然后配置 mix mode 的环境：

export MIX_MODE=1
如果当前系统为Ubuntu/Debian系统：
sudo -E dpkg -i sophon-rpc_3.3.0_$arch.deb
如果当前系统为Centos系统：
sudo -E rpm -ivh sophon-rpc-3.3.0-1.$arch.rpm
其他操作系统：
tar -xvf sophon-rpc_3.3.0.tar.gz
sudo cp -r sophon-rpc_3.3.0/* /
sudo chmod 766 /opt/sophon/sophon-rpc-3.3.0/firmware/fip.bin
sudo chmod 766 /opt/sophon/sophon-rpc-3.3.0/firmware/ramboot_rootfs.itb
sudo chmod 766 /opt/sophon/sophon-rpc-3.3.0/firmware/ramdisk_glibc_mix.itb
sudo cp /opt/sophon/sophon-rpc-3.3.0/firmware/* /opt/sophon/libsophon-current/data
查看bmcpu状态：cat /proc/bmsophon/card${card_id}/bmsophon${tpu_id}/bmcpu_status
如果bmcpu状态为running，执行test_reset_cpu ${tpu_id}使bmcpu恢复到未启动的idle状态。

配置好环境后，执行以下命令：

```
在指定的芯片${chip_id} 启动 mix mode：
test_pre_start_mix_cpu ${chip_id}
如果不输入${chip_id}，则默认在所有芯片上启动 mix mode

根据要启动的芯片配置 host 侧网卡的 ip：
sudo ifconfig veth${chip_id} 192.192.${chip_id}.3 up

启动 mix mode 后，等待大约 20s，执行以下命令登录设备：
ssh root@192.192.${chip_id}.2 密码：bitmain
注意，第一次连接板卡可能会因为动态 mac 导致连接失败，按照系统给出的提示删除对应文件即可。

如果需要关闭芯片${chip_id} 的 mix mode，则执行以下命令：
test_reset_cpu ${chip_id}
如果需要关闭所有芯片的 mix mode，执行以下命令：
test_reset_all_cpu

其中${chip_id} 为芯片 id，可通过设置${chip_id} 来在不同芯片上启动 mix mode 模式。
```

如果出现板卡 ip 不能 ping 通，或者 ssh 界面卡住的情况，则需要用 ifconfig 命令检查 veth${chip_id} 网卡 ip 是否还存在，如不存在需要重新进行配置为 192.192.${chip_id}.3。如果想永久固定虚拟网卡 ip 则需要按照以下步骤操作：

```
如果使用的是 Ubuntu 系统，可以在 Ubuntu 网络管理界面里把网卡改成固定 ip：
sudo vim /etc/netplan/01-network-manager-all.yaml （文件名字可能为 01-net 开头的其他名字）
修改其中的内容如下：
```

（下页继续）

(续上页)

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
        veth0:
            dhcp4: no
            addresses: [192.192.0.3/24]
            optional: yes
        veth1:
            dhcp4: no
            addresses: [192.192.1.3/24]
            optional: yes
        veth2:
            dhcp4: no
            addresses: [192.192.2.3/24]
            optional: yes
其中 veth 的数量可以根据启动芯片 mix mode 的数量改变
保存退出后执行:
sudo netplan apply
之后网卡 veth{chipid} 就会被固定为 192.192.{chipid}.3
```

```
如果使用的是 fedora 系统，需要执行以下命令:
sudo nmcli con add type ethernet con-name 'veth{chipid}' ifname veth{chipid} ipv4.method
manual ipv4.addresses 192.192.{chipid}.3/24 gw4 192.192.{chipid}.3
之后网卡 veth{chipid} 就会被固定为 192.192.{chipid}.3
```

重启服务器后，需要重新执行 test_pre_start_mix_cpu ${chip_id} 命令启动 mix mode。如果要恢复 PCIe 模式，则需要参考 “LIBSOPHON 使用手册”，重新安装 libsophon。

如果想在 mix mode 中联通外部网络，则执行以下命令:

```
在 mix mode 上:
route add default gw 192.192.${chip_id}.3
在 host 上:
打开服务器的本地转发功能:
cat /proc/sys/net/ipv4/ip_forward
0 为关闭，1 为打开。如果为 0，则执行:
sudo sysctl -w net.ipv4.ip_forward=1
配置将虚拟网卡的请求都发往外网:
sudo iptables -A FORWARD -i ${veth} -o ${eth} -j ACCEPT
sudo iptables -A FORWARD -i ${eth} -o ${veth} -m state --state ESTABLISHED,RELATED
-j ACCEPT
sudo iptables -t nat -A POSTROUTING -o ${eth} -j MASQUERADE
其中${veth} 为 host 侧对应芯片的虚拟网卡，${eth} 为 host 侧联通外部网络的网卡
可通过 ifconfig 查看对应网卡名字
```

4.3 系统定制

我们提供了一个 BSP SDK 以便您对 mix mode 中的内核和操作系统进行定制，然后生成自己的 mix mode 镜像。

SDK 为 github 网站（https://github.com/sophgo）上发布的源码文件，bootloarder-arm64、linux-arm64 和 gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu。代码链接如下：

bootloarder-arm64：https://github.com/sophgo/bootloader-arm64/tree/v23.09-LTS

linux-arm64：https://github.com/sophgo/linux-arm64/tree/v23.09-LTS

gcc 工具：https://github.com/sophgo/host-tools/tree/master/gcc/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu

代码下载完毕后将看到如下文件：

```
top
├─── bootloader-arm64
│    ├─── scripts
│    │    └─── envsetup.sh → 编译脚本入口
│    ├─── trusted-firmware-a → TF-A 源代码
│    ├─── u-boot → u-boot 源代码
│    └─── ramdisk → mix mode 文件系统
├─── gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu → 交叉编译工具链
└─── linux-arm64 → kernel 源代码
```

推荐在 Ubuntu 20.04 系统下进行交叉编译，不支持 X86_64 以外的架构。请预留至少 10GB 空闲磁盘空间，并请先安装必要的一些工具：

```
sudo apt install cmake libssl-dev fakeroot dpkg-dev device-tree-compiler u-boot-tools
```

进入到 BSP SDK 后，执行如下命令即可编译出 mix mode 镜像：

```
source bootloader-arm64/scripts/envsetup.sh
build_fip
build_kernel r
build_ramdisk glibc mix
```

因为脚本中使用了 sudo，编译过程中可能会提示您输入当前用户密码。第一次编译时可能遇到各种问题，如结果不符合预期，请仔细检查编译 log，如果有遇到提示某某工具找不到的话，用 apt install 安装即可。

编译结果在 install/soc_bm1684 目录下，重点有如下两个文件：

```
fip.bin → TF-A 及 uboot；

ramdisk_glibc_mix.itb → kernel 和 ramdisk、dtb 打包在一起；
```

获取 fip.bin 和 ramdisk_glibc_mix.itb 之后，可按照 4.2 节的内容启动 mix mode。