Sophon and SDK FAQ User Guide

发布 3.0.0

SOPHGO

2024 年 04 月 26 日

目录

发布记录

版本	发布日期	说明
V0.1.0	2022.05.12	第一次发布。

CHAPTER 1

声明

法律声明

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
V2.0.0	2019.09.20	第一次发布。
V2.0.1	2019.11.16	V2.0.1 版本发布。
V2.0.3	2020.05.07	V2.0.3 版本发布。
V2.2.0	2020.10.12	V2.2.0 版本发布。
V2.3.0	2021.01.11	V2.3.0 版本发布。
V2.3.1	2021.03.09	V2.3.1 版本发布。
V2.3.2	2021.04.01	V2.3.2 版本发布。
V2.4.0	2021.05.23	V2.4.0 版本发布。
V2.5.0	2021.09.02	V2.5.0 版本发布。
V2.6.0	2021.01.30	V2.6.0 版本修正后发布。
V2.7.0	2022.03.16	V2.7.0 版本发布, 20220531 发布补丁版本。
V3.0.0	2022.07.16	V3.0.0 版本发布。

CHAPTER 2

基础概念常见问题

2.1 算丰系列产品有哪些硬件形态？

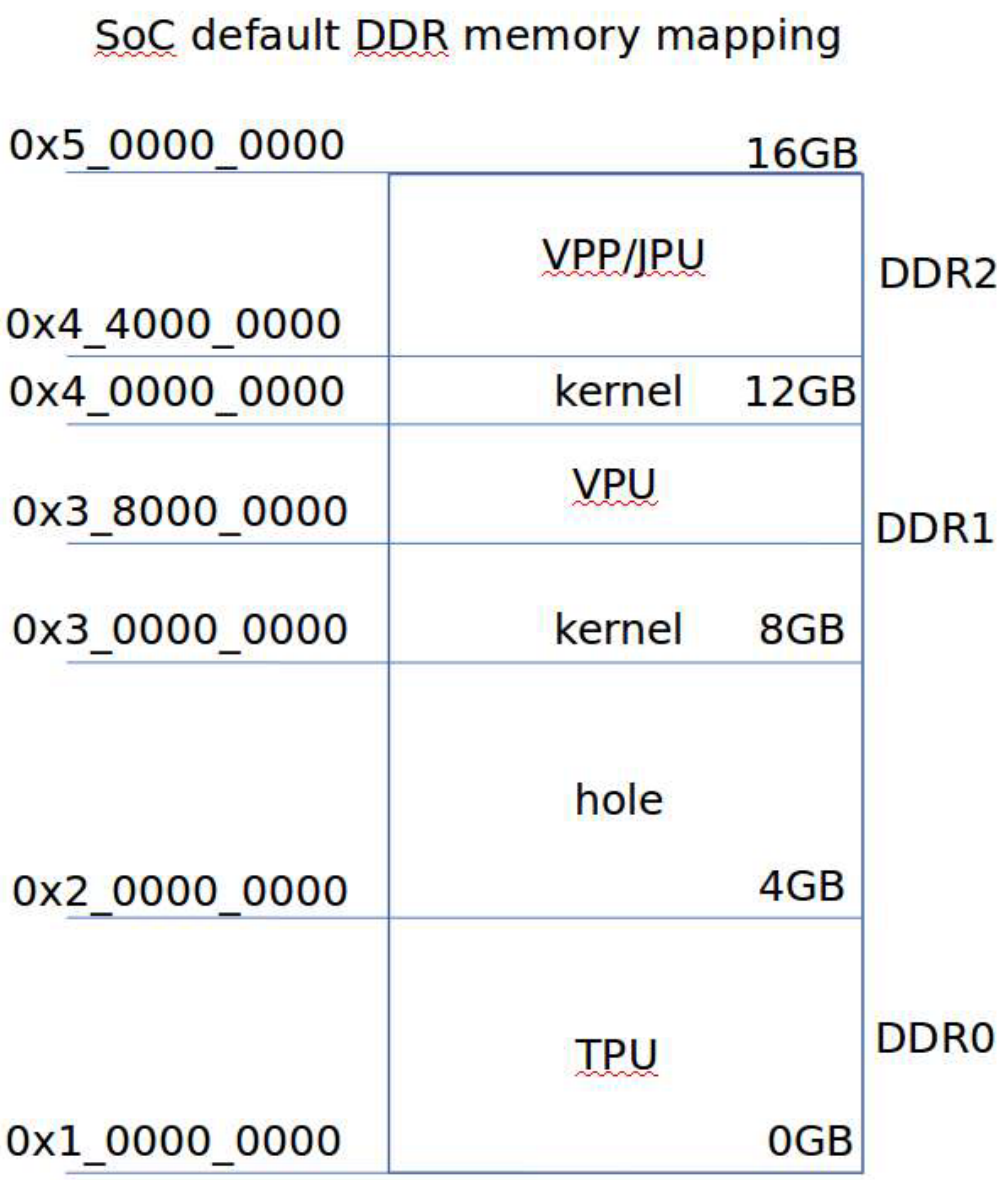
答：芯片 BM1684、模组 SM5、边缘计算盒子 SE5、PCIe 加速卡 SC5H/SC5+、服务器 SG6。

2.2 BM1684 中的设备内存？

内存是 BM1684 应用调试中经常会涉及的重要概念，特别地，有以下 3 个概念需要特别区分清楚：Global Memory、Host Memory、Device Memory。

- **全局内存（Global Memory）：**指 BM1684 的片外存储 DDR，通常为 12GB，最大支持定制为 16GB。PCIe 模式：4GB TPU 专用 + 4GB VPU 专用 + 4GB VPP/A53 专用；SoC 模式：4GB A53 专用 + 4GB TPU 专用 + 4GB VPP/VPU 专用。
- **设备内存（Device Memory）和系统内存（Host Memory）：**根据 BM168x 产品类型或工作模式的不同，设备内存和系统内存具有不同的含义：
  - SoC 模式（SE5）Host Memory 是芯片上主控 Cortex A53 的内存 Device Memory 是划分给 TPU/VPP/VPU 的设备内存
  - PCIe 模式（SC5/SM5）Host Memory 是主机的内存 Device Memory 是 PCIe 板卡的设备内存
  - CModel 模式：SophonSDK 中提供的 BM1684 软件模拟器环境，可在没有 TPU 硬件的情况下，验证模型转换编译

2.2.1 SoC 内存映射分区表



2.2.2 各个内存间数据搬运速度说明

系统内共有三个内存分别为 DDR0-DDR2，其中 DDR0 是交错的。各个子系统访问规则如下：

	jpu	decode	encode	vpp	gdma
DDR0	可读写	不能访问	不能访问	可写	可读写
DDR1	可读写	可读写	可读写	可读写	可读写
DDR2	可读写	可读写	可读写	可读写	可读写

DDR0 的速度最快，DDR1-DDR2 次之，速度排列大致如下：

源	目的	速度
DDR0	DDR0	快
DDR0	DDR1	快
DDR0	DDR2	快
DDR1	DDR1	慢
DDR1	DDR2	快
DDR2	DDR2	慢

DDR 到 TPU(Local mem) 速度分析

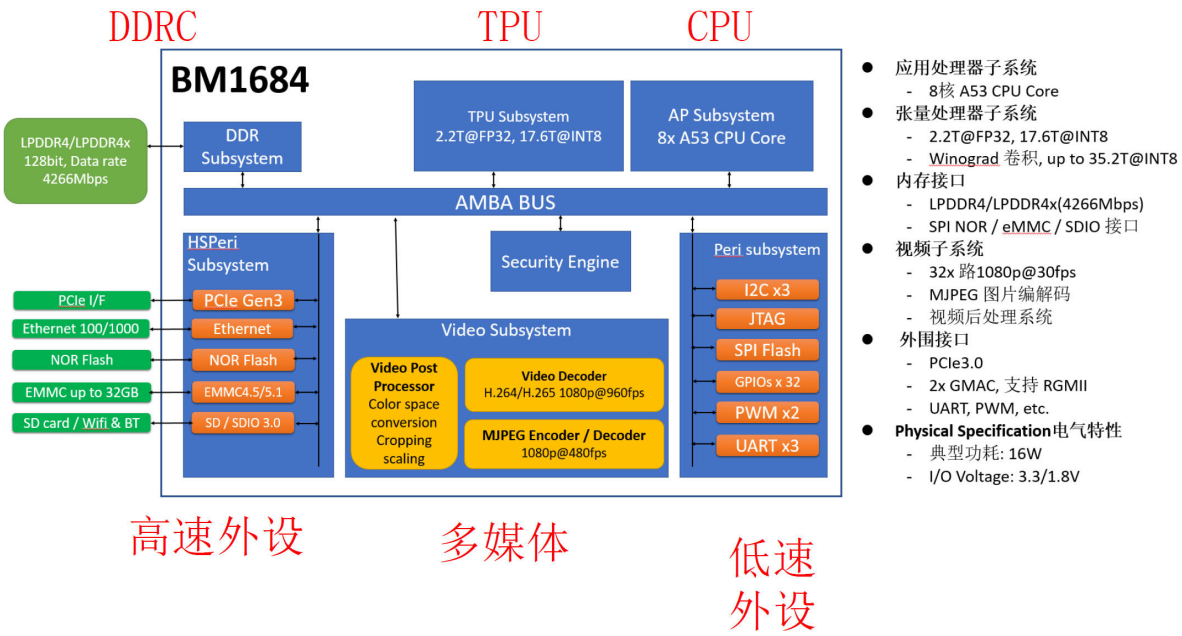
源	目的	速度
DDR0	Local mem	最快
DDR1	Local mem	快
DDR2	Local mem	快

SOC 下内存带宽速度参考数据如下：

```
./test_ddr_bindwidth 0x800000
Src:(addr=105100000, heap_id=0), Dst:(addr=105900000,heap_id=0), size:0x800000 byte.
Cost time:633 us, Bandwidth:12638.23 MB/s
Src:(addr=105100000, heap_id=0), Dst:(addr=440000000,heap_id=1), size:0x800000 byte.
Cost time:598 us, Bandwidth:13377.93 MB/s
Src:(addr=440000000, heap_id=1), Dst:(addr=105100000,heap_id=0), size:0x800000 byte.
Cost time:601 us, Bandwidth:13311.15 MB/s
Src:(addr=440000000, heap_id=1), Dst:(addr=440800000,heap_id=1), size:0x800000 byte.
Cost time:1264 us, Bandwidth:6329.11 MB/
```

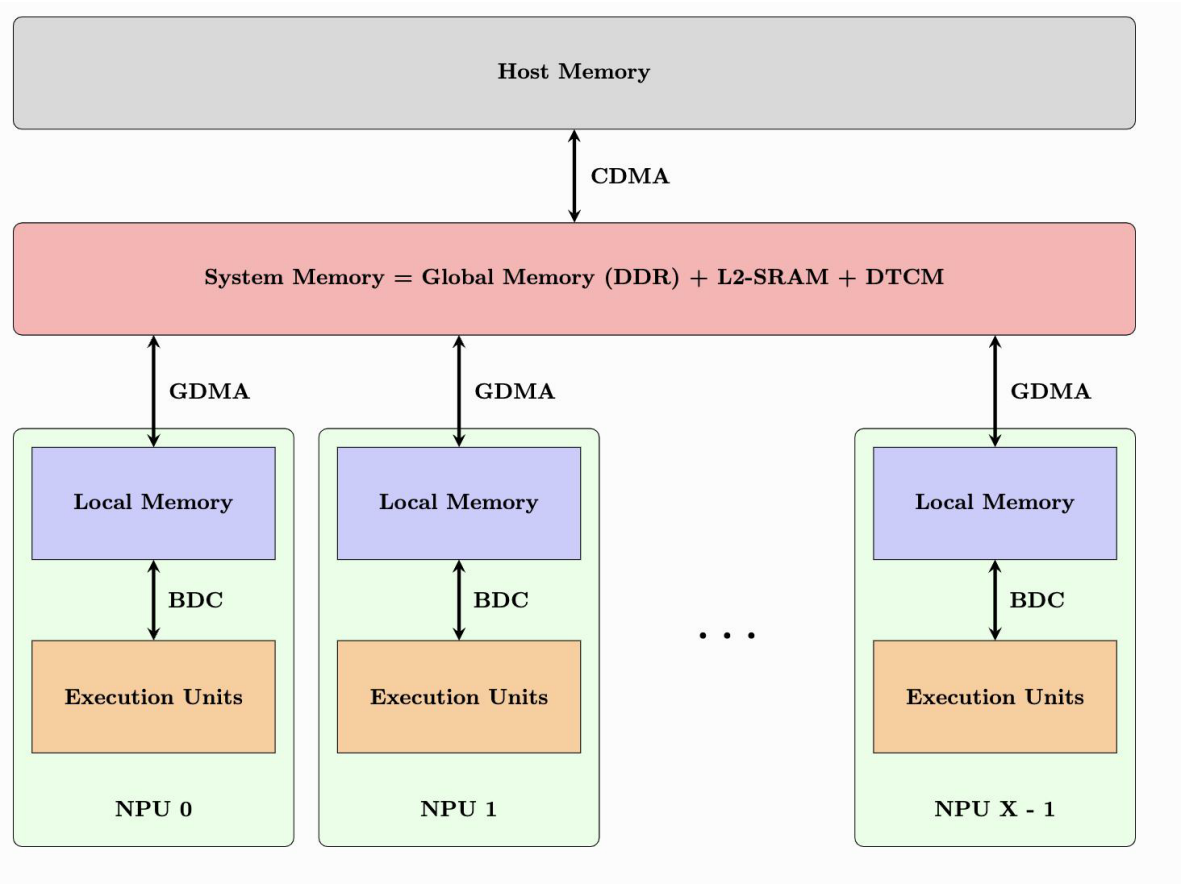
2.3 BM1684 的基本架构？

BM1684包含的模块概述



- 应用处理器子系统
  - 8核 A53 CPU Core
- 张量处理器子系统
  - 2.2T@FP32, 17.6T@INT8
  - Winograd 卷积, up to 35.2T@INT8
- 内存接口
  - LPDDR4/LPDDR4x(4266Mbps)
  - SPI NOR / eMMC / SDIO 接口
- 视频子系统
  - 32x 路1080p@30fps
  - MJPEG 图片编解码
  - 视频后处理系统
- 外围接口
  - PCIe3.0
  - 2x GMAC, 支持 RGMII
  - UART, PWM, etc.
- **Physical Specification**电气特性
  - 典型功耗: 16W
  - I/O Voltage: 3.3/1.8V

2.3.1 BM1684 TPU 的基本架构？



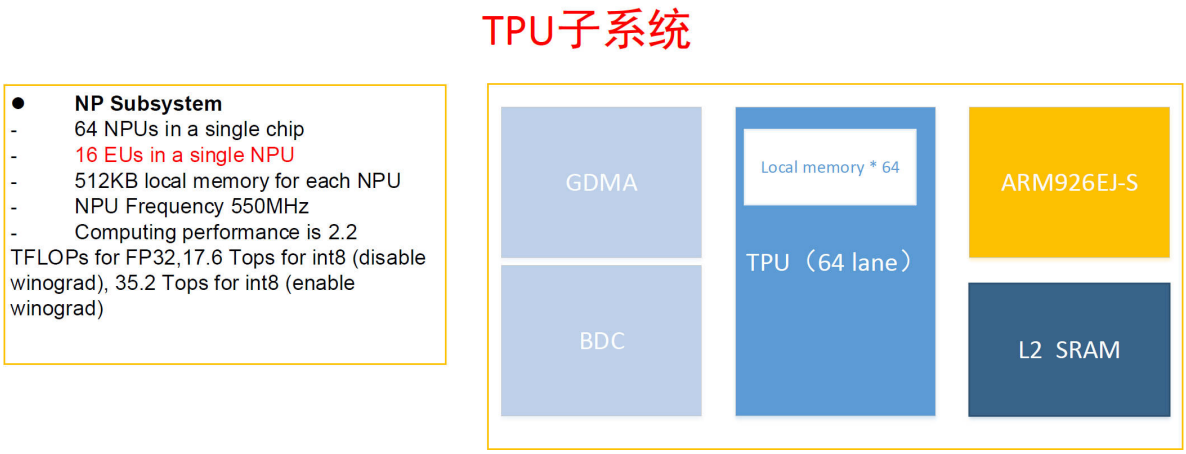
BM1684 TPU 是 SIMD(Single Instruction Mulit Data) 单指令多数据流众核架构，内部主要包括 BDC（SIMD 的控制器，控制 EU 做运算）和 GDMA（memory 间的数据搬运）。1684 内部还有 1 个 32 位 MCU ARM9，在运行动态编译生成的 bmodel 时将发送指令给 TPU 完成运算；EU 运算单元 1024 个 = 16 个 * 64 个 NPU；Local Memory 32MB = 512KB * 64 个 NPU，用于存放 EU 原子操作的输入输出数据。

2.3.2 什么是 DTCM？

答：DTCM 是 TPU 内部的 MCU ARM9 的高速缓存空间（512KB）。

2.3.3 TPU 的峰值算力是多少？

答：



峰值算力：

- FP32 峰值算力 = 64 * 16 * 2(FP32 MAC) * 2 * 0 55G / 1024 = 2.2 TOPS
- INT8 峰值算力 = 64 * 16 * 16(INT8 MAC) * 2 * 0 55G / 1024 = 17.6 TOPS
- 如果 enable winograd INT8 的算力可以提高一倍，winograd 要求卷积核必须是 3*3，会增大神经网络 weight 权值的体积。

注解:

NPU_NUM 1684 芯片中 NPU 运算单元个数值为 64；
EU_NUM 1684 芯片中每个 NPU 运算单元中 EU 个数为 16 个，每个 EU 有 2 个 FP32 的 MAC 单周期可以做一个乘法加上一个加法；
每个 EU 有 16 个 INT8 MAC，单周期可以做一个乘法加上一个加法。
FP32 MAC 和 INT8 MAC 同时只能有一种在工作。

这里乘以 2 的原因是每个周期可以做一个乘法加上一个加法，0.55 是指工作频率是 550MHz

2.4 名词解释

术语	说明
BM1684	算能科技面向深度学习领域推出的第三代张量处理器
BM1684X	算能科技面向深度学习领域推出的第四代张量处理器
VPU	BM168x 芯片中的解码单元
VPP	BM168x 芯片中的图形运算加速单元
JPU	BM168x 芯片中的图像 jpeg 编解码单元
SophonSDK	算能科技基于 BM168x 芯片的原创深度学习开发工具包
PCIE Mode	BM168x 的一种工作形态，芯片作为加速设备来进行使用，客户算法运行于 x86 主机
SoC Mode	BM168x 的一种工作形态，芯片本身作为主机独立运行，客户算法可以直接运行其上
CModel	BM168x 软件模拟器，包含于 SophonSDK 中，在不具备 TPU 硬件设备的情况下，可用于验证编译及完成模型转换
arm_pcie Mode	BM168x 的一种工作形态，搭载芯片的板卡作为 PCIe 从设备插到 ARM cpu 的服务器上，客户算法运行于 ARM cpu 的主机上
BMCompiler	面向算能科技 TPU 处理器研发的深度神经网络的优化编译器，可以将深度学习框架定义的各种深度神经网络转化为 TPU 上运行的指令流
BMRuntime	TPU 推理接口库
BMCV	图形运算硬件加速接口库
BMLib	在内核驱动之上封装的一层底层软件库，设备管理、内存管理、数据搬运、API 发送、A53 使能、功耗控制
UFamework(ufw)	算能自定义的基于 Caffe 的深度学习推理框架，用于将模型与原始框架解耦以便验证模型转换精度和完成量化
BMNetC	面向 Caffe 的 BMCompiler 前端工具
BMNetD	面向 Darknet 的 BMCompiler 前端工具
BMNetM	面向 MxNet 的 BMCompiler 前端工具
BMNetO	面向 ONNX 的 BMCompiler 前端工具
BMNetP	面向 PyTorch 的 BMCompiler 前端工具
BMNetT	面向 TensorFlow 的 BMCompiler 前端工具
BMNetU	INT8 量化模型的 BMCompiler 前端工具
BMPaddle	面向 Paddle Paddle 的 BMCompiler 前端工具
Umodel	算能自定义的 UFamework 下的模型格式，为量化模型时使用的中间模型格式
BModel	面向算能 TPU 处理器的深度神经网络模型文件格式，其中包含目标网络的权重（weight）、TPU 指令流等
BMLang	面向 TPU 的高级编程模型，用户开发时无需了解底层 TPU 硬件信息
OKKernel(BMKernel)	基于 TPU 原子操作 (根据芯片指令集封装的一套接口) 的开发库，需熟悉芯片架构、存储细节
SAIL	支持 Python/C++ 接口的 Sophon Inference 推理库，是对 BMCV、BMDecoder、BMLib、BMRuntime 等的进一步封装
winograd	一种卷积的加速算法

CHAPTER 3

环境配置常见问题

3.1 开发环境配置常见问题

3.1.1 如何安装和使用 SDK？

答：

1. 确认准备 Ubuntu 主机，安装好 docker
2. 【宿主机】加载 docker 开发镜像：

```
# 以下docker镜像包名称可能会随发布版本不同而有所变化，请注意修改
docker load -i <docker_dev_image_file>
```

3. 【宿主机】打开终端，执行如下命令解压缩文件：

```
# 以下SDK包名称可能会随发布版本不同而有所变化，请注意修改
tar zxf sophonsdk3_v<x.y.z>.tar.gz
```

4. 【宿主机】如果是 SC5 PCIe 加速卡，需要在宿主机安装驱动 (**如果为 x86 平台环境，请使用 docker；其他平台环境无需使用 docker**)：
   1) 打开终端执行 lspci | grep 1684 检查卡是否能够识别，正常情况应该输出如下信息：

```
01:00.0 Processing accelerators: Bitmain Technologies Inc. BM1684, Sophon Series Deep␣
↪Learning Accelerator (rev 01)
```

   2) 根据平台类型执行不同脚本安装驱动程序，注意使用 sudo root 权限安装

```
cd {$SDK_PATH}/scripts
x86平台:     sudo ./install_pcie_driver.sh
Mips64平台:     sudo ./install_driver_mips64.sh
Arm_pcie平台:     sudo ./install_driver_arm_pcie.sh
```

3) 检查是否安装成功执行 ls /dev/bm* 看看是否有/dev/bm-sohponX (X 表示 0-N)，如果有表示安装成功。正常情况下输出如下信息：

```
/dev/bmdev-ctl  /dev/bm-sophon0
```

5. 【宿主机】进入解压缩后的文件夹，执行脚本启动 docker：

```
# <***>中的内容可能会随发布版本不同而变化，请注意修改
./docker_run_<***>sdk.sh
```

6. 【docker 容器内】安装库：

```
cd <sdk_path>/scripts
./install_lib.sh nntc
```

7. 【docker 容器内】设置环境变量：

```
source envsetup_pcie.sh(有SC5卡) 或source envsetup_cmodel.sh（没有SC5卡）
```

8. 按照用户手册和文档操作即可

3.1.2 如何在 docker 容器内使用宿主机上的显示设备显示图像？

解决方案 1：启动容器时添加配置选项，使用宿主机显示器显示画面目前的主流图像界面服务 X11 支持客户端/服务端（Client/Server）的工作模式，只要在容器启动的时候，将『unix: 端口』或『主机名: 端口』共享给 docker，docker 就可以通过端口找到显示输出的地方，和 linux 系统共用显示设备：

1. 宿主机安装 x11-server：

```
sudo apt-get install x11-xserver-utils
```

2. 开放权限，允许所有用户，当然包括 docker, 访问 X11 的显示接口：

```
xhost +
```

3. 修改启动 docker 容器脚本，增加选项：

```
-v /tmp/.X11-unix:/tmp/.X11-unix \     #共享本地unix端口
-e DISPLAY=unix$DISPLAY \              # 修改环境变量DISPLAY
-e GDK_SCALE \                         # 与显示效果相关的环境变量
-e GDK_DPI_SCALE
```

4. 重新启动 docker 容器

解决方案 2：已经启动的容器，通过网络 IP 连接到 x11 server 显示图像（本地或局域网内其他主机都可以）若不想重新启动容器，可以通过 IP 地址来映射显示设备从而显示图像：

1. 查询 x11 server 主机 IP：假设为 192.168.150.100
2. 设置 x11 server 权限：

```
sudo gedit /etc/lightdm/lightdm.conf #增加一行 xserver-allow-tcp=true
sudo systemctl restart lightdm
xhost + #注意加号前应有空格
```

> **注解：** 如果您使用的系统的桌面管理器不是 lightdm，那么配置文件可能会不一样。比如 ubuntu18.04 以上默认使用 gdm3，其配置文件为/etc/gdm/custom.conf，您需要在 [security] 下增加 AllowRemoteAutoLogin=true，[xhmcp] 下增加 Enable=true, Port=177。重启 gdm 服务的命令为：service gdm restart

3. 在 docker 容器中设置 DSIPLAY 变量：

```
export DISPLAY= 192.168.150.100:0.0
```

测试是否成功

您可以在 docker 中安装以下显示时钟的小程序来验证显示设备已被正确配置

```
sudo apt-get install xarclock # 安装这个小程序
xarclock # 运行，如果配置成功，会显示出一个小钟表动画
```

说明：以上操作主要针对 Linux，若您想使用 windows 或者 Mac 主机作为显示设备，原理是相似的，请查找类似解决方案配置即可。

3.1.3 关于 ABI0 和 ABI1 版本的问题？

答：每个操作系统都会为运行在该系统下的应用程序提供应用程序二进制接口（Application Binary Interface，ABI）。ABI 包含了应用程序在这个系统下运行时必须遵守的编程约定。ABI 总是包含一系列的系统调用和使用这些系统调用的方法，以及关于程序可以使用的内存地址和使用机器寄存器的规定。ABI 的版本与操作系统、硬件设备有相关。SDK 原始包中提供了 ABI0 和 ABI1 两个版本的 so 库，在执行./install_lib.sh nntc 时，脚本会根据当前系统环境保留相应 ABI 版本的 so 库。

3.1.4 关于 CentOS 与程序 ABI 版本的问题？

3.1.4.1 在 CentOS7.6 上使用 gcc 6.5 编译程序链接 BM-OpenCV 库时报找不到符号链接的错误

```
thirdparty/lib/common -luuid -lxml2 -L../../../thirdparty/lib/sophgo/decode -lbmion -lbmjpuapi -lbmjpulite -lbmvideo -lbmvppapi -lbmvpuapi -lbmvpulite -lyuv -L../
../../thirdparty/lib/sophgo/sail -lsail -L../../../thirdparty/lib/sophgo/bmnn -lbmrt -lbmlib -lbmcv -lstdc++fs -lpthread
../../bin/engine/lib/module/libObjectPositionModule.so: undefined reference to `cv::dnn::dnn4_v20190122::Net::forward(cv::_OutputArray const&, std::vector<std::st
ring, std::allocator<std::string> > const&)'
../../bin/engine/lib/module/libDebugModule.so: undefined reference to `cv::putText(cv::_InputOutputArray const&, std::string const&, cv::Point_<int>, int, double,
 cv::Scalar_<double>, int, int, bool)'
../../bin/engine/lib/module/libObjectPositionModule.so: undefined reference to `cv::dnn::dnn4_v20190122::Net::getUnconnectedOutLayersNames() const'
../../bin/engine/lib/module/libObjectPositionModule.so: undefined reference to `cv::dnn::dnn4_v20190122::Net::setInput(cv::_InputArray const&, std::string const&,
 double, cv::Scalar_<double> const&)'
../../bin/engine/lib/module/libCommon.so: undefined reference to `cv::imencode(std::string const&, cv::_InputArray const&, std::vector<unsigned char, std::allocat
or<unsigned char> >&, std::vector<int, std::allocator<int> > const&)'
../../bin/engine/lib/module/libCommon.so: undefined reference to `cv::error(int, std::string const&, char const*, char const*, int)'
```

答：Centos 下直接安装的编译器，所有版本下默认都是使用 D_GLIBCXX_USE_CXX11_ABI=0，与我们的 BM-OpenCV 编译时使用的 ABI 版本（ABI=1）不兼容，因此，需要使用我们提供的 x86-linux-gcc 编译器，以确保 ABI 兼容，编译器请联系技术支持获取。

3.1.4.2 BM-OpenCV 的 abi 和客户的其他库的 abi 不一样，是否可以提供各种 abi 版本的 BM-OpenCV，或者 BM-OpenCV 的源码，自行进行编译？

答：BM-OpenCV 目前没有源码开放的计划；SDK 中其他 C++ 的库比较小，都提供了双份（ABI0 和 ABI1），BM-OpenCV 目前只有 ABI1 版本的。由于涉及 SDK 整个全家桶的发布，为了控制 SDK 的体积，暂时不提供 ABI0 版本的 BM-OpenCV；建议客户还是使用我们提供的编译器将自己的程序重新编译一遍。（那如果客户的依赖是直接 apt 安装的，客户不想都下载源码重新编译，应该怎么办?）

3.1.5 在 ARM 主机上编译 SDK/examples/SSD_object 中的示例，程序编译最后阶段输出 local symbol 的信息，是否是编译成功?

```
[root@localhost cpp_cv_bmcv_bmrt]# make -f Makefile.arm_pcie clean && make -f Makefile.arm_pcie
rm -f ssd300_cv_bmcv_bmrt.arm_pcie
aarch64-linux-gnu-g++ main.cpp ssd.cpp -g -O2 -Wall -std=c++11 -w -I../../../include/opencv/opencv4 -I../../../include/ffmpeg -I../../../include -I../../../include/bmruntime -I../../../include/bmlib -I../../../i
nclude/third_party/boost/include -I../../../NeuralNetwork/include -DCONFIG_LOCAL_MEM_ADDRWIDTH=19 -lbmrt -lbmlib -lbmcv -ldl -lopencv_core -lopencv_imgproc -lopencv_imgcodecs -lopencv_videoio -lbmvideo -lswresam
ple -lswscale -lavformat -lavutil -lprotobuf -lgflags -lglog -lboost_system -lboost_filesystem -lpthread -lbmjpuapi -lbmjpulite -Wl,-rpath=../../../lib/bmnn/arm_pcie -Wl,-rpath=../../../lib/opencv/arm_pcie -Wl,-
rpath=../../../lib/ffmpeg/arm_pcie -Wl,-rpath=../../../lib/decode/arm_pcie -Wl,-rpath=../../../lib/thirdparty/soc -Wl,-rpath=../../../lib/sys/aarch64 -L../../../lib/thirdparty/soc -L../../../lib/bmnn/arm_pcie -L
../../../lib/sys/aarch64 -L../../../lib/opencv/arm_pcie -L../../../lib/ffmpeg/arm_pcie -L../../../lib/decode/arm_pcie -o ssd300_cv_bmcv_bmrt.arm_pcie
/usr/bin/ld: ../../../lib/bmnn/arm_pcie/libbmrt.so: .dynsym local symbol at index 3 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/bmnn/arm_pcie/libbmrt.so: .dynsym local symbol at index 4 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/bmnn/arm_pcie/libbmrt.so: .dynsym local symbol at index 5 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/bmnn/arm_pcie/libbmrt.so: .dynsym local symbol at index 6 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/bmnn/arm_pcie/libbmrt.so: .dynsym local symbol at index 7 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/bmnn/arm_pcie/libbmrt.so: .dynsym local symbol at index 8 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/bmnn/arm_pcie/libbmrt.so: .dynsym local symbol at index 9 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_core.so: .dynsym local symbol at index 3 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_core.so: .dynsym local symbol at index 4 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_core.so: .dynsym local symbol at index 5 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_core.so: .dynsym local symbol at index 6 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_core.so: .dynsym local symbol at index 7 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_core.so: .dynsym local symbol at index 8 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_core.so: .dynsym local symbol at index 9 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgproc.so: .dynsym local symbol at index 3 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgproc.so: .dynsym local symbol at index 4 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgproc.so: .dynsym local symbol at index 5 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgproc.so: .dynsym local symbol at index 6 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgproc.so: .dynsym local symbol at index 7 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgproc.so: .dynsym local symbol at index 8 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgproc.so: .dynsym local symbol at index 9 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgcodecs.so: .dynsym local symbol at index 3 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgcodecs.so: .dynsym local symbol at index 4 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgcodecs.so: .dynsym local symbol at index 5 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgcodecs.so: .dynsym local symbol at index 6 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgcodecs.so: .dynsym local symbol at index 7 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgcodecs.so: .dynsym local symbol at index 8 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_imgcodecs.so: .dynsym local symbol at index 9 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_videoio.so: .dynsym local symbol at index 3 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_videoio.so: .dynsym local symbol at index 4 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_videoio.so: .dynsym local symbol at index 5 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_videoio.so: .dynsym local symbol at index 6 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_videoio.so: .dynsym local symbol at index 7 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_videoio.so: .dynsym local symbol at index 8 (>= sh_info of 3)
/usr/bin/ld: ../../../lib/opencv/arm_pcie/libopencv_videoio.so: .dynsym local symbol at index 9 (>= sh_info of 3)
[root@localhost cpp_cv_bmcv_bmrt]#
```

答：这主要是由于 arm 主机上 g++ 版本过高导致的，不影响运行。

3.1.6 如何在开发环境中的 Python 环境下使用我们提供的 BM-OpenCV?

答：只需设置好 PYTHONPATH 环境变量。不同的架构平台，请注意修改对应的路径。

```
export PYTHONPATH=$PYTHONPATH:/workspace/lib/opencv/x86/opencv-python
```

3.1.7 CentOS 宿主机上在官方 ubuntu 开发 docker 中使用 sail 模块报找不到符号错误？

```
root@localhost:/workspace/examples# ls
Resnet_classify  SSD_object    YOLOv3_object  calibration  multimedia  okkernel
RetinaFace       YOLOX_object  YOLOv5_object  centernet    nntc        sail
root@localhost:/workspace/examples# cd YOLOv5_object/
root@localhost:/workspace/examples/YOLOv5_object# ls
README.md  cpp  data  python
root@localhost:/workspace/examples/YOLOv5_object# cd python/
root@localhost:/workspace/examples/YOLOv5_object/python# ls
__init__.py  bmrt_pytorch.py  pytorch.py  sail.py  utils.py
root@localhost:/workspace/examples/YOLOv5_object/python# python3.7 sail.py
Traceback (most recent call last):
  File "sail.py", line 5, in <module>
    import sophon.sail as sail
ImportError: /root/.local/lib/python3.7/site-packages/sophon/sail.cpython-37m-x86_64-linux-gnu.so: undefined symb
ol: _ZN2cv7imwriteERKSsRKNS_11_InputArrayERKSt6vectorIiSaIiEE
root@localhost:/workspace/examples/YOLOv5_object/python#
root@localhost:/workspace/examples/YOLOv5_object/python#
root@localhost:/workspace/examples/YOLOv5_object/python#
root@localhost:/workspace/examples/YOLOv5_object/python#
```

答：

1. 排查是否环境变量没有设置好，是否有执行 source envsetup_xxx.sh；
2. 环境要求的 ABI 版本和 so 文件的 ABI 版本不匹配：具体来说，SDK 里原始会存放 ABI0 和 ABI1 2 个版本的库，./install_lib.sh nntc 这个脚本，会根据现场环境，去选择一个合适的版本，然后把多余的删掉，并重命名相关库。centos 和 ubuntu 用的 ABI 版本不一样，如果在 centos 的宿主机上执行过这个脚本，那么 SDK 中的库就变成只适用于 centos 了，这个时候在 ubuntu 的 docker 中就无法使用这个库了。请重新解压原

始的 SDK 文件，挂载到 docker 内部后重新执行环境配置命令（./install_lib.sh nntc 和 source envsetup_xxx.sh 以及 SAIL 的安装）。

3.2 常用工具有哪些？

3.2.1 bm-smi 等设备管理工具

答：

我们提供 bm-smi 工具、proc 文件系统接口和 sysfs 文件系统接口用于获取设备状态信息。

- bm-smi 以界面或者文本的形式显示设备状态信息，如设备的温度、风扇转速等信息；也可使能、禁用或者设置设备的某些功能，如 led、ecc 等。
- proc 文件系统接口：在/proc 节点下创建设备信息节点，用户通过 cat 或者编程的方式读取相关节点获取设备温度、版本等信息。
- sysfs 文件系统接口：用来获取 TPU 的利用率等信息。

注解： bm-smi 侧重于以界面的形式直观显示设备信息，proc 文件系统接口侧重于为用户提供编程获取设备信息的接口。

3.2.2 bm_model.bin

答：通过 bm_model.bin 工具，可以查看 bmodel 文件的参数信息，可以将多个网络 bmodel 分解成多个单网络的 bmodel，也可以将多个网络的 bmodel 合并成一个 bmodel，具体请查看：https://doc.sophgo.com/docs/3.0.0/docs_latest_release/nntc/html/usage/bmodel.html。

3.2.3 bmrt_test

答：bmrt_test 是基于 bmruntime 接口实现的对 bmodel 的正确性和实际运行性能的测试工具，具体请查看：https://doc.sophgo.com/docs/3.0.0/docs_latest_release/nntc/html/usage/bmrt_test.html。

3.2.4 view_demo

答：使用 view demo，可以通过 calibration 可视化分析工具查看网络量化误差，此工具通过运行 fp32 和 int8 网络，并对其每层的输出进行比较，以图形化界面形式直观显示每层数据的量化损失。具体请查看：https://doc.sophgo.com/docs/3.0.0/docs_latest_release/calibration-tools/html/module/chapter4.html#id17。

3.2.5 BMProfile

答：BMProfile 是将用户产生的 profile 信息以可视化的方式展示出来。主要作用是辅助用户进行网络的性能分析，查看 Graph、SubGraph、layer、算子、指令等各个级别的性能数据和 Memory 使用情况。此外，还可以对静态网络的指令进行分析过滤，以分析出越界读写等问题。具体使用请查看：https://doc.sophgo.com/docs/3.0.0/docs_latest_release/nntc/html/usage/bmprofile.html。

3.3 如何查看版本信息？

答：

SDK 版本：${SDK_PATH}/release_version.txt；
PCIe 模式固件版本：cat /proc/bmsophon/card0/versions；
SOC 模式固件版本：cat /system/data/buildinfo.txt。

3.4 使用 bm-smi 查看信息时发现，没有程序运行，但是设备内存被占用？

```
+-----------------------------------------------------------------------------------------------------------+
| SDK Version:    2.6.0             Driver Version:  2.6.0
+-----------------------------------------+-----------------------------------------------------------------
|card  Name      Mode        SN          |TPU  boardT  chipT   TPU_P  TPU_V  ECC  CorrectN  Tpu-Util
|12V_ATX  MaxP boardP Minclk Maxclk  Fan|Bus-ID      Status   Currclk   TPU_C   Memory-Usage
|=========================================+=================================================================
| 0  1684-SC5+   PCIE HQDZKC5BJAAAC0348 | 0    35C     37C    1.5W   616mV  OFF    N/A          0%
|2074mA   75W   29W   75M     550M   N/A|000:40:00.0 Active    550M      2.4A   178MB/11379MB
|                                       |-----------------------------------------------------------------
|                                       | 1    38C     36C    1.9W   615mV  OFF    N/A          0%
|                                       |000:40:00.1 Active    550M      3.1A   178MB/11379MB
|                                       |-----------------------------------------------------------------
|                                       | 2    37C     36C    1.2W   616mV  OFF    N/A          0%
|                                       |000:40:00.2 Active    550M      1.9A   178MB/11379MB
+=========================================+=================================================================

+-----------------------------------------------------------------------------------------------------------+
| Processes:                                                                                   TPU Memory
|  TPU-ID       PID   Process name                                                             Usage
+-----------------------------------------------------------------------------------------------------------+
```

答：这是正常的。从 SDK2.6.0 开始 bm-smi 中显示的设备内存增加了 VPU 对设备内存的使用，这里的 178MB 是安装完驱动以后就分配了的空间，即使没有程序在运行，也会一直存在。

CHAPTER 4

设备使用常见问题

4.1 通用问题

4.1.1 如果遇到 bm send api failed，TPU 无法正常工作怎么办？

答：

1. 检查 TPU 是否为 100% 利用率，杀掉占用进程；
2. 复位 TPU：执行 bm-smi –dev=0x0 –recovery；
3. 如果仍然不行，请将设备断电以后再重新上电。

4.2 智算卡常见问题

4.2.1 SC5+ 在主机运行过热的情况

答：SC5+ 主机选择，请注意主机散热环境，SC5+ 散热角度服务器选型建议：

1. SC5+ 被动卡进风口温度与需求风量对应表

表 4.1: SC5+ 被动卡进风口温度与需求风量

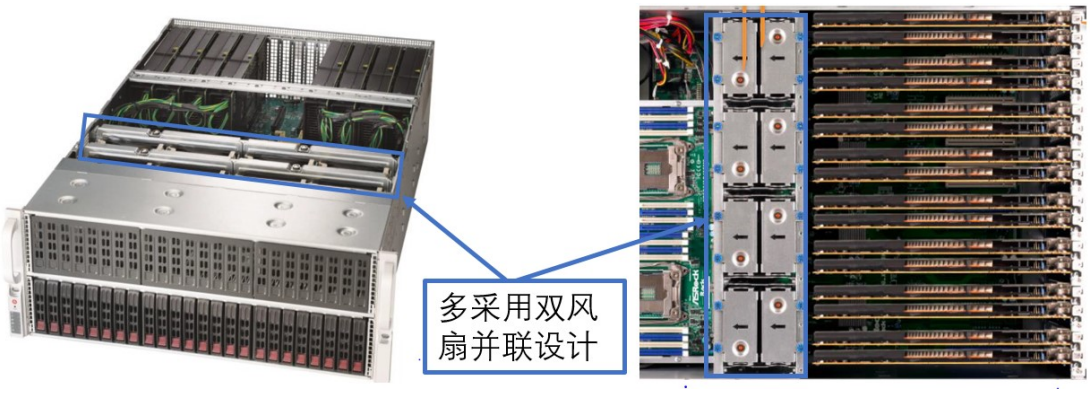
卡进风口平均温度（℃）	30℃ 以下	35℃	40℃	45℃	50℃	55℃
最低 Airflow（cfm）	3.5	4.3	5.3	6.7	8.6	10.5

注解:

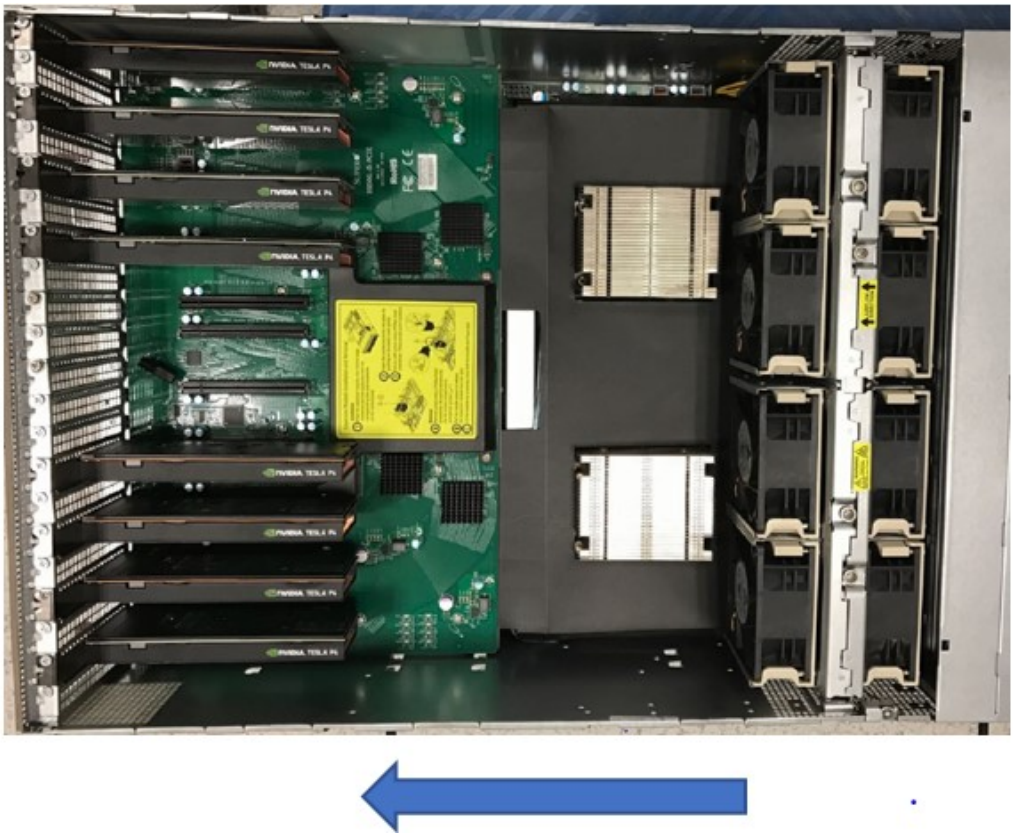
1) 表中对应温度下的风量为需求的最小风量要求，具体风量是否满足需系统厂商实测确定

2) 卡进风口温度在 25 度以下时，只要智能卡接通电源，无论是 idle 或满载状态下，系统都需要提供与 30 度环境一样的最小风量

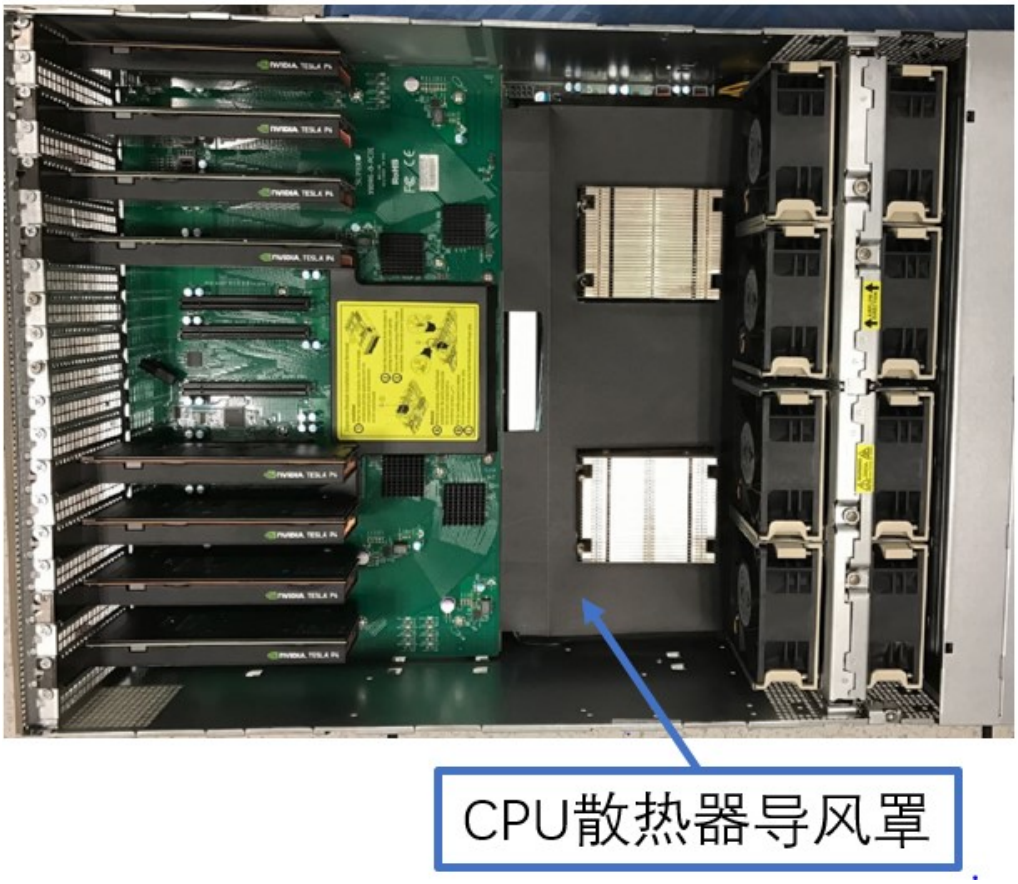
2. 建议选择搭配高风量及风压（或风扇并联）的服务器，便于智能卡散热
   a. 1U 机型的风扇建议选用：4056 风扇或两个 4028 风扇并联
   b. 2U 机型的风扇建议选用：8056 风扇或两个 8038 风扇并联
   c. 4U 机型的风扇建议选用：2 组 8056 风扇 (或两个 8038 并联)



3. 风道设计
   a. 建议最好选用前进后出的风道设计



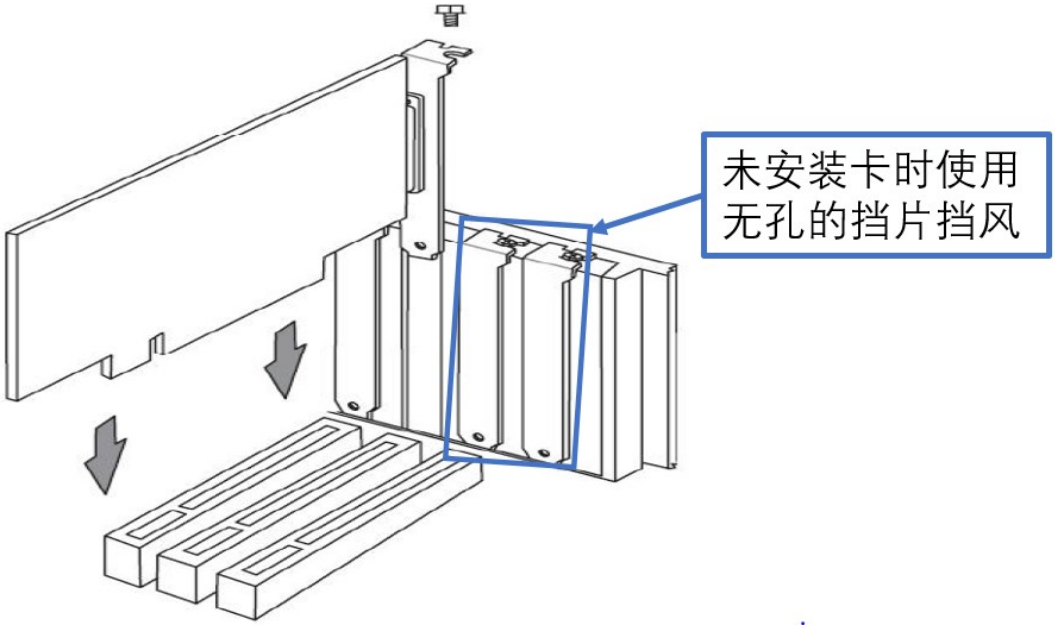
b. 导风罩设计：建议智能卡上做专用导风罩设计，与 CPU 导风罩分离；如下图所示，CPU 散热器为一个导风罩，智能卡需单独设计导风罩，导风罩设计以最便于散热为目的



c. 选用服务器时卡进风口最好不要有高发热原机器件，如下图，CPU 为高发热器件，其散热器需要选择高度矮一些的，避免挡住卡进风口

4. 系统出风口设计

   确保任何空的扩展槽上均安装一个挡片，一个未装挡片的开放扩展槽会降低系统散热效果，由此可能引起过热而影响系统性能或导致部件损害



4.2.2 SC5 安装

4.2.2.1 SC5 快速安装指南

- 在主机关机状态下将 SC5 安装至 PCIe 插槽，需要插到位，SC5 从 PCIe 接口直接供电，不需要外接 ATX 2x3 电源。注意，服务器最好是拔掉 220V 电源再安装，某些服务器，在 220V 输入时，PCIe 插槽有漏电。
- 安装完后，先不要盖好后盖，power on 上电，从上侧观察 LED 状态，如下图所示，4 个灯的状态，蓝蓝绿红，为 SC5 上电正常。
- 进入系统，输入命令 lspci | grep 1684，正常识别的话，会看到 1684 设备，如下图所示。

```
lspci | grep 1684
06:00.0 Processing accelerators: Device 1e30:1684 (rev 01)
06:00.1 Processing accelerators: Device 1e30:1684 (rev 01)
06:00.2 Processing accelerators: Device 1e30:1684 (rev 01)
```

- 如没有 1684 设备，请再次确认 LED 状态是否正常；
- 如 LED 异常，关机，断电，断开 220V，重新上电，再次观察 LED 状态，如正常，请操作步骤 4
- 如 LED 正常，lspci 命令下，仍无法识别设备，请收集硬件信息，和 OS 版本，反馈给我们。

4.2.2.2 PCIe 模式驱动安装的步骤与注意事项？

答：安装驱动成功的前提是主机需要能够扫描到我们的板卡，通过 lspci 可以查看，我们一个三芯卡 SC5+ 有 3 个 function，一个 SC5H 有一个 function。lspci 能显示我们的设备是安装驱动的必要条件，如果主机发现不了设备，则驱动安装会异常。

4.2.2.3 内核更新与驱动重装

答：内核版本升级会导致重启后不能自动加载驱动：例如当前内核版本是：uname -r 4.15.0-142-generic 我们安装完驱动后，驱动放在下面目录：/lib/modules/4.15.0-142-generic/kernel/drivers/pci/bmsophon.ko 如果内核版本升级到了 4.15.0-150-generic，则系统启动时会去/lib/modules/4.15.0-150-generic/kernel/drivers/pci/ 目录下去找 bmsophon.ko，会失败，解决方法是重新装驱动，所以我们建议禁止内核自动升级。

4.2.2.4 编译驱动过程报 gcc/make/linux header 找不到的错误？

答：build-essential、linux-headers-generic 这些在插有板卡的 host 机器上要先安装好，否则会报 gcc/make/linux header 找不到的错误。

4.2.2.5 驱动安装的其他注意事项？

答：

1. 存放 sdk 的路径中不能有空格：如，/home/abc/foo bar/bmnnsdk2-bm1684_v2.4.0/scripts，这个路径中存在“foo bar”这样的带空格的目录，会影响驱动安装，请一定不要建带空格的目录；
2. 某些发行版，例如 centos7.4 已经不再维护，没有 Linux header，导致编译失败，建议安装 centos7.9 ；
3. 某些内核版本导致我们的驱动编译不过，需要我们改驱动源码。这种情况下，安装驱动过程中会有失败 log，请将 log 发给我们分析。；
4. 板卡温度过热，或者板卡和卡槽接触不良导致安装失败，会有 kernel 错误 log；这种情况下，请把 kernel log(dmesg 命令显示的 log) 发给我们分析；
5. 板卡个体故障原因导致安装失败，建议更换板卡；这种情况下，请把 kernel log(dmesg 命令显示的 log) 发给我们分析；
6. lspci 必须能发现我们的设备，才能出现设备节点，如果没有出现设备节点，请用上面命令检查系统是否有扫描到 PCIE 设备。

4.2.3 SC5+ 三芯卡无法被识别？

该问题很可能跟散热问题相关，因散热不满足要求芯片温度过高而造成：

1. 看一下是 PC 还是标准服务器，如果是 PC，通常风道和风量不能符合三芯卡的被动散热要求，建议加装额外的风扇进行散热，或者采用 SC5H 单芯片主动散热卡；
2. 标准服务器，先看一下插卡的槽位是否是标准的 X16 槽位，X8 槽位的功率支持通常最大只有 45W，不建议使用；
3. 看一下服务器的风扇是否直对 PCIE 卡的进风口，中间有否物理阻挡；
4. 将服务器的上盖盖好，如果开盖，风扇的风会散逸，导致散热效果差；
5. 看一下服务器的所有 PCIE 槽位都加装了挡片，没有加装挡片的需要加装挡片；
6. 通过 BIOS 调整服务器的风扇转速到最大档位；
7. 建议将服务器放置于有空调的房间或者标准机房环境进行测试；
8. 针对三芯卡的散热，如果必须使用散热条件不足的工控机、PC、非 GPU 型服务器等，建议由主机厂商增加导风罩为加速卡设立专用风道，便于卡片散热；
9. 风扇 1）为 12V 供电，2pin 风扇，最大电流 0.36A，不可调速；风扇 2）和 3）为 12V 供电，4pin 风扇，最大电流 0.68A，可调速；安装方式：风扇需要安装到卡的入风口处，风扇出风口与卡进风口的距离建议最好不要超过 3mm，如果条件许可的话，建议风扇出风口与卡进风口的地方密封，防止漏风。

4.2.3.1 在 X86 服务器主机上安装卸载驱动的方法:

```
# 安装:
$ cd <sdk_path>/scripts
$ sudo ./install_driver_pcie.sh

# 安装成功后/dev/会有bmdev-ctl和bm-sophonX(X=0/1/2/3/4/5/6)这样的设备节点出现

# 卸载:
$ cd <sdk_path>/scripts $ sudo./remove_driver_pcie.sh
```

4.2.3.2 在 ARM64(飞腾/鲲鹏) 服务器主机上安装卸载驱动的方法:

```
# 安装:
$ cd <sdk_path>/scripts
$ sudo ./install_driver_arm_pcie.sh

# 安装成功后/dev/会有bmdev-ctl和bm-sophonX(X=0/1/2/3/4/5/6)这样的设备节点出现

# 卸载:
$ cd <sdk_path>/scripts $ sudo./remove_driver_arm_pcie.sh
```

4.2.3.3 在 MIPS3000/4000 服务器主机上安装卸载驱动的方法:

```
# 安装:
$ cd <sdk_path>/scripts
$ sudo ./install_driver_mips64.sh

# 安装成功后/dev/会有bmdev-ctl和bm-sophonX(X=0/1/2/3/4/5/6)这样的设备节点出现

# 卸载:
$ cd <sdk_path>/scripts $ sudo./remove_driver_mips64.sh
```

驱动安装完后，系统会有 sophon 模块，可以通过下面命令确认:

```
lsmod | grep bm
bmsophon 2752512 0
```

如果安装驱动不成功的话，是不会出现 bmsophon 模块的

4.2.4 PCIe 模式如何查看 VPU 内存分配情况？

答：PCIe VPU 内存使用情况：

```
bm-smi --opmode=display_memory_detail
```

4.2.5 如何判断机器上是否有 sophon 设备

答:

1. 卡插入卡槽后，lspci | grep 1684，正常识别的话，会看到 1684 设备

   ```
   类似这样的 06:00.0 Processing accelerators: Device 1e30:1684 (rev 01)
   ```

2. 卡被识别，驱动安装成功后，系统会有 sophon 模块

   ```
   lsmod | grep bm 会输出：类似 bmsophon 2752512 0
   ```

3. 卡被识别，驱动安装成功后，会有/dev/bmdev-ctl /dev/bm-sophon0 节点

   ```
   /dev/bmdev-ctl
   /dev/bm-sophon0节点
   ```

4. 卡被识别，驱动安装成功后，获取设备数量中可使用如下代码

   ```
   bm_dev_getcount(int *count)
   ```

4.3 智算盒子（模组）常见问题

4.3.1 刷机问题

4.3.1.1 手动升级 SM5/SE5 固件的方式

SoC 模型下有多种更新固件的方式：

1. **文件替换**直接更新 kernel、预编译的 SDK 库以及 bootloader 等文件：文件替换方式是指在 SoC 系统中直接通过替换对应文件的方式分别升级 bootloader、kernel 和 SDK 等其它软件。这种方式有一定的风险，如不同软件组件之间的版本匹配、文件损坏等。请参考《智算模组 SM5 软件开发指南》2.2 软件更新 b. 文件替换章节。

   注解:

   1. 升级 kernel: sudo cp emmcboot.itb /boot/;
   2. 升级预编译的库：sudo tar xzf system.tgz -C /system;
   3. 升级 bootloader：sudo flash_update -i spi_flash.bin -b 0x6000000 -f 0x0;
   4. 保存并重启使生效：sudo sync; sudo reboot.

2. 使用 **SD 卡刷**烧写整个固件：这种方式最为干净可靠，理论上只要您的 SE5/SM5 硬件没有损坏，都可以进行 SD 卡刷机，具体步骤请参考《SE5 用户手册》6.1 节系统升级或《智算模组 SM5 软件开发指南》2.2 软件更新 a.SD 卡刷机。（注意：带有预置算法应用的卡刷包和 干净系统的卡刷包 是不一样的，请在升级前核实清楚您的需求，并向技术支持获取相应卡刷包；**卡刷会重写整个 eMMC，也即您存储在 eMMC 的数据全部会丢失，请务必做好数据备份**。）
3. 通过以太网，使用 **tftp 刷机** 专用文件 升级：请参考《SM5 开发手册》7.4 使用 tftp 刷机。
4. 使用 **DDT 设备扫描工具**更新：请联系技术支持获取，目前仅提供 Windows 客户端。需要注意的是，DDT 设备扫描工具依赖于我方安装在 SE5/SM5 中的服务程序，若您使用自己定制的固件和操作系统，本方式不一定适用。

   DDT 设备扫描工具是算能科技 SE5 和 SM5 产品（以下简称产品）配套的辅助工具，它主要提供如下两种功能：

   1) 自动扫描：发现同一局域网内的所有相关 SE5 或 SM5 产品，支持 IP 地址等基础信息更改。
   2) 软件升级：支持对勾选的指定产品进行单个或者批量软件升级。

4.3.1.2 SE5 如何刷新固件

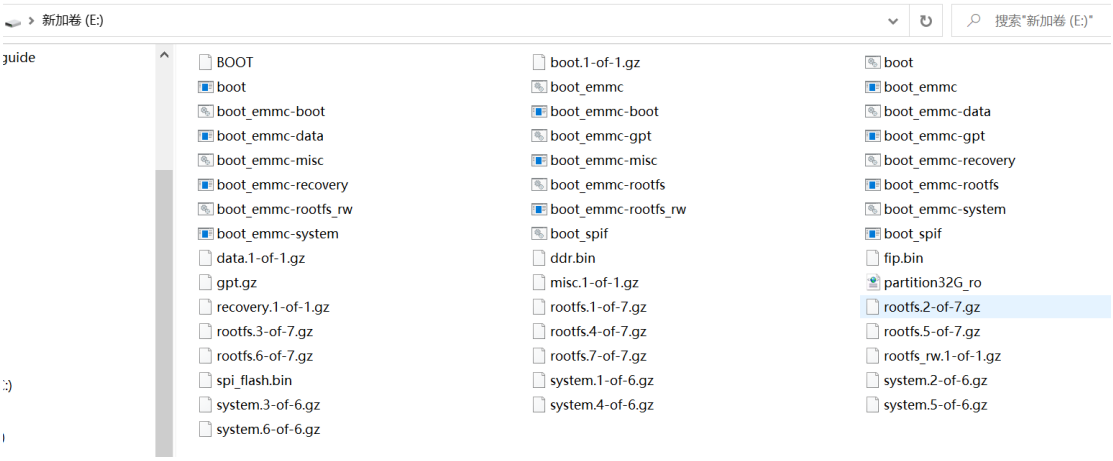
1. 准备一张 16GB 以上的 miniSD 卡和读卡器，并将卡格式为 **fat32 格式**。
   1. Ubuntu 系统（版本 16.04）可以通过界面直接格式为 **fat32** (Compatible with all system and devices(FAT))
   2. Ubuntu 系统（版本 18.04）通过命令行的方式 (下面举例)

```
df -h                          ----找到U盘的序列号，本例 "/dev/sdb1"
sudo umount /dev/sdb1          ----解除U盘挂载
sudo mkfs.vfat -F 32 /dev/sdb1  ----格式化U盘为fat32格式
```

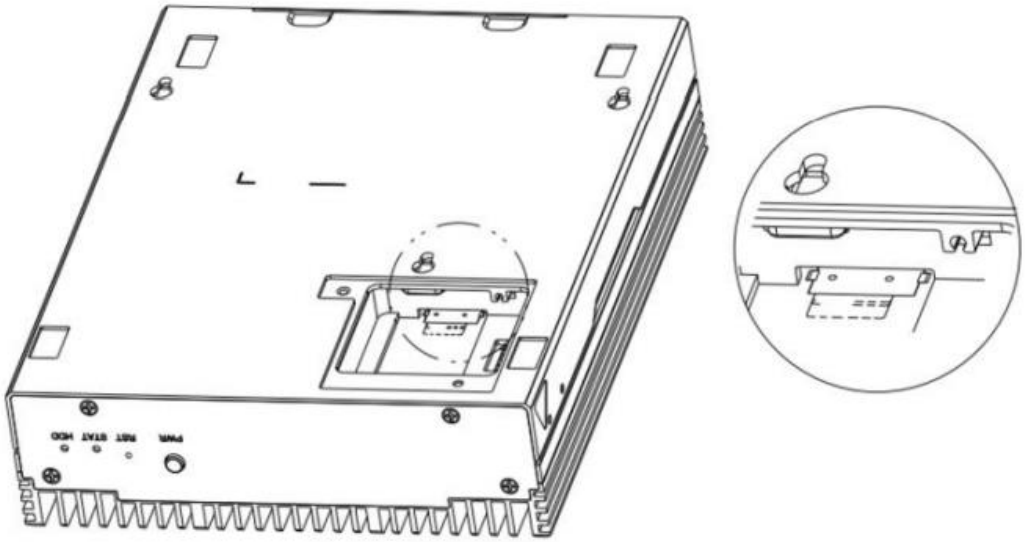
```
Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Mon Jan 10 14:23:03 2022 from 172.28.192.145
yang@yang-Vostro-3667:~$ sudo -i
[sudo] password for yang:
root@yang-Vostro-3667:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.8G     0  7.8G   0% /dev
tmpfs           1.6G  2.1M  1.6G   1% /run
/dev/sda1       458G   79G  356G  19% /
tmpfs           7.8G  904K  7.8G   1% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/loop3      768K  768K     0 100% /snap/gnome-characters/741
/dev/loop5      768K  768K     0 100% /snap/gnome-characters/761
/dev/loop4       66M   66M     0 100% /snap/gtk-common-themes/1515
/dev/loop1       56M   56M     0 100% /snap/core18/2284
/dev/loop2       44M   44M     0 100% /snap/snapd/14295
/dev/loop6      2.5M  2.5M     0 100% /snap/gnome-calculator/884
/dev/loop7      640K  640K     0 100% /snap/gnome-logs/106
/dev/loop0       43M   43M     0 100% /snap/snapd/14066
/dev/loop9       66M   66M     0 100% /snap/gtk-common-themes/1519
/dev/loop11     2.7M  2.7M     0 100% /snap/gnome-system-monitor/169
/dev/loop10      62M   62M     0 100% /snap/core20/1270
/dev/loop8       62M   62M     0 100% /snap/core20/1242
/dev/loop12     296M  296M     0 100% /snap/vlc/2344
/dev/loop14     2.7M  2.7M     0 100% /snap/gnome-system-monitor/174
/dev/loop13     128K  128K     0 100% /snap/bare/5
/dev/loop15      56M   56M     0 100% /snap/core18/2253
/dev/loop16     219M  219M     0 100% /snap/gnome-3-34-1804/77
/dev/loop17     248M  248M     0 100% /snap/gnome-3-38-2004/87
/dev/loop18     243M  243M     0 100% /snap/gnome-3-38-2004/76
/dev/loop19     219M  219M     0 100% /snap/gnome-3-34-1804/72
tmpfs           1.6G   16K  1.6G   1% /run/user/121
overlay         458G   79G  356G  19% /var/lib/docker/overlay2/06dd2e3e9f4ae95e2b285636bac4c55f0874
tmpfs           1.6G   48K  1.6G   1% /run/user/1000
/dev/sdb1        60G  107M   60G   1% /media/yang/新加卷
root@yang-Vostro-3667:~# sudo umount /dev/sdb1
root@yang-Vostro-3667:~# sudo mkfs.vfat -F 32 /dev/sdb1
mkfs.fat 4.1 (2017-01-24)
root@yang-Vostro-3667:~#
```

   3. win10 需要磁盘管理删除卷再添加新卷，用 Diskgenius 工具格式化为 **fat32**

2. 下载压缩包，并找到固件升级包（**例如：** sdcard_V5R7C01.tgz)，并将其解压后的所有文件拷贝到 miniSD 卡的根目录下：



3. 确保 SE5 处于 **断电状态**，拆卸维护窗盖板，插入 MicroSD 卡，并将设备上电，此时 STAT 灯呈现为红色（常亮）状态：



4. 等待设备升级完成，待 STAT 灯由红色（常亮）变为绿色（闪烁）状态后，将设备下电，并将 MicroSD 卡移除；
5. 升级完成，将 SE5 重新上电。

4.3.2 SE5 使用问题

4.3.2.1 SE5 支持的外围设备有哪些？

答：请参考设备支持列表：

- **2.5 机械硬盘**

供应商	型号	参数
希捷	ST2000LM015	厚度：7mm；Capacity：2TB；工作温度：0-60 度

- **2.5 SSD**

供应商	型号	参数
慧荣科技	MA619DEGMDF6	厚度：7mm；Capacity：1TB；工作温度：-40-85 度
FORE-SEE	S801S256G	厚度：7mm；Capacity：256GB；工作温度：0-70 度

- **USB WIFI 模组**

供应商	型号	参数
EDUP	EP-MS1558	300M 无线网卡
EDIMAX	EW-7822UAn	300M 无线网卡

- **USB 声卡**

供应商	型号	参数
PHILIPS	SWR1656Y/93	USB 转耳机口

- **USB 蓝牙发射器**

供应商	型号	参数
UGREEN 绿联	CM408	蓝牙版本 5.0，传输距离 10 米

- **USB 摄像头**

供应商	型号	参数
奥尼	奥尼 A10 HD1080P	黑色，1920*1080 分辨率，MAX FPS：30，像素 500 万

- **USB 读卡器**

供应商	型号	参数
暂无		

- **网络继电器**

供应商	型号	参数
泥人科技	TCP-KP-C2	工业级，2 路网络继电器，支持脱机定时，input：湿触点 x1，output：常开 X2

- **4G 模组**

供应商	型号	参数
广和通无线	NL668-CN-02	4G 通信模组，适用于多种网络制式，多频段。工作温度-30C ~ 75C
广和通无线	NL668-CN-22	4G 通信模组，适用于多种网络制式，多频段。工作温度-30C ~ 75C

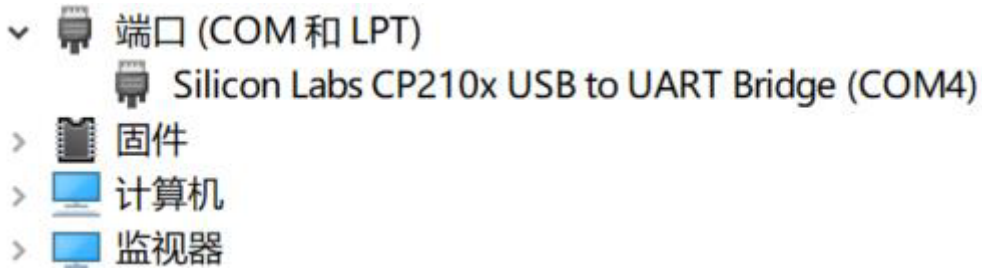
- **485 设备**

供应商	型号	参数
妙观科技有限公司	TH10S-B-H	温湿度传感器，工作温度-40-125C

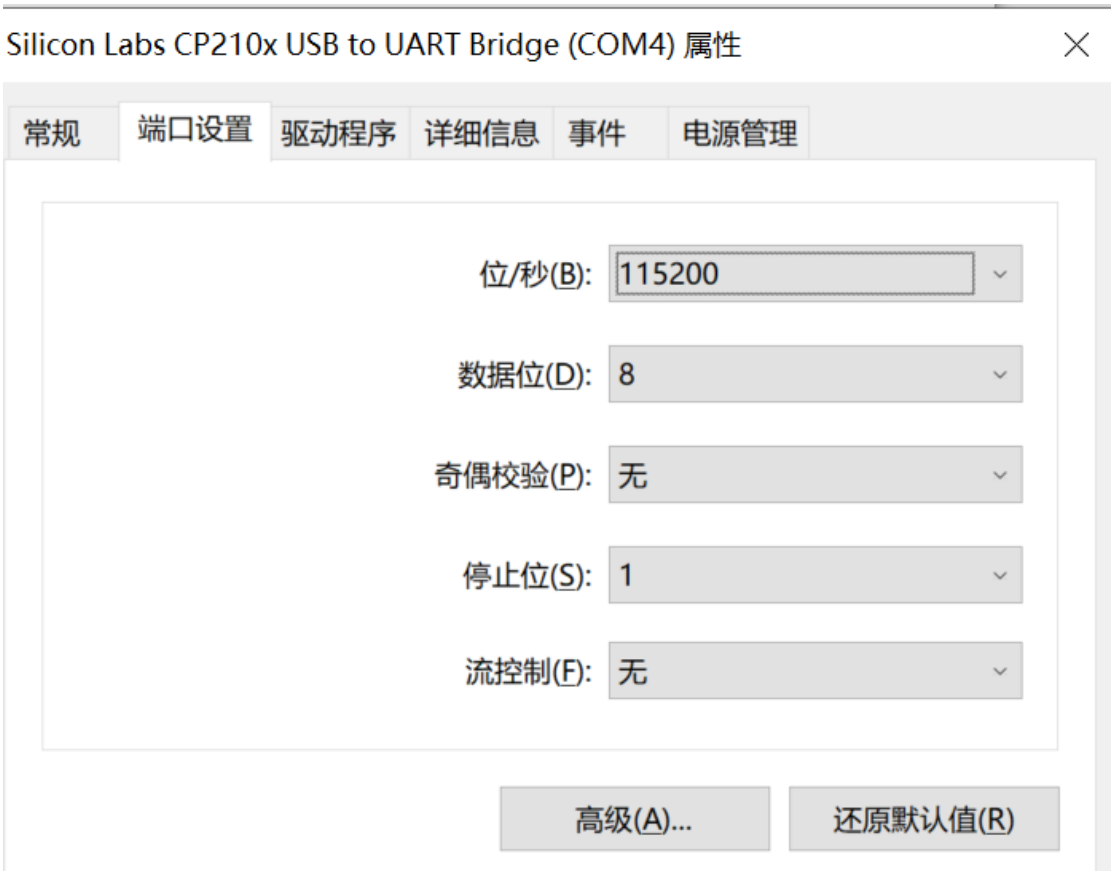
4.3.2.2 SE5 维护窗里的 Micro-USB 如何使用

1. 将 Mcro-USB 线与电脑连接：

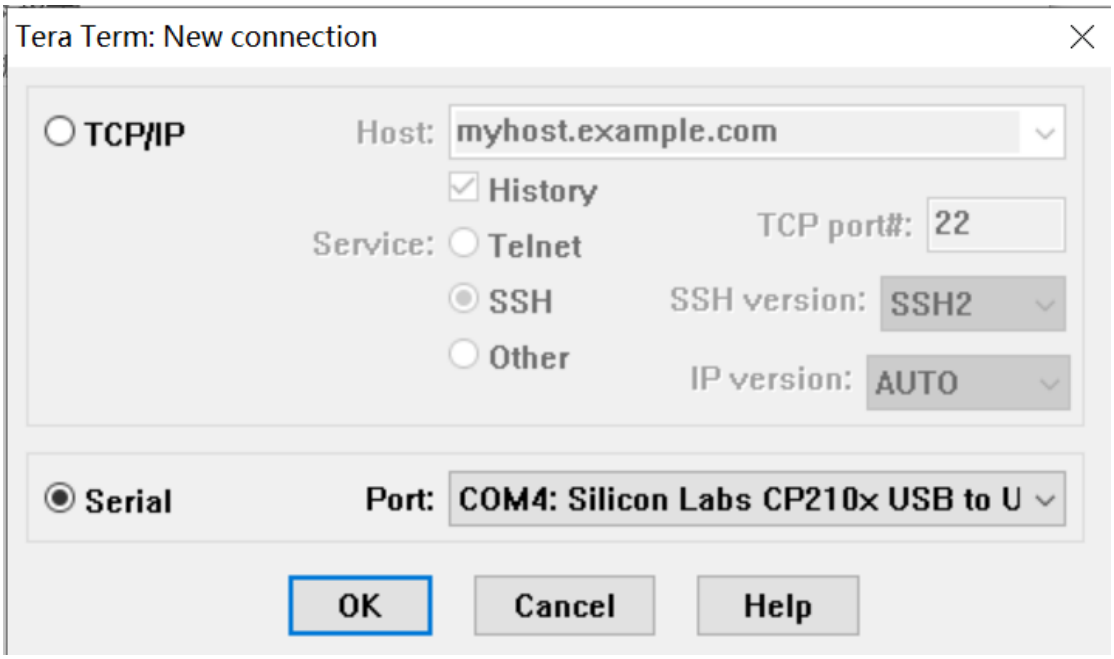
2. 从 https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers 下载 cp2102 的驱动，安装后就有串口显示：



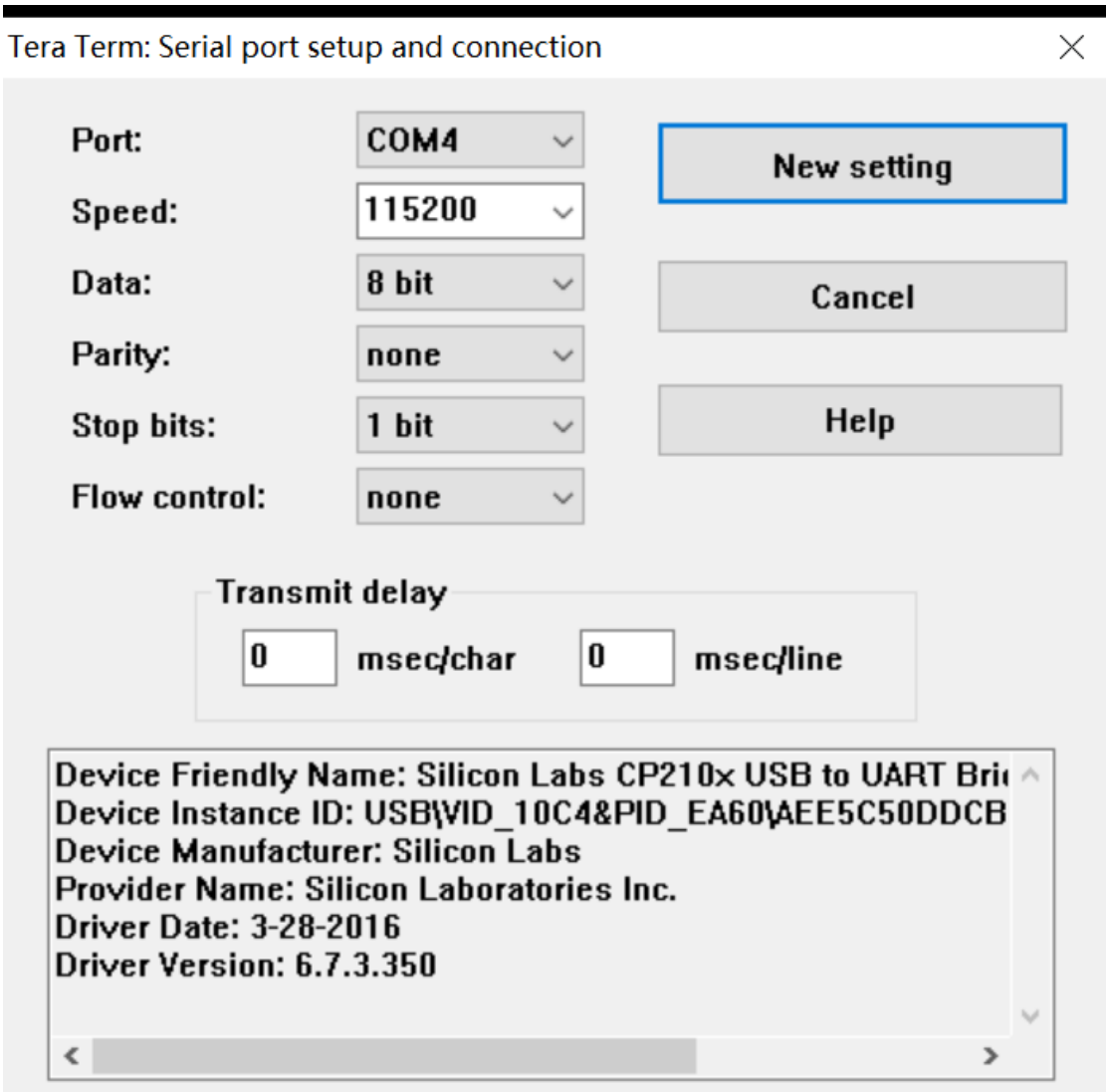
3. 配置 com 口属性，115200, 8-N-1：



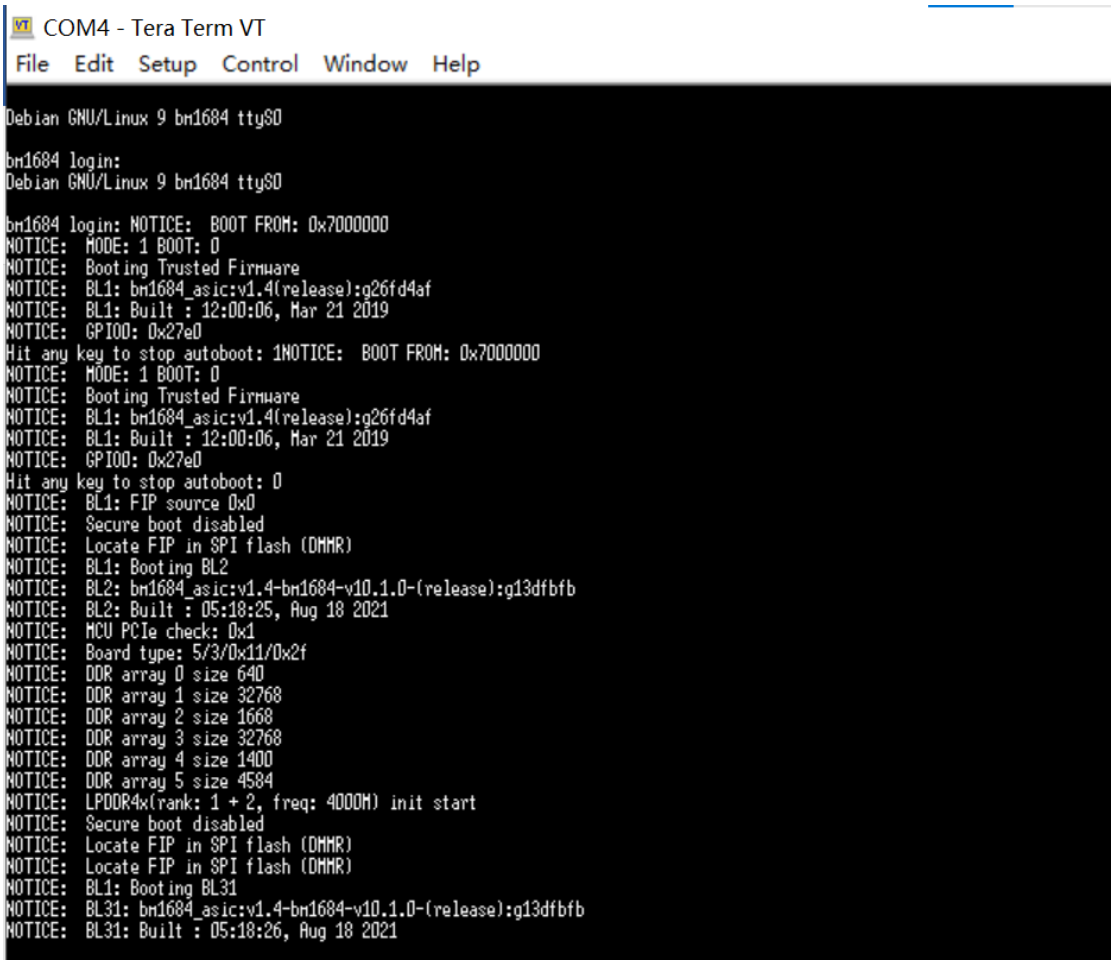
4. 下载 teraterm 软件，选择连接 comX(本例 com4)，点击 OK：



5. 在 Setup—Terminal，将 Speed 改为 115200：



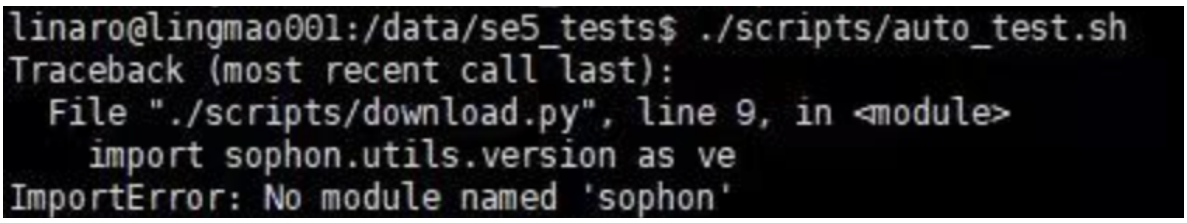
6. SE5 开机运行就会显示：



4.3.2.3 若系统故障，无法进入 SE5/SM5 的操作系统，该如何升级或恢复固件？

应当使用 SD 卡刷的方式重新刷机。

4.3.2.4 SE5 如何在 Python 中使用 SAIL？



答：SE5 已经预装在/system/lib 下，只需要设置好环境变量，然后就可以在 python 中使用 SAIL 了：

```
# SE5设置环境变量
export PATH=$PATH:/system/bin
```

（下页继续）

(续上页)

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/system/lib/:/system/usr/lib/aarch64-
↪linux-gnu
export PYTHONPATH=$PYTHONPATH:/system/lib
```

4.3.2.5 SE5 安装 openVPN？

答：只需要打开 CONFIG_TUN，直接修改文件 https://gitee.com/sophon-ai/bsp-sdk/blob/master/linux-linaro-stable/arch/arm64/configs/bitmain_bm1684_asic_defconfig，在最后面加一行 CONFIG_TUN=y，再按照 sm5 手册的第 6 章重新编译 kernel。不需要用 make menuoconfig 去找。

4.3.2.6 如何通过串口查询 SE5 的 IP 地址

答：1、宿主机为 Linux 时：使用 Ubuntu 连上 SE5 串口之后，在 Terminal 输入命令 ping bm1684.local 可以查看到盒子 IP。

2、宿主机为 Windows 时：通过串口工具 (推荐 teraterm)，需要安装驱动程序（直接在网上找 CP210x 驱动），安装好驱动之后，使用串口工具连接登录 linaro/linaro，然后使用 ip addr show 命令查看 IP。

4.3.2.7 SE5 经常自动重启，大约十几分钟到二十分钟左右重启一次？

答：

1. 盒子出厂自带库克人脸应用 gate，gate 文件系统，有个监控 sophongate 是否活着的服务，如果 10 分钟左右没活着就自己 reboot 系统了；使用/bm_bin 下 bm_switch2box 命令将盒子工作模式设置为 box 模式，关闭库克应用或重新刷机，将盒子重置为干净的系统状态；
2. 如果仍然有问题看下硬盘各分区的使用情况，之前遇到过有些分区快满的时候，就会重启。

4.3.2.8 SE5 是否可以刷 Ubuntu 系统？有没有版本要求？

答：客户没法自己随便安装系统，只能用我们发布的刷机包。目前我们使用 Debian 9.9，从 SDK2.7.0 开始，会有 official 的 ubuntu 版本同步发布。目前有一个为其他客户定制的 Ubuntu20.04，SDK2.6.0，可以下载使用：客户定制的 ubuntu 20.04 with SDK2.6.0

4.3.2.9 使用卡刷方式升级为 2.6.0 版本，刷完后原来的 web 页面不能登录？

答：这说明使用的卡刷包为干净的系统包；web 页面是库克系统 gate 版本才会有的，需要刷 gate 版本的固件才会有。目前 gate 版本的固件没有 SDK2.6.0 版本，需要刷最新的 gate 版本固件后，采用替换升级文件的方式手动升级 SDK 到 2.6.0。

4.3.2.10 SE5 盒子磁盘仍然有空间，但使用 apt 安装软件时或进行其他操作时，都会提示系统磁盘空间不足？

```
Filesystem      Size  Used Avail Use% Mounted on
overlay         5.8G  4.5G  1.2G  80% /
devtmpfs        1.8G     0  1.8G   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           1.9G  201M  1.7G  11% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mmcblk0p7   16G  9.0G  5.8G  62% /data
/dev/mmcblk0p5  5.8G  4.5G  1.2G  80% /media/root-rw
/dev/mmcblk0p4  1.6G  1.1G  428M  73% /media/root-ro
/dev/mmcblk0p6  2.0G  678M  1.2G  37% /system
/dev/mmcblk0p1  128M   37M   91M  29% /boot
/dev/mmcblk0p2  2.9G   91M  2.7G   4% /recovery
tmpfs           380M     0  380M   0% /run/user/1000
```

```
Err:1 http://mirrors.ustc.edu.cn/debian stretch/main arm64 htop arm64 2.0.2-1
  Could not open file /var/cache/apt/archives/partial/htop_2.0.2-1_arm64.deb - open (28: No space left on device) [IP:
202.141.160.110 80]
E: Failed to fetch http://mirrors.ustc.edu.cn/debian/pool/main/h/htop/htop_2.0.2-1_arm64.deb  Could not open file /var/
cache/apt/archives/partial/htop_2.0.2-1_arm64.deb - open (28: No space left on device) [IP: 202.141.160.110 80]
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?
```

答：通常这是由于 node 节点满了导致的，请使用 ‘df -i‘查看 Inodes 节点是否被用完，当磁盘中存在大量的小文件时，就容易发生这样的问题。可以使用扩大 inodes 节点数量或删除无用的小文件。

4.3.2.11 SE5 盒子如何修改 IP？执行 bm_set_ip 提示没有该命令？

答：

1. 使用我们提供的命令：/bm_bin/bm_set_ip 和/bm_bin/bm_set_ip_auto（注意：bm_set_ip 等 SE5 产品手册中描述的常用命令，2.6.0 及之前是预刷了我们带有库克人脸识别系统的 gate 版固件才有的。干净的系统固件内并没有这个命令。您可以预先备份好/bm_bin 下的这些文件，待重新刷机后拷贝到新系统中使用。2.7.0 开始干净的系统固件也将预置）；

2. 修改配置文件：对于盒子来说（debian 9 系统），修改/etc/network/interfaces.d 下的配置文件，eth0 对应 WAN 口，根据实际修改；eth1 对应 LAN 口，应保持 static 方式固定 IP 192.168.150.1 不变，方便系统维护时使用固定 IP 连接；
3. 使用 NetManager 的接口 **（注意：使用 nmcli 使能网口，必须确保/etc/network/interfaces.d/下没有相应网卡的配置文件，否则会使能失败）**：

```
# 修改IP
nmcli c add type ethernet con-name edgeboard_eth0 ifname eth0 ipv4.addresses
↪'192.168.1.254/24' ipv4.gateway 192.168.1.1 ipv4.method manual ipv4.route-
↪metric 601
nmcli c up edgeboard_eth0 ifname eth0
```

4.3.2.12 SE5 盒子如何扩展存储空间？

答：

1. 配有硬盘仓，支持 2.5 英寸 7mm 高度的 SATA 接口 HDD 或 SSD 硬盘，支持容量不超过 2TB；
2. SD 卡，SE5 背部维护仓内部有 SD 卡插槽，速率模式为 SDR104，最大容量 2TB；
3. USB 接口连接外接硬盘，可临时使用，长期不推荐。

4.3.2.13 如何实现扩展硬盘的开机自动挂载？

答：推荐使用启动脚本或者 systemd service 的方式来实现扩展硬盘的开机自动挂载；您可以修改/etc/rc.local 文件，将挂载命令放在文件的末尾 exit 0 之前。我们不建议直接修改/etc/fstab 文件，因为一旦挂载出现问题，系统将无法启动。为了安全起见，SE5 中/etc/fstab 文件不能随意修改，每次开机时只读分区的/media/root-ro/ fstab.emmc.ro 会自动替换/etc/fstab。（若您一定要修改 fstab 文件，请先 sudo mount -o remount rw /media/root-ro 将分区修改为可写；然后修改 fstab.emmc.ro 文件；修改完成后，请再将分区修改回只读属性。）（如果因挂载问题出现系统无法启动的问题，请尝试使用 SD 卡刷的方式修复。）

4.3.2.14 关于内核版本以及增加内核模块？

问：我拿到的这个盒子上 uname -r 拿到的版本和实际的内核模块版本不一致，另外我想加几个内核模块，盒子上源码目录下是空的，是不是需要你们帮忙重新编下内核

```
root@bit:/usr/src# uname -r
4.9.38-bm1684-v10.3.0-00528-g8be6792
```

```
root@bit:/usr/src# ls /lib/modules/
4.9.38-bm1684-v7.3.0-00469-g49e7e2dd
```

答：内核版本和/lib/modules 不一致，说明那个盒子应该是从某个比较老的版本，通过替换内核镜像的方式升级的，所以只有内核更新了，/lib/modules 没有更新。建议您用我们的卡刷

包重新刷一下，就一致了。如果您想添加内核模块的话，可以参考 https://gitee.com/sophon-ai/bsp-sdk，但是请了解如果您添加的 kernel config 对 kernel API 有比较大变化，会和我们以二进制形式发布的一些驱动不兼容。

4.3.2.15 SE5 盒子中使用卡刷后，使用 lsmod 查看内核模块，比之前多了 br_netfilter，它的作用是什么？和之前没有的盒子有什么区别？

```
baidu@bm1684:~$ lsmod
Module                  Size  Used by
jpu                    20480  0
vpu                    65536  0
bmtpu                2498560  0
br_netfilter           24576  0
fl2000                118784  0
```

答：安装使用 k3s/k8s 需要这个 driver，是在某个版本开始加入的。另外，如果没有这个 driver，说明盒子的固件比较老旧了，建议使用卡刷重刷一下固件，以免直接替换文件升级新版 SDK 后发现不可预期的问题。

4.3.2.16 k8s plugin 在哪里下载？

答：请联系我们的技术支持获取最新版本。

k8s plugin 1.0.8，适用于 k8s，version>=1.10

在官网资料下载可获得：https://developer.sophgo.com/site/index/material/11/74.html

4.3.2.17 SoC 中如何使用 OpenCV？导入 cv2，程序提示找不到 numpy 或 numpy 导入失败？

```
>>> import cv2
ImportError: numpy.core.multiarray failed to import
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: numpy.core.multiarray failed to import
>>>
linaro@bm1684:/data$ bm_version
```

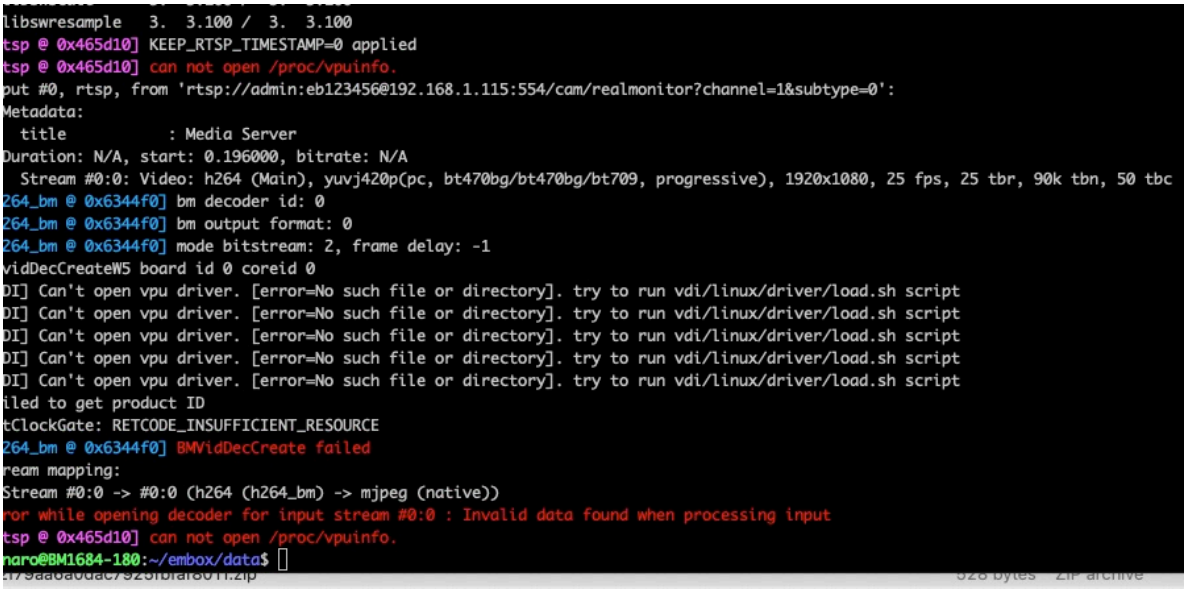
答：SoC 应当使用我们改造过的 BM-OpenCV 以获得硬件加速支持。设置环境变量即可：

```
# SE5设置环境变量
export PATH=$PATH:/system/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/system/lib/:/system/usr/lib/aarch64-
↪linux-gnu
export PYTHONPATH=$PYTHONPATH:/system/lib
```

但需要注意的是，目前 SoC 中默认没有安装 numpy 库，需要手动安装，请特别注意 numpy 的版本号，否则可能导致 cv2 导入时提示 numpy 导入失败：

```
# 安装numpy
# 注意：安装过程可能需要编译，比较耗时
sudo pip3 install numpy==1.17.2 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

4.3.2.18 使用 ffmpeg 命令时报错，找不到 vpu 驱动怎么办？



答：执行/system/data/下的 load.sh 这个脚本，查看 ko 是否可以正常加载：

1. lsmod 查看所有的 ko 是否加载，正常情况下 jpu、vpu、bmtpu、fl2000 等 4 个 ko 加载成功；
2. 对于没有加载的 ko，进入/system/data 目录下手动加载查看是否报错。若报错，原因可能是升级过内核，内核和 ko 不匹配，建议使用卡刷修复系统。

4.3.2.19 使用 hdmi 连接显示器，过一会儿不显示黑屏的问题

hdmi_status.sh

```
#!/bin/sh

count=0
service_state="init"
sleep 3

status=$(find /sys -name "hdmi_status")

path1="$status/status"
```

（下页继续）

(续上页)

```
while true
do
    if [ "$status" = "" ]; then
        status=$(find /sys -name "hdmi_status")

        path1="$status/status"

    fi
    STATUS=$(cat $path1)
    if [ $? -eq 1 ]; then
        STATUS="0"
    fi
    #echo $STATUS
    if [ $STATUS = "0" ]; then
        #delay stop service for avoid quickly hot-plug
        count=$(($count + 1))
        if [ $count -gt 60 ] && [ $service_state != "inactive" ]; then
            echo "stop hdmi service"
            systemctl stop SophonHDMI.service
            service_state="inactive"
            count=0
        fi
    else
        count=0
        if [ $service_state != "active" ]; then
            echo "start hdmi service"
            systemctl start SophonHDMI.service
            service_state="active"
        fi
    fi

    sleep 1
done
```

4.3.2.20 SoC 查看内存使用情况？

答：

1. 查看系统内存：

```
free -h
```

2. 查看 ION 内存

   NPU 内存使用情况：

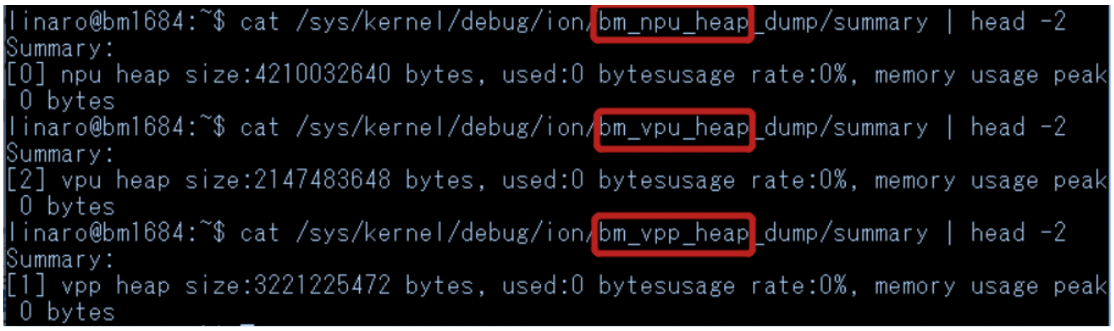
```
cat /sys/kernel/debug/ion/bm_npu_heap_dump/summary | head -2
```

   VPU 内存使用情况：

```
cat /sys/kernel/debug/ion/bm_vpu_heap_dump/summary | head -2
```

VPP 内存使用情况：

```
cat /sys/kernel/debug/ion/bm_vpp_heap_dump/summary | head -2
```



如上，通常会有 3 个 ION heap(即三块预留的内存区域)，如名字所示，分别供 TPU、VPU、VPP 使用。以上示例中只打印了每个 heap 使用信息的开头，如果完整地 cat summary 文件，可以到其中分配的每块 buffer 的地址和大小信息

4.3.2.21 SE5 盒子的配置方法

SE5 盒子有两种配置方式：

- SE5 直连电脑：
  1. 链接 SE5 WAN 口至电脑端 LAN 口
  2. 配置电脑 ip 地址至 192.168.150.1 同网段下
  3. ssh 至 192.168.150.1，初始账户名密码在机身下方

```
# 例如
ssh linaro@192.168.150.1
# 输入密码linaro
```

- SE5 通过路由器或交换机连接：
  1. 连接 SE5 端的 WAN 口至路由器或交换机
  2. 从路由器或交换机连接至电脑端 LAN 口
  3. 配置电脑 ip 至 192.168.150.1 同网段
  4. ssh 至 192.168.150.1，初始账户名密码在机身下方
  5. 查看 SE5 盒子的 WAN 口 ip 地址，以用来远程连接

4.3.3 BSP 问题

BSP 开源工程：https://gitee.com/sophon-ai/bsp-sdk

4.3.3.1 BSP SDK

所谓 SoC mode 是指 BM1684 片内的 CPU 运行嵌入式 Linux 系统，主要的业务软件运行在这个系统上，通过以太网与外部进行业务数据交互。与之相对的是 PCIe mode，即 BM1684 作为一张 PCIe 加速卡插入到主机上，主要的业务软件运行在主机上，只是把视频编解码、图像处理、神经网络推理等工作 offload 到 BM1684 上做硬件加速。BSP SDK 是 BM1684 SoC 模式产品最主要的软件开发资料：https://gitee.com/sophon-ai/bsp-sdk。使用方式请查看 README.md 文件。特别请留意里面提供了一个网盘链接，其中有一份《算丰 SM5 系列 AI 计算模组的 SOC 模式软件开发指南》（以下简称 SM5 指南），这份文档提供了很多基础的信息，在本文档中就不再赘述了。SM5 指南也可以从 官网链接 获取。本文档作为 SM5 指南的补充，适用于如下产品：

1. SM5 系列产品
2. 预装 box 版本软件（即没有人脸识别应用）的 SE5 产品（预装 gate 版本软件的 SE5 产品在很多地方会有不同，如无特别说明，通常不适用，不清楚的地方请咨询技术支持）
3. 客户采购 BM1684 芯片制作的 SoC 模式产品

4.3.3.2 SM5 使用时过热的情况

答：SM5 模组二次开发，需要设计完整的散热方案，包含散热器和风扇。单纯依靠模块自身时其不能正常工作，具体参考设计方案请参考如下说明

4.3.3.2.1 SM5MW（SM5）散热设计说明

答：

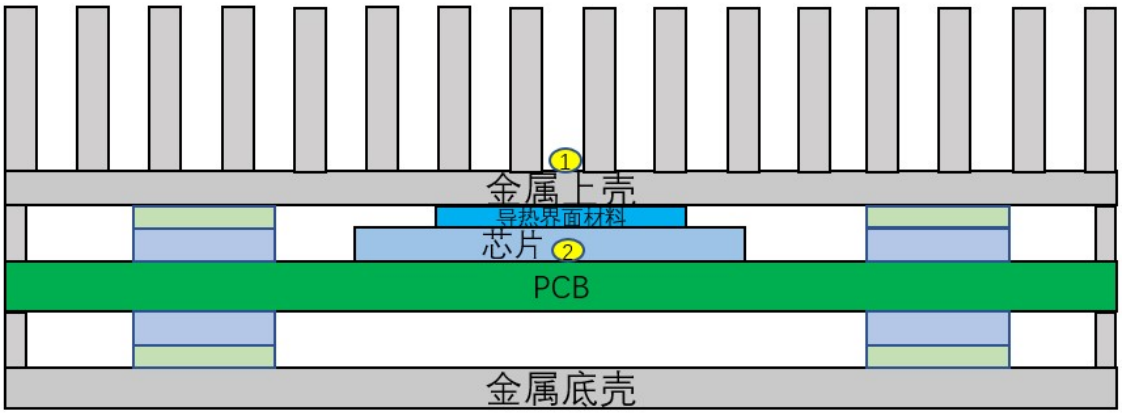


图 4.1: SM5M 结构刨面图

注意:

- 请勿拆除 SM5MW 的金属外壳，包括上壳和下壳
- 只要拆除一次散热部件，导热界面材料就不能重复使用
- 如 SM5M 结构刨面图中的 ① 号点代表 SM5MW 的壳温测试点，任何情况下使用 SM5MW，请确保该点测量温度值不超过 85℃(SM5M:75°)
- SM5M 结构刨面图中的 ② 号点代表 BM1684 结温，通过内置的温度传感器读取上报值。任何情况下使用 SM5MW，请确保该点上报温度值不超过 95℃(SM5M:85°) | SM5M 的热性能参数要求参考下表

表 4.2: SM5M 热阻

参数	说明
SM5M 热阻	工作模式：17.6TOPS
Rjc	0.5℃/W
Rjb	6.34℃/W

- 计算热阻：
  1. 为了确定 SM5M 需要的散热方案热阻 Rca，首先根据以下公式计算 Tc（即 SM5M 结构刨面图散热结构示意图中的 ① 号点温度）

     $$Tjmax = PowerxRjc + Tc$$

     说明：各参数取值如下

     a. Tjmax=105℃(SM5M:85°)（即 SM5M 结构刨面图散热结构示意图中的 ② 号点最大工作温度限值，长期工作温度建议采用 95℃(SM5M:85°)

     b. Rjc=0.5℃/W

     c. Power：SM5M 功耗需要通过底板测试得出
  2. 查看 Ta 的值。Ta=SM5M 正常工作时所要支持的最高环境温度。
  3. 根据以下公式，计算 SM5M 正常工作所需要的散热方案热阻

     $$Rca = （Tc - Ta）/Power$$

     说明：为了保证 SM5M 正常工作, 请用户确保设计的散热方案热阻在最恶劣情况下小于上述计算值

4.3.3.2.2 SM5W（SM5）散热设计说明

答:

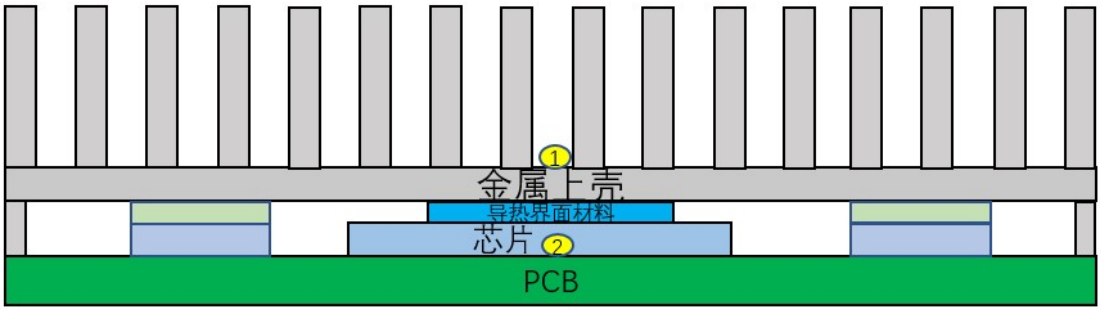


图 4.2: SM5W 结构刨面图

注意:

- 请勿拆除 SM5W 的金属外壳
- 只要拆除一次散热部件，导热材料就不能重复使用
- 如 SM5W 结构刨面图中的 ① 号点代表 SM5W 的壳温测试点，任何情况下使用 SM5W，请确保该点测量温度值不超过 85℃(SM5:75°)
- SM5W 结构刨面图中的 ② 号点代表 BM1684 结温，通过内置的温度传感器读取上报值。任何情况下使用 SM5W，请确保该点上报温度值不超过 95℃(SM5:85°) | SM5W 的热性能参数要求参考下表

表 4.3: SM5W 热阻

参数	说明
SM5W 热阻	工作模式：17.6TOPS
Rjc	0.5℃/W
Rjb	6.34℃/W

- 计算热阻:
  1. 为了确定 SM5W 需要的散热方案热阻 Rca，首先根据以下公式计算 Tc（即 SM5W 结构刨面图散热结构示意图中的 ① 号点温度）

     $$Tjmax = PowerxRjc + Tc$$

     说明：各参数取值如下

     a. Tjmax=105℃(SM5:90°)（即 SM5W 结构刨面图散热结构示意图中的 ② 号点最大工作温度限值，长期工作温度建议采用 95℃(SM5M:85°)

b. Rjc=0.5℃/W

c. Power：SM5W 功耗需要通过底板测试得出

2. 查看 Ta 的值。Ta=SM5W 正常工作时所要支持的最高环境温度。
3. 根据以下公式，计算 SM5W 正常工作所需要的散热方案热阻

$$Rca = (Tc - Ta)/Power$$

说明：为了保证 SM5W 正常工作, 请用户确保设计的散热方案热阻在最恶劣情况下小于上述计算值

4.3.3.2.3 SM5 模块散热参考设计

答:

1. 当 SM5 采用被动散热设计 (仅靠自带的散热器模块)，主机侧必须提供风扇来为模块散热，需满足的风量及风压设计要求如下表所述, 建议参数如下

表 4.4: SM5 对主机侧的风量散热要求（针对常温芯片）

入风口平均温度/℃	进风口需求最低风速/CFM	最小压降/inch H2O
70	16	0.21
60	8	0.18
50	4.2	0.09
40	3	0.07
35	2.2	0.06
<30	1.8	0.05

表 4.5: SM5 对主机侧的风量散热要求（针对宽温芯片）

入风口平均温度/℃	进风口需求最低风速/CFM	最小压降/inch H2O
70	10	0.17
60	5	0.13
50	2.8	0.08
40	2.1	0.05
35	1.6	0.04
<30	1.3	0.02

2. 当使用 SM5 作为纯被动散热组件使用时，自身带的散热器无法满足散热需求，需要客户重新设计散热器以满足散热需求，参考散热器尺寸如下

注解:

a. 以最高环境温度 70° 为例

b. 纯被动散热温度可能达到极限, 具有一定的风险, 条件允许, 建议采用主动散热设计

- 环境温度 70 度，需求的散热器最小尺寸约为：250(L)*200(W)*30(H) mm，其中齿高不能低于 25mm，齿间距不能小于 5mm
- 散热器与芯片接触的地方不能使用导热垫，推荐使用导热硅脂或液态金属导热材料
- 散热表面需要做增加辐射的表面处理 (比如阳极氧化，喷涂石墨碳等等)
- 散热器材质：除去普通的铝材散热材料外，建议与芯片接触的地方增加热管，VC 等具有较高导热能力的材质为芯片快速均匀散热

4.3.3.3 BSP 常见问题

4.3.3.3.1 日志文件太大怎么办

答：系统运行中的日志保存在/var/log 目录下，包括用户态的 syslog，内核态的 kern.log 等文件。系统默认有开启 logrotate 服务，触发周期为 1 天，当 logrotate 触发时，它会把当前的 syslog 等日志文件压缩，依次保存成 syslog.1、syslog.2.gz、syslog.3.gz 等等，序号最多到 7。所以按这个机制，如果您在一天内频繁打印了太多日志，就有可能使得日志文件过大，占满磁盘空间。如果确实有打印大量日志的需求，sudo logrotate -f /etc/logrotate.conf 可以强制触发一次 logrotate，或者可以按照下图将 logrotate 的策略从 daily 改成类似 “size 10M” 这样，就可以把 lograte 改成按照日志文件大小来触发：

```
linaro@bm1684:~$ cat /etc/logrotate.d/rsyslog
/var/log/syslog
{
        rotate 7
        daily
        missingok
        notifempty
        delaycompress
        compress
        postrotate
                invoke-rc.d rsyslog rotate > /dev/null
        endscript
}
```

但这里要注意的是，logroate 是通过/etc/cron.daily/logrotate 触发执行的，而只有当它执行时才会去检查上面这些规则文件，所以如果只在上面配置文件里修改成 hourly 或者 size 10M，并不能直接生效。需要首先 sudo mv /etc/cron.daily/logrotate /etc/cron.hourly，让 logrotate 每小时执行一次，它才有机会去检查/etc/logrotate.conf 文件，进而执行您修改后的更激进的设置。

4.3.3.3.2 如何控制看门狗

基于 BM1684 芯片的产品在板上都会有一颗 STM32 MCU，它的主要任务是给 BM1684 芯片上下电，然后顺便承担了其他一些功能，比如这里要介绍的看门狗。BM1684 和 STM32 之间有一条 I2C 总线连接，BM1684 做 master，STM32 做 slave，BM1684 通过发送 I2C 消息来做踢狗的动作。BM1684 在每个 CPU 核上绑定一个线程，只有当所有线程都活着时才会周期性踢狗，即任何一个 CPU 核挂死都会引起看门狗超时，STM32 会复位 BM1684。可以通过如下命令来控制这个看门狗：

echo ‘enable’ > /dev/bm-wdt-0	启用看门狗功能
echo ‘disable’ > /dev/bm-wdt-0	禁用看门狗功能
echo ‘auto’> /dev/bm-wdt-0	启动内核线程自动周期性踢狗的动作
echo ‘manual’ > /dev/bm-wdt-0	关闭内核线程自动周期性踢狗的动作
echo ‘kick’> /dev/bm-wdt-0	手动触发一次踢狗
echo ‘timeout 30’ > /dev/bm-wdt-0	设置看门狗超时时间，超过这个时间没有收到踢狗消息，看门狗就复位 BM1684
echo ‘interval 20’ > /dev/bm-wdt-0	设置内核线程自动踢狗的周期

4.3.3.3.3 windows 操作 SD 卡分区导致 SD 卡无法升级问题

答：

串口 log 上可以看到：

```
mmc_set_clock max 12000000, min 200000, set 50000000
switch to partitions #0, OK
mmc1 is current device
** No partition table - mmc 1 **
mmc_set_clock max 125000000, min 200000, set 1
```

还有一种情况是用 windows 分区不正确，串口 log 上会卡在 SD init 就没了：

```
NOTICE:  BL1: bm1684_asic:v1.4(release):g26fd4af
NOTICE:  BL1: Built : 12:00:06, Mar 21 2019
NOTICE:  GPIO0: 0x27e0
Hit any key to stop autoboot: 1NOTICE:  BOOT FROM: 0x7000000
NOTICE:  MODE: 1 BOOT: 0
NOTICE:  Booting Trusted Firmware
NOTICE:  BL1: bm1684_asic:v1.4(release):g26fd4af
NOTICE:  BL1: Built : 12:00:06, Mar 21 2019
NOTICE:  GPIO0: 0x27e0
Hit any key to stop autoboot: 0
NOTICE:  SD initializing 100000000Hz
```

这两种情况需要：

- Ubuntu 系统可以使用 disk 工具/fdisk 添加新卷，然后格式成 fat；
- Win10 需要用磁盘管理（也可用 Diskgenius 工具）删除卷再添加新卷，然后格式成 fat。

确保 SD 在 SE5/SM5 中使用 sudo fdisk -l 显示成：

```
Disk /dev/mmcblk1: 1.92 GiB, 2041577472 bytes, 3987456 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc8d245dd

Device         Boot Start     End Sectors  Size Id Type
/dev/mmcblk1p1       2048 3987455 3985408  1.9G  b W95 FAT32
```

从 v10.4.0 版本之后的 u-boot 已经可以兼容这种 SD 卡文件系统，无需进行特别处理。

4.3.3.3.4 不同规格的 BM1684 Soc mode 产品区别

答：为了适应不同算力需求的应用场景，BM1684 SoC mode 产品提供了如下几种规格：

	DDR 容量	CPU 频率	TPU 频率	VPU / JPU / VPP 频率
16 路产品	总计 12GB，其中 TPU：4GB VPU：2GB VPP：3GB	2.3GHz	550MHz (0.62V)	640MHz
8 路产品	总计 6GB，其中 TPU：1.2GB VPU：1GB VPP：1.7GB	2.3GHz	330MHz (0.55V)	400MHz (Video decoder/JPEG/VPP 只有一半硬件单元)
4 路产品	总计 4GB，其中 TPU：1GB VPU：0.5GB VPP：0.7GB	2.3GHz	330MHz (0.55V)	400MHz (Video decoder/JPEG/VPP 只有一半硬件单元)

4.3.3.3.5 ion_allocate_buffer failed 错误如何处理

答：如果程序运行时打印了类似这种 log：

```
kernel: [4] ion alloc (3133440,2,0) dma_buf_fd failed fd:-24
kernel: [4] ion_ioctl ion alloc failed, fd=-24, from yolov5s_demo
run.sh[8326]: [ion_malloc:386] ioctl ION_IOC_ALLOC failed. [error=Too many open files].
run.sh[8326]: [ion_allocate_buffer:132] ion_malloc failed!
run.sh[8326]: [VDI] fail to vdi_allocate_dma_memory size=3133440
run.sh[8326]: AllocateDecFrameBuffer:1081 fail to allocate frame buffer
run.sh[8326]: [VDI] invalid buffer to free address = 0x0
run.sh[8326]: InstIdx 3: BMVidDecSeqInitW5 failed Error code is 0xffffffff
```

可见关键字 “fd=-24”、“error=Too many open files”。这个是因为我们使用 ION 管理内存，而 ION 又依赖于 dma_buf，后者会对每个分出来的 buffer 分配一个 fd 进行管理。系统对同时处在打开状态的 fd 总数是有限制的。

如果有遇到这个问题，请执行 ulimits -n，检查 “open files” 那一行是否为 20480。如果不是的话，请再检查/etc/security/limits.conf 文件，是否有如下两行：

```
#*              soft    core            0
#root           hard    core            100000
#*              hard    rss             10000
#@student       hard    nproc           20
#@faculty       soft    nproc           20
#@faculty       hard    nproc           50
#ftp            hard    nproc           0
#ftp            -       chroot          /ftp
#@student       -       maxlogins       4

*               soft    nofile          20480
*               hard    nofile          20480
```

如果没有的话，请添加后重启，对在 shell 里敲命令启动的进程应该就会生效了。

如果上述方法没有解决问题，请先通过 ps 命令获取业务进程 ID，然后 cat /proc/$pid/limits，检查 “Max open files” 那一行是否为 20480，如果不是的话，请在直接启动您业务程序的 shell 脚本（比如上面 log 示例里的 run.sh）开头添加 “ulimit -n 20480”。重新运行业务，通过 cat /proc/$pid/limits 复核是否修改成功。

如果以上方法都检查了没问题，那可能您的确申请了太多 buf，可以通过上述方法继续增大 max open files 数量。但最大也不应超过 cat /proc/sys/fs/file-max 显示的数量。

4.3.3.3.6 如何修改 IP 地址

答：SM5 默认搭载的是 Debian 系统，双网口中的 eth0 默认是动态获取 IP 的（即 DHCP），eth1 是固定成了 192.168.150.1。后者的配置是通过/etc/network/interfaces.d/eth1 文件完成的，这个文件是我们对 Debian 原始网络配置唯一的修改。即：如果您删掉这个文件，那么 eth1 会变成跟 eth0 一样的 DHCP 获取 IP；如果您想要把 eth0 也固定 IP，就依样在/etc/network/interafces.d 文件夹下创建一个 eth0 文件。重启后生效。注意最好不要把两个网卡配置成同一网段，可能会有奇奇怪怪的问题。

如果 SM5 搭载的是 Ubuntu 系统，则配置文件在/etc/netplan/01-netcfg.yaml 文件中，修改完后可以通过 sudo netplan apply 命令生效。更多详细配置信息，可以查询 netplan 工具的使用，与 PC 版 Ubuntu 系统是一样的。

4.3.3.3.7 使用 K3S 遇到问题

答：使用 k3s，出现 k3s 管理的 pods 开机启动会 CrashLoopBackOff，使用 journalctl -u k3s 查看日志，发现这个错误：

```
kmod_search_moddep() could not open moddep file '/lib/modules/4.9.38-bm1684-v10.3.0-00528-
↪g8be6792/modules.dep.bin'
而盒子上是/lib/modules/4.9.38-bm1684-v7.3.0-00469-g49e7e2dd。
```

```
-- Logs begin at Tue 2022-04-19 16:02:14 CST, end at Tue 2022-04-19 17:44:02 CST. --
pr 19 16:02:18 BM1684-180 modprobe[1125]: modprobe: ERROR: ../libkmod/libkmod.c:586 kmod_search_moddep() could not open moddep file '/lib/modules/4.9.38-bm1684-v10.3.0-00528-g8be6792/modules.dep.bin'
pr 19 16:02:18 BM1684-180 modprobe[1125]: modprobe: FATAL: Module br_netfilter not found in directory /lib/modules/4.9.38-bm1684-v10.3.0-00528-g8be6792
pr 19 16:02:18 BM1684-180 modprobe[1183]: modprobe: ERROR: ../libkmod/libkmod.c:586 kmod_search_moddep() could not open moddep file '/lib/modules/4.9.38-bm1684-v10.3.0-00528-g8be6792/modules.dep.bin'
pr 19 16:02:18 BM1684-180 modprobe[1183]: modprobe: FATAL: Module overlay not found in directory /lib/modules/4.9.38-bm1684-v10.3.0-00528-g8be6792
pr 19 16:02:27 BM1684-180 k3s[1199]: time="2022-04-19T16:02:27.148969000+08:00" level=info msg="Starting k3s v1.18.10+k3s1 (6fa97306)"
pr 19 16:02:27 BM1684-180 k3s[1199]: time="2022-04-19T16:02:27.154891840+08:00" level=info msg="Cluster bootstrap already complete"
pr 19 16:02:27 BM1684-180 k3s[1199]: time="2022-04-19T16:02:27.365673380+08:00" level=info msg="Kine listening on unix://kine.sock"
```

此错误是由于使用部分升级的方式，升级了 kernel。但是文件系统没有升级导致的不匹配。需使用 sd 卡刷机的方式全升级。

4.3.3.3.8 SE5 上使用 QT 输出图形界面

答：SE5 上的 HDMI 输出并没有使用标准的 framebuffer 驱动，所以公版的 QT 并不能直接使用，请从 https://github.com/sophon-ai-algo/sophon-qt 获取我们修改过的 QT 版本进行开发，此版本的 QT 可以正确输出到 SE5 的 HDMI 上。

4.3.3.3.9 apt update 时出现 public key 无效提示

答：如果在 apt update 时遇到如下错误：

```
The following signatures couldn't be verified because the public key is not available: NO_
↪PUBKEY 6AF0E1940624A220
```

可以参考如下步骤解决：

1. 到 OpenPGP Keyserver 网站：https://keyserver.ubuntu.com/，在文本框里输入错误提示里 “NO_PUBKEY” 后面那串字符，要加上 “0x” 前缀，形如 “0x6AF0E1940624A220”，点击 Search Key 按钮，得到结果如下图

2. 点击 “pub” 后面那个链接，如上图红框部分，得到对应的 key，如下图：

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: Hockeypuck ~unreleased
Comment: Hostname:

xo0ESXTUHwEEAMtdNPmcgQcoPN3JcUcRrmdmlchJSmX6gj28OamOgE3Nxp3XgkDd
g/vLFPv6Tk8zIMxQnvuSpuGlYGp3x8atcKlQAlEHncAo27Vlio6pk8jG+qipDBKq
7X7FyXE6X9Peg/k7t7eXMLwH6ZJFN6IEmvPRTsiiQEd/dXRRuIRhPHirABEBAAHN
G0xhdW5jaHBhZCBQUEEgZm9yIFRlYWxhdHJpWMK2BBMBAgAgBQJJdNQfAhsDBgsJ
CAcDAgQVAggDBBYCAwECHgECF4AACgkQavDhlAYkoiC8mAQAmaxr4Kw/R2WZKde7
MfbTPy7O9YoL/NQeThYGwxX6ICVr0IZUj9nxFQ/vtmhZ59p53bpdR8jpPXjdDwjZ
IIlxTf72Fky6Ri3/zsC4YRD6idS4c4L50dTy74W6IabCt8GQLtJy5YASlEp5OGwR
NptRSFxVE59LuOPRo2kvLIAa0DfCXgQQEQgABgUCS+lLwgAKCRD8kg8z/8ZzxCI8
AP46hKf0zWyG5j7rd4hxKxMJnfKH5v2PgF7x+5TLuvJlKgD/UjT4GEEiymGk2NRC
H0t4AfbPLcvdkJJ7UymNrdx9yBzCXgQQEQgABgUCTHOBTQAKCRDr74MxVxlYMJtf
AQCHQgkq4O5hU4VVlXl+VjhIaY5UU8GjvIeYyc/gbMl6/wEAnrnPt7VJkCve4yer
pah45DRqN2n644JxeqHQjG+QfzjCwVwEEAECAAYFAkz4h5oACgkQz6EollykKXwD
9g/+K9ilT0iFTnIb+kAD/NK8rvi4XmTvW6Jxrgse27WbCVu+YGcPHzg//pt9ehI4
JufaOnNYrv/Lflvj5j3UlNrzqDeUUAMuDSz/NOgnERfoq5usuPtwD2/DzWGMQeZr
O5vdRnpj9/zyOe4mcrBIahzmfl8b6MVqCC8tXQ+kxawh0/pMDftFQTNu9sAjq3dR
Kh+LUwPI4/lCspDx6AmkDQcPlQidwqoB5rcxLWLwHGx5l+w8EU7rdwJ4XxYVrbU3
p7fV3oYlS2Q+Qlvj7ZIlxMbFA5LpSUXyvFqw70li0luZZsZrTXiKX/a92w/sBTqc
lJj9IWouB+Db/x3v7NqYH3DSxXDxbGGGEBKlRketKYE5nLlkV3rUNvcogtzHCp4l
SecTxw2v7BPDoy3PKYGwrcKCqiRsbqcEHJPm9CzRpW3rvPsod/Bi+cEI2lM968wx
hSh9hxJDL3fU9tmWBy/EVz/7/sXbvXm9yO3HupBwPTMABmINvoL9W0bq/VYbc8fY
AtdXBZcYTSqr0foDKu/zrIXQwd0+Fs2HgpzYCg/w2iKGpyQHx9PmyNelKQ8WtBpj
GsJaO3xqeO3wgJYuwLXXN0tQse0E+sU18xqP3La+7rF2VzXI2htGajgNo/uYRFAJ
cGXbY2fSoccFIqr51s5kFI5w6+3lvlpv4Jw9G4VJHXeGufDCwVwEEAECAAYFAk0w
VyMACgkQuW8jAK0Ry+6PBhAAwn/nvKe4OwLrbs5mMuFZlt76c4FWWEfn74zSl6NU
5Hzt69QtarLUFWEuaNpBIxFCFYHgZv52c7tWksYkWcnb0QvhYOw2EMdhvV86HpHE
Dmwn3oxifDp3s0+2LcPGuQiBPqRUR/T2d2LPnb/ohgufhCDENgWOcjt4GSNdDxZg
jUkkTxUZiYVsfuLmqwDmnwOXgQbmaMSZvnZtwXnHsGhwr7zDdaxhLcpeOCbQDq7o
HY7ncud+cfy6AtlXQoFOdsMavQH6lmCBN4IQyJN3qlrOfnjqayTdbARRTGejxk2V
/zdyREHTiJnd3WADmtCfme2+t93lT7YJRff0hLHVyfHvkWSvxgIYuX7qlh+c3UZY
P4qz8eQHBDIeUX3iNol7/XlHqIVEC79zESkJwA5cKn6GFotHCI9FYyeRePABk3gD
riNSNWI3dDIzoWkoJbnAzfHC0LZzcFeRkqrtMGXU5tmaQSYeY8HplmWFC0XxGUK7
dWqofQuGWDBjn6vXkA8lkKkJmIIlOhngbmHq3lUqFRjrfQbpIoLI3biVI0gO09sS
/bQvSyFfsn7N7EB6eVt5HUjI6OuX8Yymgu+iQkvFH9/NeGjKGmP/9YaQqnCZlpBZ
lQrA3uED9MY48dc6jyyVsKt/t9gJckw09MXzLy4bF8zgcF2BoydaYwly5VVXzcDb
/pk=
=UkNX
-----END PGP PUBLIC KEY BLOCK-----
```


3. 复制此页面里的所有文本，保存成一个文件放到板子上，如 pgp.key
4. 在板子上执行 sudo apt-key add pgp.key，正常应该会得到一个“OK”提示。再重新执行 sudo apt update 即可。

4.3.3.4 SM5 参考方案

SM5 作为一个半成品，需要客户进行二次开发，在 BSP 软件和外围设备上拥有很高的定制自由度，客户的需求也各有不同，所以在此我们提供一些参考方案，这些方案没有经过严格的产品化测试，客户可以酌情谨慎采用，集成到自己的方案中。请做好充足的测试后再进行实际产品的部署。

4.3.3.4.1 本地刷机方案

SM5 模组提供了 SD 卡和 tftp 两种刷机方式，详见 SM5 指南。客户可以选择两种方式来定制自己的刷机包：

1. 使用上面提到的 BSP SDK 来创建刷机包，这种拥有最大的自由度，但需要学习相关脚本的使用；
2. 将自己的程序提供给算能，由算能制作刷机包在模组出厂时刷好，这种比较适合大批量出货，不适合开发阶段频繁版本迭代；

当然客户也可以选择只刷算能的官方通用刷机包，然后再通过网络等方式把自己的业务应用部署上去。

在上述 SD 卡刷机包的基础上，这里再提供一种不依赖于 SD 卡的升级方案。这并不是一个完整的 OTA 方案，并不包含版本管理、分发等功能，仅仅只是提供拿到升级包后刷到 SM5 上这最后一公里的方案，客户仍然需要自行实现刷机包部署。也并不支持差分包的方式，主要只是为了方便把 SD 卡刷机包刷到 SM5 上。操作步骤如下：

1. 首先并不是所有的 SD 卡刷机包都支持这种方式升级，建议先与技术支持确认。主要是两个限制：a. 刷机包里包含的脚本要支持这种方式 b. 从 SM5 当在的版本，到刷机包里的版本，eMMC 上的分区布局没有发生过变化
2. 请把 SD 卡刷机包里的全部文件丢到 SM5 的/data 目录下，效果如下，建议最好能做一下 md5 校验，确保文件正确：

```
linaro@bm1684:~$ ls /data
BOOT                      data.1-of-2.gz        rootfs.5-of-18.gz
boot.1-of-2.gz            data.2-of-2.gz        rootfs.6-of-18.gz
boot.2-of-2.gz            ddr.bin               rootfs.7-of-18.gz
boot.cmd                  docker                rootfs.8-of-18.gz
boot.scr                  fip.bin               rootfs.9-of-18.gz
boot_emmc-boot.cmd        gpt.gz                rootfs_rw.1-of-2.gz
boot_emmc-boot.scr        lost+found            rootfs_rw.2-of-2.gz
boot_emmc-data.cmd        misc.1-of-2.gz        spi_flash.bin
boot_emmc-data.scr        misc.2-of-2.gz        system.1-of-15.gz
boot_emmc-gpt.cmd         partition32G_ro.xml   system.10-of-15.gz
boot_emmc-gpt.scr         recovery.1-of-2.gz    system.11-of-15.gz
boot_emmc-misc.cmd        recovery.2-of-2.gz    system.12-of-15.gz
boot_emmc-misc.scr        rootfs.1-of-18.gz     system.13-of-15.gz
boot_emmc-recovery.cmd    rootfs.10-of-18.gz    system.14-of-15.gz
boot_emmc-recovery.scr    rootfs.11-of-18.gz    system.15-of-15.gz
boot_emmc-rootfs.cmd      rootfs.12-of-18.gz    system.2-of-15.gz
boot_emmc-rootfs.scr      rootfs.13-of-18.gz    system.3-of-15.gz
boot_emmc-rootfs_rw.cmd   rootfs.14-of-18.gz    system.4-of-15.gz
boot_emmc-rootfs_rw.scr   rootfs.15-of-18.gz    system.5-of-15.gz
boot_emmc-system.cmd      rootfs.16-of-18.gz    system.6-of-15.gz
boot_emmc-system.scr      rootfs.17-of-18.gz    system.7-of-15.gz
boot_emmc.cmd             rootfs.18-of-18.gz    system.8-of-15.gz
boot_emmc.scr             rootfs.2-of-18.gz     system.9-of-15.gz
boot_spif.cmd             rootfs.3-of-18.gz     vendor_name
boot_spif.scr             rootfs.4-of-18.gz
```

3. 然后输入如下命令，请务必核对输入的字母，如果有问题了修复起来比较麻烦，最好写到一个脚本文件里来执行

```
sudo -i
echo -e "boot-recovery\n/DATA/" > /dev/mmcblk0p3
```

4. 然后读取一下看是否符合预期：

```
cat /dev/mmcblk0p3
预期要能看到：
boot-recovery
/DATA/
然后按ctrl+c结束
```

5. 最后重启系统，请注意不要直接拔电源，以免文件损坏：

```
sync
sudo reboot
```

6. 正常情况下，重启后应该可以从串口 log 看到开始刷机了，此过程中请务必不要断电（SE5 产品可以通过面板上的指示灯判断升级是否成功，请参考 SE5 产品手册；SM5 产品只能通过串口 log，或者等待几分钟后重新尝试 ssh 登入来确认）：

```
## Executing script at 310000000
writing 3.log
do_savelog 4096 bytes written in 11 ms (363.3 KiB/s)
reading rootfs.1-of-18.gz
6241270 bytes read in 158 ms (37.7 MiB/s)

Uncompressed size: 102760448 = 0x6200000


MMC write: dev # 0, block # 794624, count 200704 ... 200704 blocks written: OK

reading rootfs.2-of-18.gz
24688639 bytes read in 546 ms (43.1 MiB/s)

Uncompressed size: 102760448 = 0x6200000


MMC write: dev # 0, block # 995328, count 200704 ... 200704 blocks written: OK
```

7. 刷机完成后，SM5 会自动重启，待重新进入系统后可以通过 bm_version 命令查看版本号，确认升级是否成功。请注意/data 分区并不会被清除，里面的文件都会保存下来。

如果过程中遇到问题，请参考 SM5 指南中关于 recovery mode 的介绍，通过串口终端进入 recovery mode 的命令行后，可以通过如下命令清除升级标记

```
mdev -s
dd if=/dev/zero of=/dev/mmcblk0p3 bs=512 count=1
sync
reboot -f
```

如果升级程序还没有清除 eMMC 上的数据，上述方式应该可以使您进入原来的系统。如果 eMMC 上的数据已经被破坏，甚至连 recovery mode 都进不去了，那么只能再通过 SD 卡刷机来恢复了。

4.3.3.4.2 网络刷机方案

SM5 指南上已经介绍了 tftp 刷机，但需要手工敲命令的方式，这里再提供一个便于自动化的方案：SM5 的内核启动方式是 u-boot 去加载/boot 目录下的 boot.scr.emmc 脚本，解析里面的命令，找到内核去启动。所以我们可以通过把 tftp 升级用的命令替换进这个脚本，让 SM5 一开机就自动进入 tftp 升级。只有当 tftp 升级成功完成时，才会把这个脚本刷回正常的启动脚本；如果升级失败，重启 SM5 就可以再次进行 tftp 升级。脚本内容如下：

```
echo "set serverip 10.0.0.3; dhcp 0x310000000 \$serverip:boot.scr; set reset_after 1; source␣
↪0x310000000" > boot.txt
# 这里serverip要设置成tftp server的IP，SM5需要与这个server在同一个网关下
mkimage -A arm64 -O linux -T script -C none -a 0 -e 0 -n "Distro Boot Script" -d boot.txt boot.scr.
↪emmc
sudo cp ./boot.scr.emmc /boot
sudo reboot
```

4.3.3.4.3 图形界面

SM5 可以通过如下几种方式得到图形界面，有时可以用来方便在板上做一些开发工作或者快速原型验证：

1. 通过 PCIe 连接 SM750/768 芯片的显卡
2. 通过 USB 连接 FL2000 芯片的 HDMI dongle
3. 不连接外部显示设备，在另一台主机上通过 VNC 登录

前两种连接外部显示设备的方案，可以通过如下方式来使能图形界面（建议使用 xfce4，Debian 9 仓库里的 lxde 可能会遇到拖拽窗口时死机的问题）：

```
sudo apt update
sudo apt install xfce4
sudo reboot
待重启后应该就可以看到图形界面了，通过USB键鼠即可操作，
需要再通过如下命令关闭休眠机制，以免遇到唤醒问题
xset s off -dpms
```

两种方式相比，更推荐 SM750/768 的方案，CPU 占用率低。FL2000 方案会至少用掉一个 CPU core 来刷新图像。最后一种 VNC 的方式通过如下步骤来使能：

```
sudo apt update
sudo apt install xserver-xorg-video-dummy x11vnc xfce4
sudo cp xorg.conf /etc/X11/
sudo systemctl disable gfx-feeder
sudo reboot
sudo x11vnc -display :0 -auth /var/lib/lightdm/.Xauthority &
然后就可以在另一台主机上用$sm5_ip:0地址来VNC了
```

上述用到的 xorg.conf 文件如下：

```
Section "Device"
    Identifier  "Configured Video Device"
    Driver      "dummy"
    VideoRam 256000
EndSection

Section "Monitor"
    Identifier  "Configured Monitor"
    HorizSync 5.0 - 1000.0
    VertRefresh 5.0 - 200.0
    ModeLine "1920x1080" 148.50 1920 2448 2492 2640 1080 1084 1089 1125 +Hsync +Vsync
#   Modeline "1280x800" 24.15 1280 1312 1400 1432 800 819 822 841
EndSection

Section "Screen"
    Identifier  "Default Screen"
    Monitor     "Configured Monitor"
    Device      "Configured Video Device"
```

（下页继续）

(续上页)

```
    DefaultDepth 24
    SubSection "Display"
    Depth 24
    Modes "1920x1080"
#     Modes "1280x800"
    EndSubSection
EndSection
```

4.3.3.4.4 用 Perfetto 工具分析性能

从 2.3.1 版本开始，SM5 预装了 Perfetto 工具，可以通过如下步骤使用：

1. 写一个 Perfetto 的配置文件，放到 SM5 上。配置文件描述了要抓取什么样的系统信息。可以参考下面的 scheduling.cfg 文件写法，这个配置文件会抓取系统进程调度的信息：

```
# One buffer allocated within the central tracing binary for the entire trace,
# shared by the two data sources below.
buffers {
size_kb: 20480
fill_policy: DISCARD
}

# Ftrace data from the kernel, mainly the process scheduling events.
data_sources {
config {
    name: "linux.ftrace"
    target_buffer: 0
    ftrace_config {
    ftrace_events: "sched_switch"
    ftrace_events: "sched_waking"
    ftrace_events: "sched_wakeup_new"

    ftrace_events: "task_newtask"
    ftrace_events: "task_rename"

    ftrace_events: "sched_process_exec"
    ftrace_events: "sched_process_exit"
    ftrace_events: "sched_process_fork"
    ftrace_events: "sched_process_free"
    ftrace_events: "sched_process_hang"
    ftrace_events: "sched_process_wait"
    }
}
}

# Resolve process commandlines and parent/child relationships, to better
# interpret the ftrace events, which are in terms of pids.
data_sources {
config {
```

(下页继续)

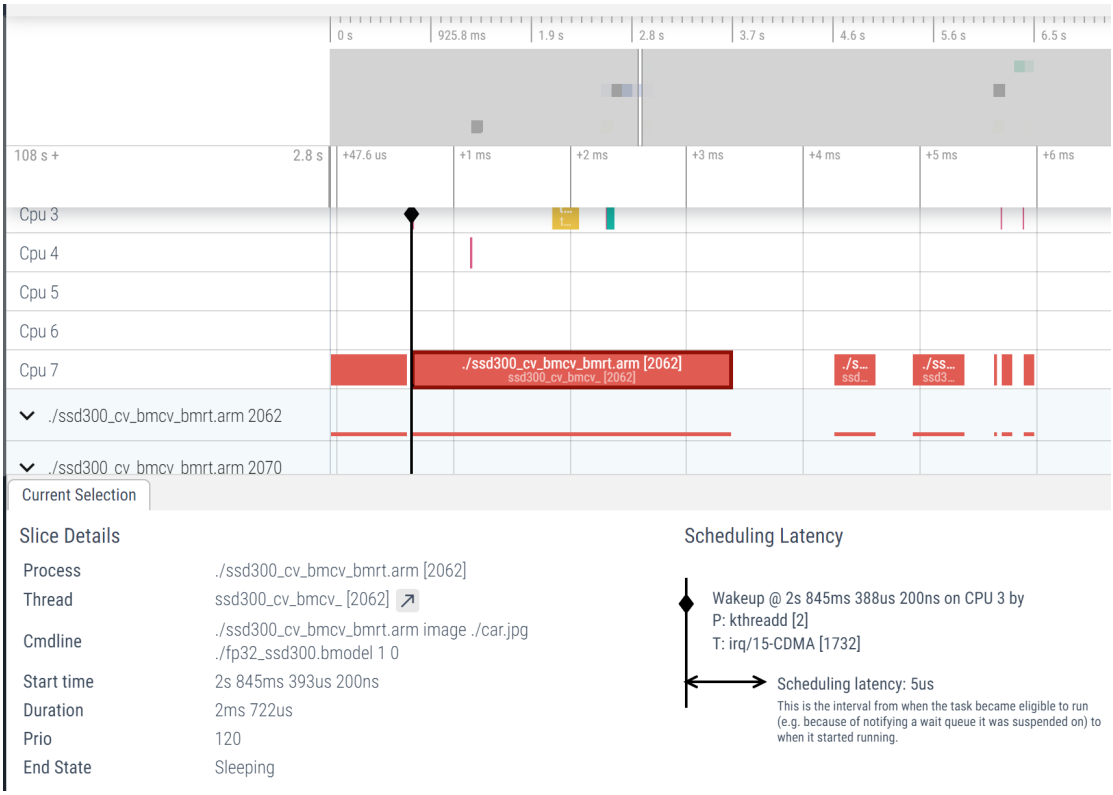
(续上页)

```
    name: "linux.process_stats"
    target_buffer: 0
}
}

# 10s trace, but can be stopped prematurely.
duration_ms: 10000
```

具体信息可以参考 Perfetto 的官方文档，https://perfetto.dev/docs/data-sources/cpu-scheduling，也可以在这里找到更多的配置文件参考：https://github.com/google/perfetto/tree/master/test/configs。

2. 在 SM5 上执行 sudo perfetto –txt -c scheduling.cfg -o scheduling.pftrace 即开始抓取，当 config 文件中定义的 duration_ms 时间到后会自动结束，或 ctrl+c 杀掉 perfetto 进程也可以结束抓取。结束后会生成一个 scheduling.pftrace 文件。打开 https://ui.perfetto.dev，点左边的 Open trace file，选择上面生成的 scheduling.pftrace 文件即可。w 键放大，s 键缩小，a 键左平移，d 键右平移，点击进程或 cpu 的色块可以在下面看到具体的描述：



如果想在应用程序里打点，跟上面的调度信息显示在同一个视图里，可以参考下面的写法，在你想要标记的段落开头调用 atrace_begin_body(“foo_bar”)，结尾调用 atrace_end_body。在配置文件中，增加一行 atrace_categories: “*”，比如加在 scheduling.cfg 中 ftrace_config 段落。

```
#include <stdio.h>
```

(下页继续)

(续上页)

```
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <math.h>

#define ATRACE_MESSAGE_LENGTH 1024

int atrace_marker_fd = -1;

#define WRITE_MSG(format_begin, format_end, name, value) { \
    char buf[ATRACE_MESSAGE_LENGTH];     \
    int pid = getpid(); \
    int len = snprintf(buf, sizeof(buf), format_begin "%s" format_end, pid, \
        name, value); \
    if (len >= (int) sizeof(buf)) { \
        /* Given the sizeof(buf), and all of the current format buffers, \
        * it is impossible for name_len to be < 0 if len >= sizeof(buf). */ \
        int name_len = strlen(name) - (len - sizeof(buf)) - 1; \
        /* Truncate the name to make the message fit. */ \
        printf("Truncated name in %s: %s\n", __FUNCTION__, name); \
        len = snprintf(buf, sizeof(buf), format_begin "%.*s" format_end, pid, \
            name_len, name, value); \
    } \
    write(atrace_marker_fd, buf, len); \
}
void atrace_begin_body(const char* name)
{
    WRITE_MSG("B|%d|", "%s", name, "");
}

void atrace_end_body()
{
    WRITE_MSG("E|%d", "%s", "", "");
}

static void atrace_init_once()
{
    atrace_marker_fd = open("/sys/kernel/tracing/trace_marker", O_WRONLY | O_
↪CLOEXEC);
    if (atrace_marker_fd < 0) {
        // try debugfs
        atrace_marker_fd = open("/sys/kernel/debug/tracing/trace_marker", O_WRONLY␣
↪| O_CLOEXEC);
        if (atrace_marker_fd < 0) {
            printf("trace_marker file not found\n");
        }
    }
}
```

(下页继续)

(续上页)

```
int main(int argc, char const *argv[])
{
    int i;

    atrace_init_once();
    if (atrace_marker_fd < 0)
        return atrace_marker_fd;

    for (i = 0; i < 10; i++) {
        atrace_begin_body("user_foo_bar");
        usleep(500 * 1000);
        atrace_end_body();
        usleep(500 * 1000);
    }
    close(atrace_marker_fd);
    return 0;
}
```

用 perfetto 抓取 trace 后，就可以找到你的 tag 叠加到时间轴上显示了，如下面 user_foo_bar 标记：

如果想在内核态驱动里打点，可以参考如下的做法：

```
#include <linux/trace_events.h>
在想要标记的段落开头调用atrace_begin_body("foo_bar")，结尾调用atrace_end_body
```

在配置文件中，增加一行 atrace_categories: “*”，比如加在 scheduling.cfg 中 ftrace_config 段落。用 perfetto 抓取 trace 后，效果跟上面用户态的一样。

4.3.3.4.5 提高业务程序的实时性

答：如果您的业务对实时性比较敏感，您可以考虑使用如下几种优化的方式：

1. Linux 内核使用虚拟内存，对用户空间的内存（包括栈、代码段、数据段以及使用函数 malloc 或 mmap 动态分配的内存）使用惰性分配的策略，如果实时进程访问的虚拟页没有映射到物理页，那么会触发页错误异常，影响实时性。所以需要尽量避免频繁申请释放内存，malloc 出来的内存可以先做一次 memset 再送进实时性敏感的流程

2. 对于有明显周期性的实时性敏感业务，可以考虑使用 dealine 调度器：

```
struct sched_attr attr;

 memset(&attr, 0, sizeof(attr));
 attr.size = sizeof(attr);
 attr.sched_policy = SCHED_DEADLINE;//设置调度类型
 attr.sched_runtime = 700000000;//设置runtime，纳秒
 attr.sched_deadline = attr.sched_period = 2000000000;//设置deadline和period，纳秒
 ret = sched_setattr(0, &attr, flags);
 if (ret < 0) {
     perror("sched_setattr failed");
     exit(-1);
 }
```

3. 打开内核抢占，在 linux-linaro-stable/arch/arm64/configs/bitmain_bm1684_asic_defconfig 文件末尾新增：

```
  # CONFIG_PREEMPT_NONE is not set
  # CONFIG_PREEMPT_VOLUNTARY is not set
  CONFIG_PREEMPT=y

重新编译内核并替换到板子上重启后，输入uname -a,应该显示：
```

```
Linux bm1684 4.9.38-bm1684-v10.4.0-00550-g4ad0f96b7016-dirty #4 SMP PREEMPT Thu␣
↪Mar 10 08:56:25 CST 2022 aarch64 GNU/Linux
（默认非抢占内核显示：Linux bm1684 4.9.38-bm1684-v10.4.0-00550-g4ad0f96b7016-dirty #5␣
↪SMP Thu Mar 10 09:44:04 CST 2022 aarch64 GNU/Linux）
```

4. 将某几个 CPU 核从系统隔离出来，然后将实时性敏感人物绑定到这些核上，可以尽量避免这些任务被系统的其它事物干扰。以 CPU5 和 6 为例，大致有如下步骤：

   1) 修改 u-boot 的内核启动参数，增加 “isolcpus=5,6” 字段，这样除了最基础的一些系统任务外，调度器不会主动把任务分配到这两个 CPU 上。可以通过如下命令确认

   ```
   cat /sys/devices/system/cpu/isolated
   ```

   2) 修改 irqbalance 的规则文件（/etcdefault/irqbalance）：

   ```
   IRQBALANCE_BANNED_CPUS="E0"␣
   ↪（原值为80，因为我们默认将PCIe的中断单独绑定到了CPU7上）
   ```

重启后生效，可以通过如下命令观察效果：

```
cat /proc/interrupts
```

3) 通过 taskset -cp 5,6 $pid 命令将制定的进程绑定到这两个 CPU 上，设置完后要等到下次这个进程被调度执行时才会生效。如果是多线程，每个线程的 ID 要单独设置（可以通过 ps -T 查看）。

```
ps -o pid,spid,psr,stat,comm -A 可以在PSR列显示进程最近一次执行时所在的CPU
```

4.3.3.4.6 4G 模块方案

我们适配的 4G 模块是广和通 NL668。您可以直接购买带 4G 模组的设备，也可以自行单独购买。

安装指南：

软件配置:

1. 检查 4G 模块服务是否存在

```
cat /etc/systemd/system/lteModemManager.service
```

```
ExecStart=/usr/bin/python3 -u /bm_services/lteModem.py
ExecReload=/bin/kill -HUP $MAINPID
WorkingDirectory=/bm_services/
```

2. 修改 “WorkingDirectory=” 后面路径下的 cfg.ini 文件：修改为 Ittefirst = true
3. 打开 4G 拨号服务：

```
sudo systemctl enable lteModemManager.service
sudo systemctl restart lteModemManager.service
```

4. 拨号成功后 sudo ifconfig 会出现 ppp0 的网卡并带有 IP
5. route -n 会看到：

```
linaro@bm1684:/etc/systemd/system$ sudo route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         *.*.*.*     0.0.0.0         UG    0      0        0  ppp0
```

由于现在自动拨号服务支持中国电信，联调，移动的 SIM 卡。但是可能没有测试完全，有可能会有没有覆盖的 sim 卡类型。如果出现需要修改 lteModem.py 以支持

4.3.3.4.7 如何增加板类型

答：客户使用我们发布的 BSP SDK 开发，新增板类型时可以参考本节操作：

1. 客户修改控制单板 PCB 版本和 BOM 版本的硬件电阻。MCU 代码会采集电压值确定硬件的版本，并将该值写入到指定地址，让 BM1684 可以获取
2. 由于当前 arm-trusted-firmware 部分代码未开源，需要联系我们提供适配新的 PCB 版本和 BOM 版本的 bl2.bin。获取后替换 bsp-sdk/install/soc_bm1684_asic/prebuilt/bl2.bin
3. 修改 BSP SDK 中的 u-boot 目录下代码，在 bsp-sdk/u-boot/include/configs/bm1684-asic.h 中增加新的板类型，然后修改 bsp-sdk/u-boot/board/bitmain/bm1684/vexpress64.c 中的 pcb_info 数组和 dtb_mapping 数组，增加该 pcb 对应的板类型及该板类型在 u-boot 中使用的 dtb，如下图所示

```
yang@yang-MS-7A59:~/gitee/bsp-sdk$ git diff u-boot/board/bitmain/bm1684/vexpress64.c
diff --git a/u-boot/board/bitmain/bm1684/vexpress64.c b/u-boot/board/bitmain/bm1684/vexpress64.c
index 26336189b..af1ae8188 100644
--- a/u-boot/board/bitmain/bm1684/vexpress64.c
+++ b/u-boot/board/bitmain/bm1684/vexpress64.c
@@ -240,6 +240,7 @@ const struct {
        {0x17,  SE5L_BOARD_LP1},
        {0x18,  SE5L_BOARD_LP2},
        {0x19,  SE5L_BOARD_LP3},
+       {0x1c,  SA5_BOARD_DRANK},
 };

 static int setup_pcb(void)
@@ -413,6 +414,7 @@ struct dtb_mapping {
        {SE5W_BOARD_LP,         6},
        {SA6_BOARD,             7},
        {SC5PRO_BOARD,          7},
+       {SA5_BOARD_DRANK,   1},
        {-1, -1},
 };

yang@yang-MS-7A59:~/gitee/bsp-sdk$ git diff u-boot/include/configs/bm1684-asic.h
diff --git a/u-boot/include/configs/bm1684-asic.h b/u-boot/include/configs/bm1684-asic.h
index c4a6308da..e4d868869 100644
--- a/u-boot/include/configs/bm1684-asic.h
+++ b/u-boot/include/configs/bm1684-asic.h
@@ -72,6 +72,7 @@ enum {
        SE5L_BOARD_LP1, // 1+1+4+4=6GB, DDR0 interleave, SE5 lite, full function, passive
        SE5L_BOARD_LP2, // 1+1+1+1=4GB, DDR0 interleave, SE5 lite, full function, passive
        SE5L_BOARD_LP3, // 1+1+4+4=6GB, DDR0 interleave, SE5 lite, full function, active
+       SA5_BOARD_DRANK = 33, // 16GB

        /*
         * the actual index is a 16 bits unsigned integer, higher 16bits are
yang@yang-MS-7A59:~/gitee/bsp-sdk$
```

4. 修改 BSP SDK 中的 linux-linaro-stable 目录下代码，新增对应的 dts，并修改 makefile。以增加 SA5 16G DDR 板类型为例，在 bsp-sdk/linux-linaro-stable/arch/arm64/boot/dts/bitmain 目录下新增 bm1684_asic_hds_drank.dts 文件，内容为相应的配置

```
#include "bm1684.dtsi"
#include "bm1684_mm_16g.dtsi"
#include "bm1684_asic_hds.dtsi"
```

然后在 bsp-sdk/linux-linaro-stable/arch/arm64/boot/dts/bitmain/Makefile 目录新增 bm1684_asic_hds_drank.dts 的编译选项，如下图所示

```
yang@yang-MS-7A59:~/gitee/bsp-sdk$ git diff linux-linaro-stable/arch/arm64/boot/dts/bitmain/Makefile
diff --git a/linux-linaro-stable/arch/arm64/boot/dts/bitmain/Makefile b/linux-linaro-stable/arch/arm64/boot/dts/bitmain/
Makefile
index 97dc7a1eb..3046844c7 100644
--- a/linux-linaro-stable/arch/arm64/boot/dts/bitmain/Makefile
+++ b/linux-linaro-stable/arch/arm64/boot/dts/bitmain/Makefile
@@ -22,7 +22,8 @@ dtb-$(CONFIG_ARCH_BM1684_ASIC) += bm1684_asic.dtb bm1684_asic_hds.dtb bm1684_asi
                                bm1684_asic_boxm_lite_lp2.dtb \
                                bm1684_asic_boxm_lite_lp3.dtb \
                                bm1684_asic_mix_mode.dtb \
-                               bm1684_asic_boxm_mm2.dtb
+                               bm1684_asic_boxm_mm2.dtb \
+                               bm1684_asic_hds_drank.dtb


 always          := $(dtb-y)
yang@yang-MS-7A59:~/gitee/bsp-sdk$
```

5. 修改 BSP SDK 中的 bsp-sdk/ramdisk/build/bm1684_asic/workspace/multi.its.base，新增 dts 的 fdt 和 config 描述。如下图所示，在红色框上面，绿色部分为新增代码。

```
diff --git a/ramdisk/build/bm1684_asic/workspace/multi.its.base b/ramdisk/build/bm1684_asic/workspace/multi.its.base
index 43871459c..e85352894 100644
--- a/ramdisk/build/bm1684_asic/workspace/multi.its.base
+++ b/ramdisk/build/bm1684_asic/workspace/multi.its.base
@@ -312,6 +312,17 @@
                        };
                };

+               fdt@1c {
+                       description = "for SA5 2+2 rank, 16GB";
+                       data = /incbin/("./bm1684_asic_hds_drank.dtb");
+                       type = "flat_dt";
+                       arch = "arm64";
+                       compression = "none";
+                       hash@1 {
+                               algo = "sha1";
+                       };
+               };
+
                fdt-pcb7-mm0 {
                        description = "custom for SM5 1+2 rank, larger memory for kernel";
                        data = /incbin/("./bm1684_asic_modm_mm0.dtb");
@@ -671,6 +682,18 @@
                        };
                };

+               config@1c {
+                       description = "for SA5 2+2 rank, 16GB";
+                       kernel = "kernel@1";
+                       ramdisk = "ramdisk@1";
+                       fdt = "fdt@1c";
+                       signature@1 {
+                               algo = "sha1,rsa2048";
+                               key-name-hint = "vboot";
+                               sign-images = "fdt", "ramdisk", "kernel";
+                       };
+               };
+
                config-pcb7-mm0 {
                        description = "custom for SM5 1+2 rank, larger memory for kernel";
                        kernel = "kernel@1";
~
~
yang@yang-MS-7A59:~/gitee/bsp-sdk$
```

4.3.3.4.8 定制化 rootfs

答:

1. x86 环境安装 qemu-user-static

```
sudo apt-get install qemu-user-static
```

2. 准备 arm linux 根文件系统 arm linux 根文件系统可以直接下载干净的版本（从我们 gitee 获取）或者从运行的 arm 单板上进行备份获取。arm 单板上备份根文件系统命令:

```
sudo tar -cvpzf rootfs.tgz --exclude=/proc --exclude=/mnt --exclude=/sys --exclude=/rootfs.
↪tgz /
```

解压根文件系统命令:

```
tar -xvpfz rootfs.tgz -C ./rootfs
```

3. 构建虚拟机

创建 rootfs 目录，在 rootfs 下执行根文件系统解压命令，解压完成后，copy qemu 执行命令到 arm 文件系统中

```
sudo cp /usr/bin/qemu-arm-static usr/bin/ sudo cp /usr/bin/qemu-aarch64-static usr/bin/
```

在 rootfs 录下创建 proc、sys、host 目录在 rootfs 所在目录下创建 ch-mount.sh 文件

```
#!/bin/bash

function mnt() {
    echo "MOUNTING"
    sudo mount -t proc /proc ${2}proc
    sudo mount -t sysfs /sys ${2}sys
    sudo mount -o bind /dev ${2}dev
    sudo mount -o bind /run ${2}run
    sudo mount --bind / ${2}host
    #sudo mount -vt tmpfs shm ${2}dev/shm
    #sudo mount -t /dev/shm ${2}dev/shm
    sudo chroot ${2}
}

function umnt() {
    echo "UNMOUNTING"
    sudo umount ${2}proc
    sudo umount ${2}sys
    #sudo umount ${2}dev/shm
    sudo umount ${2}dev
    sudo umount ${2}run
    sudo umount ${2}host
}


if [ "$1" == "-m" ] && [ -n "$2" ] ;
then
    mnt $1 $2
elif [ "$1" == "-u" ] && [ -n "$2" ];
then
    umnt $1 $2
else
    echo ""
    echo "Either 1'st, 2'nd or both parameters were missing"
    echo ""
    echo "1'st parameter can be one of these: -m(mount) OR -u(umount)"
    echo "2'nd parameter is the full path of rootfs directory(with trailing '/')"
    echo ""
    echo "For example: ch-mount -m /media/sdcard/"
    echo ""
    echo 1st parameter : ${1}
    echo 2nd parameter : ${2}
```

执行 ch-mount.sh，创建虚拟机

```
sudo ./ch-mount.sh -m rootfs/
```

虚拟机准备完毕，可以在虚拟的 arm 环境上进行相应的操作，创建用户、编译 arm 版

本、安装软件……

4. 卸载虚拟机在虚拟机环境中执行 exit 退出。然后执行命令卸载挂载的相关文件。

```
sudo ./ch-mount.sh -u rootfs/
```

4.3.3.4.9 使用 eFuse 和 SPACC 进行加解密

答：BM1684 芯片内的 eFuse 的基础使用请参考 SM5 手册的 3.6 节。这里仅介绍其中一个特殊的 secure key 区域。用户可以在 secure key 和其副本区域分别烧写同一个密钥，然后在地址 1 中的 bit[0] 和 bit[1] 烧写 1 以使能 secure key 功能，这样之后软件将不再能从 secure key 区域读出密钥原文，这个密钥只能被 SPACC 硬件模块用作 AES、DES 加解密之用。以上操作都是不可逆的，即 secure key 无法被改写，这个功能也无法被关闭。

因为早期硬件上的一个小问题，欲使用此功能的话，请先做如下检查：

```
#在板子上执行
sudo busybox devmem 0x50010004
#返回值是个16进制字串，如0x00018A05
#只看最后一位数字，如果是1，则可以正常使用此功能，
#如果是5，需要对板子做一下rework，请联系技术支持返厂修改
```

在烧写时，如果先烧写了 secure key enable bit，则 secure key 区域马上变成不能被读出（总是读出为 0，但仍可以写入）。如果先烧写了 secure key，此时它仍可以被读出，直到烧写 secure key enable bit 后才变成不能被读出。

Secure Key 在 eFuse 中的字节序如下，我们以 128bit 的 AES key 举例。现有一 128bit 的 AES key 如下：

```
0x00 0x01 0x02 0x03 0x10 0x11 0x12 0x13 0x20 0x21 0x22 0x23 0x30 0x31 0x32 0x33
```

在 eFuse 中的存储顺序为:

地址	内容 (HEX)
2	0x03020100
3	0x13121110
4	0x23222120
5	0x33323230

如果想要使用底层接口编程读写 eFuse（这个并不是必须的，也可以直接使用 SPACC 提供的接口，详见后面 SPACC 的部分），打开/dev/bm_efuse 节点，通过如下两个 ioctl 对 eFuse 原始数据进行读写。ioctl 参数为一个结构体，其中 addr 即为 0 到 127 的 eFuse 地址，val 用于存放读取和写入的 32bit 值：

```
#define EFUSE_IOCTL_READ    _IOWR('y', 0x20, struct efuse_ioctl_data)
#define EFUSE_IOCTL_WRITE   _IOWR('y', 0x21, struct efuse_ioctl_data)

struct efuse_ioctl_data {
```

（下页继续）

(续上页)

```
    uint32_t addr;
    uint32_t val;
};
```

读取:

```
int file = open("/dev/bm_efuse", O_RDWR);
struct efuse_ioctl_data data;
int ret = 0;

data.addr = address;
ret = ioctl(file, EFUSE_IOCTL_READ, &data);
if (ret < 0) {
    ERR("EFUSE_IOCTL_READ fail,errno=0x%x", errno);
    close(file);
    return ret;
}
close(file);
return data.val;
```

写入:

```
int file = open("/dev/bm_efuse", O_RDWR);
struct efuse_ioctl_data data;
int ret = 0;

data.addr = address;
data.val = val;
ret = ioctl(file, EFUSE_IOCTL_WRITE, &data);
if (ret < 0)
    ERR("EFUSE_IOCTL_WRITE fail ,errno=0x%x\n", errno);

close(file);
```

SPACC 是 BM1684 芯片内置的一个硬件加速模块，支持 AES、DES、SM4 和 SHA1、SHA256、Base64。这里介绍 SPACC 搭配 eFuse 使用的一个特殊功能：客户可以在 eFuse 的 secure key 区域烧录一把密钥，这把密钥写入后是不能再被软件读出的，只能被 SPACC 用作加解密运算。结合客户自定义 ID 使用，可以用作产品授权等功能。以下是 eFuse 和 SPACC 结合使用时的 API。

1. 查看 secure key 使能状态和客户自定义 ID

```
#查看secure key使能状态
sudo cat /sys/devices/platform/50028000.efuse/secure-key

#查看客户自定义ID
sudo cat /sys/devices/platform/50028000.efuse/uid
```

2. 烧写 secure key 和客户自定义 ID 为方便使用，烧写 secure key 和客户自定义 ID，可以不适用前述的 RAW API，而是使用下面的 nvmem 接口，重启后生效：

```
#切换到root用户
sudo -i

#解锁eFuse烧录限制
echo 0 > /sys/devices/platform/50028000.efuse/nvmem-lock

#烧录key
echo 000102030405060708090a0b0c0d0e0f111213141516171819101a1b1c1d1e1f > /sys/devices/
↪platform/50028000.efuse/secure-key

#烧录客户自定义ID
echo 0123456789abcdef0123456789abcdef > /sys/devices/platform/50028000.efuse/uid
```

如您的 key 为 128bit 长度，可仅烧写 128bit，如：

```
echo 000102030405060708090a0b0c0d0e0f > /sys/devices/platform/50028000.efuse/secure-key
```

客户自定义 ID 可烧录少于 128bit 的任意长度，如烧录 4byte ID：

```
echo 01234567 > /sys/devices/platform/50028000.efuse/uid
```

3. 如何使用 SPACC SPACC 已经集成到 Linux AF_ALG Crypto Framework 中，关于如何使用 Linux Crypto Framework，可参考 Linux Crypto Framework 。代码可参考 libkcapi ，也可直接使用 libkcapi。如需使用 secure key 加密，请在调用 socket 函数时，将 salg_name 写成带有 secure key 后缀的 cipher 名称，如：

```
struct sockaddr_alg sa = {
        .salg_family = AF_ALG,
        .salg_type = "skcipher",
        .salg_name = "cbc(aes)"
};
```

更改为

```
struct sockaddr_alg sa = {
        .salg_family = AF_ALG,
        .salg_type = "skcipher",
        .salg_name = "cbc(aes-secure-key)"
};
```

如您使用 libkcapi，请在调用初始化函数时，将 ciphername 参数设置为带有 secure-key 后缀的 cipher 类型，如：

```
kcapi_cipher_init(&handle,"cbc(aes)", 0);
```

更改为：

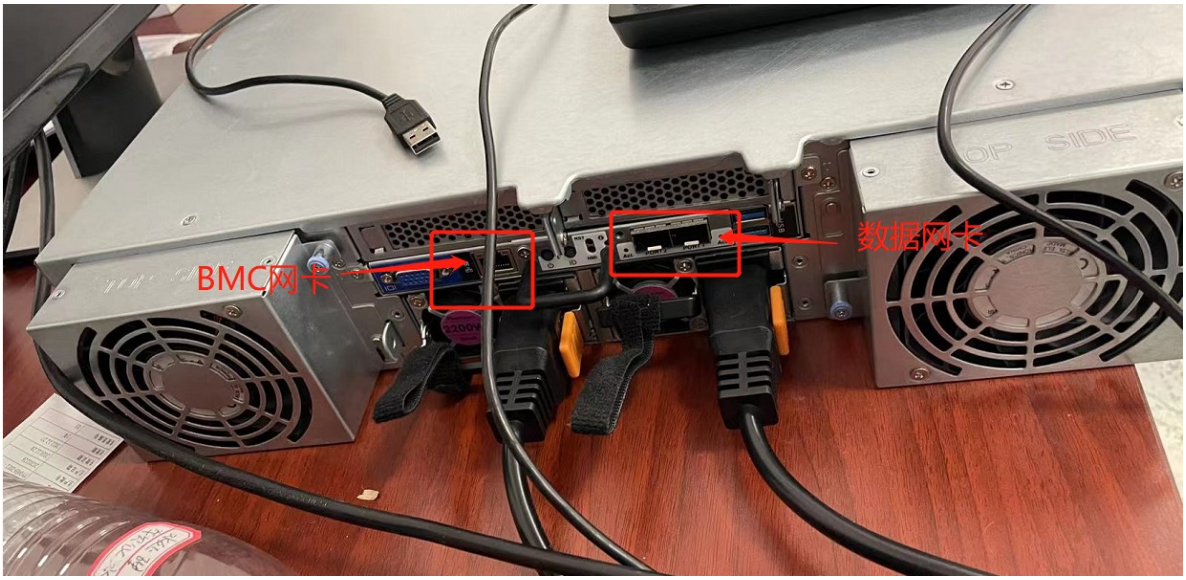
```
kcapi_cipher_init(&handle,"cbc(aes-secure-key)", 0);
```

4.4 智算服务器常见问题

4.4.1 SG6 服务器常见问题

4.4.1.1 SG6 服务器有两个网卡，但是有一个电口和两个光口，具体应该怎么连接？

答：SG6 光口才能实现操作系统联网；电口是 BMC 管理口，插上网线之后有一个网页可以对服务器进行管理。



4.4.1.2 SG6-10 的 BMC 密码忘记了，如何重置？

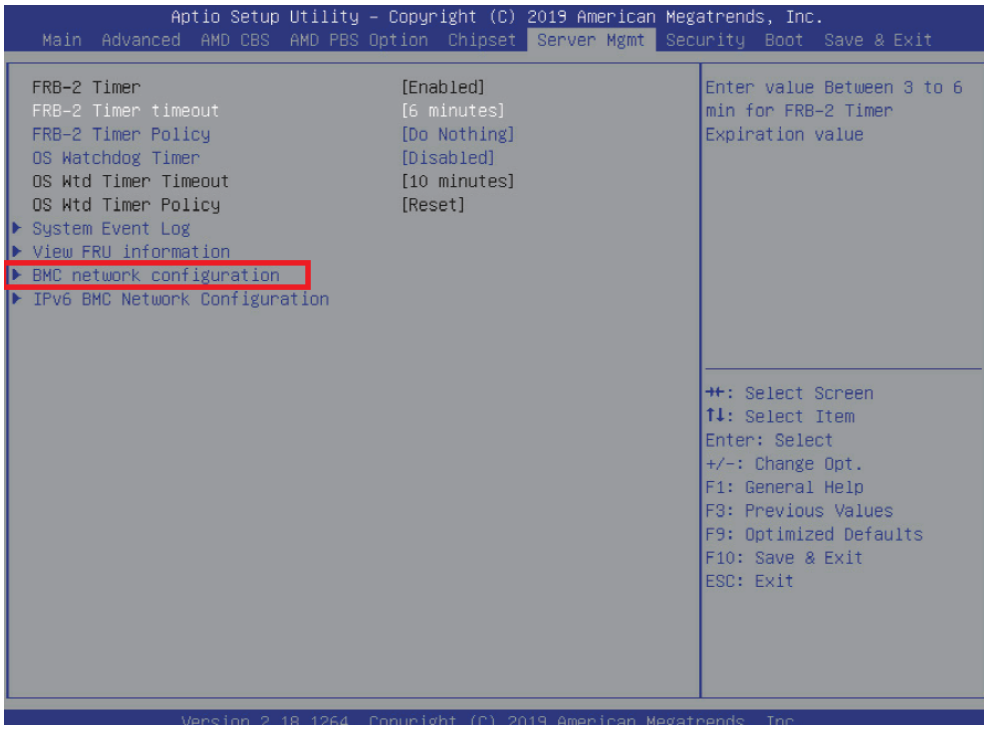
答：

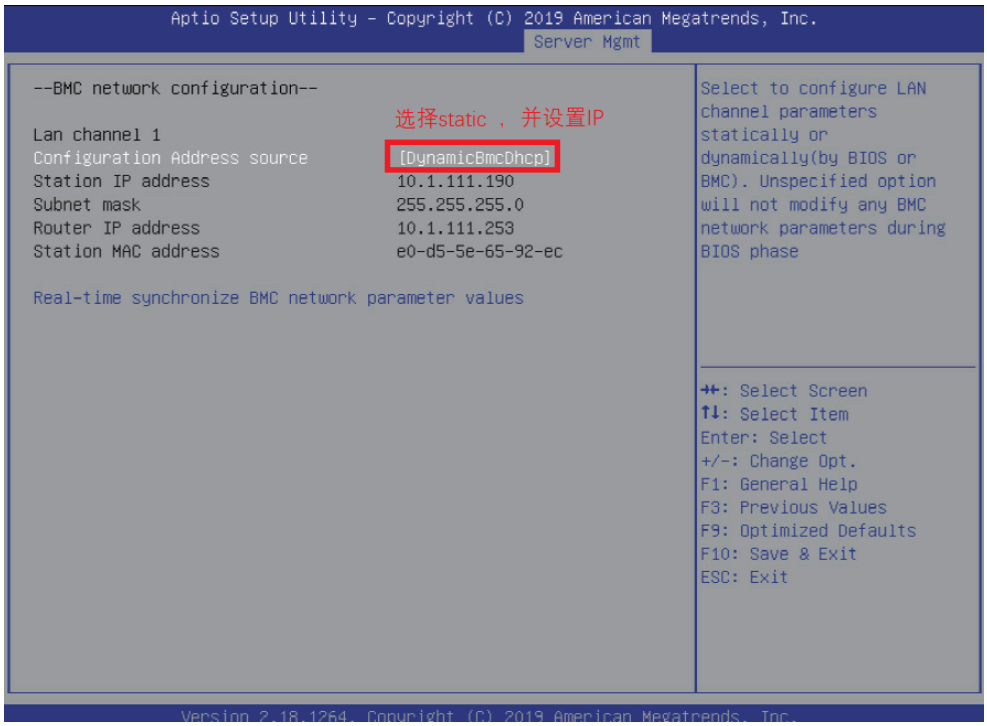
1. 通常默认出货默认密码是 admin1234 或 password ，如果不对或忘记密码可以按下面步骤进行密码重置；
2. 确认服务器可以联网，已经安装操作系统（以 ubuntu 系统为例）；
3. 在操作系统下运行 sudo apt-get install ipmitool -y 安装 ipmitool 工具；
4. 在操作系统下运行 sudo ipmitool user list 1，查看当前 BMC 的用户列表；一般我们默认用的是 admin 用户，查看并记住 admin 用户的序号；
5. 在操作系统下运行 sudo ipmitool user set password 2 admin1234（其中 2 是需要重置密码的用户的序号，后面的 password 是你要设置的密码）
6. 此时就可以在电脑通过浏览器访问 bmc，用 admin 用户和重置的密码登陆 BMC。

4.4.1.3 SG6-10 的风扇噪音太高，怎样能把噪音降低？

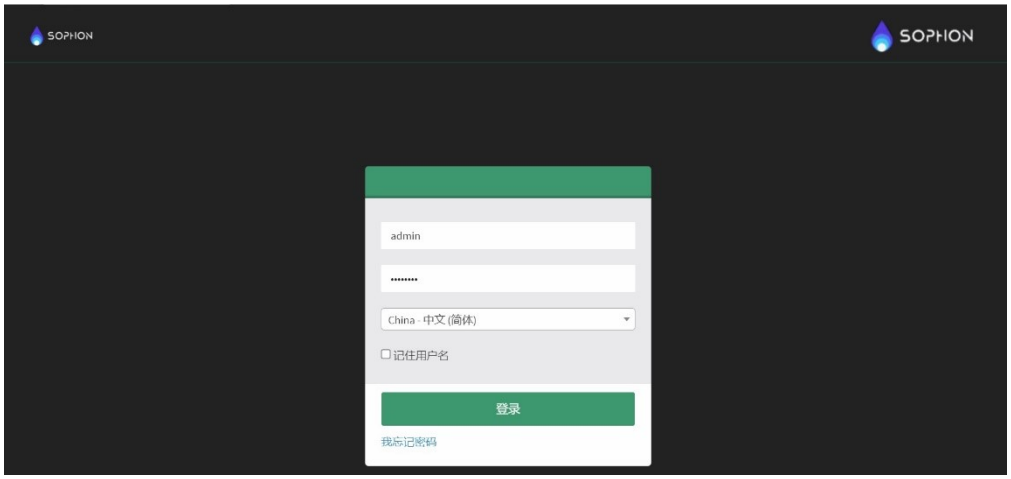
答：

1. 首先确认噪音大是否是在启动阶段。服务器在启动过程，未进入操作系统前风扇转速是全速，进入操作系统并加载加速卡驱动后，BMC 才能获取到卡的温度，根据加速卡温度进行风扇自动调速；
2. 确认操作系统是否已经安装加速卡的驱动程序。需要安装并加载加速卡的驱动后，BMC 才能获取到卡的温度，根据加速卡温度进行风扇自动调速；
3. 确认 BMC 的风扇策略是否设置正确，需要选择并运行 default 的风扇策略。确认和设置运行 default 风扇策略的步骤如下（如已可以登陆 BMC WEB 管理页面，直接跳至 b 步骤）：1) 将服务器的管理网口连上网络。服务器开机启动时会在屏幕左下方显示 BMC 的 IP 地址。BMC 网口默认是 DHCP 方式，如果局域网无 DHCP 服务，则在开机启动时，按 DEL 进入 BIOS 设置，在如下界面选择并设置 BMC 静态 IP：

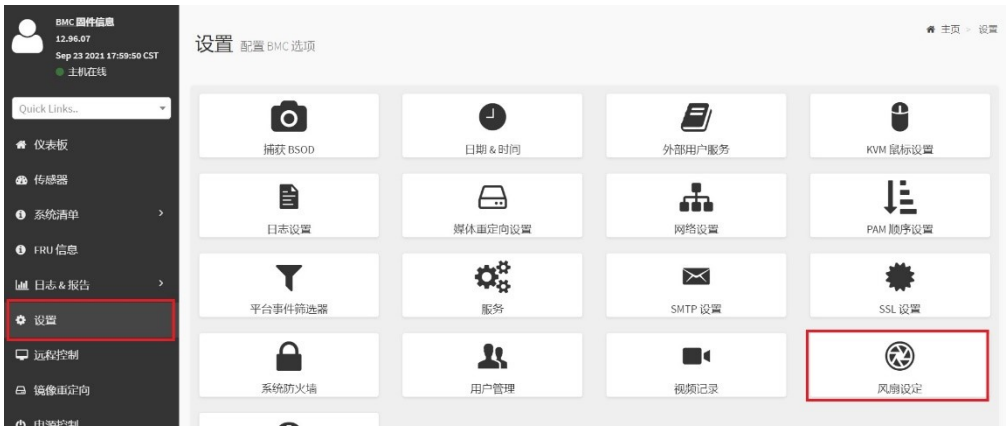




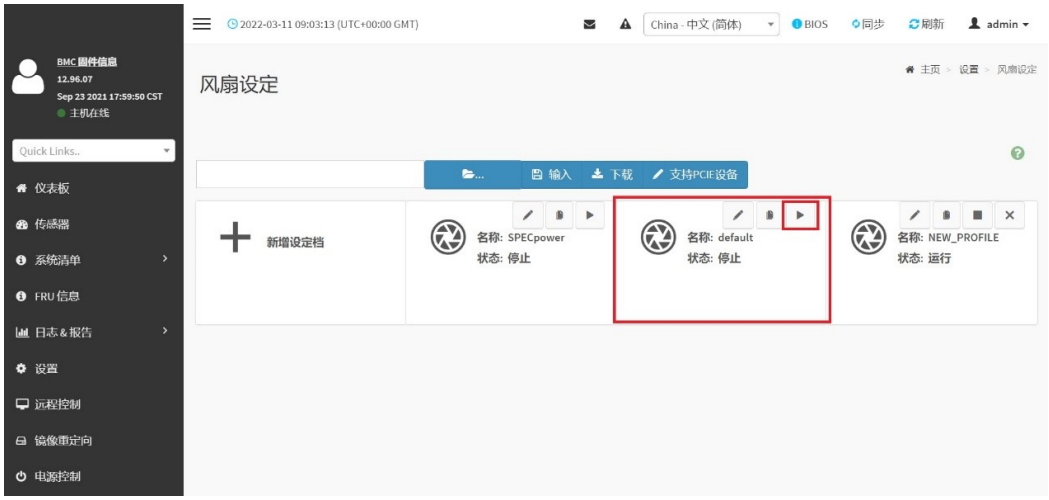
2) 在笔记本或其他电脑用浏览器通过 web 方式登陆 BMC，用户名为 admin，密码为 admin1234 或 password:



3) 登陆后，在左侧浏览树选择 < 设置 >，在右侧选择 < 风扇设定 >:



4) 确认目前是否运行的是 default 风扇策略，如果不是，则按下图点击 default 策略上的运行标志，运行 default 风扇策略:



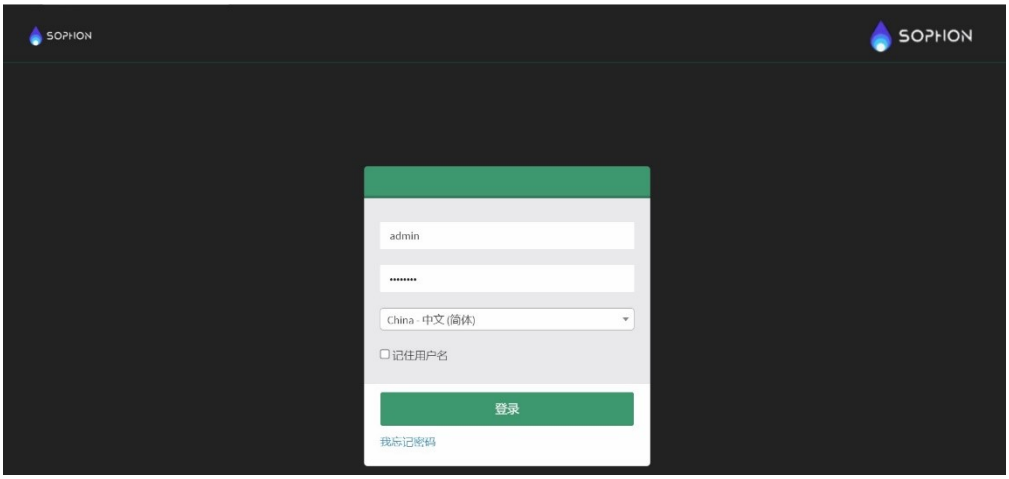
5) 在弹窗的窗口点击 < 确认 > 即可运行 default 风扇策略:

6) 此时风扇是根据服务器内多个温度传感器及加速卡温度进行自动调速的。

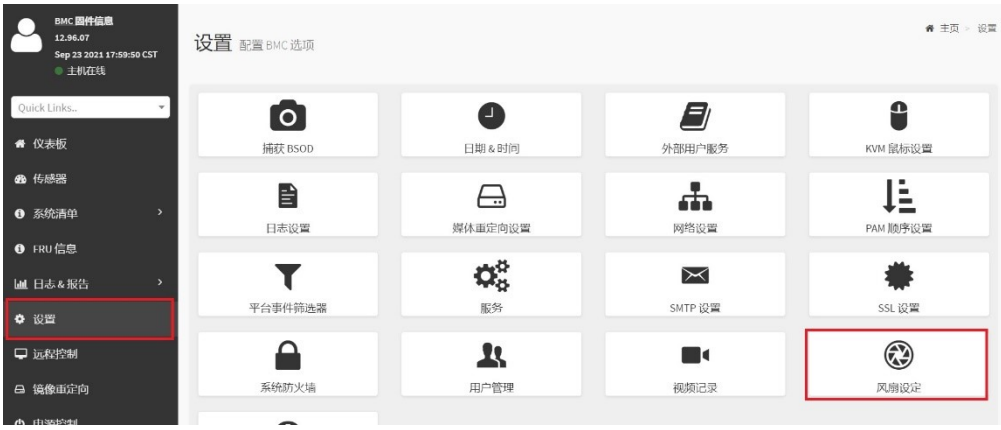
4.4.1.4 SG6-10 在配置 10 张加速卡时，位于 CPU 后方两个半高 PCIE 槽位的加速卡温度偏高，如何能调高服务器 CPU 前部的两个风扇转速？

答：运行 default 风扇策略时，CPU 前部的两个风扇转速默认是根据 CPU 温度、主板温度、硬盘温度来进行调速的，在环境温度太高时可能会出现加速卡温度偏高的情况。可根据下面步骤，设置 CPU 前部的两个风扇转速。

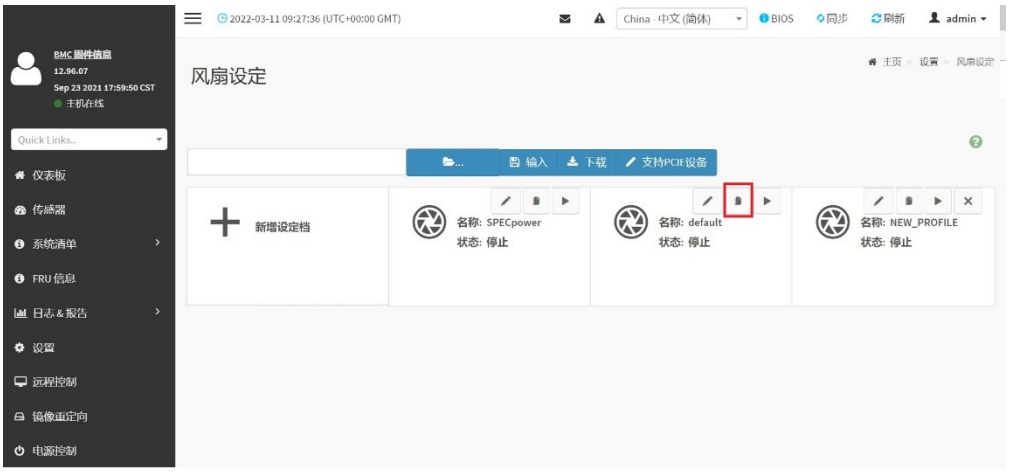
1. 在笔记本或其他电脑用浏览器通过 web 方式登陆 BMC，用户名为 admin，密码为 admin1234 或 password：



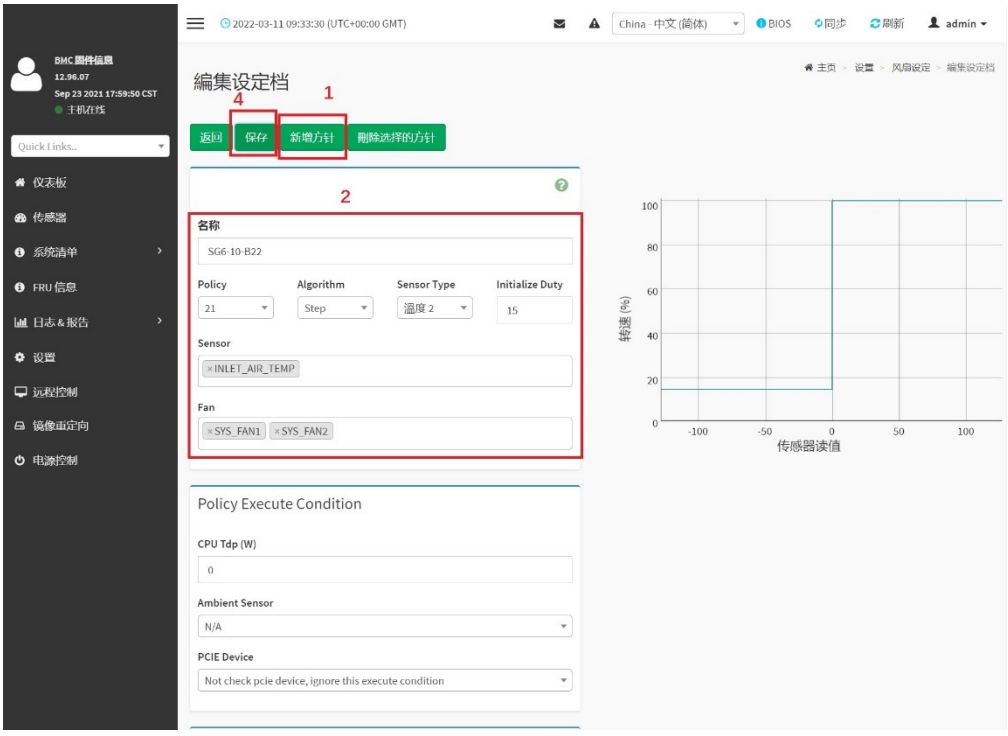
2. 登陆后，在左侧浏览树选择 < 设置 >，在右侧选择 < 风扇设定 >：



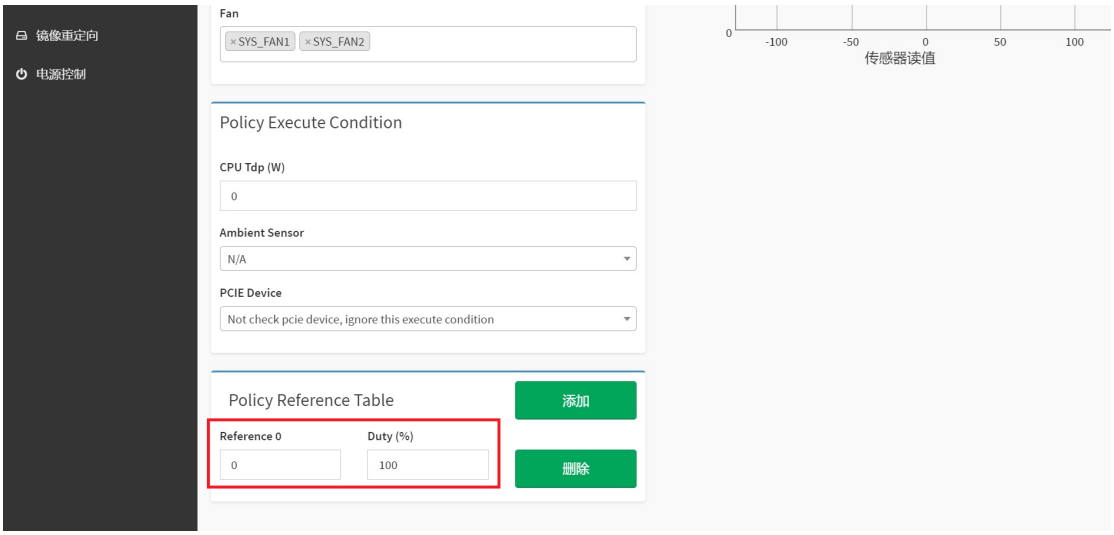
3. 点击下图 default 风扇策略的复制标志，新建一个风扇策略：



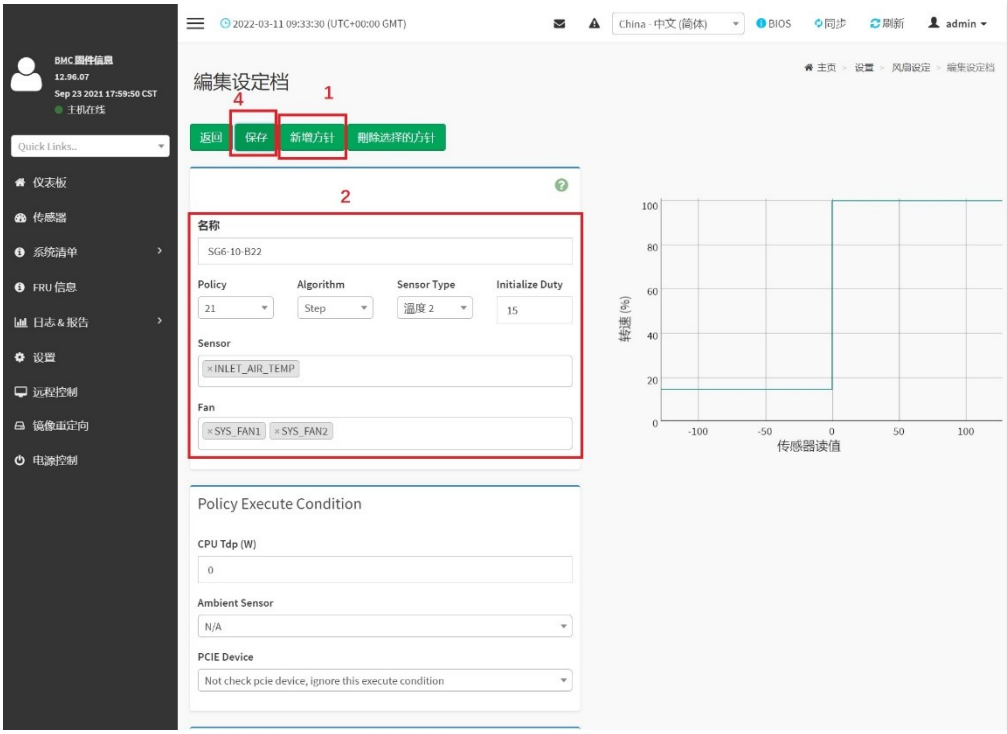
4. 点击下图红色标记 <1> 处，新增一个 Policy，在 <2> 处，按下图选择设置：



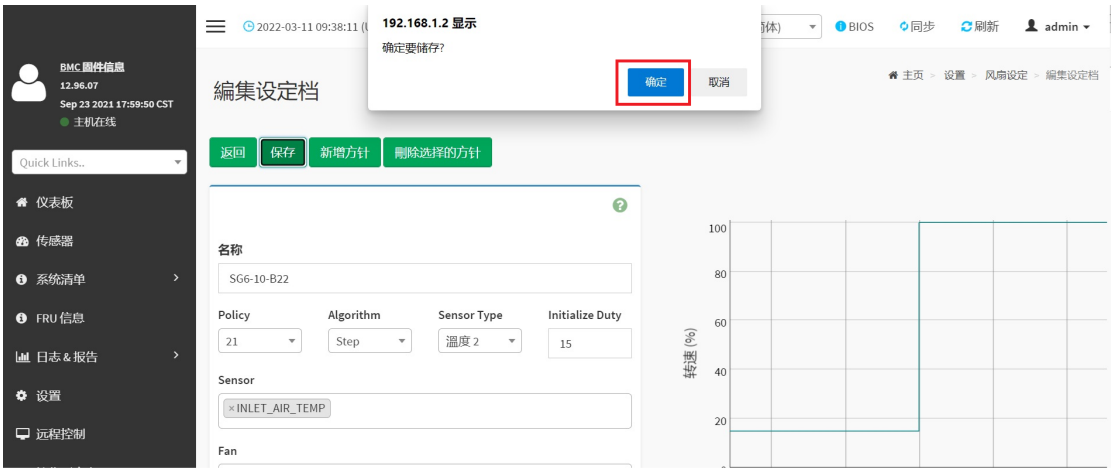
5. 在下方 reference 处填 0，Duty 处填写期望的风扇转速百分比，100 为全速。10 卡配置时如考虑 35 度环温，建议设置 100：



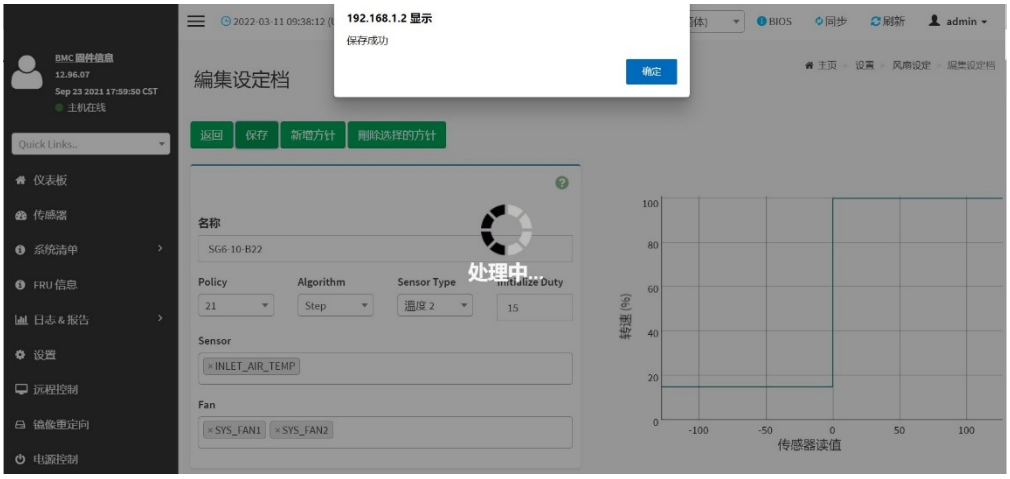
6. 点击步骤下图的 <4> 保存:



7. 点击确认:

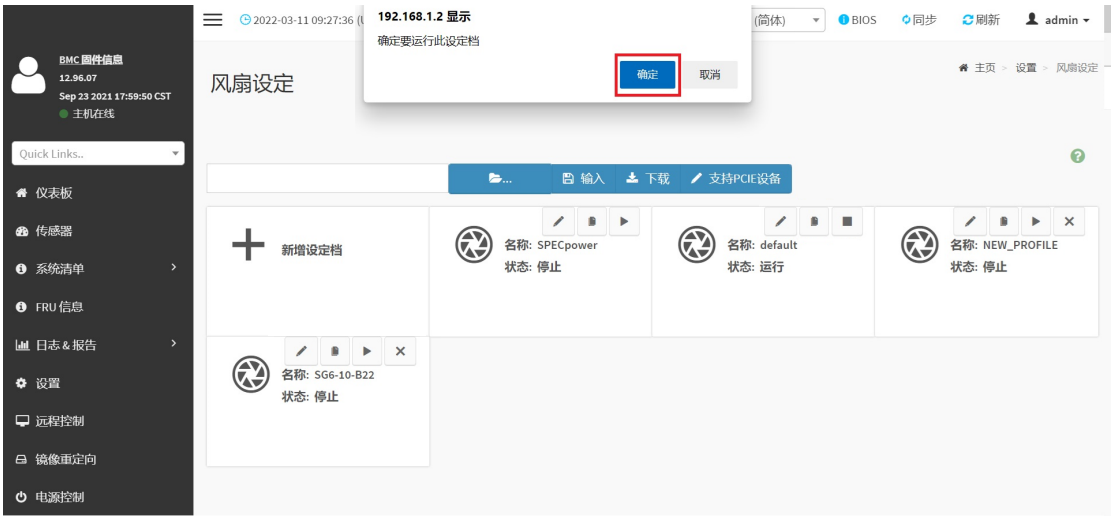


8. 提示保存成功，点击确认：



9. 点击新增加风扇策略上的运行标志，运行该新增策略：

10. 点击确认，确认运行：



11. CPU 前部的两个风扇转速按设置值或更转速值运行（此两风扇对应区域温度传感器检测到温度偏高时）。

CHAPTER 5

模型转换及量化常见问题

开发文档：《NNToolChain 用户开发手册 》、《量化工具用户开发手册 》

5.1 基本使用

5.1.1 pytorch 模型转换需要注意的事项？

- 什么是 JIT（torch.jit）?

  答：JIT（Just-In-Time）是一组编译工具，用于弥合 PyTorch 研究与生产之间的差距。它允许创建可以在不依赖 Python 解释器的情况下运行的模型，并且可以更积极地进行优化。

- 支持什么格式的 pytorch 模型?

  答：Sophon 的 PyTorch 模型编译工具 BMNETP 只接受 PyTorch 的 JIT 模型 JIT 模型（TorchScript 模型）。

- 如何得到 JIT 模型?

  答：在已有 PyTorch 的 Python 模型（基类为 torch.nn.Module）的情况下，通过 torch.jit.trace 得到；traced_model=torch.jit.trace(python_model, torch.rand(input_shape))，然后再用 traced_model.save(‘jit.pt’) 保存下来。

- 为什么不能使用 torch.jit.script 得到 JIT 模型?

  答：BMNETP 暂时不支持带有控制流操作（如 if 语句或循环）的 JIT 模型，但 torch.jit.script 可以产生这类模型，而 torch.jit.trace 却不可以，仅跟踪和记录张量上的操作，不会记录任何控制流操作。

- 为什么不能是 GPU 模型?

  答：BMNETP 的编译过程不支持。

- 如何将 GPU 模型转成 CPU 模型?

  答：在加载 PyTorch 的 Python 模型时，使用 map_location 参数：torch.load(python_model, map_location = 'cpu')。

5.1.2 paddle 模型 *.pdmodel*.pdpot 如何转换为 bmodel ?

- 静态图保存 *.pdmodel 和 *.pdpot 与 *.pdmodel 和 *.pdiparams 的区别?

  答：其主要差别在于保存结果的应用场景：.pdmodel，.pdparams，.pdopt 三种由 save 接口保存 (2.0 的 paddle.static.save 或者 1.8 的 fluid.io.save)。.pdmodel 和 *.pdiparams 由 save_inference_model 接口保存（2.0 的 paddle.static.save_inference_model 或者 1.8 的 fluid.io.save_inference_model）

  - save 接口:

    该接口用于保存训练过程中的模型和参数，一般包括 *.pdmodel，.pdparams，.pdopt 三个文件。其中 *.pdmodel 是训练使用的完整模型 program 描述，区别于推理模型，训练模型 program 包含完整的网络，包括前向网络，反向网络和优化器，而推理模型 program 仅包含前向网络，.pdparams 是训练网络的参数 dict，key 为变量名，value 为 Tensor array 数值，.pdopt 是训练优化器的参数，结构与 *.pdparams 一致

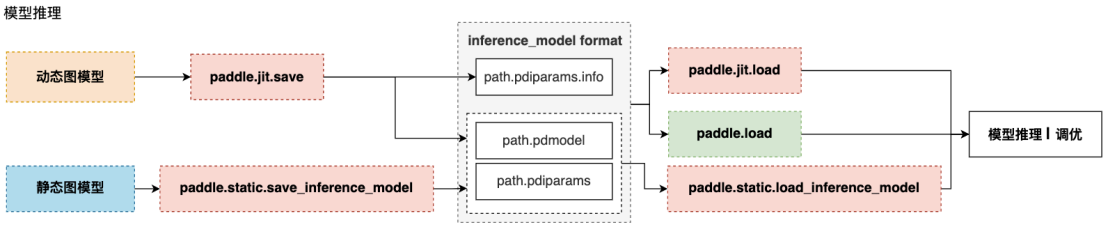
  - save_inference_model 接口:

    该接口用于保存推理模型和参数，2.0 的 paddle.static.save_inference_model 保存结果为 *.pdmodel 和 *.pdiparams 两个文件，其中 *.pdmodel 为推理使用的模型 program 描述，.pdiparams 为推理用的参数，这里存储格式与.pdparams 不同（注意两者后缀差个 i），*.pdiparams 为二进制 Tensor 存储格式，不含变量名。1.8 的 fluid.io.save_inference_model 默认保存结果为 __model__ 文件，和以参数名为文件名的多个分散参数文件，格式与 2.0 一致。

  详情请参考：模型保存与载入-使用文档-PaddlePaddle 深度学习平台

- 如何生成 *.pdmodel 和 *.pdiparams?

  答：



- 动态图存储模型结构和参数根据训练模式不同，分为两种情况：

  1. 动转静训练 + 模型 & 参数存储

     动转静训练相比直接使用动态图训练具有更好的执行性能，训练完成后，直接将目标 Layer 传入 paddle.jit.save 存储即可。

  2. 动态图训练 + 模型 & 参数存储

     动态图模式相比动转静模式更加便于调试，如果仍需要使用动态图直接训练，也可以在动态图训练完成后调用 paddle.jit.save 直接存储模型和参数。

- 静态图推理模型 & 参数保存

  保存/载入静态图推理模型，可以通过 paddle.static.save/load_inference_model 实现。静态图导出推理模型需要指定导出路径、输入、输出变量以及执行器。save_inference_model 会裁剪 Program 的冗余部分，并导出两个文件：path_prefix.pdmodel 、path_prefix.pdiparams。注意事项请参考：模型保存与载入-使用文档-PaddlePaddle 深度学习平台

- checkpoint（persistable 模型）转换为 inference 模型保存

  在任务的训练阶段，通常我们会保存一些 checkpoint（persistable 模型），这些只是模型权重文件，不能直接被预测引擎直接加载预测，所以我们通常会在训练完之后，找到合适的 checkpoint 并将其转换为 inference 模型。需要构建训练引擎，之后保存 inference 模型即可。在 paddleclas 模型库的 tools/export_model.py 中提供了完整的示例，只需执行下述命令即可完成转换：

  ```
  python tools/export_model.py \
    --m=模型名称 \
    --p=persistable 模型路径 \
    --o=model和params保存路径
  ```

  详情请参考：分类预测框架—PaddleClas 文档

· bmpaddle 转换 Paddle Paddle 模型

答：

BMPADDLE 是针对 Paddle Paddle 的模型编译器，可以将模型文件（inference.pdmodel，inference.pdiparams）编译成 BMRuntime 所需的文件。而且在编译的同时，可选择将每一个操作的 NPU 模型计算结果和 CPU 的计算结果进行对比，保证正确性。命令行格式：

```
python3 -m bmpaddle [--model=<path>] \
                    [--input_names=<string>] \
                    [--shapes=<string>] \
                    [--descs=<string>] \
                    [--output_names=<string>] \
                    [--net_name=<name>] \
                    [--opt=<value>] \
                    [--dyn=<bool>] \
                    [--outdir=<path>] \
```

（下页继续）

(续上页)

```
                [--target=<name>] \
                [--cmp=<bool>] \
                [--mode=<string>] \
                [--enable_profile=<bool>] \
                [--list_ops]
```

python 模式:

```
import bmpaddle
## compile fp32 model
bmpaddle.compile(
  model = "/path/to/model(directory)", ## Necessary
  outdir = "xxx", ## Necessary
  target = "BM1684", ## Necessary
  shapes = [[x,x,x,x],[x,x,x]], ## Necessary
  net_name = "name", ## Necessary
  input_names=["name1","name2"], ## Necessary, when .h5 use None
  output_names=["out_name1","out_name2"], ## Necessary, when .h5 use None
  opt = 2, ## optional, if not set, default equal to 1
  dyn = False, ## optional, if not set, default equal to False
  cmp = True, ## optional, if not set, default equal to True
  enable_profile = True ## optional, if not set, default equal to False
  )
```

注意:

1. –model 参数指定到模型所在文件夹那一级，但要特别注意，PaddlePaddle 模型有 2 种：组合式 (combined model) 和非复合式 (uncombined model)；组合式就是 *model + 权重，__model__ 文件夹下有很多文件，每一个文件是一层，这种模型必须用 _model_ ；如果是非组合式，一定要把模型名称修改为 *.pdmodel 和 *.pdiparams
2. shapes 和 descs 中的变量顺序、名称要和实际模型一致，不能写错。
3. 对于模型中带 nms 操作的，并且 cmp==true 时 descs 参数必填；对于 int32 类型，范围不能填重复的值，比如 608*608 的输入，要填 608,609，但生效的就是 608；对于 float 类型，则没有这个限制；对于不填的输入，其取值范围默认都是 0-1。

5.1.3 模型转换失败怎么办？

答:

1. 检查命令行输入参数有没有错误，这个一般会有打印提示；
2. 不支持的算子需要使用 BMLang 或者 OKKernel 开发，也可以联系 FAE 来解决；
3. 如果是 pytorch 模型，是不是没有做 trace?
4. 使用转换工具编译模型时，设置环境变量 export BMCOMPILER_STAT_ERR=1，然后加上–v 4，保存更详细的日志，提供给我方技术人员进一步排查；

5. 有时转换失败是因为误差比对超过了允许的阈值而导致编译过程中断，目前比对误差阈值设置为误差在 0.01 之内，但也不排除有些模型有很多的累加或除法操作，由于尾差累计导致超出这个范围；可以加上–cmp False 关闭比对，最终到业务层面上验证转换后的模型精度是否符合要求；
6. 2.7.0 以后的 sdk 可以通过 BMCOMPILER_STAT_ERR=1，来看每层的数据相似度，个别超过误差不会中断完整编译过程。

5.1.4 如何使用 BMLang 开发自定义的算子？

BMLang 是算能科技面向用户推出的针对 BM168x TPU 的一套高级编程接口库，使得用户可以快速的基于 TPU 硬件开发自定义算子，甚至整个神经网络。BMLang 的基本元素是：张量数据（bmlang::Tensor）和计算操作（bmlang::Operator）。用户需要使用 bmlang::Tensor 和 bmlang::Operation 来编写 C++ 代码，然后在程序最后使用 bmlang::compile 或 bmlang::compile_with_check 来生成 TPU 可以运行的二进制文件 bmodel，与普通网络编译产生的 BMODEL 文件一样，依赖于 BMRuntime 接口载入与执行。

5.1.5 是否支持模型的在线编译？

答：不支持，也不推荐这样做，因为模型编译的过程是特别耗时的。因此我们采用了离线编译生成 BModel，在线推理时直接加载 BModel 运行的方式。

5.2 fp32 模型转换

5.2.1 fp32 模型的输出和原始模型输出差异比较大怎么办？

答：

1. 对于两个模型采用同样的输入，看看输出是否一致，比如输入都是用全部是 0.1 的矩阵，填充 input tensor 的内存空间，然后做推理，比较输出数据的差异；
2. 设置 export BMRT_SAVE_IO_TENSORS=1，运行 bmruntime 的时候会输出两个文件 input_ref_data.dat.bmrt 和 output_ref_data.dat.bmrt，用这个和理想的输入输出作对比排查；比较数据时尽量使用二进制比较，不要通过打印的数据比较，因为打印的输入格式不一样，也会导致数据显示不一样。

5.2.2 使用 bmpaddle 转换模型时应该如何填写参数？descs 参数是选填还是必填？

答：示例：

```
python3 -m bmpaddle --model=model/ --input_names="image,im_size" --shapes="[1, 3, 608, 608],
↪[1, 2]" --target="BM1684" --cmp=true  --
descs="[1,int32,608,609]"
```

注意:

1. –model 参数到模型所在文件夹那一级;paddle 模型有 2 种:组合式 (combined model) 和非复合式 (uncombined model);组合式就是 __model__ + 权重,__model__ 文件夹下有很多文件,每一个文件是一层,这种模型名称必须用 __model__;如果是非组合式,要用.pdmodel 和.pdiparams;
2. shapes 和 descs 中的变量顺序、名称要和实际模型一致,不能写错;
3. 关于 descs 参数是选填还是必填?对于模型中带 nms 的目标检测网络,并且 cmp==true 时 descs 必填,并且对于多输入时的其他输入(比如 paddle 检测模型中通常都会有的 im_size 参数),必须写明。对于没有填的输入,默认会产生 0-1 的随机数,比如输入图像,可以不填;当类型为 int32 时,不能填重复的值,故写 608,609,但生效的就是 608;float 类型没这个限制;
4. paddle-ocr-detection 开比对会有误差累计,报错。另外有的模型算子中有很多累加或除法,由于误差累计会导致超出允许的比对误差阈值,转换中断报错;还有的有排序操作,小误差会导致顺序不同。这些都会导致转换中断,可以关闭 cmp,不进行数据比对,到业务层面验证转换后模型的精度。

5.3 int8 模型量化

5.3.1 int8 的输出和 fp32 模型输出差异比较大怎么办?

答:

1. 检查前后处理是否有问题,int8 网络输入输出一般需要做 scale 处理,看看是否遗漏?
2. 通过量化可视化工具分析 int8 的输出和 fp32 的输出的精度差异,做个输出曲线对比;通过曲线查找出导致误差较大的层,通过更换该层的量化方式调整结果,找到最优量化方式,若仍然不能满足精度要求,则可将该层设置为 float 输出,不进行量化。

5.3.2 关于模型量化工具 calibration_use_pb 指定校准参数的问题?

calibration_use_pb 这个工具和校准相关的参数:

-accuracy_opt
-per_channel
-th_method

1. 这三个参数是互斥的存在,还是可以组合使用?

   答:可以组合使用,accuracy_opt 针对 depthwise 的卷积,如果没有 depthwise 可以不选,跟 th_method 不冲突。

2. per_channel 这个参数指定成 true 之后，还需要其他的参数配合吗？比如：first_conv_enable_per_channel

   答：per_channel 可以单独使用。

3. th_method 这个参数可以同时指定多个吗？比如同时指定：-th_method=SYMKL, -th_method=JSD

   答：th_method 一次只能指定一种。

5.3.3 如何提高模型的量化效率和精度？

答：推荐策略是：

1. 对于检测和分类模型，使用自动量化工具，量化成功后精度测试；
2. 对于其他类模型，手动指定量化参数，量化成功后精度测试；
3. 自动量化失败或精度不够的，反馈给我们的技术支持。

5.3.4 关于量化方法的问题？

问：手动指定量化参数时，需要尝试 18 种量化方法的组合，并逐个测试精度。是否可以提供相关量化参数的说明及推荐使用方法？我们希望能根据模型结构缩小量化方法的搜索范围。

答：kl 和 symkl 和 jsd 算法比较相似可能差别不大，bert 一般用 max。多数网络 max 表现一般，但是各种算法和网络结构和用户训练结果都有关系，所以先默认 kl 量化，下降多的话再尝试其他组合可能省些时间。

问：这些量化方法对量化后模型的性能有影响吗？

答：几种门限算法对性能没影响，accuracyopt 会把 depthwise 卷积用浮点，一般会变慢，perchannel 也会变慢，这两个是为了提高精度。

5.3.5 可以使用已有的量化表（比如 TensorRT 量化后得到的量化表）作为输入来完成 BModel 模型的量化吗？

答：目前不支持，主要是我们前端转换成 umodel 之后网络的 layer 的名字都对不上了。

5.3.6 YOLOv3 的 darknet 模型先转为 caffe 模型后再转为 fp32bmodel，模型输出和原始模型输出存在偏差？

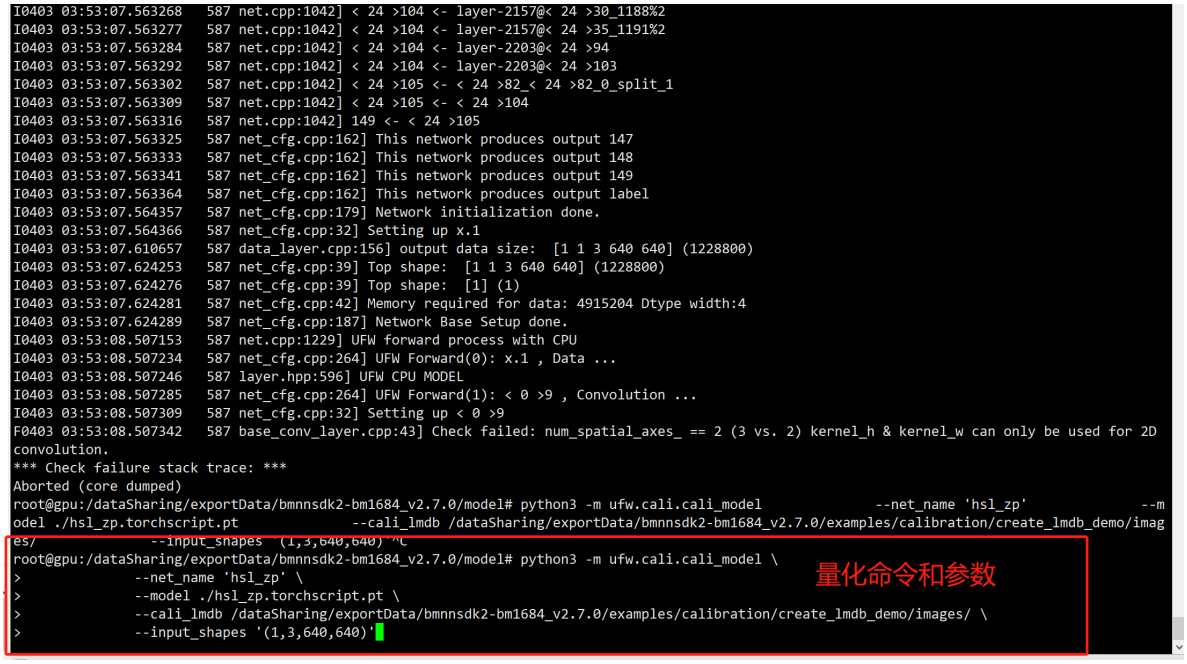
答：darknet 模型转 caffe 模型过程中可能存在偏差，建议直接用 darknet 模型转为 fp32bmodel。将 YOLOv3 的 darknet 模型直接转为 bmodel，推理结果没有偏差。

5.3.7 使用 bmnetd 编译 Darknet 出现段错误 Unknown error 27620053 ?

答：编译模型段错误时，检查 xxx.cfg 文件，如果在 windows 系统下保存或修改过，由于和 linux 系统对换行符表示不同，会导致解析出错引起段错误，可以用命令转换：

```
dos2unix xxx.cfg
```

5.3.8 使用 SDK2.7.0_20220316 auto_calib 工具时报错 base_conv_layer.cpp43] Check failed: num_spatial_axes_ == 2 (3 vs. 2) kernel_h & kernel_w can only be used for 2D convolution



答：关于使用 SDK 2.7.0_20220316 自动量化工具时报错的问题：

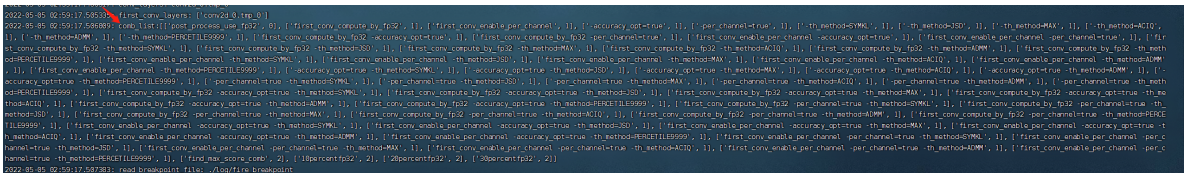
1. 使用 auto_calib ufw.calib.cali_model 为了兼容之前就版本 SDK 生成的 LMDB 文件，要求 LMDB 必须是 3, 640, 640 这种的，而不能是 1 3 640 640
2. 删除 examples/calibration/create_lmdb/convert_imageset.py 中 cv_img = np.expand_dims(cv_img,axis=0) 这一句，重新生成 lmdb。新生成 lmdb 时，请先删除掉原先生成的 lmdb 那个文件夹，否则无法覆盖原先生成的。刚才您应该是已经有之前生成的文件了，所以修改后没生效。
3. 如果 auto_calib ufw.calib.cali_model 的 preprocess 参数可以满足模型预处理要求，那么使用时可以直接使用图片集，这个不会涉及 LMDB 文件数据的维度。

补充说明：使用手动分步方式量化模型时，也要注意，如果使用的是 1 3 640 640 这样的 lmdb，那么修改使用 ufw.tools.xx_to_umodel 生成的 fp32umodel prototxt 时，prototxt 里要 batch_size 要设置为 0（如果在使用 ufw.tools.xx_to_umodel 时，指定了-D=" ${lmdb_dst_dir} 那么程序自动会在生成的 protoxt 里将 batch size 设置为 0）；如果使用的是 3 640 640 这样的 lmdb，那么 fp32umodel prototxt 里 batch_size 要设

置为 1。具体说明，请参考 https://doc.sophgo.com/docs/docs_latest_release/calibration-tools/html/module/chapter4.html#using-lmdb

5.3.9 一键量化会不会遍历 KL、MAX 这些量化策略，策略的细节有哪些？

答：autocalib 会遍历所有量化策略，循环进行量化和测试精度的过程，叠加每次进行多次 iteration 的推理，选优提高量化精度。目前搜索策略有：



5.3.10 量化策略里的 MAX 方法是指 min-max 吗？

答：MAX 是用最大值作为量化阈值，不是 min-max。

CHAPTER 6

多媒体用户常见问题

开发文档：《多媒体用户开发手册 》

BM-FFmpeg 开源仓库：https://gitee.com/sophon-ai/bm_ffmpeg

6.1 4K 图片的问题

答：有些客户需要的图片较大如 8K 等，由于 VPP 只支持 4K 大小的图片，所以通过 opencv 读取图片后，会自动保持比例缩放到一个 4K 以内的尺寸。

6.2 Opencv 读取图片后，cvMat 转为 bmimage，之后，调用 bmcv_image_vpp_convert 做缩放或者颜色空间转换，得到的图片不一致

原因分析：opencv 内部的转换矩阵和 bmcv_image_vpp_convert 使用的转换矩阵不一致，需要调用 bmcv_image_vpp_csc_matrix_covert，并且指定 CSC_YPbPr2RGB_BT601 来进行转换才能保持一致。

6.3 Opencv imread 读取图片性能问题。

原因分析：如果碰到图片小于 16x16 大小的图片，或者 progressive 格式的 jpeg，芯片不能实现加速，结果走了 CPU 的路径，导致客户发现图片解码并没有加速。

6.4 VideoWriter.write 性能问题，有些客户反应，存文件慢。

解析：就目前来看看采用 YUV 采集，然后编码 10-20ms 之间写入一帧数据属于正常现象。

6.5 Ffmpeg 的阻塞问题

原因分析：如果没有及时释放 avframe，就会导致阻塞，vpu 内部是循环 buffer。

6.6 关于什么时候调用 uploadMat/downloadMat 接口的问题。

解析：当创建了一个 cv::Mat, 然后调用 cv::bmcv 里面的接口进行了一些处理后，设备内存的内容改变了，这时候需要 downloadMat 来同步一下设备内存和系统内存。当调用了 cv::resize 等 opencv 原生的接口后，系统内存的内容改变了，需要 uploadMat，使设备内存和系统内存同步。

6.7 opencv 下如何获取视频帧的 timestamp ?

答：opencv 原生提供了获取 timestamp 的接口，可以在 cap.read() 每一帧后获取当前帧的 timestamp.

代码如下：

```
Mat frame;
cap.read(frame);
int64_t timestamp = (int64_t)cap.getProperty(CAP_PROP_TIMESTAMP);    *//
↪ 获取timestamp，返回值为double类型*
```

6.8 SA3 opencv 下 videocapture 经常 5 分钟左右断网的解决方案

答：最近因为路由板软件版本有问题，在 udp 方式下经常会发生 RTSP 数据连上后 3-5 分钟就”connection timeout”的问题，这个问题最终解决方案是更新最新的路由板软件。验证方法可以用 TCP 测试下，如果 TCP 没有问题可以确认是这类问题。

使用 TCP 的方式见下面：

```
export OPENCV_FFMPEG_CAPTURE_OPTIONS="rtsp_transport;tcp"
```

执行应用（如果用 sudo 执行，需要 sudo -E 把环境变量带过去）注意：最新的 middleware-soc 将使用 TCP 作为默认协议，对原来客户需要使用 UDP 传输协议的，需要引导客户按照下面方式进行设置

使用 UDP 方式：

```
export OPENCV_FFMPEG_CAPTURE_OPTIONS="rtsp_transport;udp"
```

执行应用（如果用 sudo 执行，需要 sudo -E 把环境变量带过去）

UDP 适用的环境：当网络带宽比较窄，比如 4G/3G 等移动通信系统，此时用 udp 比较合适
TCP 适用的环境：网络带宽足够，对视频花屏要求比较高，对延时要求较小的应用场景，适合 TCP

6.9 如何获取 rtsp 中原来的 timestamp

答：opencv 中默认获取的 rtsp 时间戳是从 0 开始的，如果想获取 rtsp 中的原始时间戳，可以用环境变量进行控制，然后按照问题 1 进行获取即可

```
export OPENCV_FFMPEG_CAPTURE_OPTOINS="keep_rtsp_timestamp;1"
```

6.10 如何判断视频花屏的原因

答：这里提的视频花屏是长时间的花屏，对于偶尔的花屏有可能是网络数据传输错误导致的，此类不属于应用代码可控的方位。如果视频出现长时间的花屏，很大概率是由于视频帧读取不及时，导致内部缓存满以后，socket recv buffer 溢出导致的。

1. 将加大 rmem_max 到 2M，如果此时花屏消失，说明应用的数据处理抖动很大，应该要加大 buffer queue 进行平滑

   ```
   echo 2097152 > /proc/sys/net/core/rmem_max
   ```

2. 用 netstat -na, 一般是一下格式，找到 rtsp 的那个端口（udp 在应用中会有打印，tcp 的话可以看目标 rtsp 地址），这里的 Recv-Q, Send-Q 在正常情况应该都是 0，或者不满的，如果 Recv-Q 经常有很大的数，就说明 overflow 了。一般 Send- Q 不会出问题，如果这个也很大的话，那么很可能 network driver 驱动挂死了。

   ```
   Proto Recv-Q Send-Q Local Address Foreign Address State
   tcp 0 0 0.0.0.0:111 0.0.0.0:\* LISTEN
   ```

6.11 无法连接 rtsp ?

答：可以通过 ffmpeg 固有命令来进行连接测试：（url 为 rtsp 连接地址）

```
ffmpeg -rtsp_transport tcp -i url -f rawvideo -y /dev/null
或者
ffmpeg -rtsp_transport udp -i url -f rawvideo -y /dev/null
若以上无法连接成功，请检查网络。
```

6.12 确认解码器是否能正常工作：（url 为文件名或者 rtsp 连接地址）

答：

```
ffmpeg -i url -f rawvideo -y /dev/null
```

6.13 确认解码器和 vpp 的 OpenCV 接口是否正常工作：

答：

```
vidmulti number_of_instances url1 url2
```

6.14 解码不正确或者无法解码的最终调试手段

答：如果经常各种调试后，在现场仍然无法解决问题，可以通过打开环境变量，把问题发生前后的数据 dump 下来，供后续进行进一步分析

在 PCIE 模式下

```
export BMVID_PCIE_DUMP_STREAM_NUM=1000
```

dump 的数据在/data/下（需要创建该文件夹并且有写权限），dump 的数据根据 core index 和 instance index 存储。

在 SOC 模式下

```
echo "0 0 1000 100" > /proc/vpuinfo
```

(dump 第 1 个 core 的第 1 个 instance 的码流数据)

这个配置会在两个文件之间循环存储 1000 帧数据，当问题发生的时候，把这两个发生前后的那个 10000 帧文件拷贝回来就可以。两个文件的存储位置在/data/core_%dinst%d_stream%d.bin. （注意：PCIE 需要提前准备/data/目录的写权限）

6.15 判断 rtsp 是否正常工作

答:

方法一: 通过 vlc 播放视频 (推荐), 分别设置 tcp, udp 方式

方法二: 在高密度服务器上用 vidmutil 播放, vidmutil 默认是 udp 方式, 通过设置环境变量使用 tcp 方式。

```
export OPENCV_FFMPEG_CAPTURE_OPTIONS="rtsp_transport;tcp|buffer_size;
↪1024000|max_delay;50000"
sudo -E ./vidmulti 8
```

6.16 高密度服务器播放 rtsp 流出现断连情况验证

答: 可以使用 vlc 播放相同的视频, 在相同的时间下, 看 vlc 播放是否有断连的情况, 注意设置 vlc 的缓冲区大小。

6.17 验证当前 rtsp 服务输出的视频是否有花屏

答: 使用 vlc 播放视频, 持续一段时间, 看视频是否有花屏

6.18 查看 rtsp 服务是否实时推流

答: 通过 rtspserver 日志, 查看当前播放的文件是否正在发送。

6.19 对于 cvQueryFrame 等老的 opencv 接口支持状况

答:

有些客户采用旧版 opencv 的 C 接口, 接口列表如下

```
void cvReleaseCapture( CvCapture** pcapture )

IplImage* cvQueryFrame( CvCapture* capture )

int cvGrabFrame( CvCapture* capture )

IplImage\* cvRetrieveFrame( CvCapture* capture, int idx )

double cvGetCaptureProperty( CvCapture* capture, int id )

int cvSetCaptureProperty( CvCapture* capture, int id, double value )
```

(下页继续)

(续上页)

```
int cvGetCaptureDomain( CvCapture* capture)

CvCapture * cvCreateCameraCapture (int index)

CvCapture * cvCreateFileCaptureWithPreference (const char * filename, int apiPreference)

CvCapture * cvCreateFileCapture (const char * filename)
```

对于这些接口，大部分都是支持的，只有返回值是 iplImage 的接口无法支持，这是因为我们硬件底层的 ion 内存类型是保存在 MAT 的 uMatData 类型中的，而 iplIMage 类型没有 uMatData 数据结构。

因此对于客户目前使用 cvQueryFrame 接口的，建议客户基于 cap.read 接口封一个返回值为 Mat 数据的 C 函数接口，不要直接调用 opencv 老版的接口。

6.20 对于 VPP 硬件不支持的 YUV 格式转换，采取什么样的软件方式最快？

答：

建议采用内部增强版版本的 libyuv。

相比较原始版本，增加了许多 NEON64 优化过的格式转换 API 函数。其中包含许多 JPEG YCbCr 相关的函数。

位置：middleware-soc/bm_opencv/3rdparty/libyuv/

6.21 OpenCV 中的 BGR 格式，在 libyuv 中对应的那个格式？OpenCV 中的 RGB 格式呢？

答：

- OpenCV 中的 BGR 格式，在 libyuv 中对应的格式为 RGB24
- OpenCV 中的 RGB 格式，在 libyuv 中对应的格式为 RAW。

6.22 若是采用 libyuv 处理 JPEG 方面的输出或者输入，需要注意什么事项？

答:

若是处理 jpeg 方面的输出或者输入，需要使用 J400，J420，J422，J444 等字样的函数，不然输出结果会有色差。

原因是 JPEG 的格式转换矩阵跟视频的不一样。

6.23 高密度及小盒子支持 gb28181 协议，部署步骤如下

答:

1. 启动 sip 代理

   sip 代理部署在外部 x86 或者主控板上，sip 代理配置文件 [GB28181.cfg] 说明参考 24

   启动命令，

   ```
   ./ctrlCenter 192.168.193.62 8081
   ```

   ctrlCenter 为可执行程序。

   192.168.193.62 是本机 ip。

   8081 是本机端口号，给 web 访问用。

   启动成功打印日志:

   ```
   ~/sip_proxy/ctrlCenter/build$ ./ctrlCenter 192.168.193.62 8089
   10:13:36.572 critsec !Mutex created
   10:13:36.572 critsec !Mutex: thread thr0x7f6cb6bcc740 is waiting (mutex owner=)
   10:13:36.572 critsec Mutex acquired by thread thr0x7f6cb6bcc740 (level=1)
   10:13:36.572 critsec Mutex released by thread thr0x7f6cb6bcc740 (level=0)
   10:13:36.572 critsec Mutex: thread thr0x7f6cb6bcc740 is waiting (mutex owner=)
   10:13:36.572 critsec Mutex acquired by thread thr0x7f6cb6bcc740 (level=1)
   10:13:36.572 critsec Mutex released by thread thr0x7f6cb6bcc740 (level=0)
   10:13:36.572 os_core_unix.c pjlib 2.5.5 for POSIX initialized
   ip addr 192.168.193.62
   http start at port 8089
   ```

2. 相机或下及平台注册到 sip 代理

   具体注册方式参考下级平台或相机配置。

   设备注册成功打印日志:

   ```
   UserAgentRegistration 34020000001310101202
   -----------[UA 34020000001310101202 register] ---------
   new device id is 34020000001310101202
   ```

3. 获取前端设备列表

   http://192.168.193.62:8081/sipproxy?type=getdevicelist

   返回结果

```
{"devidelist": [{"id": "34010000001310000009"}{"id": "34010000001310000010"}{"id":
↪"34020000001310101202"}]}
```

   34010000001310000009 等为设备 20 位编码。

4. 高密度配置端口映射，若为小盒子跳过此步

   配置端口映射，播放一路国标流需要映射两个 UDP 端口，并且端口映射中 out_start_port 与 in_start_port 要相同。gb28181 地址中的【媒体接收端的视频流端口】需要做端口映射，并且需要做一个对应 rtcp 的端口映射【媒体接收端的视频流端口+1】。

   具体配置方法参考 [内部路由器配置使用说明]

5. 设备上获取国标视频流

   与打开 rtsp 流的方式相同

   · 方式一：可以在高密度核心板或小盒子上输入命令

   udp 实时流：

```
sudo /system/bin/vidmulti  1 gb28181://34020000002019000001:123456@35.26.240.99:5666?
↪deviceid=35018284001310090010#localid=12478792871163624979#localip=172.10.18.201
↪#localmediaport=20108
```

   udp 回放流：

```
sudo /system/bin/vidmulti  1 gb28181_playback://34020000002019000001:123456@35.26.240.
↪99:5666?deviceid=35018284001310090010#devicetype=3#localid=12478792871163624979
↪#localip=172.10.18.201#localmediaport=20108#begtime=20191018160000
↪#endtime=20191026163713
```

   tcp 实时流：

```
sudo /system/bin/vidmulti 1 gb28181://34020000002019000001:123456@35.26.240.99:5666?
↪deviceid=35018284001310090010#localid=12478792871163624979#localip=172.10.18.201
```

   tcp 回放流：

```
sudo /system/bin/vidmulti  1 gb28181_playback://34020000002019000001:123456@35.26.240.
↪99:5666?deviceid=35018284001310090010#devicetype=3#localid=12478792871163624979
↪#localip=172.10.18.201#begtime=20191018160000#endtime=20191026163713
```

   gb28181url 地址参数说明参考 21

   · 方式二：与 rtsp 地址使用用法的，需要获取国标流时，只需把 rtsp 地址换成 gb28181url。

6.24 配置文件 [GB28181.cfg] 说明

答:

```
#local info

localID=34020000002000000001      sip代理的20位编码
localRealm=34020000
localPasswd=123456                  sip代理的密码
localIp=192.168.193.62              sip代理的ip
localPort=5070                      sip代理端口号
```

6.25 ffmpeg&opencv 支持 gb28181 协议，传入的 url 地址形式如下

答:

udp 实时流地址

```
gb28181://34020000002019000001:123456@35.26.240.99:5666?deviceid=35018284001310090010
↪#localid=1247879287116362497 9#localip=172.10.18.201
↪#localmediaport=20108340200000020190000 01:123456@35.26.240.
↪99:5666：sip服务器国标编码:sip服务器的密码@sip服务器的ip地址:sip服务器的port
deviceid：前段设备20位编码
localid：本地20位编码，可选项
localip：本地ip，可选项.不设置会获取 eth0 的ip，如果没有eth0需要手动设置
localmediaport：媒体接收端的视频流端口，需要做端口映射，映射两个端口(rtp:11801,
↪rtcp:11802)，两个端口映射的in和out要相同.同一个核心板端口不可重复。
```

udp 回放流地址

```
gb28181_playback://34020000002019000001:123456@35.26.240.99:5666?deviceid=\
↪35018284001310090010#devicetype=3#localid=1247879287116362497 9#localip=172.10.18.201
↪#localmediaport=20108#begtime=20191018160000
↪#endtime=20191026163713340200000020190000 01:123456@35.26.240.
↪99:5666：sip服务器国标编码:sip服务器的密码@sip服务器的ip地址:sip服务器的port
deviceid：前段设备20位编码
devicetype：录像存储累类型
localid：本地20位编码，可选项.不设置会获取 eth0 的ip，如果没有eth0需要手动设置
localip：本地ip，可选项
localmediaport：媒体接收端的视频流端口，需要做端口映射，映射两个端口(rtp:11801,
↪rtcp:11802)，两个端口映射的in和out要相同.同一个核心板端口不可重复。
begtime：录像起始时间
endtime：录像结束时间
```

tcp 实时流地址

```
gb28181://34020000002019000001:123456@35.26.240.99:5666?deviceid=35018284001310090010
↪#localid=1247879287116362497 9#localip=172.10.18.2013402000000020190000 01:123456@35.26.240.
↪99:5666：sip服务器国标编码:sip服务器的密码@sip服务器的ip地址:sip服务器的port
```

(下页继续)

(续上页)

```
deviceid: 前段设备20位编码
localid: 本地20位编码，可选项
localip: 本地ip，是可选项.不设置会获取 eth0 的ip，如果没有eth0需要手动设置
```

tcp 回放流地址

```
gb28181_playback://34020000002019000001:123456@35.26.240.99:5666?
↪deviceid=35018284001310090010#devicetype=3#localid=1247879287116362497 9#localip=172.10.
↪18.201#begtime=20191018160000#endtime=20191026163713340200000002019000001:123456@35.
↪26.240.99:5666: sip服务器国标编码:sip服务器的密码@sip服务器的ip地址:sip服务器的port
deviceid: 前段设备20位编码
devicetype :录像存储累类型
localid :本地20位编码，可选项
localip :本地ip，可选项. 不设置会获取 eth0 的ip，如果没有eth0需要手动设置
begtime :录像起始时间
endtime :录像结束时间
```

注意:

1. 流媒体传输默认是 udp 方式，如果使用 tcp 方式获取实时流或回放流，需要显示的指定。

   Ffmpeg 指定 tcp 方式为接口调用通过 av_dict_set 设置 gb28181_transport_rtp 为 tcp。

   Opencv 指定方式是设置环境变量

   ```
   export OPENCV_FFMPEG_CAPTURE_OPTIONS="gb28181_transport_rtp;tcp"
   ```

2. 如果使用 udp 方式外部无法访问到内部 ip/port，localmediaport 需要做端口映射，端口映射需要两个 rtp 和 rtcp。

6.26 现在 opencv 中默认是使用 ION 内存作为 MAT 的 data 空间，如何指定 Mat 对象基于 system memory 内存去创建使用？

答:

```
using namespace cv;

Mat m; m.allocator = m.getDefaultAllocator();     // get system allocator
```

然后就可以正常调用各种 mat 函数了，如 m.create() m.copyto()，后面就会按照指定的 allocator 来分配内存了。

```
m.allocator = hal::getDefaultAllocator();  // get ion allocator
```

就又可以恢复使用 ION 内存分配器来分配内存。

6.27 FFMPEG JPEG example for encode and transcode.

答:

- **a jpeg encoder example for command line**

```
ffmpeg -c:v jpeg_bm -i src/5.jpg -c:v jpeg_bm  -is_dma_buffer 1 -y 5nx.jpg
```

- **mjpeg transcoder examples for command line**

```
ffmpeg -c:v jpeg_bm -num_extra_framebuffers 2 -i in_mjpeg.avi -c:v jpeg_bm -is_dma_
↪buffer 1 -y test_avi.mov

ffmpeg -c:v jpeg_bm -num_extra_framebuffers 2 -i in_mjpeg.mov -c:v jpeg_bm -is_dma_
↪buffer 1 -y test_mov.mov
```

6.28 How to read bitstream from input buffer in FFMPEG?

答:

There is an example in FFMPEG source code. bm_ffmpeg/doc/examples/avio_reading.c (or http://www.ffmpeg.org/doxygen/trunk/doc_2examples_2avio_reading_8c-example.html)

This case makes libavformat demuxer access media content through a **custom AVIOContext read callback** instread of file, rtsp, etc protocols defined in FFMPEG.

This is an example in middleware-soc about using avio + jpeg_bm to decode still jpeg picture. (bm_ffmpeg/doc/examples/avio_decode_jpeg.c)

6.29 从内存读取图片，用 AVIOContext *avio =avio_alloc_context()，以及 avformat_open_input() 来初始化，发现初始化时间有 290ms；但是如果从本地读取图片，只有 3ms。为啥初始化时间要这么长？怎样减少初始化时间？

答:

```
ret = avformat_open_input(&fmt_ctx, NULL, NULL, NULL);
```

这里是最简单的调用。因此，avformat 内部会会读取数据，并遍历所有的数据，来确认 avio 中的数据格式。

若是避免在这个函数中读取数据、避免做这种匹配工作。在已经知道需要使用的 demuxer 的前提下，譬如，已知 jpeg 的 demuxer 是 mjpeg，可将代码改成下面的试试。

```
AVInputFormat\* input_format = av_find_input_format("mjpeg");

ret = avformat_open_input(&fmt_ctx, NULL, input_format, NULL);
```

6.30 how to know the jpeg demuxer name supported in FFMPEG?

答:

```
root@linaro-developer:~# ffmpeg -demuxers | grep jpeg
D jpeg_pipe piped jpeg sequence
D jpegls_pipe piped jpegls sequence
D mjpeg raw MJPEG video
D mjpeg_2000 raw MJPEG 2000 video
D mpjpeg MIME multipart JPEG
D smjpeg Loki SDL MJPEG
```

6.31 how to know all bm hardware decoder names supported in FFMPEG?

答:

```
root@linaro-developer:/home/sophgo/test# ffmpeg -decoders | grep _bm
V·····. avs_bm bm AVS decoder wrapper (codec avs)
V·····. cavs_bm bm CAVS decoder wrapper (codec cavs)
V·····. flv1_bm bm FLV1 decoder wrapper (codec flv1)
V·····. h263_bm bm H.263 decoder wrapper (codec h263)
V·····. h264_bm bm H.264 decoder wrapper (codec h264)
V·····. hevc_bm bm HEVC decoder wrapper (codec hevc)
V·····. jpeg_bm BM JPEG DECODER (codec mjpeg)
V·····. mpeg1_bm bm MPEG1 decoder wrapper (codec mpeg1video)
V·····. mpeg2_bm bm MPEG2 decoder wrapper (codec mpeg2video)
V·····. mpeg4_bm bm MPEG4 decoder wrapper (codec mpeg4)
V·····. mpeg4v3_bm bm MPEG4V3 decoder wrapper (codec msmpeg4v3)
V·····. vc1_bm bm VC1 decoder wrapper (codec vc1)
V·····. vp3_bm bm VP3 decoder wrapper (codec vp3)
V·····. vp8_bm bm VP8 decoder wrapper (codec vp8)
V·····. wmv1_bm bm WMV1 decoder wrapper (codec wmv1)
V·····. wmv2_bm bm WMV2 decoder wrapper (codec wmv2)
V·····. wmv3_bm bm WMV3 decoder wrapper (codec wmv3)
```

6.32 How to show the decoder info, for example, jpeg_bm, in FFMPEG?

答:

root@linaro-developer:/home/sophgo/test# ffmpeg -h decoder=jpeg_bm
Decoder jpeg_bm [BM JPEG DECODER]:
General capabilities: avoidprobe
Threading capabilities: none
jpeg_bm_decoder AVOptions:
-bs_buffer_size <int> .D.V····· the bitstream buffer size (Kbytes) for bm jpeg decoder (from 0 to INT_MAX) (default 5120)
-chroma_interleave <flags> .D.V····· chroma interleave of output frame for bm jpeg decoder (default 0)
-num_extra_framebuffers <int> .D.V····· the number of extra frame buffer for jpeg decoder (0 for still jpeg, at least 2 for motion jpeg) (from 0 to INT_MAX) (default 0)

6.33 How to show the encoder info, for example, jpeg_bm, in FFMPEG?

答:

root@linaro-developer:/home/sophgo/test# ffmpeg -h encoder=jpeg_bm
Encoder jpeg_bm [BM JPEG ENCODER]:
General capabilities: none
Threading capabilities: none
Supported pixel formats: yuvj420p yuvj422p yuvj444p
jpeg_bm_encoder AVOptions:
-is_dma_buffer <flags> E..V····· flag to indicate if the input frame buffer is DMA buffer (default 0)

6.34 A jpeg encoder example for calling api function

答:

```
AVDictionary* dict = NULL;

av_dict_set_int(&dict, "is_dma_buffer", 1, 0);// this means the input frame for encoder is as␣
↪that in note 2.

ret = avcodec_open2(pCodecContext, pCodec, &dict);
```

6.35 example to set decoder jpeg_bm when calling ffmpeg api for decoding still jpeg picture

答:

```
AVDictionary* dict = NULL;

/* The output of bm jpeg decoder is chroma-interleaved,for example, NV12 */

av_dict_set_int(&dict, "chroma_interleave", 1, 0);

/* The bitstream buffer is 3100KB(less than 1920x1080x3),
/* assuming the max dimension is 1920x1080 */
av_dict_set_int(&dict, "bs_buffer_size", 3100, 0);
/* Extra frame buffers: 0 for still jpeg, at least 2 for mjpeg */
av_dict_set_int(&dict, "num_extra_framebuffers", 0, 0);

ret = avcodec_open2(pCodecContext, pCodec, &dict);
```

6.36 example to set decoder jpeg_bm when calling ffmpeg api for decoding motion jpeg picture

答:

```
AVDictionary* dict = NULL;

/* The output of bm jpeg decoder is chroma-separated, for example, YUVJ420 */

av_dict_set_int(&dict, "chroma_interleave", 0, 0);

/* The bitstream buffer is 3100KB,
/* assuming the max dimension is 1920x1080 */
av_dict_set_int(&dict, "bs_buffer_size", 3100, 0);
/* Extra frame buffers: 0 for still jpeg, at least 2 for mjpeg */
av_dict_set_int(&dict, "num_extra_framebuffers", 2, 0);

ret = avcodec_open2(pCodecContext, pCodec, &dict);
```

6.37 BM1684 解码性能对于 H264/H265 有差别吗？如果调整码率的话，最多可以解多少路呢？有没有对应的数据参考？

答:

264,265 是解码路数相同的。

码率对解码帧率会有影响，这个变化就需要实测，我们说的 BM1684 解码能达到 960fps 是针对监控码流而言的，这类监控码流没有 B 帧，场景波动较小，码率基本在 2~4Mbps。如果是电影或者其他码率很高的，比如 10Mbps，20Mbps 甚至更多，是会有明显影响的，具体多大这个需要实测。

分辨率对于解码帧率的影响，是可以按照比例来换算的。我们说的 960fps 是针对 1920x1080 HD 分辨率而言的。

6.38 是否可以通过抽帧来提高 BM1684 的解码路数

答:

我们 opencv 中提供的抽帧，是在解码出来的结果中做的，并不是只解 I/P 帧的抽帧概念。这里的抽帧解码主要是保证出来帧数的均匀，使得后续的分析处理是等间隔的进行，这是为后续模型分析比较复杂的时候，不能达到每帧都检测而设计的解决方案，但并不能达到增加解码路数的效果。

这里顺便解释下，为什么不提供只解 I/P 帧的抽帧功能。如果只解 I、P 帧的话，抽帧的间隔就完全取决于码流的编码结构，这样是比较难控制住性能，比如监控码流中的没有 B 帧，那其实就相当于没有抽帧了。如果客户可以控制编码端，那更切合实际的做法是直接降低编码端的编码帧率，比如降到 15fps，那样解码路数就可以直接提升；反之，如果客户没有办法控制编码端，那么同样的，只解 IP 帧的抽帧方式就也无法达到增加解码路数的效果。

6.39 是否支持 avi，f4v，mov，3gp，mp4，ts，asf，flv，mkv 封装格式的 H264/H265 视频解析？

答：我们使用 ffmpeg 对这些封装格式进行支持，ffmpeg 支持的我们也支持。经查，这些封装格式 ffmpeg 都是支持的。但是封装格式对于 H265/264 的支持，取决于该封装格式的标准定义，比如 flv 标准中对 h265 就没有支持，目前国内的都是自己扩展的。

6.40 是否支持 png, jpg, bmp, jpeg 等图像格式

答: Jpg/jpeg 格式除了有 jpeg2000 外, 自身标准还有很多档次, 我们采用软硬结合的方式对其进行支持。对 jpeg baseline 的除了极少部分外, 都用硬件加速支持, 其他的 jpeg/jpg/bmp/png 采用软件加速的方式进行支持。主要的接口有 opencv/ffmpeg 库。

6.41 Valgrind 内存检查为什么有那么多警告，影响到应用的调试了

答:

我们的版本发布每次都会用 valgrind 检查一遍内存泄漏问题，如果有内存泄露问题我们会检查修正的。之所以没有去掉有些警告，是因为这些警告大部分都是内存没有初始化，如果对这些内存每个都加上初始化，会明显导致速度性能下降，而且我们确认后续操作是在硬件对其赋值后再进行的，对于此类警告，我们就不会去消除。

为了避免警告过多对上层应用调试造成影响，建议使用 valgrind 的 suppression 功能，可以通过过滤配置文件，来避免我们模块产生的 valgrind 警告，从而方便上层应用调试自身的程序。

6.42 使用 opencv 的 video write 编码，提示物理内存 (heap2) 分配失败

答:

确认 heap2 设置的大小，如果 heap2 默认大小是几十 MB，需要设置 heap2 size 为 1G。目前出厂默认配置是 1G。

```
Update_boot_info 可查询heap2 size

update_boot_info -heap2_size=0x40000000 -dev=0x0 设置heap2 size为1G。设置后重装驱动。
```

6.43 Bm_opencv 的 imread jpeg 解码结果和原生 opencv 的 imread jpeg 结果不同，有误差

答: 是的。原生 opencv 使用 libjpeg-turbo, 而 bm_opencv 采用了 bm168x 芯片中的 jpeg 硬件加速单元进行解码。

误差主要来自解码输出 YUV 转换到 BGR 的过程中。1）YUV 需要上采样到 YUV444 才能进行 BGR 转换。这个 upsample 的做法没有标准强制统一，jpeg-turbo 提供了默认 Fancy upsample，也提供了快速复制上采样的算法，原生 opencv 采用默认的 fancy upsample；而 BM168x 硬件加速单元采用快速复制的算法。2）YUV444 到 BGR 的转换是浮点运算，浮点系数精度的不同会有 +/-1 的误差。其中 1）是误差的主要来源。

这个误差并不是错误，而是双方采用了不同的 upsample 算法所导致的。

6.44 如何查看 vpu/jpu 的内存、使用率等状态

答:

在 pcie 模式下，可以用下面的方法查看:

```
cat /proc/bmsophon/card0/bmsophon0/media

cat /proc/bmsophon/card0/bmsophon0/jpu
```

在 soc 模式下，可以用下面的方法查看:

```
cat /proc/vpuinfo

cat /proc/jpuinfo
```

6.45 视频支持 32 路甚至更多的时候，报视频内存不够使用，如何优化内存使用空间

答:

在 PCIE 板卡下，视频内存有 3G，一般来说支持 32 路甚至更多的路数都绰绰有余。但在 soc 模式下，视频内存的默认配置是 2G，正常使用在 16 路是绰绰有余的，但在 32 路视频需要在应用层面上仔细设计，不能有任何的浪费。

如果解码使用的是 FFMPEG 框架，首先保证视频输出格式使用压缩格式，即 output_format 101。Opencv 框架的话，内部已经默认使用压缩格式了;

其次如果应用在获取到解码输出 avFrame 后，并不是直接压入队列，而是转换到 RGB 或者非压缩数据后再缓存的话，可以用 av_dict_set extra_frame_buffer_num 为 1（默认为 5）。Opencv 内部在最新版本中会默认优化。

最后，如果以上优化过后，仍然不够的话，在 soc 模式下可以考虑更改 dtb 设置，给视频挪用分配更多的内存，当然相应的，其他模块就要相应的减少内存。这个要从系统角度去调配。

6.46 Opencv 中 mat 是如何分配设备内存和系统内存的？

答:

因为受设计影响，这个问题细节比较多，主要从三方面能解释。

1. 在 soc 模式下，设备内存和系统内存是同一份物理内存，通过操作系统的 ION 内存进行管理，系统内存是 ION 内存的 mmap 获得。在 pcie 模式下，设备内存和系统内存是两份物理内存，设备内存指 BM168x 卡上的内存，系统内存指服务器上操作系统内存。如果用户想只开辟系统内存，和开源 opencv 保持一致，可以参见 FAQ26 的回答。

2. 在 sophon opencv 中默认会同时开辟设备内存和系统内存，其中系统内存放在 mat.u->data 或 mat.data 中，设备内存放在 mat.u->addr 中。只有以下几种情况会不开辟设备内存，而仅提供系统内存：
   - 当 data 由外部开辟并提供给 mat 的时候。即用以下方式声明的时候：

     ```
     Mat mat(h, w, type, data); 或 mat.create(h, w, type, data);
     ```

   - 在 soc 模式下，当 type 不属于（CV_8UC3, CV_32FC1, CV_32FC3）其中之一的时候。这里特别注意 CV_8UC1 是不开辟的，这是为了保证我们的 opencv 能够通过开源 opencv 的 opencv_test_core 的一致性验证检查。
   - 当宽或者高小于 16 的时候。因为这类宽高，硬件不支持
3. 在 BM1682 和 BM1684 的 SOC 模式下，mat 分配的 CV_8UC3 类型的设备内存会自动做 64 对齐，即分配的内存大小一定是 64 对齐的（注意：仅对 soc 模式的 CV_8UC3 而言，且仅对 BM1684/BM1682 芯片）。在 PCIE 模式下，分配的内存是 byte 对齐的。

6.47 ffmpeg 中做图像格式/大小变换导致视频播放时回退或者顺序不对的情况处理办法

答：ffmpeg 在编码的时候底层维护了一个 avframe 的队列作为编码器的输入源，编码期间应保证队列中数据有效，如果在解码后需要缩放或者像素格式转换时候需要注意送进编码器的 avframe 的数据有效和释放问题。

在例子 ff_bmcv_transcode 中从解码输出 src-avframe 转换成 src-bm_image 然后做像素格式转换或者缩放为 dst-bm_image 最后转回 dst-avframe 去编码的过程中 src-avframe、src-bm_image 的设备内存是同一块，dst- avframe、dst-bm_image 的设备内存是同一块。在得到 dst-bm_image 后即可释放 src_avframe 和 src-bm_image 的内存（二者释放其中一个即可释放设备内存），作为编码器的输入 dts-bm_image 在转换成 dst- avframe 之后其设备内存依然不能被释放（常见的异常情况是函数结束 dts-bm_image 的引用计数为 0 导致其被释放），如果 dst-bm_image 被释放了此时用 dst-avframe 去编码结果肯定会有问题。

解决方法是 dst-bm_image 的指针是 malloc 一块内存，然后将其传给 av_buffer_create，这样就保证在函数结束的时候 dst-bm_image 引用计数不会减 1，释放的方法是将 malloc 的 dst-bm_image 指针通过 av_buffer_create 传给释放的回调函数，当 dst- avframe 引用计数为 0 的时候会调用回调函数将 malloc 的指针和 dst-bm_image 的设备内存一起释放。详见例子 ff_bmcv_transcode/ff_avframe_convert.cpp。

6.48 启动设备首次执行某个函数慢，重启进程再次运行正常

现象：设备上电后第一次执行程序，函数处理时间长，再次执行程序，运行正常。

解决：先做个验证，如果不重启可复现，就说明是文件 cache 导致的变慢。

1. 上电后第一次执行慢，第二次执行正常，之后进入 root 用户
2. 清除 cache echo 3 > /proc/sys/vm/drop_caches
3. 再次执行程序，运行慢，即可确定是 cache 导致的。

6.49 Opencv mat 创建失败，提示"terminate called after throwing an instance of 'cv::Exception' what(): OpenCV(4.1.0) ⋯⋯matrix.cpp:452: error: (-215:Assertion failed) u != 0 in function 'creat'"

答：

这种错误主要是设备内存分配失败。失败的原因有两种：

1. 句柄数超过系统限制，原因有可能是因为句柄泄漏，或者系统句柄数设置过小，可以用如下方法确认：

   查看系统定义的最大句柄数：

   ```
   ulimit -n
   ```

   查看当前进程所使用的句柄数：

   ```
   lsof -n|awk '{print $2}'|sort|uniq -c|sort -nr|more
   ```

2. 设备内存不够用。可以用如下方法确认：
   - SOC 模式下

     ```
     cat /sys/kernel/debug/ion/bm_vpp_heap_dump/summary
     ```

   - PCIE 模式下，bm-smi 工具可以查看设备内存空间

解决方案：在排除代码本身的内存泄漏或者句柄泄漏问题后，可以通过加大系统最大句柄数来解决句柄的限制问题：ulimit -HSn 65536

设备内存不够就需要通过优化程序来减少对设备内存的占用，或者通过修改 dts 文件中的内存布局来增加对应的设备内存。详细可以参考 SM5 用户手册中的说明。

6.50 opencv 转 bm_image 的时候，报错 "Memory allocated by user, no device memory assigned. Not support BMCV!"

答：这种错误通常发生在 soc 模式下，所转换的 Mat 只分配了系统内存，没有分配设备内存，而 bm_image 要求必须有设备内存，因此转换失败。而在 pcie 模式下，接口内部会自动分配设备内存，因此不会报这个错误。

会产生这类问题的 Mat 通常是由外部分配的 data 内存 attach 过去的，即调用 Mat(h, w, data) 或者 Mat.create(h,w, data) 来创建的，而 data!=NULL, 由外部分配。

对于这种情况，因为 bm_image 必须要求设备内存，因此解决方案有

1. 新生成个 Mat，默认创建设备内存，然后用 copyTo() 拷贝一次，把数据移到设备内存上，再重新用这个 Mat 来转成 bm_image
2. 直接创建 bm_image，然后用 bm_image_copy_host_to_device, 将 Mat.data 中的数据拷贝到 bm_image 的设备内存中。

6.51 Opencv 用已有 Mat 的内存 data，宽高去创建新的 Mat 后，新 Mat 保存的图像数据错行，显示不正常

答：保存的图像错行，通常是由于 Mat 中 step 信息丢失所造成。

一般用已有 Mat 去生成一个新 Mat，并且要求内存复用时，可以直接赋值给新的 Mat 来简单实现，如 Mat1 = Mat2.

但在某些情况下，比如有些客户受限于架构，函数参数只能用 C 风格的指针传递，就只能用 Mat 中的 data 指针，rows，cols 成员来重新恢复这个 Mat。这时候就需要注意 step 变量的设置，在默认情况下是 AUTO_STEP 配置，即每行数据没有填充数据。但是在很多种情况下，经过 opencv 处理后，会导致每行出现填充数据。如，

1. soc 模式下，我们的 Mat 考虑执行效率，在创建 Mat 内存时每行数据会做 64 字节对齐，以适配硬件加速的需求（仅在 soc 模式下）
2. opencv 的固有操作，如这个 Mat 是另一个 Mat 的子矩阵（即 rect 的选定区域），或者其他可能导致填充的操作。

因此，按照 opencv 定义，通用处理方式就是在生成新的 Mat 的时候必须指定 step，如下所示:

```
cv::Mat image_mat = cv::imread(filename,IMREAD_COLOR,0);
cv::Mat image_mat_temp(image_mat.rows,image_mat.cols,CV_8UC3,image_mat.data,image_
↪mat.step[0]);
cv::imwrite("sophgo1.jpg",image_mat_temp);
```

6.52 在 soc 模式下客户用 ffmpeg 解码时拿到 AVframe 将 data[0-3] copy 到系统内存发现 copy 时间是在 20ms 左右而相同数据量在系统内存两块地址 copy 只需要 1-3ms

答：上述问题的原因是系统在 ffmpeg 中默认是禁止 cache 的，因此用 cpu copy 性能很低，使能 cache 就能达到系统内存互相 copy 同样的速度。可以用以下接口使能 cache.

```
av_dict_set_int(&opts, "enable_cache", 1, 0);
```

但是这样直接 copy 数据保存是非常占用内存、带宽和 cpu 算力的，我们推荐采用零拷贝的方式来实现原始解码数据的保存：

1. 推荐使用 extra_frame_buffer_num 参数指定增大硬件帧缓存数量，可以根据自己的需要选择缓存帧的数量。这个方式的弊端，一个是占用解码器内存，可能减少视频解码的路数；另一个是当不及时释放，当缓存帧全部用完时，会造成视频硬件解码堵塞。

   ```
   av_dict_set_int(&opts, "extra_frame_buffer_num", extra_frame_buffer_num, 0);
   ```

2. 推荐使用 output_format 参数设置解码器输出压缩格式数据，然后使用 vpp 处理输出非压缩 yuv 数据（当需要缩放，crop 时，可以同步完成），然后直接零拷贝引用非压缩 yuv 数据。这种方式不会影响到硬件解码性能，并且可以缓存的数据空间也大很多。

   ```
   av_dict_set_int(&opts, "output_format", 101, 0);
   ```

6.53 在 opencv VideoCapture 解码视频时提示：maybe grab ends normally, retry count = 513

上述问题是因为在 VideoCapture 存在超时检测，如果在一定时间未收到有效的 packet 则会输出以上 log，此时如果视频源是网络码流可以用 vlc 拉流验证码流是否正常，如果是文件一般是文件播放到末尾需调用 VidoeCapture.release 后重新 VideoCapture.open

6.54 [问题分析] 客户反馈碰到如下错误提示信息"VPU_DecRegisterFrameBuffer failed Error code is 0x3"，然后提示 Allocate Frame Buffer 内存失败。

这个提示信息表示：分配的解码器缓存帧个数，超过了最大允许的解码帧。导致这个问题的原因有可能是：

1. 不支持的视频编码格式，比如场格式，此时可以用 FAQ14 的方法，把码流数据录下来，提交给我们分析。
2. 设置了过大的 extra_frame_buffer_num。理论上，extra_frame_buffer_num 不能超过 15，超过了以后就有可能不能满足标准所需的最大缓存帧数。因为

大部分编码码流并没有用到最大值，所以 extra_frame_buffer_num 大于 15 的时候，对大部分码流仍然是可以工作的。

目前发现可能导致这个问题的原因有上述两种，后续有新的案例继续增补

6.55 SOC 模式下，opencv 在使用 8UC1 Mat 的时候报错，而当 Mat 格式为 8UC3 的时候，同样的程序完全工作正常。

这个问题碰到的客户比较多，这次专门设立一个 FAQ 以便搜索。其核心内容在 FAQ46 “Opencv 中 mat 是如何分配设备内存和系统内存的”有过介绍，可以继续参考 FAQ46

在 soc 模式下，默认创建的 8UC1 Mat 是不分配设备内存的。因此当需要用到硬件加速的时候，比如推理，bmcv 操作等，就会导致各种内存异常错误。

解决方案可以参看 FAQ26 “如何指定 Mat 对象基于 system memory 内存去创建使用”，指定 8UC1 Mat 在创建的时候，内部使用 ion 分配器去分配内存。如下所示。

```
cv::Mat gray_mat;
gray_mat.allocator = hal::getAllocator();
gray_mat.create(h, w, CV_8UC1);
```

6.56 调用 bmcv_image_vpp_convert_padding 接口时，报缩放比例超过 32 倍的错：“vpp not support: scaling ratio greater than 32”。

bm1684 的 vpp 中硬件限制图片的缩放不能超过 32 倍。即应满足 dst_crop_w <= src_crop_w * 32，src_crop_w <= dst_crop_w * 32，dst_crop_h <= src_crop_h * 32 , src_crop_h <= dst_crop_h * 32。

此问题原因可能是：

1. 输入 crop_rect 中的 crop_w，crop_h 与输出 padding_attr 中的 dst_crop_w ，dst_crop_h 缩放比例超过了 32 倍。
2. crop_rect，padding_attr 值的数量应与 output_num 的数量一致。

6.57 [问题分析] 程序提示 "VPU_DecGetOutputInfo decode fail framdIdx xxx error(0x00000000) reason(0x00400000), reasonExt(0x00000000)" 是可能什么问题，这里 reason 的具体数值可能不同

这个提示通常是由码流错误造成的，提示的含义是第 xxx 帧解码错误，错误原因为….。这里具体原因对于上层应用来说，不用关心，只需知道这是由码流错误导致的即可。

进一步分析，导致码流错误的原因通常可以分为两类，我们要有针对的进行处理。因为一旦频繁出现这种提示，说明解码出来的数据是不正确的，这时候有可能是各种马赛克或者图像花，对于后续的处理会造成各种异常情况，所以我们必须尽量减少这种情况的发生。

1. 网络情况导致的丢包。这时候可以用我们的测试程序 vidmulti 验证下，如果 vidmulti 没有解码错误，那么可以排除这种情况。如果确认网络丢包的话，要分辨下是否网络带宽本身就不够，如果本身带宽不够，那没有办法，只能降低视频码流的码率。如果带宽是够的，要检查下网线。当码流连接数超过 20 多路的时候，这时候有可能已经超出百兆了，这时网线也必须换到 CAT6，与千兆网相匹配
2. 解码性能达到上限造成丢包。这种情况发生在流媒体环境中，对于文件播放是不会发生的。这时也可以用我们的 vidmulti 跑一下，作为比较。如果 vidmulti 也发生错误，说明性能确实到了上限了，否则说明应用本身还有优化的空间。

6.58 [问题分析] 程序提示"coreIdx 0 InstIdx 0: VPU interrupt wait timeout"，这是怎么回事?

这个提示表示视频解码或者编码中断超时。这个提示只是警告，会再次尝试，因此只要没有连续出现就可以忽略。这种情况一般是由解码数据错误导致或者负荷过重产生的。例如在板卡情况下，由于板卡数据交换过于频繁，造成解码或者编码数据传输堵塞，使得中断超时。

6.59 采用 TCP 传输码流的时候如果码流服务器停止推流，ffmpeg 阻塞在 av_read_frame

这是因为超时时间过长导致的，可以用一下参数设置超时时间减短。

av_dict_set(&options, "stimeout", "1000000", 0);

6.60 [问题分析] 当用 ffmpeg jpeg_bm 解码超大 JPEG 图片的时候，有时候会报 "ERROR:DMA buffer size(5242880) is less than input data size(xxxxxxx)"，如何解决？

在用 FFMPEG 的 jpeg_bm 硬件解码器解码 JPEG 图片的时候，默认的输入 buffer 是 5120K。在拿到 JPEG 文件前提前分配好输入缓存，在 MJPG 文件解码时可以避免频繁地创建和销毁内存。当出现默认输入 buffer 大小比输入 jpeg 文件小的时候，可以通过下面的命令来调大输入缓存。

av_dict_set_int(&opts, "bs_buffer_size", 8192, 0); //注意：bs_buffer_size 是以 Kbyte 为单位的

6.61 调用 bmcv_image_vpp_basic 接口时，csc_type_t csc_type 和 csc_matrix_t* matrix 该如何填？

bmcv 中 vpp 在做 csc 色彩转换时，默认提供了 4 组 601 系数和 4 组 709 系数，如 csc_type_t 所示。

1. csc_type 可以填为 CSC_MAX_ENUM，matrix 填 NULL，会默认配置 601 YCbCr 与 RGB 互转系数。
2. csc_type 可以填 csc_type_t 中参数，如 YCbCr2RGB_BT709，matrix 填 NULL，会按照所选类型配置对应系数。
3. csc_type 可以填 CSC_USER_DEFINED_MATRIX，matrix 填自定义系数。会按照自定义系数配置。

csc_matrix_t 中系数参考如下公式：

Y = csc_coe00 * R + csc_coe01 * G + csc_coe02 * B + csc_add0;

U = csc_coe10 * R + csc_coe11 * G + csc_coe12 * B + csc_add1;

V = csc_coe20 * R + csc_coe21 * G + csc_coe22 * B + csc_add2;

由于 1684 vpp 精度为 10 位，整数处理。

csc_coe 与 csc_add 的计算方法为: csc_coe = round（浮点数 * 1024）后按整数取补码。

csc_coe 取低 13bit，即 csc_coe = csc_coe & 0x1fff，csc_add 取低 21bit，即 csc_add = csc_add & 0x1fffff。

举例如下：

floating-point coe matrix => fixed-point coe matrix

0.1826 0.6142 0.0620 16.0000 => 0x00bb 0x0275 0x003f 0x004000

6.62 [问题分析] 不同线程对同一个 bm_imag 调用 bm_image_destroy 时，程序崩溃。

bm_image_destroy(bm_image image) 接口设计时，采用了结构体做形参，内部释放了 image.image_private 指向的内存，但是对指针 image.image_private 的修改无法传到函数外，导致第二次调用时出现了野指针问题。

为了使客户代码对于 sdk 的兼容性达到最好，目前不对接口做修改。建议使用 bm_image_destroy（image）后将 image.image_private = NULL，避免多线程时引发野指针问题。

6.63 cv::Mat 如何转换为 bm_image ?

答:

1. 如果你使用 BM-OpenCV 创建 cv::Mat ，想把它变成 bm_image，那么你需要创建一个具备设备内存的 Mat，也就是使用含有 dev_id 的参数的构造函数（也就是创建 1 个我们改造过的 Mat），这样你才能把这个 Mat 通过 toBMI 转换为 bm_image，并且这个转换过程不会发生数据拷贝，bm_image 只会引用 Mat 的设备内存;
2. 如果你使用 data 指针构造没有设备内存的原始 cv::Mat，这个 cv::Mat 不能调用 toBMI 转换为 bm_image，如果你确实有一个原始 OpenCV 的 cv::Mat，要转换为 bm_image，那么你应当：创建一个有设备内存的 Mat，然后使用 Mat.copyTo 将 openCV 的 cv::Mat 的系统内存拷贝到有设备内存的 Mat 的系统内存空间，然后使用 cv::bmcv::uploadMat 将有设备内存的 Mat 的系统内存同步到其设备内存，然后再调用 toBMI 将其转换为 bm_image。

6.64 如何将 host 上的 bgr planar cv::Mat 变成 host 上的 BGR packed cv::Mat ?

答：首先，cv::Mat 支持的都是 packed 的数据，你的 BGR plannar 的数据，怎么会生成 cv::Mat，如果你真有一组 BGR plannar 的数据，那你不应该将它直接赋值给 cv::Mat，这样容易引起混乱，需要自己时刻谨记这个 cv::Mat 里的数据不是 packed。OpenCV 的标准用法，当你想从 BGR packed 得到 planar，应当使用 cv::split 分成 3 个 cv::Mat；当你想从 BGR planar 到 packed，那你应当有 3 个通道的 cv::Mat，然后使用 cv::Merge 合成 1 个 cv::Mat。但是如果你觉得 openCV 实现的 split 和 merge 不够高效，那么可以考虑使用 libyuv 中基于 ARM NEON 指令做过优化的 SplitRGB 和 MergeRGB。

6.65 使用 NV12 原始数据，创建 bm_image 的注意事项？

答：NV12 有 2 个 plane，stride 应为二维数据，调用 bm_image_create() 或者不指定 stride，内部自己计算；或者指定 stride。指定 stride 时，必须指定完整的 stride，注意 NV12 的 stride 应为二维数组，并指定 2 个 plane 的 stride。由于 nv12 是 2 个 plane，因此，如果要使用 bm_malloc_device_byte 和 bm_image_attach 的话，需要分别调用 2 次 bm_malloc_device_byte 分别为每个 plane 申请设备内存，这样会得到 2 个 bm_device_mem_t，把他们写成 1 个数组，然后再用这个数组来调用 bm_image_attach。因此，建议使用 bm_image_alloc_dev_mem 来申请内存，而不是 bm_malloc_device_byte。

6.66 是否可以提供 OpenCV contrib 库？

答：OpenCV contrib 库中包含一些有专利的算法、以及尚不稳定的新算法等，我们正式发布的 SDK 只有常用的那部分，contrib 部分如有需要只能单独提供。

1. 可在外发 FTP 下载：

   ftp://QA@172.28.141.75/opencv_contrib

2. OpenCV4.1 contrib pcie 版本：

   链接：https://pan.baidu.com/s/1Jy_uG9RqHz8yqdVgcoI_xg
   提取码：kih3

6.67 BMCV 相关接口，输入和输出可以使同一个 bm_image 吗？

答：不可以。bmcv_image_resize, bmcv_image_convert_to, bmcv_image_vpp_convert 等这些接口，input 和 output 的 bm_image 应当是不同的内存地址对象，不支持 in-place 操作。

6.68 cv::bmcv::resize 看代码底层调用的是 bmcv_image_resize，cv::resize 用的是 cpu 吗，处理的是 mat 中 cpu 内存中的那部分数据吗？还有 1 个 cv::hal::resize, bmcv::hwResize，这些是什么关系啊。cv::bmcv::resize 和 cv::resize 是否支持输入和输出是同一个对象，原地进行转换呢？

答：

cv::resize 在 soc 下使用 bmcv::hwResize 进行加入，如果加速失败，继续走软件 resize; 在 pcie 下使用软件 resize。
cv::hal::resize 是 opencv 的底层实现，会尝试使用 opencl。
cv::bmcv::resize 不能 in-place 操作。
cv::resize 如果走标准 opencv，可以 in-place 操作。

6.69 关于 BM-OpenCV 中 GB28181 接口，说的是接国标流是吧？本身支持转国标流功能吗？

答：不支持，目前 BM-OpenCV 和 BM-OpenCV 支持 GB28181 国标流的解析，具体使用方案请查看《多媒体用户开发手册 》。

6.70 如何进行编解码性能测试？是否有参考程序？

答：请参考 SDK 目录下 examples/multimedia/ocv_vidmulti 目录。可以进行编解码性能测试。如果是 SoC 模式，vidmulti 程序已经编译好放置在/system/bin/下；PCIE 模式的话请自行编译测试。

6.71 BM1684 芯片的编解码性能数据是怎样的？

答：BM1684 有 4 个 VPU 硬核和 4 个 JPU 硬核，具体视频和图片的编解码的速度与实际情况有关，要以实测为准。

视频解码的速度与输入视频码流的格式有很大关系，不同复杂度的码流的解码速度有比较大的波动，比如码率、GOP 结构，分辨率等，都会影响到具体的测试结果。一般来说，针对视频监控应用场景，BM1684 产品单芯片可以支持到 32 路 HD 高清实时解码。码率和 GOP 结构对解码速度的影响因具体情况而异，需要实测；分辨率对于解码帧率的影响，可以按照比例来换算。

视频编码的速度与编码的配置参数有很大关系，不同的编码配置下，即使相同的视频内容，编码速度也不是完全相同的。一般来说，BM1684 产品单芯片最高可以支持到 2 路 HD 高清实时编码。

视频解码	H.264/H.265:1080P @960fps
视频解码格式	CIF / D1 / 720P / 1080P / 4K(3840 × 2160) / 8K(8192 × 4096) / 8192x8192
视频编码	1080P@50fps
视频编码格式	CIF / D1 / 720P / 1080P / 4K(3840 × 2160)
图片编解码	· SoC: 600 fps (yuv420@1080p) · PCIe: 700 fps (yuv420@1080p)
图片编解码分辨率	最大分辨率 32768 x 32768

6.72 BM1684 编解码性能是同时支持 32 路解码和 2 路编码吗？内存大小和内存带宽会不会成为瓶颈？

答：编解码是不同的硬件单元，可以同时支持。按照目前的内存布局，32+2 应该就在内存带宽的临界位置。

在 PCIe 板卡下，视频内存有 3G，一般来说支持 32 路甚至更多的路数都绰绰有余。

但在 SoC 模式下，视频内存的默认配置是 2G（另外的给 VPP 使用），正常使用的话支持 16 路是绰绰有余的，但要支持 32 路视频解码时需要在应用层面上仔细设计，不能有任何的浪费。

6.73 解码会占用多少内存？使用 vpp 进行图像处理，最大可能会消耗多少内存？

答：解码过程的内存占用与码流格式、压缩率、码率有关，需以实测为准。

vpp 模块本身是没有内存积累的，内存就是一张输入图像和一张输出图像空间。

6.74 前处理时图片数据格式转换需要 HWC 转 CHW 和 NCHW，1684 是否相关接口可以使用？

答：BMCV 中提供了针对图像的处理接口，可以实现图像数据的 NHWC 与 NCHW 以及通道顺序的转换，也就是 packed 和 planar、BGR 和 RGB 的转换，但没有针对 tensor 的通用的 NHWC 和 NCHW 的转换。但应该可以满足模型前处理时 ide 图片数据格式转换需求。请查看：bmcv_image_storage_convert，bmcv_image_vpp_basic，bmcv_image_vpp_convert，bmcv_image_vpp_convert_padding、bmcv_image_convert_to 等接口。

6.75 硬编的画面输出是绿屏

答：h264_bm/265_bm 硬编后，画面是绿屏。本质是编码前的原始数据没加载到。要考虑原始数据在设备内存，还是系统内存中。一般如果原始数据在主机内存的要设置 is_dma_buffer=0；反之，在设备内存时，设置 is_dma_buffer=1。此参数默认为 1。

6.76 解码器卡住

答：如果解码器正常运行一段时间后，卡住了，可能是由于当前程序里的解码帧数量达到了限制。这个值可以通过 extra_frame_buffer_num 设置，默认为 5。所以在处理完 AVFrame 后，需要及时释放，防止占用过多解码帧。

6.77 使用 bmcv_image_vpp_resize_padding 时报错提示" vpp input image param err" ?

答：bmcv 库中带有 vpp 前缀的接口与是使用硬件 VPP 实现的，会对输入输出图像格式有一定的要求，比如缩放比例不能超过 32 倍等。具体请查看《BMCV 用户手册》。

6.78 bmcv 库中是否有和 OpenCV 相对应的 cvtColor、subtract、bitwise_and、findContours 等这几个方法 ?

答：cvtColor 请参见 bmcv_image_storage_convert，substract 请参见 bmcv_image_absdiff 和 bmcv_image_add_weighted，bitwise_and 和 findContours 无 BMCV 实现。

6.79 SoC 模式使用 cv::Mat 的数据地址初始化另外一个 cv::Mat 时有可能会出现乱码 ?

答：SoC 模式使用 cv::Mat 的数据地址初始化另外一个 cv::Mat 时需要指定 _step，即

```
cv::Mat image_temp_1(image_ost.rows,image_ost.cols,CV_8UC3,image_ost.data,image_ost.
↪step[0]);
```

FFF< HEAD 详细原因可参考 https://doc.sophgo.com/docs/docs_latest_release/multimedia_guide/html/guide/Multimedia_Guide_zh.html#opencv ======= 详细原因可参考 https://doc.sophgo.com/docs/3.0.0/docs_latest_release/multimedia_guide/html/guide/Multimedia_Guide_zh.html#opencv FFF> b77b821 (fix desc error & delete demo intro from sail-doc)

6.80 SoC 模式对 cv::Mat 的内存进行操作

答：SoC 模式下面，cv::Mat 的内存需要 64 字节对齐，所以 cv::Mat 的内存长度不是 width*3*height，而是 step*height，如果原始图像满足 64 字节对齐，那么 step=width*3

6.81 Sophon gate 人脸应用中 gate_webserver.py 的 sophonface 是如何导入的？

```
gate_webserver.py   subprocess.py   config.py   run_grab_server.py   run_ws_server.sh
import glv
glv.init()
glv.set_value("LOG_FILE", "/data/log/SophonSDK.log")
#memory_printer
#import tracemalloc
#tracemalloc.start()
from flask import Flask
import subprocess
from sophonface.web_server.server import get_api_blueprint, get_static_blueprint, get_default_ssl_contex
from sophonface import Mode, FDModel, FRModel
import threading
from ConfigDataSingletone import ConfigDataSingletone
from match_manager import MatchManager
from OEMconfig import OEMConfigSingletone
import os
#from set_mq_config_apis import SetMqConfig
from get_mq_msg_thread import MqClientService
#from upload_device_info_import DeviceInfoUploadAPI
from pretty_score import PrettyScore
import logger
from ImageToFile import EventImageSave
from DeviceManager import DeviceManager
from videortsp_control import VideoManagerAPI
```

答：sophonface 通过 PYTHONPATH 路径的方式直接指定，run_gateway.sh 中，在 PYTHONPATH 中添加了/system/SophonFaceSDK。

```
insmod /system/data/jpu.ko
insmod /system/data/vpu.ko
export LD_LIBRARY_PATH=/system/SophonFaceSDK/sophonface/video_detector:/system/lib:$LD_LIBRARY_PATH
export GLOG_minloglevel=6
export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
export OPENCV_FFMPEG_CAPTURE_OPTIONS="gb28181_transport_rtp;tcp|rtsp_transport;tcp|buffer_size;1024000|max_delay;500000|stimeout;20000000|analyzeduration;100|
probesize;5000"
export PYTHONPATH=$PYTHONPATH:/system/lib:/system/data/sophon_gate/SophonWelTV:/system/data/sophon_gate/SophonIPC:/system/data/sophon_gate/SophonCommon:/syste
m/data/sophon_gate/SophonGate/sophon_gate:/system/data/sophon_gate/SophonFaceCapCam:/system/data/sophon_gate/SophonFaceCapCam/capture_camera:/system/data/soph
on_gate/SophonFogSys:/system/data/sophon_gate/MqClient:/system/data/sophon_gate/Util:/system/data/sophon_gate/SophonDevice:/system/data/sophon_gate/SophonConf
ig:/system/data/sophon_gate/SophonFaceCapCam/capture_camera:/system/SophonFaceSDK:/system/SophonFaceSDK/sophonface/video_detector/
#PYTHONIOENCODING=utf-8 python3 sophon_gate/main_service.py --mode=9m16b2bmodel
ulimit -n 20480
count=$(ps -ef |grep SophonShellDaemon.py |grep -v "grep" |wc -l)

echo "start gate_webserver"
gunicorn -b 0.0.0.0:5556 -w 100 -k gthread --threads 25 --graceful-timeout 1 --error-logfile=/var/log/gunicorn.log gate_webserver:app
root@tzf:~# ls /system/SophonFaceSDK/
run_server.pyc  sophonface  tests
```

6.82 OpenCV 的 imread 接口读取进来的 JPG 图片尺寸问题

答：

软解 JPG 保持原尺寸。硬解 JPG 的话，会使用到 JPU，虽然 JPU 默认最大尺寸是 32768*32768，但 VPP 最大支持尺寸是 4096*4096。这里为满足后续 VPP 宽高同时小于 4K 的要求，会自动将图片宽高缩小到 4K 以下。例如，当前 w*h=5000*3000，则会同时将宽高除以 2，得到 2500*1500，此时宽高均满足小于 4K，则停止 resize。否则会继续宽高除以 2 的操作，最多会进行 3 次下采样 (32768 / 8 = 4096)。

6.83 如何将 bm_image 转为 cv.Mat?

答:

1. Python 中 cv.Mat 就是一个 numpy.array，请参考以下代码:

```
from sophon.sail import sail
import numpy as np

# engine = sail.Engine(model_path, device_id, io_mode)
# handle = engine.get_handle()

bm_image = sail.BMImage()

# 如果bm_image的format是FORMAT_BGR_PACKED或FORMAT_RGB_PACKED,
# 需要先转换成FORMAT_BGR_PLANAR或FORMAT_RGB_PLANAR,
# 再送入bm_image_to_tensor转Tensor，如
# image_bgr_planar = sail.BMImage(handle, bm_image.height(), bm_image.width(), sail.
↪Format.FORMAT_BGR_PLANAR, bm_image.dtype())
# sail.Bmcv(handle).convert_format(bm_image, image_bgr_planar)
# 转换到FORMAT_BGR_PLANAR，再将image_bgr_planar通过bm_image_to_
↪tensor转换成Tensor

result_tensor = sail.Bmcv(handle).bm_image_to_tensor(bm_image) #␣
↪传入handle，初始化sail.Bmcv
result_numpy = result_tensor.asnumpy()
np_array_temp = result_numpy[0]
np_array_t = np.transpose(np_array_temp, [1, 2, 0])
mat_array = np.array(np_array_t, dtype=np.uint8)
```

2. C++ 中使用 cv::bmcv::toMat() 接口。

6.84 rtsp 流使用 ffmpeg 和 opencv 可以正常解码，但是使用 sail.Decoder 无法正常解码

答：定位问题的原因是 extra_frame_buffer_num 给的太大。分别分析几个矛盾的现象如下：#. ffmpeg 下能够正常执行，而 python sail 下不能执行：这是因为 ffmpeg 下默认 extra_frame_buffer_num 为 5 #. 录下码流后，离线文件 python sail 可以工作，rtsp 下 python sail 不能工作：这是因为 python sail 只在 rtsp 下设置 extram_buffer_num 为 20 个

解码器内部每个 instance 最多允许 32 个 frame buffer，因为这个摄像头码流要求的缓存 frame buffer 比较多，加上 20 个 extra frame buffer 后，超过了 32 个，所以导致 register frame buffer 失败。

目前在 opencv 中配置 extra_frame_buffer_num 为 3，ffmpeg 默认为 5，这个配置基本不影响到速度，如果需要直接用解码器解出来的 frame buffer 做缓存，可以试着配置为 10，20 这个数字太大了。

sail 在 2.7.0(20220412_224617) 及以后的版本将 extram_buffer_num 设置为 5，并且添加其它了其它设置，保持和 opencv 一致

CHAPTER 7

程序优化常见问题

7.1 前后处理加速

- 编解码使用 bm_ffmpeg/bm_opencv
- 前处理中图片处理部分尽量采用 bmcv 来进行加速，优先使用带有 vpp 的接口
- 必要时候使用 libyuv，基于 ARM neon 加速 (SoC)
- nms 等后处理使用硬件加速接口

7.2 模型编译

- 精度满足的条件下，使用轻量级的模型
- 打开 Winograd 选项 (使能 Winograd 需要在 calibration_use_pb 和 bmnetu 命令中同时打开相应选项;Winograd 优化仅针对卷积核为 3x3 的卷积运算)

7.3 推理加速

- 推理的时候采用 4N batch 模式
- 使用 INT8 量化模型推理
- 对动态网络使能 cache

7.4 bmlang 开发

- 使用尽可能少的操作
- 让数据访问只发生在本地存储
  1. 尽量少使用全局操作；
  2. 集中且连续的使用本地操作来完成计算；
  3. 编程中相邻本地操作尽可能是生产者消费者关系。
- 科学切割数据，合理的 Tensor 数据排布将提升 TPU 计算单元使用率
- 使能数据切分优化（尽量设置 shape 不大于 4 维，若数据很大，在设置 shape 时优先将大数设置在 S2 维度，其次 S0 维度）

7.5 流程优化

- 优化模型流水线，多线程方式优化，VPU/JPU/VPP/CPU/TPU 的并行
- 减少不必要的内存拷贝
- 视频流抽帧处理，我们的 ffmpeg 支持只解关键帧的设置