3.4. Install SDK🕚

contents

Note

Development environment and operation environment:

Development environment refers to the environment used for development processes such as model transformation or validation and program compilation; The running environment refers to the running environment in which the Sophon device is used to deploy algorithm applications on a platform with Sophon devices. The development and run environments may be unified (e.g., x86 hosts with SC5 accelerator cards, which are both development and run environments), or they may be separate (e.g., x86 hosts as development environment transform models and compilers, and SE5 box deployments run the final algorithm applications).

However, whether you are using a product in SoC mode or PCIe mode, you will need an x86 host as your development environment, and your environment can be any system platform we have tested and supported.
In PCIe mode, you need to insert the PCIe accelerator card into the host. If you need to compile and quantify the model, you can configure the docker environment according to the instructions and compile and quantify the model in the docker environment. If you need to perform model verification, program compilation, and algorithm application deployment, install libsophon and sophon-mw in sequence as instructed. You can also add the library packages and other programming code you need to build your own production environment.

If it is SoC mode, you need to configure the docker environment on an x86 host according to the instructions, and conduct compilation and quantification of the model in the docker environment, or install libsophon and sophon-mw according to the instructions, and conduct cross-compilation of the program. After the program is compiled, You need to manually copy the compiled program to the target system (SE5/SM5) for execution.

Typical development environment:

- Linux development environment
    1. An x86 host with Ubuntu16.04/18.04/20.04 installed has at least 12GB of memory
    2. Install Docker: refer to Official Tutorial
    3. Download the SophonSDK development kit

3.4.1. Environment Configuration -Linux

We provide tpu-mlir and tpu-nntc environment for users to compile and quantify the model on x86 host, and libsophon environment for users to develop and deploy applications. PCIe users can compile, quantify and deploy the model based on tpu-nntc and libsophon. SoC users can compile and quantify the model and cross-compile programs based on tpu-mlir or tpu-nntc and libsophon on x86 hosts, and copy the compiled programs to the SoC platform (SE microserver /SM module) for execution during deployment. For configuration of relevant running environment, please refer to the section of Environment Configuration. You can also customize your own development environment based on the one we provide.

3.4.1.1. Decompress the SDK package

```
sudo apt-get install p7zip
sudo apt-get install p7zip-full
7z x Release_<date>-public.zip
cd Release_<date>-public
```

3.4.1.2. tpu-mlir environment initialization

1.Docker install

If docker is installed, skip this section.

```
# install docker
sudo apt-get install docker.io
# The docker command is executed without root permission
# Create a docker user group. If there is a docker group, an error will be reported
sudo groupadd docker
# Add the current user to the docker group
sudo gpasswd -a ${USER} docker
# Restart the docker service
sudo service docker restart
# Switch the current session to a new group or log in again to restart the X session
newgrp docker
```

2.Extract the compressed package to tpu-mlir

```
cd tpu-mlir_<date>_<hash>
mkdir tpu-mlir
tar zxvf tpu-mlir_v<x.y.z>-<hash>-<date>.tar.gz --strip-components=1 -C tpu-mlir
```

3.Create a docker container and go to Docker

```
cd tpu-mlir
#If there is no corresponding image in the current system, the image is automatically downloaded from the docker hub. Here, map the upper-level directory of tpu-mlir to the /workspace directory in docker. The mapping between ports 8001 and 8001 is used (port numbers are used when using the ufw visualization tool).
#If the port is occupied, replace it with another port that is not occupied.
docker run -v $PWD/..:/workspace -p 8001:8001 -it sophgo/tpuc_dev:latest
```

4.Initialize the software environment

```
cd /workspace/tpu-mlir
source scripts/envsetup.sh
./build.sh
```

3.4.1.3. tpu-nntc environment initialization

1. **Docker install**

If docker is installed, skip this section.

```
# install docker
sudo apt-get install docker.io
# The docker command is executed without root permission
# Create a docker user group. If there is a docker group, an error will be reported
sudo groupadd docker
# Add the current user to the docker group
sudo gpasswd -a ${USER} docker
# Restart the docker service
sudo service docker restart
# Switch the current session to a new group or log in again to restart the X session
newgrp docker
```

Note

Hint: You need to logout the system and then log back in again to use docker without the need for sudo.

2. **Extract the compressed package to tpu-nntc**

```
cd tpu-nntc_<date>_<hash>
mkdir tpu-nntc
tar zxvf tpu-nntc_v<x.y.z>-<hash>-<date>.tar.gz --strip-components=1 -C tpu-nntc
```

Note

Hint: <date> indicates the date, <x.y.z> indicates the version, and <hash> indicates the hash value. Unless otherwise specified, the following contents are subject to this rule.

3. **Create a docker container and go to Docker**

```
cd tpu-nntc
#If there is no corresponding image in the current system, the image is automatically downloaded from the docker hub. Here, map the upper-level directory of tpu-nntc to the /workspace directory in docker. The mapping between ports 8001 and 8001 is used (port numbers are used when using the ufw visualization tool).
#If the port is occupied, replace it with another port that is not occupied.
docker run -v $PWD/..:/workspace -p 8001:8001 -it sophgo/tpuc_dev:latest
```

4. **Initialize the software environment**

```
cd /workspace/tpu-nntc
source scripts/envsetup.sh
```

Attention

Note that if you re-enter after docker stop, you need to re-source the environment variables.

3.4.1.4. libsophon environment construction

Attention

libsophon offers different types of installation on different Linux distributions, so choose one based on your system and do not mix multiple installation methods on one machine.

1. If you have installed an old driver of the BM1684 SDK, uninstall the driver by performing the following steps:

```
# Go to the scripts folder in the SDK installation directory and run the following command
sudo ./remove_driver_pcie.sh
```

2. If you are using a Debian/Ubuntu system, the installation package consists of three files:sophon-driver_<x.y.z>_$arch.deb、sophon-libsophon_<x.y.z>_$arch.deb、sophon-libsophon-dev_<x.y.z>_$arch.deb

Note

In the preceding command, <x.y.z> indicates the version number and $arch indicates the hardware architecture of the current server. You can run the following command to obtain the arch of the current server. Generally, the hardware architecture of x86_64 machines is amd64, and that of arm64 machines is arm64. Select the installation file you want to install.

```
uname -m
```

You can perform the following steps to install it:

```
cd libsophon_<date>_<hash>
# Install the dependency library only once
sudo apt install dkms libncurses5
sudo dpkg -i sophon-*.deb
# You can run the following command on the terminal or log out and log in to the current user to run the bm-smi command
source /etc/profile
```

Attention

Check whether the driver is installed successfully:

Run `ls /dev/bm*` to see if there is /dev/bm-sohponx (X means 0-N). If there is, the installation is successful. Normally, the following information is displayed: /dev/bmdev-ctl /dev/bm-sophon0

3. If you use other Linux systems, the installation package consists of only one file: libsophon_<x.y.z>_$arch.tar.gz. You can install it by referring to Libsophon User Manual.

For other questions, please refer to Libsophon User Manual

3.4.1.5. sophon-mw environment construction

Make sure libsophon is installed when installing sophon-mw.

1. If you are using a Debian/Ubuntu system, the installation package consists of four files:sophon-mw-sophon-ffmpeg_<x.y.z>_$arch.deb、sophon-mw-sophon-ffmpeg-dev_<x.y.z>_$arch.deb、sophon-mw-sophon-opencv_<x.y.z>_$arch.deb、sophon-mw-sophon-opencv-dev_<x.y.z>_$arch.deb，请选择您对应的安装文件参考如下步骤进行安装：

```
cd sophon-mw_<date>_<hash>
# Must install sophon-mw-sophon-ffmpeg, then install sophon-mw-sophon-opencv
sudo dpkg -i sophon-mw-sophon-ffmpeg_<x.y.z>_*.deb sophon-mw-sophon-ffmpeg-dev_<x.y.z>_*.deb
sudo dpkg -i sophon-mw-sophon-opencv_<x.y.z>_*.deb sophon-mw-sophon-opencv-dev_<x.y.z>_*.deb
# To use the installation tool, run the following command on the terminal, or log out and log in to the current user
source /etc/profile
```

2. If you use another Linux system, the installation package sophon-mw_<x.y.z>_$arch.tar.gz can be installed by referring to the Sophon-mw User Manual.

For other questions, please refer to Sophon-mw User Manual.

3.4.1.6. Cross-compilation environment construction

If you want to set up a cross-compilation environment using SophonSDK, you need to use the gcc-aarch64-linux-gnu tool chain and package the header files and libraries that your program depends on into the soc-sdk directory.

1. First install the toolchain:

```
sudo apt-get install gcc-aarch64-linux-gnu g++-aarch64-linux-gnu
```

2. Decompress libsophon_soc_<x.y.z>_aarch64.tar.gz in the sophon-img package and copy everything lib and include to the soc-sdk folder.

```
cd sophon-img_<date>_<hash>
```

```
# Create the root directory for the dependent files
mkdir -p soc-sdk
# Decompress libsophon_soc_${x.y.z}_aarch64.tar.gz in the sophon-img release package, where x.y.z is the version number
tar -zxf libsophon_soc_<x.y.z>_aarch64.tar.gz
# Copy the relevant library directory and header directory to the dependent file root
cp -rf libsophon_soc_<x.y.z>_aarch64/opt/sophon/libsophon-<x.y.z>/lib ${soc-sdk}
cp -rf libsophon_soc_<x.y.z>_aarch64/opt/sophon/libsophon-<x.y.z>/include ${soc-sdk}
```

3. Decompress Sophon-mw-soc _<x.y.z>_aarch64.tar.gz in sophon-mw and copy all contents lib and include in sophon-mw to the soc-sdk folder.

```
cd sophon-mw_<date>_<hash>
# Decompress Sophon-mw-soc_ <x.y.z>_aarch64.tar.gz in the sophon-mw package, where x.y.z is the version number
tar -zxf sophon-mw-soc_<x.y.z>_aarch64.tar.gz
# Copy the ffmpeg and opencv library and header directories to the dependent file root
cp -rf sophon-mw-soc_<x.y.z>_aarch64/opt/sophon/sophon-ffmpeg_<x.y.z>/lib ${soc-sdk}
cp -rf sophon-mw-soc_<x.y.z>_aarch64/opt/sophon/sophon-ffmpeg_<x.y.z>/include ${soc-sdk}
cp -rf sophon-mw-soc_<x.y.z>_aarch64/opt/sophon/sophon-opencv_<x.y.z>/lib ${soc-sdk}
cp -rf sophon-mw-soc_<x.y.z>_aarch64/opt/sophon/sophon-opencv_<x.y.z>/include ${soc-sdk}
```

4. If you need to use a third-party library, you can use qemu to build a virtual environment installation on x86 and copy the header and library files into the soc-sdk directory, as shown in the sophon-mw User Manual.

3.4.1.7. Verification

You can run the following command to verify that the cross-compilation toolchain in the development environment is configured successfully:

```
which aarch64-linux-gnu-g++
# Terminal output
# /usr/bin/aarch64-linux-gnu-g++
```

If the terminal prints the path to aarch64 compilation, the cross-compilation toolchain is correct and the development environment is working.

If you need to use the SAIL module, on non-SOC platforms you need to install the pip package according to the python version you are using. Please refer to the installation instructions in the SAIL User Development Manual. If you are using the SAIL module in the SoC platform, you only need to set the environment variables.

3.4.2. Environment Configuration -SoC

3.4.2.1. Development environment configuration

For SoC mode, model transformation also needs to be completed in docker development container; C/C++ programs are recommended to be compiled on x86 hosts using the cross-compilation toolchain to generate executable files, and then copied to the SoC target platform to run. The docker development container is configured according to the previous section.

If you want to directly compile C/C++ programs on SoC, you need to install sophon-libsophon-dev_<x.y.z>_arm64.deb by running the following command:

```
sudo dpkg -i sophon-soc-libsophon-dev_<x.y.z>_arm64.deb
```

To install sophon-mw-soc-sophon-ffmpeg-dev_< X.Y.Z >_arm64.deb, sophon-mw-soc-sophon-opencv-dev_< X.Y.Z > _arm64.deb toolkit, use the following command to install:

```
sudo dpkg -i sophon-mw-soc-sophon-ffmpeg-dev_<x.y.z>_arm64.deb
sudo dpkg -i sophon-mw-soc-sophon-opencv-dev_<x.y.z>_arm64.deb
```

Attention

attention

1. SE microserver no longer presets the face capture application gate, and the gate application will not be maintained. In the future, we will also upgrade the default system of SE microserver to Ubuntu 20.04, and have a web interface for querying and configuring basic information. It also uses qt to write a simple interface for IP configuration. You can connect the HDMI interface to the display for viewing, and use the keyboard and mouse integrated suite to operate accordingly.
2. The built-in operating system of the SE microserver does not have a desktop system. You need to use ssh to log in to the terminal of the microserver for operation development.

3.4.2.2. Operating environment configuration

For the SoC platform, the corresponding libsophon, Sophon-Opencv and Sophon-ffmpeg runtime packages have been integrated internally under /opt/sophon/. Just set the environment variables.

```
# Setting environment variables
export PYTHONPATH=$PYTHONPATH:/opt/sophon/sophon-opencv_<x.y.z>/opencv
```