CANN
6.3.RC2
软件安装指南

文档版本	01
发布日期	2023-10-11

华为技术有限公司

版权所有 © 华为技术有限公司 2023。 保留一切权利。

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

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司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，华为公司对本文档内容不做任何明示或暗示的声明或保证。

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。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

 版权所有 © 华为技术有限公司

 版权所有 © 华为技术有限公司

 版权所有 © 华为技术有限公司

1 安装须知

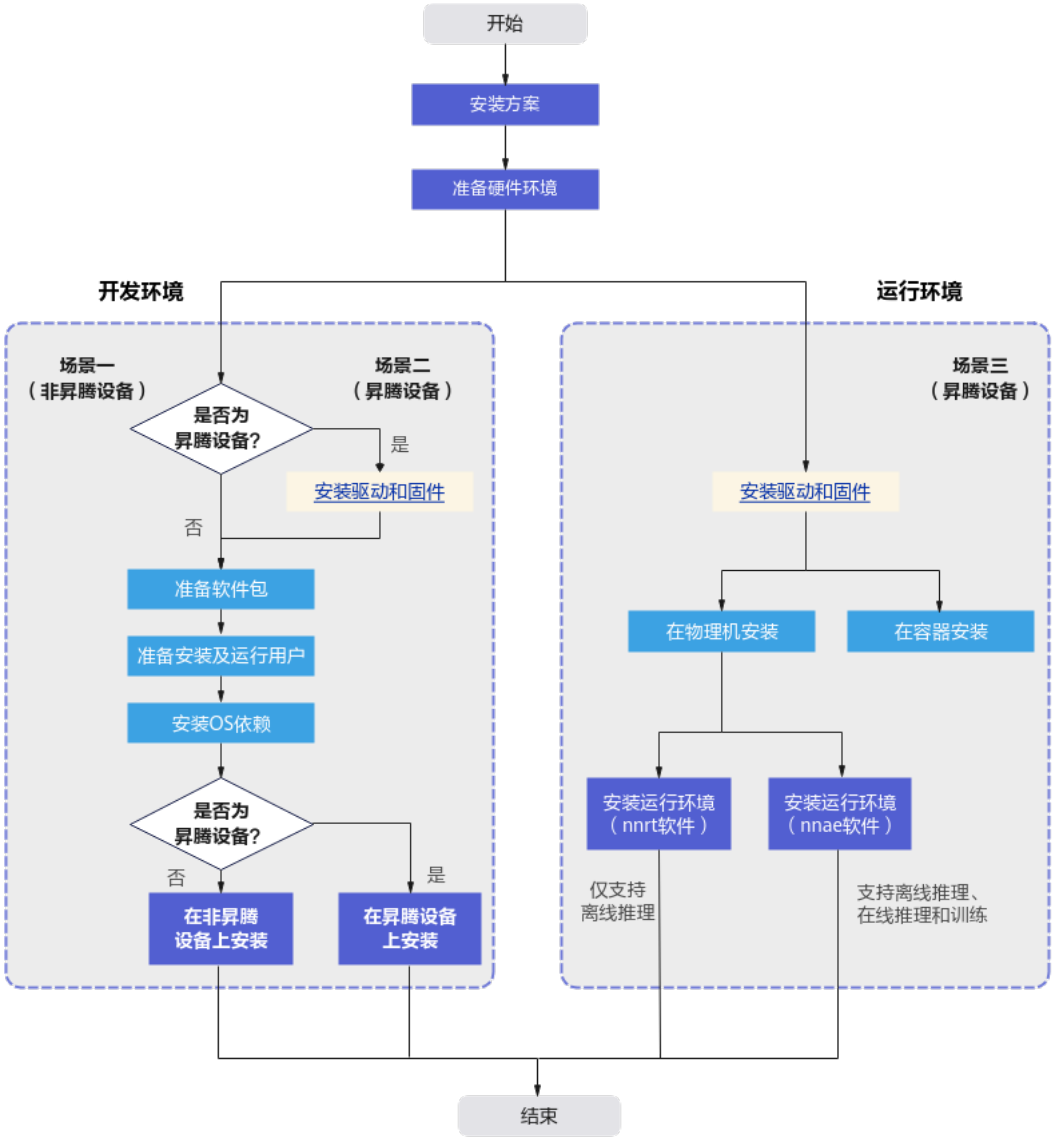
CANN（Compute Architecture for Neural Networks）是华为公司针对AI场景推出的异构计算架构，通过提供多层次的编程接口，支持用户快速构建基于昇腾平台的AI应用和业务。

本文档主要用于指导用户安装CANN开发或运行环境，文档中包含的硬件产品具体介绍请参考《**昇腾产品形态说明**》。

- **开发环境：主要用于代码开发、编译、调测等开发活动。**
  - **（场景一）在非昇腾AI设备上安装开发环境，仅能用于代码开发、编译等不依赖于昇腾设备的开发活动（例如ATC模型转换、算子和推理应用程序的纯代码开发）。**
  - **（场景二）在昇腾AI设备上安装开发环境，支持代码开发和编译，同时可以运行应用程序或进行训练脚本的迁移、开发&调试。**
- **运行环境：在昇腾AI设备上运行用户开发的应用程序或进行训练脚本的迁移、开发&调试。**

本文档指导用户使用命令行方式进行安装，如需使用ascend-deployer工具安装，请参见《**ascend-deployer用户指南**》，使用SmartKit工具进行安装，请参考《**SmartKit Computing 用户指南**》。

 版权所有 © 华为技术有限公司



 版权所有 © 华为技术有限公司

2 安装方案

2.1 非昇腾设备

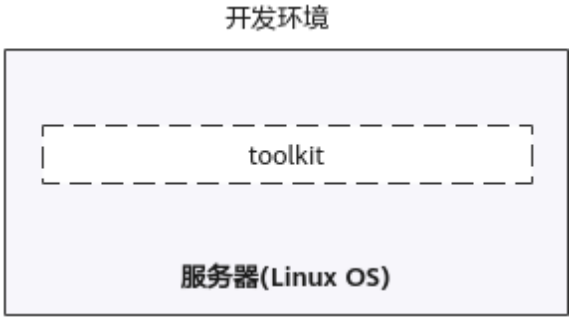
对于非昇腾AI设备，仅支持安装纯开发环境，安装软件如**图2-1**所示。

非昇腾AI设备无需安装固件与驱动，仅能用于代码开发、编译等不依赖于昇腾设备的开发活动。

软件包说明：

toolkit：开发套件包。主要用于用户开发应用、自定义算子和模型转换。开发套件包包含开发应用程序所需的库文件和开发辅助工具等。

图 2-1 开发环境



版权所有 © 华为技术有限公司

说明

若开发环境为x86_64架构，运行环境为aarch64架构，开发环境上可以只部署x86_64开发套件包，后续编译应用时需要使用交叉编译工具链及aarch64架构的库文件。

2.2 Atlas 200I SoC A1

Atlas 200I SoC A1（Atlas 200I SoC A1核心板）开发和运行环境安装参考如下。

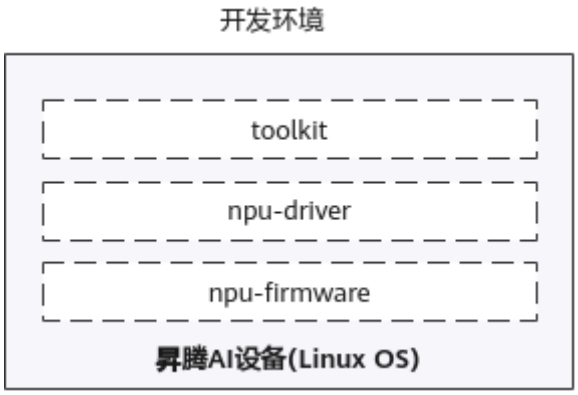
- 开发环境

  开发环境安装软件如图2-2所示。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - toolkit：开发套件包。主要用于用户开发应用、自定义算子和模型转换。开发套件包包含开发应用程序所需的AscendCL库文件、开发工具（如ATC模型转换工具）等。

  图 2-2 开发环境

- 运行环境

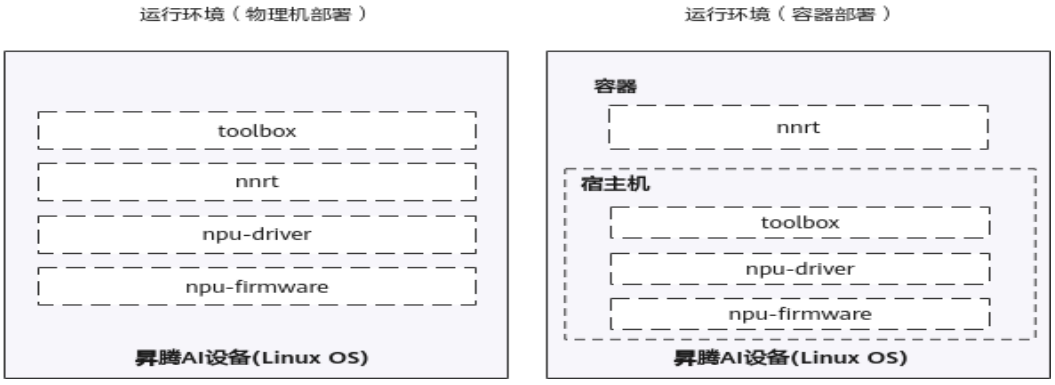
  运行环境安装软件如图2-3所示。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - nnrt：离线推理引擎包，用于应用程序的模型推理。仅支持离线推理。

 版权所有 © 华为技术有限公司

图 2-3 运行环境



说明

离线推理：是基于原有AI框架模型转换OM模型，不依赖于AI框架执行推理的场景。

2.3 Atlas 300I Pro、Atlas 300V Pro、Atlas 300V、Atlas 300I Duo

Atlas 300I Pro（Atlas 300I Pro 推理卡）、Atlas 300V Pro（Atlas 300V Pro 视频解析卡）、Atlas 300V（Atlas 300V 视频解析卡）、Atlas 300I Duo（Atlas 300I Duo 推理卡）开发和运行环境安装参考如下：

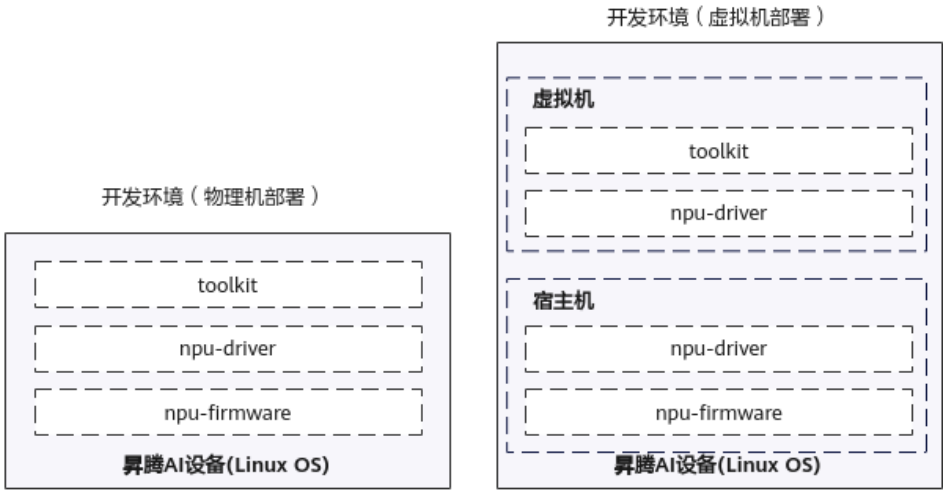
- 开发环境

  开发环境安装软件如图2-4所示。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - toolkit：开发套件包。主要用于用户开发应用、自定义算子和模型转换。开发套件包包含开发应用程序所需的AscendCL库文件、开发工具（如ATC模型转换工具）等。

图 2-4 开发环境



版权所有 © 华为技术有限公司

- 运行环境

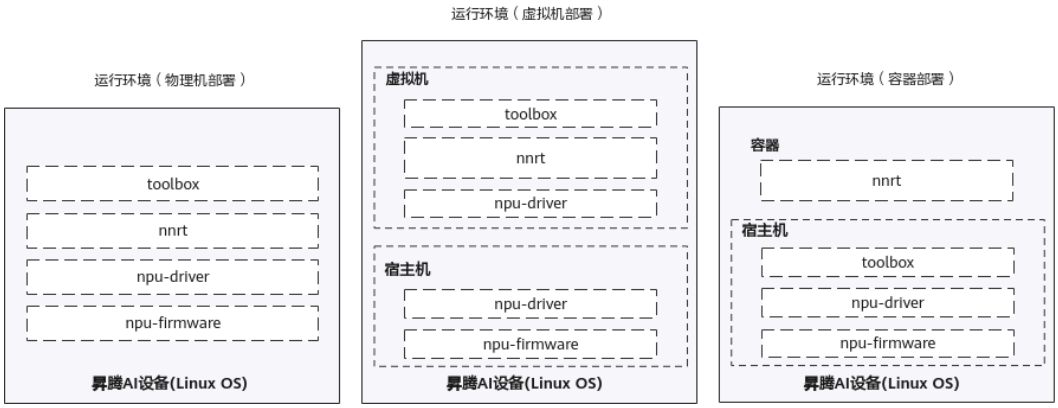
  运行环境安装软件如图2-5所示。

  如需进行容器部署，请先参考本手册安装驱动、固件，然后直接从AscendHub上拉取镜像（推荐）或参考《MindX DL Ascend Docker Runtime用户指南》自行构建容器镜像。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - nnrt：离线推理引擎包，用于应用程序的模型推理。仅支持离线推理。
  - toolbox：实用工具包。主要包含ascend-dmi工具。

  图 2-5 运行环境



说明

离线推理：是基于原有AI框架模型转换OM模型，不依赖于AI框架执行推理的场景。

2.4 A500 Pro-3000

A500 Pro-3000（Atlas 500 Pro 智能边缘服务器 （型号 3000））开发或运行环境安装参考如下：

- 开发环境

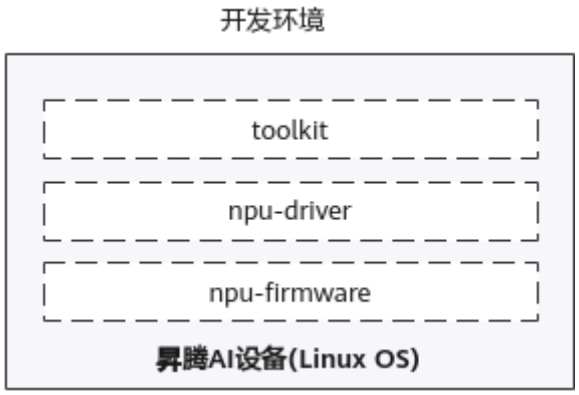
  开发环境安装软件如图2-6所示。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - toolkit：开发套件包。主要用于用户开发应用、自定义算子和模型转换。开发套件包包含开发应用程序所需的AscendCL库文件、开发工具（如ATC模型转换工具）等。

 版权所有 © 华为技术有限公司

图 2-6 开发环境



- 运行环境

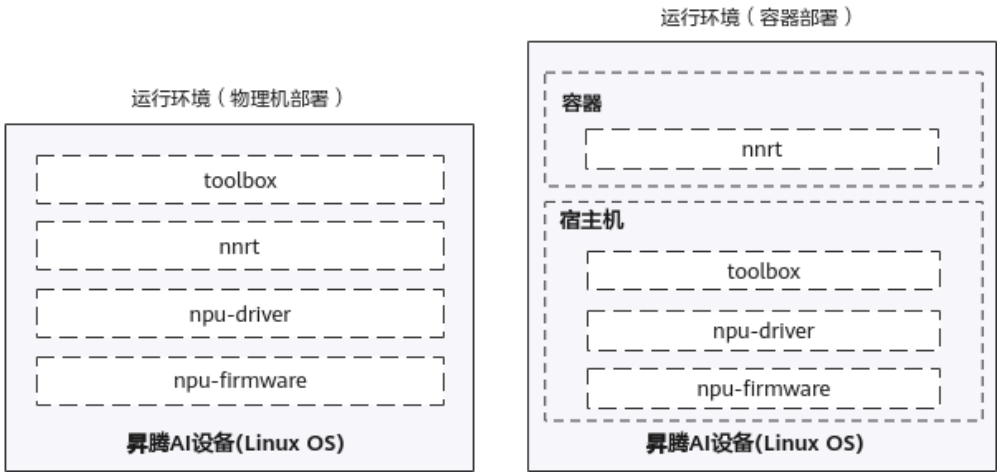
  运行环境安装软件如图2-7所示。

  如需进行容器部署，请先参考本手册安装驱动、固件，然后直接从AscendHub上拉取镜像（推荐）或参考《MindX DL Ascend Docker Runtime用户指南》自行构建容器镜像。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - nnrt：离线推理引擎包，用于应用程序的模型推理。仅支持离线推理。
  - toolbox：实用工具包。主要包含ascend-dmi工具。

图 2-7 运行环境



说明

离线推理：是基于原有AI框架模型转换OM模型，不依赖于AI框架执行推理的场景。

2.5 A800-3000、A800-3010

A800-3000（Atlas 800 推理服务器（型号 3000））、A800-3010（Atlas 800 推理服务器（型号 3010））开发或运行环境安装参考如下：

 版权所有 © 华为技术有限公司

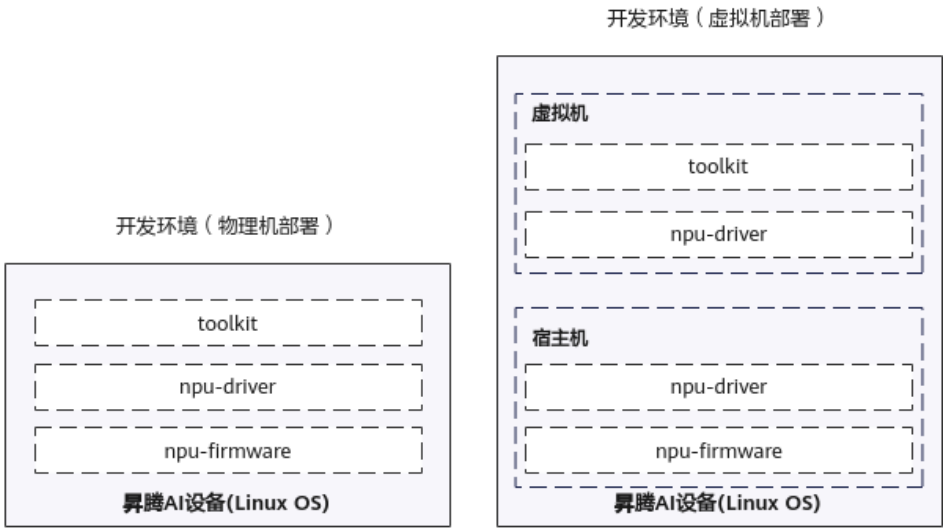
- 开发环境

  开发环境安装软件如图2-8所示。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - toolkit：开发套件包。主要用于用户开发应用、自定义算子和模型转换。开发套件包包含开发应用程序所需的AscendCL库文件、开发工具（如ATC模型转换工具）等。

  图 2-8 开发环境



- 运行环境

  运行环境安装软件如图2-9所示，其中nnae软件包支持离线推理。

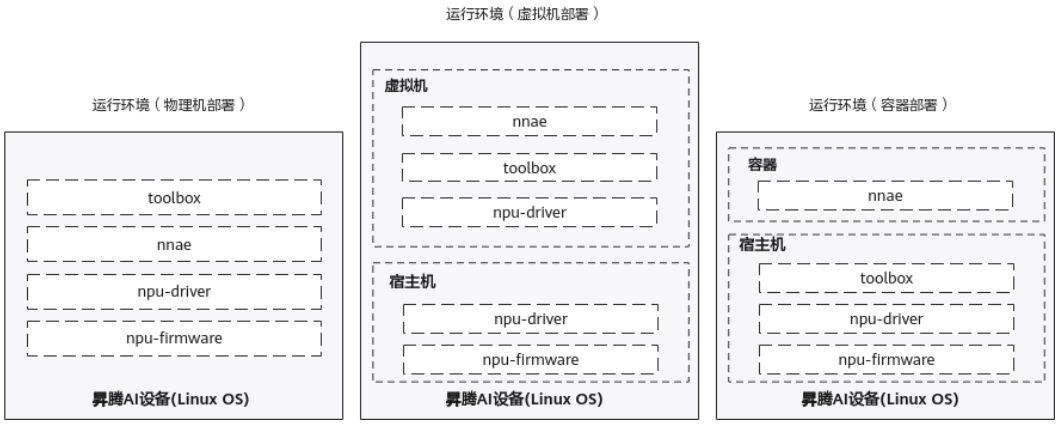
  如需进行容器部署，请先参考本手册安装驱动、固件，然后直接从AscendHub上拉取镜像（推荐）或参考《MindX DL Ascend Docker Runtime用户指南》自行构建容器镜像。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - nnae：深度学习引擎包。支持离线推理。
  - toolbox：实用工具包。主要包含ascend-dmi工具。

版权所有 © 华为技术有限公司

图 2-9 运行环境



说明

离线推理：是基于原有AI框架模型转换OM模型，不依赖于AI框架执行推理的场景。

2.6 A300T-9000、Atlas 300T Pro、A800-9000、A800-9010、A900-9000

A300T-9000（Atlas 300T 训练卡（型号 9000））、Atlas 300T Pro（Atlas 300T Pro 训练卡（型号：9000））、A800-9000（Atlas 800 训练服务器（型号 9000））、A800-9010（Atlas 800 训练服务器（型号 9010））、A900-9000（Atlas 900 AI集群（型号 9000））开发或运行环境安装参考如下：

- 开发环境

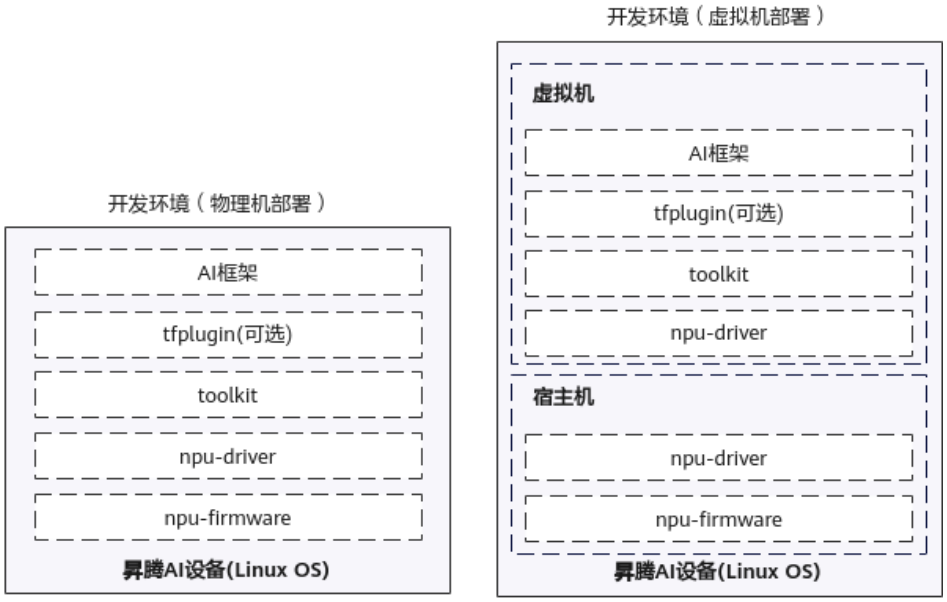
  开发环境安装软件如图2-10所示。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - toolkit：开发套件包。主要用于用户开发应用、自定义算子和模型转换。开发套件包包含开发应用程序所需的AscendCL库文件、开发工具（如ATC模型转换工具）等。
  - tfplugin（可选）：插件包，对接上层框架TensorFlow的适配插件。在线推理或训练场景下若使用深度学习框架TensorFlow，需要安装该软件包。
  - AI框架：深度学习框架。如MindSpore、TensorFlow、PyTorch等。

版权所有 © 华为技术有限公司

图 2-10 开发环境



- 运行环境

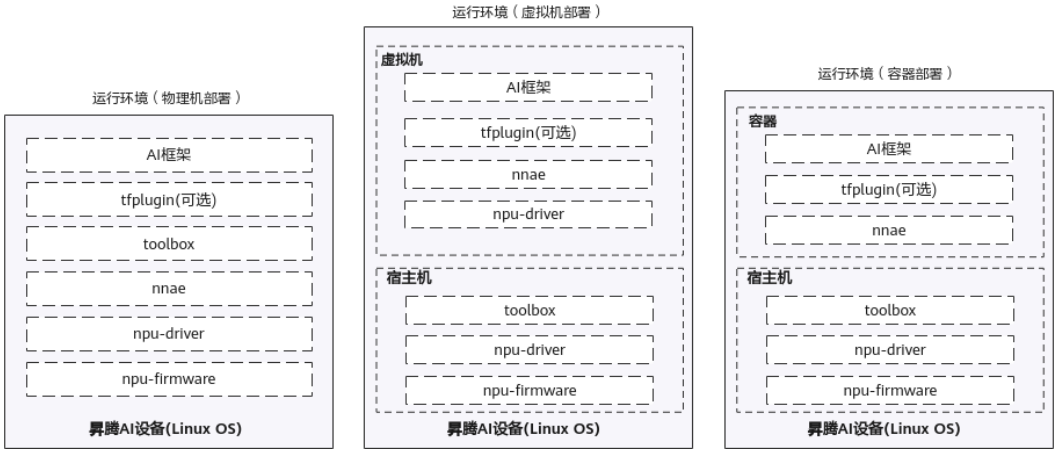
  运行环境安装软件如图2-11所示。

  如需进行容器部署，请先参考本手册安装驱动、固件，然后直接从AscendHub上拉取镜像（推荐）或参考《MindX DL Ascend Docker Runtime用户指南》自行构建容器镜像。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - nnae：深度学习引擎包。支持离线推理、在线推理、训练。
  - toolbox：实用工具包。主要包含ascend-dmi工具。
  - tfplugin（可选）：插件包，对接上层框架TensorFlow的适配插件。在线推理或训练场景下若使用深度学习框架TensorFlow，需要安装该软件包。
  - AI框架：深度学习框架。如MindSpore、TensorFlow、PyTorch等。

图 2-11 运行环境



版权所有 © 华为技术有限公司

说明

- 离线推理：是基于原有AI框架模型转换OM模型，不依赖于AI框架执行推理的场景。
- 在线推理：是将原有AI框架做推理的应用快速迁移至昇腾AI处理器上，依赖于AI框架执行推理的场景。

2.7 Atlas 300T A2、Atlas 800T A2、Atlas 900 A2 PoD

Atlas 300T A2（Atlas 300T A2 训练卡）、Atlas 800T A2（Atlas 800T A2 训练服务器）和Atlas 900 A2 PoD（Atlas 900 A2 PoD 集群基础单元）开发或运行环境安装参考如下：

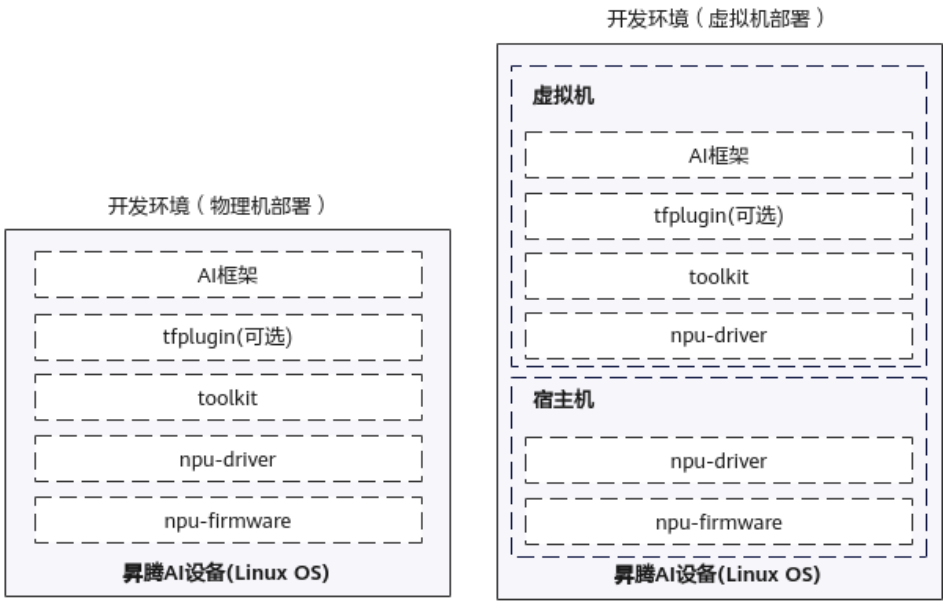
- 开发环境

  开发环境安装软件如图2-12所示。

  软件包说明：

  - npu-firmware：固件安装包。
  - npu-driver：驱动安装包。
  - toolkit：开发套件包。主要用于用户开发应用、自定义算子和模型转换。开发套件包包含开发应用程序所需的AscendCL库文件、开发工具（如ATC模型转换工具）等。
  - tfplugin（可选）：插件包，对接上层框架TensorFlow的适配插件。在线推理或训练场景下若使用深度学习框架TensorFlow，需要安装该软件包。
  - AI框架：深度学习框架。如MindSpore、TensorFlow、PyTorch等。

  图 2-12 开发环境

- 运行环境

  运行环境安装软件如图2-13所示。

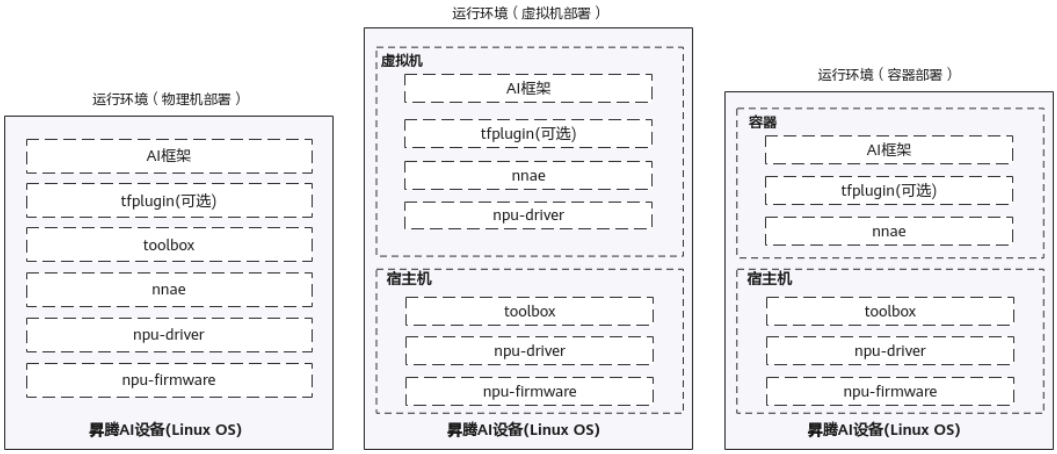
  如需进行容器部署，请先参考本手册安装驱动、固件，然后直接从AscendHub上拉取镜像（推荐）或参考《MindX DL Ascend Docker Runtime用户指南》自行构建容器镜像。

 版权所有 © 华为技术有限公司

软件包说明：

- npu-firmware：固件安装包。
- npu-driver：驱动安装包。
- nnae：深度学习引擎包。支持离线推理、在线推理、训练。
- toolbox：实用工具包。主要包含ascend-dmi工具。
- tfplugin（可选）：插件包，对接上层框架TensorFlow的适配插件。在线推理或训练场景下若使用深度学习框架TensorFlow，需要安装该软件包。
- AI框架：深度学习框架。如MindSpore、TensorFlow、PyTorch等。

图 2-13 运行环境



说明

在线推理：是将原有AI框架做推理的应用快速迁移至昇腾AI处理器上，依赖于AI框架执行推理的场景。

 版权所有 © 华为技术有限公司

3 准备硬件环境

请根据表3-1，参考对应指导文档安装硬件产品和操作系统。

表 3-1 准备硬件环境

产品型号	参考手册
Atlas 200I SoC A1	• Atlas 200I SoC A1安装的相关操作具体请参考《**Atlas 200I SoC A1 核心板 用户指南**》。 • 操作系统要求请参考《**Atlas 200I Soc A1 核心板 openEuler 20.03 LTS SP3 操作系统 安装指导书**》。
Atlas 300I Pro	• Atlas 300I Pro 推理卡的安装方法可参见各服务器用户指南。 • 操作系统要求请参考《**Atlas 300I Pro 推理卡 用户指南**》。
Atlas 300V Pro	• Atlas 300V Pro 视频解析卡的安装方法可参见各服务器用户指南。 • 操作系统要求请参考《**Atlas 300V Pro 视频解析卡 用户指南**》。
Atlas 300V	• Atlas 300V 视频解析卡的安装方法可参见各服务器用户指南。 • 操作系统要求请参考《**Atlas 300V 视频解析卡 用户指南**》。
Atlas 300I Duo	• Atlas 300I Duo 推理卡的安装方法可参见各服务器用户指南。 • 操作系统要求请参考《**Atlas 300I Duo 推理卡 用户指南**》。
Atlas 300T A2	• Atlas 300T A2 训练卡安装的相关操作具体请参考《**Atlas 300T A2 训练卡 用户指南**》。 • 操作系统要求请参考《**Atlas 300T A2 训练卡 用户指南**》。

 版权所有 © 华为技术有限公司

产品型号	参考手册
A300T-9000	• Atlas 300T 训练卡（型号：9000）安装的相关操作具体请参考《**Atlas 300T 训练卡 用户指南（型号9000）**》。 • 操作系统要求请参考《**Atlas 300T 训练卡 用户指南（型号9000）**》。
Atlas 300T Pro	• Atlas 300T Pro 训练卡（型号：9000）安装的相关操作具体请参考《**Atlas 300T Pro 训练卡 用户指南（型号9000）**》。 • 操作系统要求请参考《**Atlas 300T Pro 训练卡 用户指南（型号 9000）**》。
A500 Pro-3000	• Atlas 500 Pro 智能边缘服务器（型号：3000）安装上架、服务器基础参数配置、安装操作系统等操作请参见《**Atlas 500 Pro 智能边缘服务器 用户指南（型号 3000）**》。 • 操作系统要求请参考《**Atlas 500 Pro 智能边缘服务器 用户指南（型号 3000）**》。
A800-3000	• Atlas 800 推理服务器（型号：3000）安装上架、服务器基础参数配置、安装操作系统等操作请参见《**Atlas 800 推理服务器 用户指南（型号 3000）**》。 • 操作系统要求请参考《**Atlas 800 推理服务器 用户指南 (型号 3000)**》。
A800-3010	• Atlas 800 推理服务器（型号：3010）安装上架、服务器基础参数配置、安装操作系统等操作请参见《**Atlas 800 推理服务器 用户指南（型号 3010）**》。 • 操作系统要求请参考《**Atlas 800 推理服务器 用户指南 (型号 3010)**》。
A800-9000	• Atlas 800 训练服务器（型号：9000）安装上架、服务器基础参数配置、安装操作系统等操作请参见《**Atlas 800 训练服务器 用户指南 (型号9000, 风冷)**》或《**Atlas 800 训练服务器 用户指南 (型号9000, 液冷)**》。 • 操作系统要求请参考《**Atlas 800 训练服务器 用户指南 (型号9000, 风冷)**》或《**Atlas 800 训练服务器 用户指南 (型号9000, 液冷)**》。
Atlas 800T A2	• Atlas 800T A2 训练服务器安装上架、服务器基础参数配置、安装操作系统等操作请参见《**Atlas 800T A2 训练服务器 用户指南**》。 • 操作系统要求请参考《**Atlas 800T A2 训练服务器 用户指南**》。
A800-9010	• Atlas 800 训练服务器（型号：9010）安装上架、服务器基础参数配置、安装操作系统等操作请参见《**Atlas 800 训练服务器 用户指南 (型号9010)**》。 • 操作系统要求请参考《**Atlas 800 训练服务器 用户指南 (型号9010)**》。

版权所有 © 华为技术有限公司

产品型号	参考手册
A900-9000	• Atlas 900 AI集群（型号：9000）安装上架、服务器基础参数配置、安装操作系统等操作，请根据集群配置参见对应的手册： – 《**Atlas 900 PoD 用户指南 (型号9000, 交流)**》 – 《**Atlas 900 计算节点 用户指南 (液冷)**》 • 操作系统要求请参考《**Atlas 900 PoD 用户指南 (型号9000, 交流)**》或《**Atlas 900 计算节点 用户指南 (液冷)**》。
Atlas 900 A2 PoD	• Atlas 900 A2 PoD 集群基础单元安装上架、服务器基础参数配置、安装操作系统等操作，请参见《**Atlas 900 A2 PoD 集群基础单元 用户指南（AICC场景）**》 • 操作系统要求请参见《**Atlas 900 A2 PoD 集群基础单元 用户指南（AICC场景）**》。

版权所有 © 华为技术有限公司

说明

操作系统镜像请从官网获取，示例如下：

- Ubuntu 18.04.1：
  - aarch64

    从Ubuntu官网**http://old-releases.ubuntu.com/releases/18.04.1/**下载**ubuntu-18.04.1-server-arm64.iso**。
  - x86_64

    从Ubuntu官网**http://old-releases.ubuntu.com/releases/18.04.1/**下载**ubuntu-18.04.1-server-amd64.iso**。
- CentOS 7.6
  - aarch64

    从CentOS官网**https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/**下载**CentOS-7-aarch64-Everything-1810.iso**。
  - x86_64

    从CentOS官网**http://vault.centos.org/7.6.1810/isos/x86_64/**下载**CentOS-7-x86_64-DVD-1810.iso**。
- Ubuntu 22.04
  - aarch64

    从Ubuntu官网**http://old-releases.ubuntu.com/releases/22.04/**下载**ubuntu-22.04-live-server-arm64.iso**。
  - x86_64

    从Ubuntu官网**http://old-releases.ubuntu.com/releases/22.04/**下载**ubuntu-22.04-live-server-amd64.iso**。
- openEuler 22.03
  - aarch64

    从openEuler官网**https://repo.openeuler.org/openEuler-22.03-LTS/ISO/aarch64/**下载**openEuler-22.03-LTS-aarch64-dvd.iso**
  - x86_64

    从openEuler官网**https://repo.openeuler.org/openEuler-22.03-LTS/ISO/x86_64/**下载**openEuler-22.03-LTS-x86_64-dvd.iso**。

 版权所有 © 华为技术有限公司

4 安装驱动和固件

请根据表4-1参考对应文档进行安装操作。

说明

首次安装请按照“**驱动->固件**”的顺序，分别安装驱动和固件；覆盖安装请按照“**固件->驱动**”的顺序，分别安装固件和驱动。

表 4-1 驱动和固件安装指导

产品型号	参考文档
Atlas 200I SoC A1	请参见《**Atlas 200I SoC A1核心板 23.0.RC2 NPU驱动和固件安装指南**》。
Atlas 300I Pro	请参见《**Ascend 310P 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》。
Atlas 300V Pro	请参见《**Ascend 310P 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》。
Atlas 300V	请参见《**Ascend 310P 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》。
Atlas 300I Duo	请参见《**Ascend 310P 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》。
Atlas 300T A2	请参见《**Ascend 910B 23.0.RC2 NPU驱动和固件安装指南**》。
A300T-9000	请参见《**Ascend 910 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》。
Atlas 300T Pro	请参见《**Ascend 910 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》。
A500 Pro-3000	请参见所配置标卡对应的文档。
A800-3000	请参见所配置标卡对应的文档。
A800-3010	请参见所配置标卡对应的文档。
A800-9000	请参见《**Ascend 910 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》。
Atlas 800T A2	请参见《**Ascend 910B 23.0.RC2 NPU驱动和固件安装指南**》
A800-9010	请参见《**Ascend 910 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》。

版权所有 © 华为技术有限公司

产品型号	参考文档
A900-9000	请参见《**Ascend 910 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》。
Atlas 900 A2 PoD	请参见《**Ascend 910B 23.0.RC2 NPU驱动和固件安装指南**》。

 版权所有 © 华为技术有限公司

5 CANN 软件包支持的 OS 清单

CANN软件包支持的OS清单如表5-1所示，请执行以下命令查询当前操作系统，如果查询的操作系统版本不在列表中，请替换为支持的操作系统。

```
uname -m && cat /etc/*release
```

表 5-1 支持系统列表

产品型号	支持的操作系统
Atlas 200I SoC A1	openEuler 20.03
A800-9000	Ubuntu 18.04.1、Ubuntu 18.04.5、Ubuntu 20.04 EulerOS 2.8、EulerOS 2.10、EulerOS 2.11 openEuler 20.03、openEuler 22.03 CentOS 7.6、CentOS 8.2 BCLinux 7.6、BCLinux 7.7 Kylin V10、Kylin V10 SP1、Kylin V10 SP2 UOS20 1020e
A800-9010	Ubuntu 18.04.1、Ubuntu 18.04.5、Ubuntu 20.04 openEuler 20.03、openEuler 22.03 CentOS 7.6、CentOS 8.2 Debian 9.9、Debian 10.0 BCLinux 7.6 Kylin V10 SP1
A800-3000+A300T-9000 A800-3000+Atlas 300T Pro	Ubuntu 18.04.1、Ubuntu 18.04.5、Ubuntu 20.04 EulerOS 2.8 openEuler 20.03、openEuler 22.03 CentOS 7.6、CentOS 8.2 Kylin V10 SP1 UOS20 1020e

版权所有 © 华为技术有限公司

产品型号	支持的操作系统
A800-3000+Atlas 300I Pro A800-3000+Atlas 300V Pro	Ubuntu 18.04.5、Ubuntu 20.04 EulerOS 2.9、EulerOS 2.10、EulerOS 2.11 openEuler 20.03、openEuler 22.03 CentOS 7.6、CentOS 8.5 Kylin V10 SP1、Kylin V10 SP2
A800-3010+A300T-9000 A800-3010+Atlas 300T Pro	Ubuntu 18.04.1、Ubuntu 18.04.5、Ubuntu 20.04 openEuler 20.03、openEuler 22.03 CentOS 7.6、CentOS 8.2 Debian 9.9、Debian 10.0 Kylin V10 SP1
A800-3010+Atlas 300I Pro A800-3010+Atlas 300V Pro	Ubuntu 20.04 EulerOS 2.9、EulerOS 2.10 openEuler 20.03、openEuler 22.03 CentOS 7.6 SLES 12.5 Kylin V10 SP1
A800-3000 + Atlas 300V A800-3010 + Atlas 300V	Ubuntu 20.04 CentOS 7.8 openEuler22.03
A500 Pro-3000+Atlas 300I Pro A500 Pro-3000+Atlas 300V Pro	Ubuntu 20.04 EulerOS 2.10 openEuler 20.03、openEuler 22.03 CentOS 7.6 Linx 6.0.90、Linx 6.0.100 Kylin V10 SP1
A500 Pro-3000+A300V	Linx 6.0.100 Kylin V10 SP1
Atlas 300I Duo	Ubuntu 20.04 CentOS 7.8
Atlas 800T A2	openEuler 22.03 Kylin V10 SP2 BCLinux-for-Euler-21.10

版权所有 © 华为技术有限公司

产品型号	支持的操作系统
Atlas 900 A2 PoD	EulerOS 2.10 openEuler 22.03 Kylin V10 SP2 BCLinux-for-Euler-21.10
Atlas 800-3000 + Atlas 300T A2	Ubuntu 22.04 openEuler 22.03

版权所有 © 华为技术有限公司

6 安装开发环境

6.1 准备软件包

下载软件包

软件安装前，请参考**表6-1**获取所需软件包和对应的数字签名文件，各软件包版本号需要保持一致。

下载本软件即表示您同意**华为企业软件许可协议**的条款和条件。

表 6-1 CANN 软件包

名称	软件包	说明	获取链接
开发套件包	Ascend-cann-toolkit_*{version}*_linux-*{arch}*.run	• 主要用于用户开发应用、自定义算子和模型转换。开发套件包包含开发应用程序所需的库文件、开发工具如ATC模型转换工具。 • 请根据CPU架构（x86_64、aarch64）获取对应的软件包。	**获取链接**

版权所有 © 华为技术有限公司

名称	软件包	说明	获取链接
框架插件包	Ascend-cann-tfplugin_*{version}*_linux-*{arch}*.run	（可选）插件包，对接上层框架TensorFlow的适配插件。 在线推理或训练场景下若使用深度学习框架TensorFlow，需要获取该软件包。	**获取链接**

说明

*{version}*表示软件版本号，*{arch}*表示CPU架构。

软件数字签名验证

为了防止软件包在传递过程或存储期间被恶意篡改，下载软件包时需下载对应的数字签名文件用于完整性验证。

在软件包下载之后，请参考《 OpenPGP签名验证指南 》，对从Support网站下载的软件包进行PGP数字签名校验。如果校验失败，请不要使用该软件包，先联系华为技术支持工程师解决。

使用软件包安装/升级之前，也需要按上述过程先验证软件包的数字签名，确保软件包未被篡改。

运营商客户请访问：**http://support.huawei.com/carrier/digitalSignatureAction**

企业客户请访问：**https://support.huawei.com/enterprise/zh/tool/pgp-verify-TL1000000054**

6.2 准备安装及运行用户

- 运行用户：实际运行推理业务或执行训练的用户。
- 安装用户：实际安装软件包的用户。
  - 若使用root用户安装，支持所有用户运行相关业务。
  - 若使用非root用户安装，则安装及运行用户必须相同。
    - 已有非root用户，则无需再次创建。
    - 若想使用新的非root用户，则需要先创建该用户，请参见如下方法创建。

说明

- 如果安装驱动时未携带“--install-for-all”，并且CANN软件包运行用户为非root，则该CANN软件包运行用户所属的属组必须和驱动运行用户所属属组相同；如果不同，请用户自行添加到驱动运行用户属组。
- 运行用户不建议为root用户属组，权限控制可能存在安全风险，请谨慎使用。

创建非root用户操作方法如下，如下命令请以root用户执行。

 版权所有 © 华为技术有限公司

1. 创建非root用户。
   groupadd *usergroup*
   useradd -g *usergroup* **-d /home/**_username_ **-m** *username* **-s /bin/bash**
2. 设置非root用户密码。
   passwd *username*

说明

- 创建完运行用户后， 请勿关闭该用户的登录认证功能。
- 设置的口令需符合口令复杂度要求（请参见**C.4 口令复杂度要求**）。密码有效期为90天，您可以在/etc/login.defs文件中修改有效期的天数，或者通过chage命令来设置用户的有效期，详情请参见**A.9 设置用户有效期**。

6.3 安装依赖

6.3.1 安装前必读

安装CANN软件前需安装相关依赖。

本章节以Ubuntu 18.04、CentOS 7.6和SLES 12.5为例，详述依赖安装操作。其他系统可参考这三种系统进行安装。

- Ubuntu、Debian、UOS20、UOS20 SP1、Linx系统可参考Ubuntu 18.04进行安装。
- EulerOS、openEuler、CentOS、BCLinux、Kylin、UOS20 1020e系统可参考CentOS 7.6进行安装。
- SLES系统可参考SLES 12.5进行安装。

6.3.2 依赖列表

说明

针对用户自行安装的开源软件，如Python、numpy等，请使用稳定版本（尽量使用无漏洞的版本）。

Ubuntu 18.04

表 6-2 依赖信息

名称	版本要求
Python	CANN支持Python3.7.*x*（3.7.0~3.7.11）、Python3.8.*x*（3.8.0~3.8.11）、Python3.9.*x*（3.9.0~3.9.7）。 TensorFlow1.15支持Python3.7.*x*（3.7.0~3.7.11），TensorFlow2.6.5和PyTorch框架支持Python3.7.x（3.7.5~3.7.11）、Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。
cmake	>=3.5.1

 版权所有 © 华为技术有限公司

名称	版本要求
make	-
gcc g++	• 离线推理场景 要求4.8.5版本及以上gcc。 • 在线推理、训练、Ascend Graph开发场景 要求7.3.0版本及以上gcc，若gcc版本低于7.3.0，可参考A.7 安装7.3.0版本gcc进行安装。
zlib1g zlib1g-dev libsqlite3-dev openssl libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran	无版本要求，安装的版本以操作系统自带的源为准。
numpy	>=1.14.3
decorator	>=4.4.0
sympy	>=1.4
cffi	>=1.12.3
protobuf	>=3.11.3
attrs pyyaml pathlib2 scipy requests psutil absl-py	无版本要求，安装的版本以pip源为准。

 版权所有 © 华为技术有限公司

CentOS 7.6

表 6-3 依赖信息

名称	版本限制
Python	CANN支持Python3.7.*x*（3.7.0~3.7.11）、Python3.8.*x*（3.8.0~3.8.11）、Python3.9.*x*（3.9.0~3.9.7）。 TensorFlow1.15支持Python3.7.*x*（3.7.0~3.7.11），TensorFlow2.6.5和PyTorch框架支持Python3.7.x（3.7.5~3.7.11）、Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。
cmake	>=3.5.1
make	-
gcc gcc-c++	• 离线推理场景 要求4.8.5版本及以上gcc。 • 在线推理、训练、Ascend Graph开发场景 要求7.3.0版本及以上gcc，若gcc版本低于7.3.0，可参考**A.7 安装7.3.0版本gcc**进行安装。
unzip zlib-devel libffi-devel openssl-devel pciutils net-tools sqlite-devel lapack-devel gcc-gfortran python3-devel（openEuler系统需要安装）	无版本要求，安装的版本以操作系统自带的源为准。
numpy	>=1.14.3
decorator	>=4.4.0
sympy	>=1.4
cffi	>=1.12.3
protobuf	>=3.11.3

版权所有 © 华为技术有限公司

名称	版本限制
attrs pyyaml pathlib2 scipy requests psutil absl-py	无版本要求，安装的版本以pip源为准。

Suse 12SP5

表 6-4 依赖信息

类别	版本限制
Python	CANN支持Python3.7.*x*（3.7.0~3.7.11）、Python3.8.*x*（3.8.0~3.8.11）、Python3.9.*x*（3.9.0~3.9.7）。 TensorFlow1.15支持Python3.7.*x*（3.7.0~3.7.11），TensorFlow2.6.5和PyTorch框架支持Python3.7.x（3.7.5~3.7.11）、Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。
cmake	>=3.5.1
make	-
gcc	离线推理场景： 要求4.8.5版本及以上gcc。
gcc-c++	
unzip zlib-devel libffi-devel openssl-devel pciutils net-tools gdbm-devel	无版本要求，安装的版本以操作系统自带的源为准。
numpy	>=1.14.3
decorator	>=4.4.0
sympy	>=1.4
cffi	>=1.12.3
protobuf	>=3.11.3

 版权所有 © 华为技术有限公司

类别	版本限制
attrs pyyaml pathlib2 scipy requests psutil gnureadline absl-py	无版本要求，安装的版本以pip源为准。

6.3.3 安装步骤（Ubuntu 18.04）

检查源

安装过程需要下载相关依赖，请确保安装环境能够连接网络。

请在root用户下执行如下命令检查源是否可用。

```
apt-get update
```

如果命令执行报错或者后续安装依赖时等待时间过长甚至报错，则检查网络是否连接或者把“/etc/apt/sources.list”文件中的源更换为可用的源或使用镜像源（以配置华为镜像源为例，可参考**华为开源镜像站**）。

检查 root 用户的 umask

1. 以root用户登录安装环境。
2. 检查root用户的umask值。
   ```
   umask
   ```
3. 如果umask不等于0022，请执行如下操作配置，在该文件的最后一行添加umask 0022后保存。
   a. 在任意目录下执行如下命令，打开**.bashrc**文件：
      ```
      vi ~/.bashrc
      ```
      在文件最后一行后面添加umask 0022内容。
   b. 执行**:wq!**命令保存文件并退出。
   c. 执行**source ~/.bashrc**命令使其立即生效。

说明

依赖安装完成后，请用户恢复为原umask值（删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

配置安装用户权限

可使用root或非root用户（该非root用户需与软件包安装用户保持一致）安装依赖，如果使用非root用户安装，可能需要用到提权命令，请用户自行获取所需的sudo权限。使用完成后请取消涉及高危命令的权限，否则有sudo提权风险。

 版权所有 © 华为技术有限公司

安装依赖

步骤1 检查系统是否安装python依赖以及gcc等软件。

分别使用如下命令检查是否安装gcc，make以及python依赖软件等。

```
gcc --version
g++ --version
make --version
cmake --version
dpkg -l zlib1g| grep zlib1g| grep ii
dpkg -l zlib1g-dev| grep zlib1g-dev| grep ii
dpkg -l libsqlite3-dev| grep libsqlite3-dev| grep ii
dpkg -l openssl| grep openssl| grep ii
dpkg -l libssl-dev| grep libssl-dev| grep ii
dpkg -l libffi-dev| grep libffi-dev| grep ii
dpkg -l unzip| grep unzip| grep ii
dpkg -l pciutils| grep pciutils| grep ii
dpkg -l net-tools| grep net-tools| grep ii
dpkg -l libblas-dev| grep libblas-dev| grep ii
dpkg -l gfortran| grep gfortran| grep ii
dpkg -l libblas3| grep libblas3| grep ii
```

若分别返回如下信息则说明已经安装，进入下一步（以下回显仅为示例，版本要求请以**6.3.2 依赖列表**为准）。

```
gcc (Ubuntu 7.3.0-3ubuntu1~18.04) 7.3.0
g++ (Ubuntu 7.3.0-3ubuntu1~18.04) 7.3.0
GNU Make 4.1
cmake version 3.10.2
zlib1g:arm64   1:1.2.11.dfsg-0ubuntu2 arm64       compression library - runtime
zlib1g-dev:arm64 1:1.2.11.dfsg-0ubuntu2 arm64       compression library - development
libsqlite3-dev:arm64 3.22.0-1ubuntu0.3 arm64       SQLite 3 development files
openssl  1.1.1-1ubuntu2.1~18.04.6 arm64   Secure Sockets Layer toolkit - cryptographic utility
libssl-dev:arm64 1.1.1-1ubuntu2.1~18.04.6 arm64     Secure Sockets Layer toolkit - development files
libffi-dev:arm64 3.2.1-8      arm64       Foreign Function Interface library (development files)
unzip          6.0-21ubuntu1 arm64       De-archiver for .zip files
pciutils       1:3.5.2-1ubuntu1 arm64       Linux PCI Utilities
net-tools      1.60+git20161116.90da8a0-1ubuntu1 arm64       NET-3 networking toolkit
libblas-dev:arm64 3.7.1-4ubuntu1 arm64       Basic Linear Algebra Subroutines 3, static library
gfortran       4:7.4.0-1ubuntu2.3 arm64       GNU Fortran 95 compiler
libblas3:arm64 3.7.1-4ubuntu1 arm64     Basic Linear Algebra Reference implementations, shared library
```

否则请执行如下安装命令（如果只有部分软件未安装，则如下命令修改为还未安装的软件即可）：

说明

- 如果使用root用户安装依赖，请将**步骤1至步骤2**命令中的sudo删除。
- 如果python及其依赖是使用非root用户安装，则需要执行**su - *username***命令切换到非root用户继续执行**步骤1至步骤3**。
- libsqlite3-dev需要在python安装之前安装，如果用户操作系统已经安装满足版本要求的python环境，在此之后再安装libsqlite3-dev，则需要重新编译python环境。

```
sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev
unzip pciutils net-tools libblas-dev gfortran libblas3
```

步骤2 检查系统是否安装满足版本要求的python开发环境（具体要求请参见**6.3.2 依赖列表**，此步骤以环境上需要使用python 3.7.*x*为例进行说明）。

执行命令**python3 --version**，如果返回信息满足python版本要求，则直接进入下一步。

否则可参考如下方式安装python3.7.5。

1. 使用wget下载python3.7.5源码包，可以下载到安装环境的任意目录，命令为：
   ```
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```

 版权所有 © 华为技术有限公司

2. 进入下载后的目录，解压源码包，命令为：
```
tar -zxvf Python-3.7.5.tgz
```
3. 进入解压后的文件夹，执行配置、编译和安装命令：
```
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
make
sudo make install
```
其中“--prefix”参数用于指定python安装路径，用户根据实际情况进行修改。“--enable-shared”参数用于编译出libpython3.7m.so.1.0动态库。“--enable-loadable-sqlite-extensions”参数用于加载libsqlite3-dev依赖。

本手册以--prefix=/usr/local/python3.7.5路径为例进行说明。执行配置、编译和安装命令后，安装包在/usr/local/python3.7.5路径，libpython3.7m.so.1.0动态库在/usr/local/python3.7.5/lib/libpython3.7m.so.1.0路径。

4. 设置python3.7.5环境变量。
```
#用于设置python3.7.5库文件路径
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
#如果用户环境存在多个python3版本，则指定使用python3.7.5版本
export PATH=/usr/local/python3.7.5/bin:$PATH
```
通过以上export方式设置环境变量，该种方式设置的环境变量只在当前窗口有效。您也可以通过将以上命令写入~/.bashrc文件中，然后执行**source ~/.bashrc**命令，使上述环境变量永久生效。注意如果后续您有使用环境上其他python版本的需求，则不建议将以上命令写入到~/.bashrc文件中。

5. 安装完成之后，执行如下命令查看安装版本，如果返回相关版本信息，则说明安装成功。
```
python3 --version
pip3 --version
```

步骤3 安装前请先使用**pip3 list**命令检查是否安装相关依赖，若已经安装，则请跳过该步骤；若未安装，则安装命令如下（如果只有部分软件未安装，则如下命令修改为只安装还未安装的软件即可）。

- 请在安装前配置好pip源，具体可参考**A.10 配置pip源**。
- 安装前，建议执行命令**pip3 install --upgrade pip**进行升级，避免因pip版本过低导致安装失败。
- 如下命令如果使用非root用户安装，需要在安装命令后加上**--user**，例如：**pip3 install attrs --user**，安装命令可在任意路径下执行。

```
pip3 install attrs
pip3 install numpy
pip3 install decorator
pip3 install sympy
pip3 install cffi
pip3 install pyyaml
pip3 install pathlib2
pip3 install psutil
pip3 install protobuf
pip3 install scipy
pip3 install requests
pip3 install absl-py
```

如果执行上述命令时报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”，请参见**B.6 pip3 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

 版权所有 © 华为技术有限公司

> **说明**
>
> 依赖安装完成后，请用户恢复为原umask值（参考**检查root用户的umask**，删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

6.3.4 安装步骤（CentOS 7.6）

检查源

安装过程需要下载相关依赖，请确保安装环境能够连接网络。

请在root用户下执行如下命令检查源是否可用。

```
yum makecache
```

如果命令执行报错或者后续安装依赖时等待时间过长甚至报错，则检查网络是否连接或者把"/etc/yum.repos.d/xxxx.repo"文件中的源更换为可用的源或使用镜像源（以配置华为镜像源为例，可参考**华为开源镜像站**）。

> **说明**
>
> 如果执行上述命令提示“Your license is invalid”，请获取OS授权license。

检查 root 用户的 umask

1. 以root用户登录安装环境。
2. 检查root用户的umask值。

   ```
   umask
   ```

3. 如果umask不等于0022，请执行如下操作配置，在该文件的最后一行添加umask 0022后保存。
   a. 在任意目录下执行如下命令，打开**.bashrc**文件：

      ```
      vi ~/.bashrc
      ```

      在文件最后一行后面添加umask 0022内容。
   b. 执行**:wq!**命令保存文件并退出。
   c. 执行**source ~/.bashrc**命令使其立即生效。

> **说明**
>
> 依赖安装完成后，请用户恢复为原umask值（删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

配置安装用户权限

可使用root或非root用户（该非root用户需与软件包安装用户保持一致）安装依赖，如果使用非root用户安装，可能需要用到提权命令，请用户自行获取所需的sudo权限。使用完成后请取消涉及高危命令的权限，否则有sudo提权风险。

配置最大线程数

训练场景下，不同OS的最大线程数可能不满足训练要求，需执行以下命令修改最大线程数为无限制。

1. 以root用户登录安装环境。

 版权所有 © 华为技术有限公司

2. 配置环境变量，修改线程数为无限制，编辑“/etc/profile”文件，在文件的最后添加如下内容后保存退出：
```
ulimit -u unlimited
```
3. 执行如下命令使环境变量生效。
```
source /etc/profile
```

安装依赖

步骤1 检查系统是否安装python依赖以及gcc等软件。

分别使用如下命令检查是否安装gcc，make以及python依赖软件等。其中python3-devel在openEuler系统上需要安装。

```
gcc --version
g++ --version
make --version
cmake --version
rpm -qa |grep unzip
rpm -qa |grep zlib-devel
rpm -qa |grep libffi-devel
rpm -qa |grep openssl-devel
rpm -qa |grep pciutils
rpm -qa |grep net-tools
rpm -qa |grep sqlite-devel
rpm -qa |grep lapack-devel
rpm -qa |grep gcc-gfortran
rpm -qa |grep python3-devel
```

若分别返回如下信息则说明已经安装，进入下一步（以下回显仅为示例，版本要求请以**6.3.2 依赖列表**为准）。

```
gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)
g++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)
GNU Make 3.82
cmake version 3.20.5
unzip-6.0-21.el7.aarch64
zlib-devel-1.2.7-18.el7.aarch64
libffi-devel-3.0.13-18.el7.aarch64
openssl-devel-1.1.1c-15.el8.aarch64
pciutils-3.5.1-3.el7.aarch64
net-tools-2.0-0.25.20131004git.el7.aarch64
sqlite-devel-3.7.17-8.el7_7.1.aarch64
lapack-devel-3.4.2-8.el7.aarch64
gcc-gfortran-4.8.5-39.el7.aarch64
python3-devel-3.7.4-8.oe1.aarch64
```

否则请执行如下安装命令（如果只有部分软件未安装，则如下命令修改为还未安装的软件即可）：

说明

- 如果使用root用户安装依赖，请将**步骤1至步骤2**命令中的sudo删除。
- 如果python及其依赖是使用非root用户安装，则需要执行**su - *username***命令切换到非root用户继续执行**步骤1至步骤3**。
- sqlite-devel需要在python安装之前安装，如果用户操作系统已经安装满足版本要求的python环境，在此之后再安装sqlite-devel，则需要重新编译python环境。

```
sudo yum install -y gcc gcc-c++ make cmake unzip zlib-devel libffi-devel openssl-devel pciutils net-tools sqlite-devel lapack-devel gcc-gfortran python3-devel
```

如果通过上述方式安装的cmake版本低于3.5.1，则请参见**A.6 安装3.5.2版本cmake**解决。

步骤2 检查系统是否安装满足版本要求的python开发环境（具体要求请参见**6.3.2 依赖列表**，此步骤以环境上需要使用python 3.7.*x*为例进行说明）。

 版权所有 © 华为技术有限公司

执行命令**python3 --version**，如果返回信息满足python版本要求，则直接进入下一步。

否则可参考如下方式安装python3.7.5。

1. 使用wget下载python3.7.5源码包，可以下载到安装环境的任意目录，命令为：
```
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
```
2. 进入下载后的目录，解压源码包，命令为：
```
tar -zxvf Python-3.7.5.tgz
```
3. 进入解压后的文件夹，创建安装目录，执行配置、编译和安装命令：
```
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
make
sudo make install
```
其中“--prefix”参数用于指定python安装路径，用户根据实际情况进行修改，“--enable-shared”参数用于编译出libpython3.7m.so.1.0动态库，“--enable-loadable-sqlite-extensions”参数用于加载sqlite-devel依赖。

本手册以--prefix=/usr/local/python3.7.5路径为例进行说明。执行配置、编译和安装命令后，安装包在/usr/local/python3.7.5路径，libpython3.7m.so.1.0动态库在/usr/local/python3.7.5/lib/libpython3.7m.so.1.0路径。
4. 设置python3.7.5环境变量。
```
#用于设置python3.7.5库文件路径
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
#如果用户环境存在多个python3版本，则指定使用python3.7.5版本
export PATH=/usr/local/python3.7.5/bin:$PATH
```
通过以上export方式设置环境变量，该种方式设置的环境变量只在当前窗口有效。您也可以通过将以上命令写入~/.bashrc文件中，然后执行**source ~/.bashrc**命令，使上述环境变量永久生效。注意如果后续您有使用环境上其他python版本的需求，则不建议将以上命令写入到~/.bashrc文件中。
5. 安装完成之后，执行如下命令查看安装版本，如果返回相关版本信息，则说明安装成功。
```
python3 --version
pip3 --version
```

步骤3 安装前请先使用**pip3 list**命令检查是否安装相关依赖，若已经安装，则请跳过该步骤；若未安装，则安装命令如下（如果只有部分软件未安装，则如下命令修改为只安装还未安装的软件即可）。

- 请在安装前配置好pip源，具体可参考**A.10 配置pip源**。
- 安装前，建议执行命令**pip3 install --upgrade pip**进行升级，避免因pip版本过低导致安装失败。
- 如下命令如果使用非root用户安装，需要在安装命令后加上**--user**，例如：**pip3 install attrs --user**，安装命令可在任意路径下执行。

```
pip3 install attrs
pip3 install numpy
pip3 install decorator
pip3 install sympy
pip3 install cffi
pip3 install pyyaml
pip3 install pathlib2
pip3 install psutil
pip3 install protobuf
pip3 install scipy
pip3 install requests
pip3 install absl-py
```

- 如果安装numpy报错，请参考**B.5 pip3 install numpy报错**解决。

 版权所有 © 华为技术有限公司

- 如果安装scipy报错，请参考**B.4 pip3 install scipy报错**解决。

步骤4 如果仅需支持离线推理，请跳过此步骤。

CentOS7.6系统使用软件源默认安装的gcc版本为4.8.5，因此需要安装7.3.0版本gcc，安装过程请参见**A.7 安装7.3.0版本gcc**。

----结束

说明

依赖安装完成后，请用户恢复为原umask值（参考**检查root用户的umask**，删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

6.3.5 安装步骤（SLES 12.5）

检查源

安装过程需要下载相关依赖，请确保安装环境能够连接网络。

请在root用户下执行如下命令检查源是否可用。

```
zypper ref
```

如果命令执行报错或者后续安装依赖时等待时间过长甚至报错，可参考以下操作配置本地源。

请以root用户执行如下操作。

1. 挂载系统镜像iso文件（以下命令为示例）。
   ```
   mount -o loop SLE-12-SP5-Server-DVD-x86_64-GM-DVD1.iso /mnt
   ```
2. 使用ar参数添加一个mnt路径下挂好的本地源，并命名为suse（用户可自定义）。
   ```
   zypper ar -f /mnt suse
   ```
3. 清空zypper缓存，刷新源，使配置的源生效。
   ```
   zypper clean
   zypper ref
   ```

检查 root 用户的 umask

1. 以root用户登录安装环境。
2. 检查root用户的umask值。
   ```
   umask
   ```
3. 如果umask不等于0022，请执行如下操作配置，在该文件的最后一行添加umask 0022后保存。
   a. 在任意目录下执行如下命令，打开**.bashrc**文件：
      ```
      vi ~/.bashrc
      ```
      在文件最后一行后面添加umask 0022内容。
   b. 执行**:wq!**命令保存文件并退出。
   c. 执行**source ~/.bashrc**命令使其立即生效。

说明

依赖安装完成后，请用户恢复为原umask值（删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

 版权所有 © 华为技术有限公司

配置安装用户权限

可使用root或非root用户（该非root用户需与软件包安装用户保持一致）安装依赖，如果使用非root用户安装，可能需要用到提权命令，请用户自行获取所需的sudo权限。使用完成后请取消涉及高危命令的权限，否则有sudo提权风险。

安装依赖

步骤1 检查系统是否安装python依赖以及gcc等软件。

分别使用如下命令检查是否安装gcc，make以及python依赖软件等。

```
rpm -qa | grep gcc
rpm -qa | grep make
rpm -qa | grep unzip
rpm -qa | grep zlib-devel
rpm -qa | grep openssl-devel
rpm -qa | grep pciutils
rpm -qa | grep net-tools
rpm -qa | grep gdbm-devel
rpm -qa | grep libffi-devel
```

若分别返回如下信息则说明已经安装，进入下一步（以下回显仅为示例，版本要求请以**6.3.2 依赖列表**为准）。

```
gcc48-4.8.5-31.20.1.x86_64
libgcc_s1-8.2.1+r264010-1.3.3.x86_64
gcc-4.8-6.189.x86_64
libgcc_s1-32bit-8.2.1+r264010-1.3.3.x86_64
gcc-c++-4.8-6.189.x86_64
gcc48-c++-4.8.5-31.20.1.x86_64
cmake-3.5.2-20.6.1.x86_64
makedumpfile-1.6.5-1.19.x86_64
automake-1.13.4-6.2.noarch
make-4.0-4.1.x86_64
unzip-6.00-33.8.1.x86_64
zlib-devel-1.2.11-3.21.1.x86_64
zlib-devel-32bit-1.2.11-3.21.1.x86_64
zlib-devel-static-1.2.11-3.21.1.x86_64
zlib-devel-1.2.11-9.42.x86_64
zlib-devel-static-32bit-1.2.11-3.21.1.x86_64
libopenssl-devel-1.0.2p-1.13.noarch
pciutils-3.2.1-11.3.1.x86_64
pciutils-ids-2018.02.08-12.3.1.noarch
net-tools-1.60-765.5.4.x86_64
gdbm-devel-1.10-9.70.x86_64
libffi-devel-3.2.1.git259-10.8.x86_64
```

否则请执行如下安装命令（如果只有部分软件未安装，则如下命令修改为还未安装的软件即可）：

说明

- 如果使用root用户安装依赖，请将**步骤1至步骤2**命令中的sudo删除。
- 如果python及其依赖是使用非root用户安装，则需要执行**su - *username***命令切换到非root用户继续执行**步骤1至步骤3**。

```
sudo zypper install -y gcc gcc-c++ make cmake unzip zlib-devel openssl-devel pciutils net-tools gdbm-devel
```

由于本地源中缺少libffi-devel依赖，可从**opensuse镜像源**中下载libffi-devel-3.2.1.git259-10.8.x86_64.rpm、libffi7-3.2.1.git259-10.8.x86_64.rpm、libffi7-32bit-3.2.1.git259-10.8.x86_64.rpm（软件包会定时更新，请以实际rpm包名为准）并一同上传至服务器某一目录下（如“/home/test”）。

进入rpm包所在路径（如“/home/test”），执行如下命令安装所需依赖：

 版权所有 © 华为技术有限公司

```
sudo rpm -ivh *.rpm --nodeps
```

步骤2 检查系统是否安装满足版本要求的python开发环境（具体要求请参见**6.3.2 依赖列表**，此步骤以环境上需要使用python 3.7.*x*为例进行说明）。

执行命令**python3 --version**，如果返回信息满足python版本要求，则直接进入下一步。

否则可参考如下方式安装python3.7.5。

1. 使用wget下载python3.7.5源码包，可以下载到安装环境的任意目录，命令为：
   ```
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```
2. 进入下载后的目录，解压源码包，命令为：
   ```
   tar -zxvf Python-3.7.5.tgz
   ```
3. 进入解压后的文件夹，创建安装目录，执行配置、编译和安装命令：
   ```
   cd Python-3.7.5
   ./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
   make
   sudo make install
   ```
   其中“--prefix”参数用于指定python安装路径，用户根据实际情况进行修改，“--enable-shared”参数用于编译出libpython3.7m.so.1.0动态库。

   本手册以--prefix=/usr/local/python3.7.5路径为例进行说明。执行配置、编译和安装命令后，安装包在/usr/local/python3.7.5路径，libpython3.7m.so.1.0动态库在/usr/local/python3.7.5/lib/libpython3.7m.so.1.0路径。
4. 设置python3.7.5环境变量。
   ```
   #用于设置python3.7.5库文件路径
   export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
   #如果用户环境存在多个python3版本，则指定使用python3.7.5版本
   export PATH=/usr/local/python3.7.5/bin:$PATH
   ```
   通过以上export方式设置环境变量，该种方式设置的环境变量只在当前窗口有效。您也可以通过将以上命令写入~/.bashrc文件中，然后执行**source ~/.bashrc**命令，使上述环境变量永久生效。注意如果后续您有使用环境上其他python版本的需求，则不建议将以上命令写入到~/.bashrc文件中。
5. 安装完成之后，执行如下命令查看安装版本，如果返回相关版本信息，则说明安装成功。
   ```
   python3 --version
   pip3 --version
   ```

步骤3 安装前请先使用**pip3 list**命令检查是否安装相关依赖，若已经安装，则请跳过该步骤；若未安装，则安装命令如下（如果只有部分软件未安装，则如下命令修改为只安装还未安装的软件即可）。

- 请在安装前配置好pip源，具体可参考**A.10 配置pip源**。
- 安装前，建议执行命令**pip3 install --upgrade pip**进行升级，避免因pip版本过低导致安装失败。
- 如下命令如果使用非root用户安装，需要在安装命令后加上**--user**，例如：**pip3 install attrs --user**，安装命令可在任意路径下执行。

```
pip3 install attrs
pip3 install numpy
pip3 install decorator
pip3 install sympy
pip3 install cffi
pip3 install pyyaml
pip3 install pathlib2
pip3 install psutil
pip3 install protobuf
pip3 install scipy
pip3 install requests
```

 版权所有 © 华为技术有限公司

```
pip3 install gnureadline
pip3 install absl-py
```

----结束

说明

依赖安装完成后，请用户恢复为原umask值（参考**检查root用户的umask**，删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

6.4 在非昇腾设备上安装

6.4.1 安装开发套件包

前提条件

- 请参见**6.3 安装依赖**完成安装前准备。
- 安装开发套件包前请确保安装目录可用空间大于7G，如不满足请清理空间或更换安装目录。
- 如果用户获取的软件包格式为***.deb**或***.rpm**，可参考**A.2 安装和卸载CANN软件包（适用于.deb格式）**或**A.3 安装和卸载CANN软件包（适用于.rpm格式）**进行安装。

安装步骤

步骤1 以软件包的安装用户登录安装环境。

若**6.3 安装依赖**中安装依赖的用户为root用户，则软件包的安装用户可自行指定；若**6.3 安装依赖**中安装依赖的用户为非root用户，请确保软件包的安装用户与该用户保持一致。

步骤2 将获取到的开发套件包上传到安装环境任意路径（如“/home/package”）。

步骤3 进入软件包所在路径。

步骤4 增加对软件包的可执行权限。

```
chmod +x 软件包名.run
```

其中*软件包名*.**run**表示开发套件包Ascend-cann-toolkit_*{version}*_linux-*{arch}*.run，请根据实际包名进行替换。

步骤5 执行如下命令校验软件包安装文件的一致性和完整性。

```
./软件包名.run --check
```

步骤6 执行如下命令安装软件（以下命令支持**--install-path=**<*path*>等参数，具体参数说明请参见**C.1 参数说明**）。

```
./软件包名.run --install
```

 版权所有 © 华为技术有限公司

说明

- 如果以root用户安装，**不允许安装在非root用户目录下**。
- 如果用户未指定安装路径，则软件会安装到默认路径下，默认安装路径如下。
  - root用户：“/usr/local/Ascend”
  - 非root用户：“*${HOME}*/Ascend”
    其中${HOME}为当前用户目录。
- 软件包安装详细日志路径如下。
  - root用户：“/var/log/ascend_seclog/ascend_toolkit_install.log”
  - 非root用户：“*${HOME}*/var/log/ascend_seclog/ascend_toolkit_install.log”
    其中${HOME}为当前用户目录。

安装完成后，若显示如下信息，则说明软件安装成功：

```
xxx install success
```

*xxx*表示安装的实际软件包名。

----结束

6.4.2 配置环境变量

CANN软件提供进程级环境变量设置脚本，供用户在进程中引用，以自动完成环境变量设置。用户进程结束后自动失效。示例如下（以root用户默认安装路径为例）：

```
# 安装toolkit包时配置
source /usr/local/Ascend/ascend-toolkit/set_env.sh
# 其中<arch>请替换为实际架构
export LD_LIBRARY_PATH=/usr/local/Ascend/ascend-toolkit/latest/<arch>-linux/devlib/:$LD_LIBRARY_PATH
```

用户也可以通过修改~/.bashrc文件方式设置永久环境变量，操作如下：

1. 以运行用户在任意目录下执行**vi ~/.bashrc**命令，打开**.bashrc**文件，在文件最后一行后面添加上述内容。
2. 执行**:wq!**命令保存文件并退出。
3. 执行**source ~/.bashrc**命令使其立即生效。

6.4.3 配置交叉编译环境

若开发环境和运行环境上的操作系统架构不同，需在开发环境中使用交叉编译工具，并使用运行环境架构的库文件进行编译，这样编译出来的可执行文件，才可以在运行环境中执行。具体如**表6-5**所示。

 版权所有 © 华为技术有限公司

表 6-5 安装交叉编译工具

开发环境架构	运行环境架构	编译环境配置
x86_64	aarch64	• 请使用软件包的安装用户，在开发环境执行**aarch64-linux-gnu-g++ --version**命令检查是否安装g++交叉编译工具，若已经安装则可以忽略。安装命令示例如下（以下命令仅为示例，请用户根据实际情况替换，如果使用root用户安装依赖，请删除命令中的**sudo**）： **sudo apt-get install g++-aarch64-linux-gnu** • aarch64架构的库文件所在路径（以root用户默认安装路径为例）：/usr/local/Ascend/ascend-toolkit/latest/runtime/lib64/stub/aarch64。

6.5 在昇腾设备上安装

6.5.1 安装开发套件包

前提条件

- 请参见**6.3 安装依赖**完成安装前准备。
- 通过**6.1 准备软件包**章节获取开发套件包Ascend-cann-toolkit_*xxx*.run。
- 安装开发套件包前请确保安装目录可用空间大于7G，如不满足请清理空间或更换安装目录。
- 如果用户获取的软件包格式为***.deb**或***.rpm**，可参考**A.2 安装和卸载CANN软件包（适用于.deb格式）**或**A.3 安装和卸载CANN软件包（适用于.rpm格式）**进行安装。

安装步骤

步骤1 以软件包的安装用户登录安装环境。

若**6.3 安装依赖**中安装依赖的用户为root用户，则软件包的安装用户可自行指定；若**6.3 安装依赖**中安装依赖的用户为非root用户，请确保软件包的安装用户与该用户保持一致。

步骤2 将获取到的开发套件包上传到安装环境任意路径（如“/home/package”）。

步骤3 进入软件包所在路径。

步骤4 增加对软件包的可执行权限。

chmod +x *软件包名*.**run**

软件包名.**run**表示开发套件包Ascend-cann-toolkit_*{version}*_linux-*{arch}*.run，请根据实际包名进行替换。

步骤5 执行如下命令校验软件包安装文件的一致性和完整性。

./*软件包名*.**run --check**

 版权所有 © 华为技术有限公司

步骤6 执行以下命令安装软件（以下命令支持**--install-path=**<*path*>等参数，具体参数说明请参见**C.1 参数说明**）。

```
./软件包名.run --install
```

说明

- 开发套件包支持不同用户在同一开发环境安装，但安装版本必须保持一致，不同用户所属的属组必须和驱动运行用户所属属组相同；如果不同，请用户自行添加到驱动运行用户属组。
- 如果以root用户安装，**不允许安装在非root用户目录下**。
- 如果用户未指定安装路径，则软件会安装到默认路径下，默认安装路径如下。
  - root用户：“/usr/local/Ascend”
  - 非root用户：“*${HOME}*/Ascend”
    其中${HOME}为当前用户目录。
- 软件包安装详细日志路径如下。
  - root用户：“/var/log/ascend_seclog/ascend_toolkit_install.log”
  - 非root用户：“*${HOME}*/var/log/ascend_seclog/ascend_toolkit_install.log”
    其中${HOME}为当前用户目录。

安装完成后，若显示如下信息，则说明软件安装成功：

```
xxx install success
```

*xxx*表示安装的实际软件包名。

----结束

说明

在包含动态shape网络或ACLNN单算子直调的场景下须安装二进制算子包，具体操作请参考**A.1 安装、升级和卸载二进制算子包**。

6.5.2 安装框架插件包

在线推理或训练场景下若使用深度学习框架TensorFlow，需要安装框架插件包Ascend-cann-tfplugin_*xxx*.run。

请参照**6.5.1 安装开发套件包**的步骤安装框架插件包。其中软件包安装详细日志路径如下。

- root用户：“/var/log/ascend_seclog/ascend_tfplugin_install.log”
- 非root用户：“*${HOME}*/var/log/ascend_seclog/ascend_tfplugin_install.log”
  其中${HOME}为当前用户目录。

6.5.3 配置环境变量

CANN软件提供进程级环境变量设置脚本，供用户在进程中引用，以自动完成环境变量设置。用户进程结束后自动失效。示例如下（以root用户默认安装路径为例）：

```
# 安装toolkit包时配置
. /usr/local/Ascend/ascend-toolkit/set_env.sh

# 安装tfplugin包时配置
. /usr/local/Ascend/tfplugin/set_env.sh
```

用户也可以通过修改~/.bashrc文件方式设置永久环境变量，操作如下：

1. 以运行用户在任意目录下执行**vi ~/.bashrc**命令，打开**.bashrc**文件，在文件最后一行后面添加上述内容。

 版权所有 © 华为技术有限公司

2. 执行**:wq!**命令保存文件并退出。
3. 执行**source ~/.bashrc**命令使其立即生效。

6.5.4 安装深度学习框架

6.5.4.1 安装 TensorFlow

若用户仅进行离线推理，请跳过此章节。

安装前准备

- 对于x86架构，跳过安装前准备。
- 对于aarch64架构。

  由于TensorFlow依赖h5py，而h5py依赖HDF5，需要先编译安装HDF5，否则使用pip安装h5py会报错，以下步骤以root用户操作。

  - 编译安装HDF5。
    i. 访问**下载链接**下载HDF5源码包，并上传到安装环境的任意目录。
    ii. 进入源码包所在目录，执行如下命令解压源码包。
    ```
    tar -zxvf hdf5-1.10.5.tar.gz
    ```
    iii. 进入解压后的文件夹，执行配置、编译和安装命令：
    ```
    cd hdf5-1.10.5/
    ./configure --prefix=/usr/include/hdf5
    make
    make install
    ```
    iv. 配置环境变量并建立动态链接库软连接。
    1) 配置环境变量。
    ```
    export CPATH="/usr/include/hdf5/include/:/usr/include/hdf5/lib/"
    ```
    2) root用户建立动态链接库软连接命令如下，非root用户需要在以下命令前添加sudo。
    ```
    ln -s /usr/include/hdf5/lib/libhdf5.so  /usr/lib/libhdf5.so
    ln -s /usr/include/hdf5/lib/libhdf5_hl.so  /usr/lib/libhdf5_hl.so
    ```
  - 安装h5py。

    root用户下执行如下命令安装h5py依赖包。
    ```
    pip3 install Cython==0.29
    ```
    root用户下执行如下命令安装h5py。
    ```
    pip3 install h5py==2.8.0
    ```

安装 TensorFlow

说明

- TensorFlow1.15配套的Python版本是：Python3.7.x（3.7.5~3.7.11）。
- TensorFlow2.6.5配套的Python版本是：Python3.7.x（3.7.5~3.7.11）、Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。
- TensorFlow2.6.5存在漏洞，请参考**相关漏洞及其修复方案**处理。

需要安装TensorFlow才可以进行算子开发验证、训练业务开发。

- 对于x86架构：直接从pip源下载即可，系统要求等具体请参考**TensorFlow官网**。需要注意TensorFlow官网提供的指导描述有误，从pip源下载CPU版本需要显式指

 版权所有 © 华为技术有限公司

定tensorflow-cpu，如果不指定CPU，默认下载的是GPU版本。安装命令参考如下：

如下命令如果使用非root用户安装，需要在安装命令后加上**--user**，例如：**pip3 install tensorflow-cpu==1.15 --user**

- 安装TensorFlow1.15
  ```
  pip3 install tensorflow-cpu==1.15
  ```
- 安装TensorFlow2.6.5
  ```
  pip3 install tensorflow-cpu==2.6.5
  ```

- 对于aarch64架构：由于pip源未提供对应的版本，所以需要用户使用官网要求的linux_gcc7.3.0编译器编译tensorflow1.15.0或tensorflow2.6.5，编译步骤参考官网**TensorFlow官网**。特别注意点如下。

  在下载完tensorflow tag v1.15.0或tensorflow tag v2.6.5源码后需要执行如下步骤。

步骤1 下载“nsync-1.22.0.tar.gz”源码包。

1. 进入源码目录，TensorFlow1.15场景下打开“tensorflow/workspace.bzl”文件，TensorFlow2.6.5场景下打开“tensorflow/workspace2.bzl”文件，找到其中name为nsync的“tf_http_archive”定义。
   ```
   tf_http_archive(
       name = "nsync",
       sha256 = "caf32e6b3d478b78cff6c2ba009c3400f8251f646804bcb65465666a9cea93c4",
       strip_prefix = "nsync-1.22.0",
       system_build_file = clean_dep("//third_party/systemlibs:nsync.BUILD"),
       urls = [          "https://storage.googleapis.com/mirror.tensorflow.org/github.com/google/nsync/archive/1.22.0.tar.gz",
           "https://github.com/google/nsync/archive/1.22.0.tar.gz",
       ],
   )
   ```
2. 从urls中的任一路径下载nsync-1.22.0.tar.gz的源码包，保存到任意路径。

步骤2 修改“nsync-1.22.0.tar.gz”源码包。

1. 切换到nsync-1.22.0.tar.gz所在路径，解压缩该源码包。解压缩后存在“nsync-1.22.0”文件夹。
2. 编辑“nsync-1.22.0/platform/c++11/atomic.h”。

   在NSYNC_CPP_START_内容后添加如下加粗字体内容。
   ```
   #include "nsync_cpp.h"
   #include "nsync_atomic.h"

   NSYNC_CPP_START_

   #define ATM_CB_() __sync_synchronize()

   static INLINE int atm_cas_nomb_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
       int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p), &o, n, std::memory_order_relaxed, std::memory_order_relaxed));
       ATM_CB_();
       return result;
   }
   static INLINE int atm_cas_acq_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
       int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p), &o, n, std::memory_order_acquire, std::memory_order_relaxed));
       ATM_CB_();
       return result;
   }
   static INLINE int atm_cas_rel_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
       int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p), &o, n, std::memory_order_release, std::memory_order_relaxed));
       ATM_CB_();
       return result;
   ```

 版权所有 © 华为技术有限公司

```
}
static INLINE int atm_cas_relacq_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
    int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p),
&o, n, std::memory_order_acq_rel, std::memory_order_relaxed));
    ATM_CB_();
    return result;
}
```

步骤3 重新压缩“nsync-1.22.0.tar.gz”源码包。

将上个步骤中解压出的内容压缩为一个新的“nsync-1.22.0.tar.gz”源码包，保存（比如，保存在“/tmp/nsync-1.22.0.tar.gz”）。

步骤4 重新生成“nsync-1.22.0.tar.gz”源码包的sha256sum校验码。

执行如下命令后得到sha256sum校验码（一串数字和字母的组合）。

```
sha256sum /tmp/nsync-1.22.0.tar.gz
```

步骤5 修改sha256sum校验码和urls。

进入tensorflow tag源码目录，TensorFlow1.15场景下打开“tensorflow/workspace.bzl”文件，TensorFlow2.6.5场景下打开“tensorflow/workspace2.bzl”文件，找到其中name为nsync的“tf_http_archive”定义，其中“sha256=”后面的数字填写**步骤4**得到的校验码，“urls=”后面的列表第二行，填写存放“nsync-1.22.0.tar.gz”的file:///索引。

```
  tf_http_archive(
      name = "nsync",
      sha256 = "caf32e6b3d478b78cff6c2ba009c3400f8251f646804bcb65465666a9cea93c4",
      strip_prefix = "nsync-1.22.0",
      system_build_file = clean_dep("//third_party/systemlibs:nsync.BUILD"),
      urls = [
          "https://storage.googleapis.com/mirror.tensorflow.org/github.com/google/nsync/archive/
1.22.0.tar.gz",
          "file:///tmp/nsync-1.22.0.tar.gz ",
          "https://github.com/google/nsync/archive/1.22.0.tar.gz",
      ],
  )
```

步骤6 继续参考官网的“配置build”（**TensorFlow官网**）章节执行编译。

执行完**./configure**之后，需要修改 .tf_configure.bazelrc 配置文件。

添加如下一行build编译选项：

```
build:opt --cxxopt=-D_GLIBCXX_USE_CXX11_ABI=0
```

删除以下两行：

```
build:opt --copt=-march=native
build:opt --host_copt=-march=native
```

步骤7 继续执行官方的编译指导步骤（**TensorFlow官网**）即可。

步骤8 安装编译好的TensorFlow。

以上步骤执行完后会打包TensorFlow到指定目录，进入指定目录后执行如下命令安装：

如下命令如果使用非root用户安装，需要在安装命令后加上**--user**，例如：**pip3 install tensorflow-1.15.0-*.whl --user**

- TensorFlow1.15：
  ```
  pip3 install tensorflow-1.15.0-*.whl
  ```
- TensorFlow2.6.5：
  ```
  pip3 install tensorflow-2.6.5-*.whl
  ```

 版权所有 © 华为技术有限公司

步骤9 执行如下命令验证安装效果。

```
python3 -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
```

如果返回了张量则表示安装成功。

----结束

6.5.4.2 安装 PyTorch

安装前请根据**表6-6**完成相应配套版本CANN软件的安装。若用户需要详细了解CANN软件和其安装流程，可从**1 安装须知**章节开始了解手册内容。若用户仅进行离线推理，请跳过此章节。

昇腾开发PyTorch Adapter插件用于适配PyTorch框架，为使用PyTorch框架的开发者提供昇腾AI处理器的超强算力，本章节指导用户安装PyTorch框架和PyTorch Adapter插件。

安装场景

说明

- PyTorch 1.8.1/1.11.0配套的Python版本是：Python3.7.x（3.7.5~3.7.11）、Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。
- PyTorch 2.0.1配套的Python版本是：Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。

根据不同的PyTorch使用场景，用户可以选择以下三种方式安装PyTorch：

- 获取镜像下载安装PyTorch：若用户希望在容器中安装使用PyTorch，可以**点击链接**前往AscendHub昇腾PyTorch镜像仓库，根据**表6-6**选择对应版本的镜像下载使用。
- 二进制whl包安装：推荐用户使用编好的二进制whl包安装PyTorch。

  请前往**安装PyTorch环境依赖**继续安装流程。
- 编译安装：请前往**A.5 编译安装PyTorch**进行安装。

如果用户不确定自己的软件包版本，可参考**A.11 查询软件包版本信息**进行查询。

表 6-6 Ascend 配套软件

AscendPyTorch版本	CANN版本	支持PyTorch版本	AscendHub镜像版本/名称
5.0.rc2	CANN 6.3.RC2	1.8.1.post2	23.0.RC2-1.8.1
		1.11.0.post1	23.0.RC2-1.11.0
		2.0.1.rc1	-

安装 PyTorch 环境依赖

执行如下命令安装。如果使用非root用户安装，需要在命令后加**--user**，例如：**pip3 install pyyaml --user**。

 版权所有 © 华为技术有限公司

```
pip3 install pyyaml
pip3 install wheel
pip3 install typing_extensions
```

安装 PyTorch

用户可选择编译安装方式安装PyTorch。请参考**A.5 编译安装PyTorch**。

步骤1 安装官方torch包。

- x86_64

  说明

  在x86_64架构下，官方torch包使用MKL加速库。如果需要使用其他blas和lapack加速库（如openblas）来提升性能，请使用源码编译安装方式安装官方torch包，步骤请参考**A.5 编译安装PyTorch**章节。

  ```
  # 安装1.8.1版本
  pip3 install torch==1.8.1+cpu
  # 安装1.11.0版本
  pip3 install torch==1.11.0+cpu
  # 安装2.0.1版本
  pip3 install torch==2.0.1+cpu
  ```

  若执行以上命令安装cpu版本PyTorch报错，请点击下方PyTorch官方链接下载whl包。

  PyTorch 1.8.1版本：**下载链接**。

  PyTorch 1.11.0版本：**下载链接**。

  PyTorch 2.0.1版本：**下载链接**。

  执行如下安装命令：

  ```
  # 用户请根据自己实际情况修改命令中的安装包名
  pip3 install torch-1.8.1+cpu-cp37-cp37m-linux_x86_64.whl
  ```

- aarch64

  a. 获取安装包。

  - PyTorch1.8.1/1.11.0：

    进入安装目录，执行如下命令获取鲲鹏文件共享中心上对应版本的whl包。

    ```
    # 安装1.8.1版本
    wget https://repo.huaweicloud.com/kunpeng/archive/Ascend/PyTorch/torch-1.8.1-cp37-cp37m-linux_aarch64.whl
    # 安装1.11.0版本
    wget https://repo.huaweicloud.com/kunpeng/archive/Ascend/PyTorch/torch-1.11.0-cp37-cp37m-linux_aarch64.whl
    ```

  - PyTorch2.0.1：**下载链接**。

    或使用wget命令下载。

    ```
    wget https://download.pytorch.org/whl/torch-2.0.1-cp38-cp38-manylinux2014_aarch64.whl
    ```

  b. 执行如下命令安装。如果使用非root用户安装，需要在命令后加**--user**。

  ```
  # 安装1.8.1版本
  pip3 install torch-1.8.1-cp37-cp37m-linux_aarch64.whl
  # 安装1.11.0版本
  pip3 install torch-1.11.0-cp37-cp37m-linux_aarch64.whl
  # 安装2.0.1版本
  pip3 install torch-2.0.1-cp38-cp38-manylinux2014_aarch64.whl
  ```

步骤2 安装PyTorch插件torch_npu。以下命令以在aarch64架构下安装为例。

 版权所有 © 华为技术有限公司

1. 进入安装目录，执行如下命令获取PyTorch插件的whl包。

```
# 若用户在x86架构下安装插件，请将命令中文件包名中的“aarch64”改为“x86_64”。
# 安装1.8.1版本
wget https://gitee.com/ascend/pytorch/releases/download/v5.0.rc2-pytorch1.8.1/torch_npu-1.8.1.post2-cp37-cp37m-linux_aarch64.whl
# 安装1.11.0版本
wget https://gitee.com/ascend/pytorch/releases/download/v5.0.rc2-pytorch1.11.0/torch_npu-1.11.0.post1-cp37-cp37m-linux_aarch64.whl
# 安装2.0.1版本
wget https://gitee.com/ascend/pytorch/releases/download/v5.0.rc2-pytorch2.0.1/torch_npu-2.0.1rc1-cp38-cp38-linux_aarch64.whl
```

说明

- 如果下载whl包时出现**ERROR: cannot verify gitee.com's certificate**报错，可在下载命令后加上**--no-check-certificate**参数避免此问题。

  样例代码如下所示。

  ```
  wget https://gitee.com/ascend/pytorch/releases/download/v5.0.rc2-pytorch1.11.0/torch_npu-1.11.0.post1-cp37-cp37m-linux_aarch64.whl --no-check-certificate
  ```

- PyTorch 1.11版本在aarch64架构上可能出现CPU精度问题，需在Docker环境编译torch_npu，请参见**B.12 PyTorch框架在ARM CPU上算子计算结果异常**。

2. 执行如下命令安装。如果使用非root用户安装，需要在命令后加**--user**。

```
# 若用户在x86架构下安装插件，请将命令中文件包名中的“aarch64”改为“x86_64”。
# 安装1.8.1版本
pip3 install torch_npu-1.8.1.post2-cp37-cp37m-linux_aarch64.whl
# 安装1.11.0版本
pip3 install torch_npu-1.11.0.post1-cp37-cp37m-linux_aarch64.whl
# 安装2.0.1版本
pip3 install torch_npu-2.0.1rc1-cp38-cp38m-linux_aarch64.whl
```

步骤3 执行如下命令，若返回**True**则说明安装成功。

```
python3 -c "import torch;import torch_npu;print(torch_npu.npu.is_available())"
```

步骤4 安装对应框架版本的torchvision。

```
#PyTorch 1.8.1需安装0.9.1版本，PyTorch 1.11.0需安装0.12.0版本，PyTorch 2.0.1版本需安装0.15.2版本
pip3 install torchvision==0.9.1
```

----结束

安装 APEX 混合精度模块

混合精度训练是在训练时混合使用单精度（float32）与半精度(float16)数据类型，将两者结合在一起，并使用相同的超参数实现了与float32几乎相同的精度。在迁移完成、训练开始之前，基于NPU芯片的架构特性，用户需要开启混合精度，可以提升模型的性能。APEX混合精度模块是一个集优化性能、精度收敛于一身的综合优化库，可以提供不同场景下的混合精度训练支持。混合精度的介绍可参考《**PyTorch 模型迁移和训练指南**》中的“自动混合精度（AMP）”章节，APEX模块的使用介绍可参考《**PyTorch 模型迁移和训练指南**》中的“参考信息>模型套件和第三方库>APEX”章节。

推荐用户通过编译源码包安装APEX模块。

步骤1 安装依赖。

选择编译安装方式安装时需要安装系统依赖。目前支持CentOS与Ubuntu操作系统。

- CentOS

```
yum install -y patch libjpeg-turbo-devel dos2unix openblas git
yum install -y gcc==7.3.0 cmake==3.12.0 #gcc7.3.0版本及以上，cmake3.12.0版本及以上。若用户要安装1.11.0版本PyTorch，则gcc需为7.5.0版本以上。
```

 版权所有 © 华为技术有限公司

- Ubuntu
  ```
  apt-get install -y patch build-essential libbz2-dev libreadline-dev wget curl llvm libncurses5-dev libncursesw5-dev xz-utils tk-dev liblzma-dev m4 dos2unix libopenblas-dev git
  apt-get install -y gcc==7.3.0 cmake==3.12.0 #gcc7.3.0版本及以上，cmake3.12.0版本及以上。若用户要安装1.11.0版本PyTorch，则gcc需为7.5.0版本以上。
  ```

步骤2 获取昇腾适配的APEX源码。

```
git clone -b 分支名称 https://gitee.com/ascend/apex.git
```

分支名称请根据实际使用的PyTorch版本选择，例如使用PyTorch1.11.0-5.0.rc2，则apex分支也使用5.0.rc2。

步骤3 进入昇腾适配的APEX源码目录，执行命令编译生成二进制安装包。

```
cd apex
bash scripts/build.sh
```

说明

请确保NPU版本的PyTorch可以正常使用，否则会影响APEX的编译。

步骤4 执行如下命令安装。如果使用非root用户安装，需要在命令后加**--user**。

```
# 若用户在x86架构下安装，请将命令中文件包名中的“aarch64”改为“x86_64”。
pip3 install apex/dist/apex-0.1_ascend-cp37-cp37m-linux_aarch64.whl
```

----结束

6.5.4.3 安装昇思 MindSpore

若用户仅进行离线推理，请跳过此章节。

用户要使用MindSpore框架，请登录**MindSpore官网**获取安装MindSpore框架的方法。

6.5.5 安装 Python 版本的 proto

如果训练脚本依赖protobuf的Python版本进行序列化结构的数据存储（例如TensorFlow的序列化相关接口），则需要安装Python版本的proto。

步骤1 检查系统中是否存在“/usr/local/python3.7.5/lib/python3.7/site-packages/google/protobuf/pyext/_message.cpython-37m-*<arch>*-linux-gnu.so”（以参考**6.3 安装依赖**章节安装的python3.7.5为例）这个动态库，如果没有，需要按照如下步骤安装。其中*<arch>*为系统架构类型。

说明

“/usr/local/python3.7.5/lib/python3.7/site-packages”是pip安装第三方库的路径，可以使用**pip3 -V**检查。

如果系统显示：/usr/local/python3.7.5/lib/python3.7/site-packages/pip，则pip安装第三方库的路径为/usr/local/python3.7.5/lib/python3.7/site-packages。

步骤2 执行如下命令卸载protobuf。

```
pip3 uninstall protobuf
```

步骤3 下载protobuf软件包。

从**https://github.com/protocolbuffers/protobuf/releases/download/v3.11.3/protobuf-python-3.11.3.tar.gz**路径下下载3.11.3版本protobuf-python-3.11.3.tar.gz软件包（或者其他版本，保证和当前环境上安装的tensorflow兼容），并以root用户上传到所在Linux服务器任意目录并执行命令解压。

```
tar zxvf protobuf-python-3.11.3.tar.gz
```

 版权所有 © 华为技术有限公司

步骤4 以root用户安装protobuf。

进入protobuf软件包目录。

1. 安装protobuf的依赖。

   当操作系统为Ubuntu时，安装命令如下：
   ```
   apt-get install autoconf automake libtool curl make g++ unzip libffi-dev -y
   ```
   当操作系统为CentOS/BClinux时，安装命令如下：
   ```
   yum install autoconf automake libtool curl make gcc-c++ unzip libffi-devel -y
   ```
2. 为autogen.sh脚本添加可执行权限并执行此脚本。
   ```
   chmod +x autogen.sh
   ./autogen.sh
   ```
3. 配置安装路径（默认安装路径为"**/usr/local**"）。
   ```
   ./configure
   ```
   如果想指定安装路径，可参考以下命令。
   ```
   ./configure --prefix=/protobuf
   ```
   “/protobuf”为用户指定的安装路径。
4. 执行protobuf安装命令。
   ```
   make -j15        # 通过grep -w processor /proc/cpuinfo|wc -l查看cpu数，示例为15，用户可自行设置相应参数。
   make install
   ```
5. 刷新共享库。
   ```
   ldconfig
   ```
   protobuf安装完成后，会在**步骤4.3**中配置的路径下面的include目录中生成google/protobuf文件夹，存放protobuf相关头文件；在**步骤4.3**中配置路径下面的bin目录中生成protoc可执行文件，用于进行*.proto文件的编译，生成protobuf的C++头文件及实现文件。
6. 检查是否安装完成。
   ```
   ln -s /protobuf/bin/protoc /usr/bin/protoc
   protoc --version
   ```
   其中**/protobuf**为**步骤4.3**中用户配置的安装路径。如果用户未配置安装路径，则直接执行**protoc --version**检查是否安装成功。

步骤5 安装protobuf的python版本运行库。

1. 进入protobuf软件包目录的python子目录，编译python版本的运行库。
   ```
   python3 setup.py build --cpp_implementation
   ```
   说明

   这里需要编译二进制版本的运行库，如果使用**python3 setup.py build**命令无法生成二进制版本的运行库，在序列化结构的处理时性能会非常慢。
2. 安装动态库。
   ```
   cd .. && make install
   ```
   进入python子目录，安装python版本的运行库。
   ```
   python3 setup.py install --cpp_implementation
   ```
3. 检查是否安装成功。

   检查系统中是否存在“/usr/local/python3.7.5/lib/python3.7/site-packages/protobuf-3.11.3-py3.7-linux-aarch64.egg/google/protobuf/pyext/_message.cpython-37m-*<arch>*-linux-gnu.so”这个动态库。其中*<arch>*为系统架构类型。

 版权所有 © 华为技术有限公司

说明

“/usr/local/python3.7.5/lib/python3.7/site-packages”是pip安装第三方库的路径，可以使用**pip3 -V**检查。

如果系统显示：/usr/local/python3.7.5/lib/python3.7/site-packages/pip，则pip安装第三方库的路径为/usr/local/python3.7.5/lib/python3.7/site-packages。

4. 若用户在**步骤4.3**中指定了安装路径，则需在运行脚本中增加环境变量的设置：
```
export LD_LIBRARY_PATH=/protobuf/lib:${LD_LIBRARY_PATH}
```
"**/protobuf**"为**步骤4.3**中用户配置的安装路径。

5. 建立软连接。

当用户自行配置安装路径时，需要建立软连接，否则导入tensorflow会报错。命令如下：
```
ln -s /protobuf/lib/libprotobuf.so.22.0.3 /usr/lib/libprotobuf.so.22
```
其中"/protobuf"为**步骤4.3**中用户配置的安装路径。

----结束

6.5.6 配置 device 的网卡 IP

当进行分布式训练时，需要通过昇腾软件中的HCCN Tool工具配置device的网卡IP，用于多个device间通信以实现网络模型参数的同步更新。本章节只介绍使用HCCN Tool工具配置网络的命令，如果用户需要使用HCCN Tool工具的其他功能（如检查网口Link状态），请参见《**Ascend 910 23.0.RC2 HCCN Tool 接口参考（AI加速卡）**》或《**Ascend 910B 23.0.RC2 HCCN Tool 接口参考**》。

Atlas 800 训练服务器、Atlas 900 AI 集群场景

说明

判定是SMP模式还是AMP模式，请登录BMC后台执行命令“ipmcget -d npuworkmode”进行查询。

- SMP（对称多处理器）模式：

  以root用户登录到AI Server配置每个device的网卡IP。配置要求：

  - AI Server中的第0/4，1/5，2/6，3/7号网卡需处于同一网段，第0/1/2/3号网卡在不同网段，第4/5/6/7号网卡在不同网段。
  - 对于集群场景，各AI Server对应的位置的device需处于同一网段，例如AI Server1和AI Server2的0号网卡需处于同一网段，AI Server1和AI Server2的1号网卡需处于同一网段。如下命令所用IP地址为示例，配置时以实际规划IP为准。

```
hccn_tool -i 0 -ip -s address 192.168.100.101 netmask 255.255.255.0
hccn_tool -i 1 -ip -s address 192.168.101.101 netmask 255.255.255.0
hccn_tool -i 2 -ip -s address 192.168.102.101 netmask 255.255.255.0
hccn_tool -i 3 -ip -s address 192.168.103.101 netmask 255.255.255.0
hccn_tool -i 4 -ip -s address 192.168.100.100 netmask 255.255.255.0
hccn_tool -i 5 -ip -s address 192.168.101.100 netmask 255.255.255.0
hccn_tool -i 6 -ip -s address 192.168.102.100 netmask 255.255.255.0
hccn_tool -i 7 -ip -s address 192.168.103.100 netmask 255.255.255.0
```

- AMP（非对称多处理器）模式：

  AMP模式下暂不需要配置device的网卡IP。

 版权所有 © 华为技术有限公司

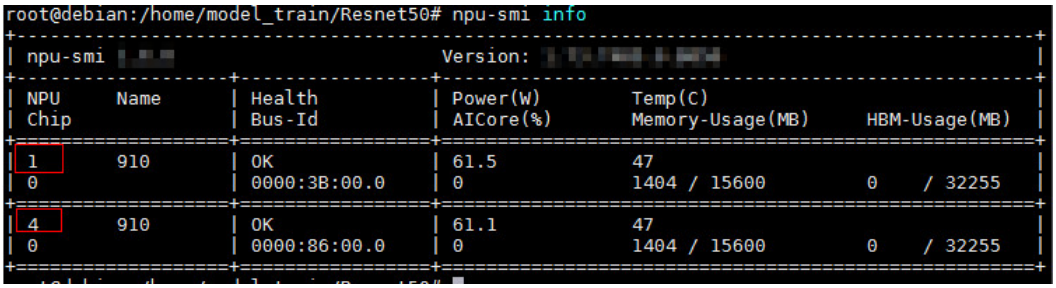
Atlas 300T 训练卡、Atlas 300T A2 训练卡场景

Atlas 300T 训练卡和Atlas 300T A2 训练卡每台服务器可以配置1或2张标卡，每张标卡对应1个Device OS，每张标卡需要配置1个地址，不同标卡配置相同网段IP地址即可。

以root用户登录到AI Server配置每个device的网卡IP，以Atlas 300T 训练卡为例，配置操作如下：

步骤1 先使用命令**npu-smi info查看**待配置device的ID，如图6-1中的NPU值，下文以NPU值为1和4为例，实际操作中以查询结果为准：

图 6-1 查看 device ID



步骤2 执行如下命令配置device的网卡IP，如下命令所用IP地址为示例，配置时以实际规划IP为准。

```
hccn_tool -i 1 -ip -s address 192.168.0.2 netmask 255.255.255.0
hccn_tool -i 4 -ip -s address 192.168.0.3 netmask 255.255.255.0
```

----结束

说明

需要确认在服务器上安装有npu-smi工具，执行**npu-smi -h**查询。

版权所有 © 华为技术有限公司

7 安装运行环境（ nnrt 软件，在物理机/虚拟机安装 ）

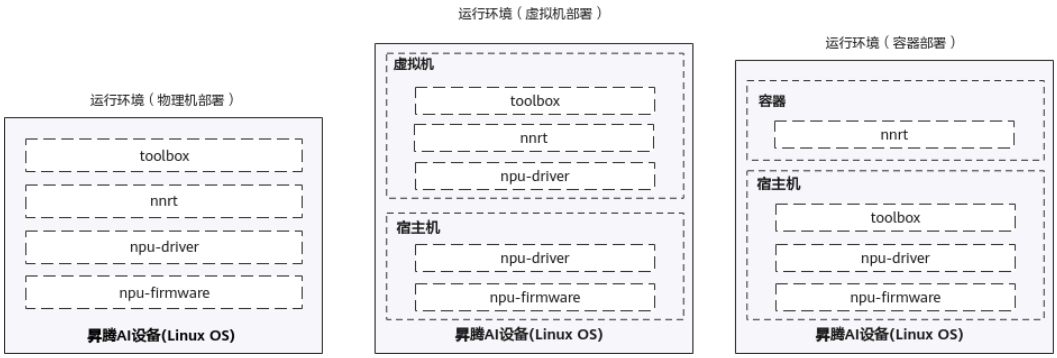
7.1 安装前必读

参考本章节安装运行环境，仅支持离线推理。

运行环境上会运行用户开发的应用程序。

运行环境（ 仅支持离线推理 ）安装软件如图7-1所示。用户可进行物理机、虚拟机和容器部署。

图 7-1 运行环境



版权所有 © 华为技术有限公司

7.2 准备软件包

下载软件包

用户可根据下表获取所需软件包和数字签名文件，下载本软件即表示您同意**华为企业软件许可协议**的条款和条件。

表 7-1 软件包

软件包类型	软件包名称	说明	获取链接
离线推理引擎包	Ascend-cann-nnrt_*{version}*_linux-*{arch}*.run	仅支持离线推理，主要包含AscendCL库、编译依赖的相关库（不包含driver包中的库），用于应用程序的模型推理。	**获取链接**
实用工具包	Ascend-mindx-toolbox_*{version}*_linux-*{arch}*.run	主要包含ascend-dmi工具。	**获取链接**

说明

*{version}*表示软件版本号，*{arch}*表示CPU架构。

软件数字签名验证

为了防止软件包在传递过程或存储期间被恶意篡改，下载软件包时需下载对应的数字签名文件用于完整性验证。

在软件包下载之后，请参考《OpenPGP签名验证指南》，对从Support网站下载的软件包进行PGP数字签名校验。如果校验失败，请不要使用该软件包，先联系华为技术支持工程师解决。

使用软件包安装/升级之前，也需要按上述过程先验证软件包的数字签名，确保软件包未被篡改。

运营商客户请访问：**http://support.huawei.com/carrier/digitalSignatureAction**

企业客户请访问：**https://support.huawei.com/enterprise/zh/tool/pgp-verify-TL1000000054**

7.3 准备安装及运行用户

- 运行用户：实际运行推理业务的用户。
- 安装用户：实际安装软件包的用户。

 版权所有 © 华为技术有限公司

- 若使用root用户安装，支持所有用户运行相关业务。
- 若使用非root用户安装，则安装及运行用户必须相同。
  - 已有非root用户，则无需再次创建。
  - 若想使用新的非root用户，则需要先创建该用户，请参见如下方法创建。

说明

- 如果安装驱动时未携带“--install-for-all”，并且CANN软件包运行用户为非root，则该CANN软件包运行用户所属的属组必须和驱动运行用户所属属组相同；如果不同，请用户自行添加到驱动运行用户属组。
- 运行用户不建议为root用户属组，权限控制可能存在安全风险，请谨慎使用。

创建非root用户操作方法如下，如下命令请以root用户执行。

1. 创建非root用户。
   groupadd *usergroup*
   useradd -g *usergroup* **-d /home/**_username_ **-m** *username* **-s /bin/bash**
2. 设置非root用户密码。
   passwd *username*

说明

- 创建完运行用户后， 请勿关闭该用户的登录认证功能。
- 设置的口令需符合口令复杂度要求（请参见**C.4 口令复杂度要求**）。密码有效期为90天，您可以在/etc/login.defs文件中修改有效期的天数，或者通过chage命令来设置用户的有效期，详情请参见**A.9 设置用户有效期**。

7.4 安装离线推理引擎包

前提条件

- 请参见**7.3 准备安装及运行用户**完成安装前准备。
- 在安装软件前请确保安装环境已安装推理卡或AI加速模块的驱动和固件。
- 通过**7.2 准备软件包**章节获取离线推理引擎包Ascend-cann-nnrt_*{version}*_linux-*{arch}*.run。
- 如果用户获取的软件包格式为***.deb**或***.rpm**，可参考**A.2 安装和卸载CANN软件包（适用于.deb格式）**或**A.3 安装和卸载CANN软件包（适用于.rpm格式）**进行安装。

安装步骤

获取并安装离线推理引擎包，详细安装步骤如下。

步骤1 以软件包的安装用户登录安装环境。

步骤2 将获取到的离线推理引擎包上传到安装环境任意路径（如“/home/package”）。

步骤3 进入软件包所在路径。

步骤4 执行如下命令增加对软件包的可执行权限。
chmod +x *软件包名*.run

 版权所有 © 华为技术有限公司

软件包名.**run**表示软件包名，例如离线推理引擎包Ascend-cann-nnrt_*{version}*_linux-*{arch}*.run。请根据实际包名进行替换。

步骤5 执行如下命令校验软件包安装文件的一致性和完整性。

```
./软件包名.run --check
```

步骤6 执行如下命令安装软件（以下命令支持**--install-path=**<*path*>等参数，具体参数说明请参见**C.1 参数说明**）。

```
./软件包名.run --install
```

说明

- 离线推理引擎包支持不同用户在同一运行环境安装，但安装版本必须保持一致，不同用户所属的属组也必须和驱动运行用户所属属组相同；如果不同，请用户自行添加到驱动运行用户属组。
- 如果以root用户安装，**不允许安装在非root用户目录下**。
- 如果用户未指定安装路径，则软件会安装到默认路径下，默认安装路径如下。
  - root用户：“/usr/local/Ascend”
  - 非root用户：“*${HOME}*/Ascend”
    其中${HOME}为当前用户目录。
- 软件包安装详细日志路径如下。
  - root用户：“/var/log/ascend_seclog/ascend_nnrt_install.log”
  - 非root用户：“*${HOME}*/var/log/ascend_seclog/ascend_nnrt_install.log”
    其中${HOME}为当前用户目录。

安装完成后，若显示如下信息，则说明软件安装成功：

```
xxx install success
```

*xxx*表示安装的实际软件包名。

----结束

7.5 安装实用工具包

若用户需要在昇腾AI设备上使用ascend-dmi工具，请安装实用工具包toolbox。

建议使用root用户请参考《**MindX ToolBox用户指南**》的安装实用工具包toolbox。

7.6 配置环境变量

nnrt等软件提供进程级环境变量设置脚本，供用户在进程中引用，以自动完成环境变量设置。用户进程结束后自动失效。示例如下（以root用户默认安装路径为例）：

```
# 安装nnrt包时配置
. /usr/local/Ascend/nnrt/set_env.sh

# 安装toolbox包时配置
. /usr/local/Ascend/toolbox/set_env.sh
```

用户也可以通过修改~/.bashrc文件方式设置永久环境变量，操作如下：

1. 以运行用户在任意目录下执行**vi ~/.bashrc**命令，打开**.bashrc**文件，在文件最后一行后面添加上述内容。
2. 执行**:wq!**命令保存文件并退出。

 版权所有 © 华为技术有限公司

3. 执行**source ~/.bashrc**命令使其立即生效。

7.7 安装后检查

通过“ascend-dmi”工具可检查设备健康信息与软硬件间的兼容性，配置完环境变量即可在任意目录使用工具。用户可以**root**用户或非**root**用户使用工具。若以非**root**用户使用工具，需要执行以下步骤加入软件包运行用户属组（若在安装软件包时使用了**--install-for-all**，则无需执行此操作）。例如软件包默认运行用户属组为HwHiAiUser（查询软件运行用户属组可执行命令**source /etc/ascend_install.info; echo ${UserGroup}**），操作如下：

1. 以**root**用户登录服务器。
2. 执行**usermod -a -G HwHiAiUser *{username}***命令，加入属组。***{username}***为非root用户名，请用户自行替换。

安装后检查具体步骤如下。

步骤1 检查设备健康状态。

执行如下命令查询健康信息。

```
ascend-dmi --dg
```

设备健康状态检查详情请参见《**MindX ToolBox用户指南**》的“ascend-dmi工具>故障诊断”章节。

步骤2 检查环境软硬件兼容性。

- 若用户安装软件包时，采用的是默认路径。

  执行如下命令检查软硬件兼容性。

  ```
  ascend-dmi -c
  ```

- 若软件包安装在用户指定目录下，用户使用兼容性检测功能时，需提供软件包安装目录。例如软件包安装在“/home/*xxx*/Ascend”目录下，命令如下：

  ```
  ascend-dmi -c -p /home/xxx/Ascend
  ```

软硬件兼容性检查详情请参见《**MindX ToolBox用户指南**》的“ascend-dmi工具>软硬件版本兼容性测试”章节。

----结束

 版权所有 © 华为技术有限公司

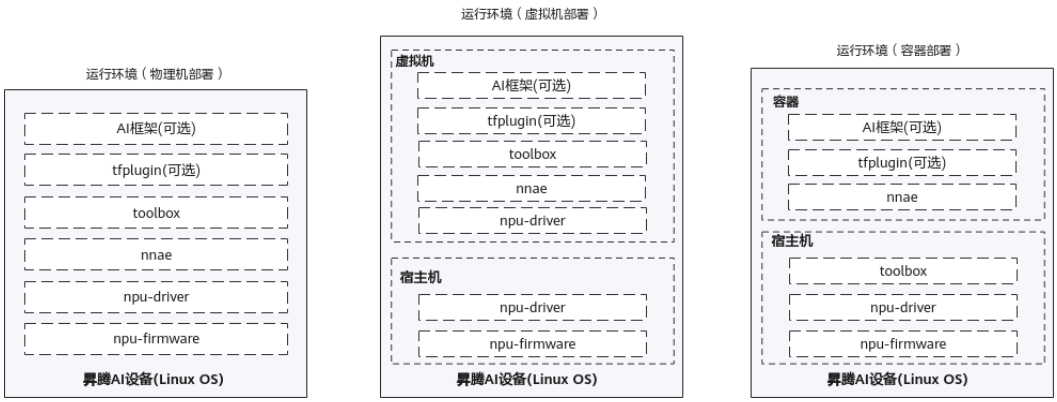
8 安装运行环境（ nnae 软件，在物理机/虚拟机安装 ）

8.1 安装前必读

参考本章节安装运行环境，支持离线推理、在线推理和训练。

运行环境安装软件如图8-1所示。用户可进行物理机和虚拟机和容器部署。

图 8-1 运行环境



版权所有 © 华为技术有限公司

8.2 准备软件包

下载软件包

用户可根据下表获取所需软件包和数字签名文件，下载本软件即表示您同意**华为企业软件许可协议**的条款和条件。

表 8-1 软件包

名称	软件包	说明	获取链接
深度学习引擎包	Ascend-cann-nnae_*{version}*_linux-*{arch}*.run	• 支持离线推理、在线推理、训练，包含FWK库Fwklib和算子库OPP组件。 • 请根据CPU架构（ x86_64、aarch64 ）获取对应的软件包。	**获取链接**
实用工具包	Ascend-mindx-toolbox_*{version}*_linux-*{arch}*.run	主要包含ascend-dmi工具。	**获取链接**
框架插件包	Ascend-cann-tfplugin_*{version}*_linux-*{arch}*.run	插件包，对接上层框架TensorFlow的适配插件。 在线推理或训练场景下若使用深度学习框架TensorFlow，需要获取该软件包。	**获取链接**

说明

*{version}*表示软件版本号，*{arch}*表示CPU架构。

软件数字签名验证

为了防止软件包在传递过程或存储期间被恶意篡改，下载软件包时需下载对应的数字签名文件用于完整性验证。

在软件包下载之后，请参考《 OpenPGP签名验证指南 》，对从Support网站下载的软件包进行PGP数字签名校验。如果校验失败，请不要使用该软件包，先联系华为技术支持工程师解决。

使用软件包安装/升级之前，也需要按上述过程先验证软件包的数字签名，确保软件包未被篡改。

运营商客户请访问：**http://support.huawei.com/carrier/digitalSignatureAction**

企业客户请访问：**https://support.huawei.com/enterprise/zh/tool/pgp-verify-TL1000000054**

版权所有 © 华为技术有限公司

8.3 准备安装及运行用户

- 运行用户：实际运行推理业务或执行训练的用户。
- 安装用户：实际安装软件包的用户。
  - 若使用root用户安装，支持所有用户运行相关业务。
  - 若使用非root用户安装，则安装及运行用户必须相同。
    - 已有非root用户，则无需再次创建。
    - 若想使用新的非root用户，则需要先创建该用户，请参见如下方法创建。

说明

- 如果安装驱动时未携带“--install-for-all”，并且CANN软件包运行用户为非root，则该CANN软件包运行用户所属的属组必须和驱动运行用户所属属组相同；如果不同，请用户自行添加到驱动运行用户属组。
- 运行用户不建议为root用户属组，权限控制可能存在安全风险，请谨慎使用。

创建非root用户操作方法如下，如下命令请以root用户执行。

1. 创建非root用户。
   groupadd *usergroup*
   useradd -g *usergroup* **-d /home/**_username_ **-m** *username* **-s /bin/bash**
2. 设置非root用户密码。
   passwd *username*

说明

- 创建完运行用户后， 请勿关闭该用户的登录认证功能。
- 设置的口令需符合口令复杂度要求（请参见**C.4 口令复杂度要求**）。密码有效期为90天，您可以在/etc/login.defs文件中修改有效期的天数，或者通过chage命令来设置用户的有效期，详情请参见**A.9 设置用户有效期**。

8.4 安装依赖

8.4.1 安装前必读

安装CANN软件前需安装相关依赖。

本章节以Ubuntu 18.04、CentOS 7.6和SLES 12.5为例，详述依赖安装操作。其他系统可参考这三种系统进行安装。

- Ubuntu、Debian、UOS20、UOS20 SP1、Linx系统可参考Ubuntu 18.04进行安装。
- EulerOS、openEuler、CentOS、BCLinux、Kylin、UOS20 1020e系统可参考CentOS 7.6进行安装。
- SLES系统可参考SLES 12.5进行安装。

 版权所有 © 华为技术有限公司

8.4.2 依赖列表

说明

针对用户自行安装的开源软件，如Python、numpy等，请使用稳定版本（尽量使用无漏洞的版本）。

Ubuntu 18.04

表 8-2 依赖信息

名称	版本要求
Python	CANN支持Python3.7.*x*（3.7.0~3.7.11）、Python3.8.*x*（3.8.0~3.8.11）、Python3.9.*x*（3.9.0~3.9.7）。 TensorFlow1.15支持Python3.7.*x*（3.7.0~3.7.11），TensorFlow2.6.5和PyTorch框架支持Python3.7.x（3.7.5~3.7.11）、Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。
cmake	>=3.5.1
make	-
gcc g++	• 离线推理场景 要求4.8.5版本及以上gcc。 • 在线推理、训练、Ascend Graph开发场景 要求7.3.0版本及以上gcc，若gcc版本低于7.3.0，可参考A.7 安装7.3.0版本gcc进行安装。
zlib1g zlib1g-dev libsqlite3-dev openssl libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran	无版本要求，安装的版本以操作系统自带的源为准。
numpy	>=1.14.3
decorator	>=4.4.0
sympy	>=1.4

版权所有 © 华为技术有限公司

名称	版本要求
cffi	>=1.12.3
protobuf	>=3.11.3
attrs pyyaml pathlib2 scipy requests psutil absl-py	无版本要求，安装的版本以pip源为准。

CentOS 7.6

表 8-3 依赖信息

名称	版本限制
Python	CANN支持Python3.7.*x*（3.7.0~3.7.11）、Python3.8.*x*（3.8.0~3.8.11）、Python3.9.*x*（3.9.0~3.9.7）。 TensorFlow1.15支持Python3.7.*x*（3.7.0~3.7.11），TensorFlow2.6.5和PyTorch框架支持Python3.7.x（3.7.5~3.7.11）、Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。
cmake	>=3.5.1
make	-
gcc gcc-c++	• 离线推理场景 要求4.8.5版本及以上gcc。 • 在线推理、训练、Ascend Graph开发场景 要求7.3.0版本及以上gcc，若gcc版本低于7.3.0，可参考**A.7 安装7.3.0版本gcc**进行安装。

版权所有 © 华为技术有限公司

名称	版本限制
unzip zlib-devel libffi-devel openssl-devel pciutils net-tools sqlite-devel lapack-devel gcc-gfortran python3-devel（openEuler系统需要安装）	无版本要求，安装的版本以操作系统自带的源为准。
numpy	>=1.14.3
decorator	>=4.4.0
sympy	>=1.4
cffi	>=1.12.3
protobuf	>=3.11.3
attrs pyyaml pathlib2 scipy requests psutil absl-py	无版本要求，安装的版本以pip源为准。

Suse 12SP5

表 8-4 依赖信息

类别	版本限制
Python	CANN支持Python3.7.*x*（3.7.0~3.7.11）、Python3.8.*x*（3.8.0~3.8.11）、Python3.9.*x*（3.9.0~3.9.7）。 TensorFlow1.15支持Python3.7.*x*（3.7.0~3.7.11），TensorFlow2.6.5和PyTorch框架支持Python3.7.x（3.7.5~3.7.11）、Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。
cmake	>=3.5.1

 版权所有 © 华为技术有限公司

类别	版本限制
make	-
gcc	离线推理场景： 要求4.8.5版本及以上gcc。
gcc-c++	
unzip zlib-devel libffi-devel openssl-devel pciutils net-tools gdbm-devel	无版本要求，安装的版本以操作系统自带的源为准。
numpy	>=1.14.3
decorator	>=4.4.0
sympy	>=1.4
cffi	>=1.12.3
protobuf	>=3.11.3
attrs pyyaml pathlib2 scipy requests psutil gnureadline absl-py	无版本要求，安装的版本以pip源为准。

8.4.3 安装步骤（ Ubuntu 18.04 ）

检查源

安装过程需要下载相关依赖，请确保安装环境能够连接网络。

请在root用户下执行如下命令检查源是否可用。

```
apt-get update
```

如果命令执行报错或者后续安装依赖时等待时间过长甚至报错，则检查网络是否连接或者把“/etc/apt/sources.list”文件中的源更换为可用的源或使用镜像源（ 以配置华为镜像源为例，可参考**华为开源镜像站** ）。

 版权所有 © 华为技术有限公司

检查 root 用户的 umask

1. 以root用户登录安装环境。
2. 检查root用户的umask值。
   ```
   umask
   ```
3. 如果umask不等于0022，请执行如下操作配置，在该文件的最后一行添加umask 0022后保存。
   a. 在任意目录下执行如下命令，打开**.bashrc**文件：
      ```
      vi ~/.bashrc
      ```
      在文件最后一行后面添加umask 0022内容。
   b. 执行**:wq!**命令保存文件并退出。
   c. 执行**source ~/.bashrc**命令使其立即生效。

说明

依赖安装完成后，请用户恢复为原umask值（删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

配置安装用户权限

可使用root或非root用户（该非root用户需与软件包安装用户保持一致）安装依赖，如果使用非root用户安装，可能需要用到提权命令，请用户自行获取所需的sudo权限。使用完成后请取消涉及高危命令的权限，否则有sudo提权风险。

安装依赖

步骤1 检查系统是否安装python依赖以及gcc等软件。

分别使用如下命令检查是否安装gcc，make以及python依赖软件等。

```
gcc --version
g++ --version
make --version
cmake --version
dpkg -l zlib1g| grep zlib1g| grep ii
dpkg -l zlib1g-dev| grep zlib1g-dev| grep ii
dpkg -l libsqlite3-dev| grep libsqlite3-dev| grep ii
dpkg -l openssl| grep openssl| grep ii
dpkg -l libssl-dev| grep libssl-dev| grep ii
dpkg -l libffi-dev| grep libffi-dev| grep ii
dpkg -l unzip| grep unzip| grep ii
dpkg -l pciutils| grep pciutils| grep ii
dpkg -l net-tools| grep net-tools| grep ii
dpkg -l libblas-dev| grep libblas-dev| grep ii
dpkg -l gfortran| grep gfortran| grep ii
dpkg -l libblas3| grep libblas3| grep ii
```

若分别返回如下信息则说明已经安装，进入下一步（以下回显仅为示例，版本要求请以**6.3.2 依赖列表**为准）。

```
gcc (Ubuntu 7.3.0-3ubuntu1~18.04) 7.3.0
g++ (Ubuntu 7.3.0-3ubuntu1~18.04) 7.3.0
GNU Make 4.1
cmake version 3.10.2
zlib1g:arm64   1:1.2.11.dfsg-0ubuntu2 arm64       compression library - runtime
zlib1g-dev:arm64 1:1.2.11.dfsg-0ubuntu2 arm64       compression library - development
libsqlite3-dev:arm64 3.22.0-1ubuntu0.3 arm64       SQLite 3 development files
openssl  1.1.1-1ubuntu2.1~18.04.6 arm64   Secure Sockets Layer toolkit - cryptographic utility
libssl-dev:arm64 1.1.1-1ubuntu2.1~18.04.6 arm64     Secure Sockets Layer toolkit - development files
libffi-dev:arm64 3.2.1-8      arm64       Foreign Function Interface library (development files)
unzip          6.0-21ubuntu1 arm64       De-archiver for .zip files
```

 版权所有 © 华为技术有限公司

```
pciutils       1:3.5.2-1ubuntu1 arm64        Linux PCI Utilities
net-tools      1.60+git20161116.90da8a0-1ubuntu1 arm64       NET-3 networking toolkit
libblas-dev:arm64 3.7.1-4ubuntu1 arm64       Basic Linear Algebra Subroutines 3, static library
gfortran       4:7.4.0-1ubuntu2.3 arm64      GNU Fortran 95 compiler
libblas3:arm64 3.7.1-4ubuntu1 arm64     Basic Linear Algebra Reference implementations, shared library
```

否则请执行如下安装命令（如果只有部分软件未安装，则如下命令修改为还未安装的软件即可）：

说明

- 如果使用root用户安装依赖，请将**步骤1至步骤2**命令中的sudo删除。
- 如果python及其依赖是使用非root用户安装，则需要执行**su - *username***命令切换到非root用户继续执行**步骤1至步骤3**。
- libsqlite3-dev需要在python安装之前安装，如果用户操作系统已经安装满足版本要求的python环境，在此之后再安装libsqlite3-dev，则需要重新编译python环境。

```
sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3
```

步骤2 检查系统是否安装满足版本要求的python开发环境（具体要求请参见**6.3.2 依赖列表**，此步骤以环境上需要使用python 3.7.*x*为例进行说明）。

执行命令**python3 --version**，如果返回信息满足python版本要求，则直接进入下一步。

否则可参考如下方式安装python3.7.5。

1. 使用wget下载python3.7.5源码包，可以下载到安装环境的任意目录，命令为：
   ```
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```
2. 进入下载后的目录，解压源码包，命令为：
   ```
   tar -zxvf Python-3.7.5.tgz
   ```
3. 进入解压后的文件夹，执行配置、编译和安装命令：
   ```
   cd Python-3.7.5
   ./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
   make
   sudo make install
   ```
   其中“--prefix”参数用于指定python安装路径，用户根据实际情况进行修改。“--enable-shared”参数用于编译出libpython3.7m.so.1.0动态库。“--enable-loadable-sqlite-extensions”参数用于加载libsqlite3-dev依赖。

   本手册以--prefix=/usr/local/python3.7.5路径为例进行说明。执行配置、编译和安装命令后，安装包在/usr/local/python3.7.5路径，libpython3.7m.so.1.0动态库在/usr/local/python3.7.5/lib/libpython3.7m.so.1.0路径。
4. 设置python3.7.5环境变量。
   ```
   #用于设置python3.7.5库文件路径
   export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
   #如果用户环境存在多个python3版本，则指定使用python3.7.5版本
   export PATH=/usr/local/python3.7.5/bin:$PATH
   ```
   通过以上export方式设置环境变量，该种方式设置的环境变量只在当前窗口有效。您也可以通过将以上命令写入~/.bashrc文件中，然后执行**source ~/.bashrc**命令，使上述环境变量永久生效。注意如果后续您有使用环境上其他python版本的需求，则不建议将以上命令写入到~/.bashrc文件中。
5. 安装完成之后，执行如下命令查看安装版本，如果返回相关版本信息，则说明安装成功。
   ```
   python3 --version
   pip3 --version
   ```

步骤3 安装前请先使用**pip3 list**命令检查是否安装相关依赖，若已经安装，则请跳过该步骤；若未安装，则安装命令如下（如果只有部分软件未安装，则如下命令修改为只安装还未安装的软件即可）。

 版权所有 © 华为技术有限公司

- 请在安装前配置好pip源，具体可参考**A.10 配置pip源**。
- 安装前，建议执行命令**pip3 install --upgrade pip**进行升级，避免因pip版本过低导致安装失败。
- 如下命令如果使用非root用户安装，需要在安装命令后加上**--user**，例如：**pip3 install attrs --user**，安装命令可在任意路径下执行。

```
pip3 install attrs
pip3 install numpy
pip3 install decorator
pip3 install sympy
pip3 install cffi
pip3 install pyyaml
pip3 install pathlib2
pip3 install psutil
pip3 install protobuf
pip3 install scipy
pip3 install requests
pip3 install absl-py
```

如果执行上述命令时报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”，请参见**B.6 pip3 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

说明

依赖安装完成后，请用户恢复为原umask值（参考**检查root用户的umask**，删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

8.4.4 安装步骤（CentOS 7.6）

检查源

安装过程需要下载相关依赖，请确保安装环境能够连接网络。

请在root用户下执行如下命令检查源是否可用。

```
yum makecache
```

如果命令执行报错或者后续安装依赖时等待时间过长甚至报错，则检查网络是否连接或者把"/etc/yum.repos.d/xxxx.repo"文件中的源更换为可用的源或使用镜像源（以配置华为镜像源为例，可参考**华为开源镜像站**）。

说明

如果执行上述命令提示“Your license is invalid”，请获取OS授权license。

检查 root 用户的 umask

1. 以root用户登录安装环境。
2. 检查root用户的umask值。
   ```
   umask
   ```
3. 如果umask不等于0022，请执行如下操作配置，在该文件的最后一行添加umask 0022后保存。
   a. 在任意目录下执行如下命令，打开**.bashrc**文件：
      ```
      vi ~/.bashrc
      ```

 版权所有 © 华为技术有限公司

在文件最后一行后面添加umask 0022内容。

b. 执行**:wq!**命令保存文件并退出。

c. 执行**source ~/.bashrc**命令使其立即生效。

说明

依赖安装完成后，请用户恢复为原umask值（删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

配置安装用户权限

可使用root或非root用户（该非root用户需与软件包安装用户保持一致）安装依赖，如果使用非root用户安装，可能需要用到提权命令，请用户自行获取所需的sudo权限。使用完成后请取消涉及高危命令的权限，否则有sudo提权风险。

配置最大线程数

训练场景下，不同OS的最大线程数可能不满足训练要求，需执行以下命令修改最大线程数为无限制。

1. 以root用户登录安装环境。
2. 配置环境变量，修改线程数为无限制，编辑“/etc/profile”文件，在文件的最后添加如下内容后保存退出：
```
ulimit -u unlimited
```
3. 执行如下命令使环境变量生效。
```
source /etc/profile
```

安装依赖

步骤1 检查系统是否安装python依赖以及gcc等软件。

分别使用如下命令检查是否安装gcc，make以及python依赖软件等。其中python3-devel在openEuler系统上需要安装。

```
gcc --version
g++ --version
make --version
cmake --version
rpm -qa |grep unzip
rpm -qa |grep zlib-devel
rpm -qa |grep libffi-devel
rpm -qa |grep openssl-devel
rpm -qa |grep pciutils
rpm -qa |grep net-tools
rpm -qa |grep sqlite-devel
rpm -qa |grep lapack-devel
rpm -qa |grep gcc-gfortran
rpm -qa |grep python3-devel
```

若分别返回如下信息则说明已经安装，进入下一步（以下回显仅为示例，版本要求请以**6.3.2 依赖列表**为准）。

```
gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)
g++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)
GNU Make 3.82
cmake version 3.20.5
unzip-6.0-21.el7.aarch64
zlib-devel-1.2.7-18.el7.aarch64
libffi-devel-3.0.13-18.el7.aarch64
openssl-devel-1.1.1c-15.el8.aarch64
pciutils-3.5.1-3.el7.aarch64
```

 版权所有 © 华为技术有限公司

```
net-tools-2.0-0.25.20131004git.el7.aarch64
sqlite-devel-3.7.17-8.el7_7.1.aarch64
lapack-devel-3.4.2-8.el7.aarch64
gcc-gfortran-4.8.5-39.el7.aarch64
python3-devel-3.7.4-8.oe1.aarch64
```

否则请执行如下安装命令（如果只有部分软件未安装，则如下命令修改为还未安装的软件即可）：

说明

- 如果使用root用户安装依赖，请将**步骤1至步骤2**命令中的sudo删除。
- 如果python及其依赖是使用非root用户安装，则需要执行**su - *username***命令切换到非root用户继续执行**步骤1至步骤3**。
- sqlite-devel需要在python安装之前安装，如果用户操作系统已经安装满足版本要求的python环境，在此之后再安装sqlite-devel，则需要重新编译python环境。

```
sudo yum install -y gcc gcc-c++ make cmake unzip zlib-devel libffi-devel openssl-devel pciutils net-tools sqlite-devel lapack-devel gcc-gfortran python3-devel
```

如果通过上述方式安装的cmake版本低于3.5.1，则请参见**A.6 安装3.5.2版本cmake**解决。

步骤2 检查系统是否安装满足版本要求的python开发环境（具体要求请参见**6.3.2 依赖列表**，此步骤以环境上需要使用python 3.7.*x*为例进行说明）。

执行命令**python3 --version**，如果返回信息满足python版本要求，则直接进入下一步。

否则可参考如下方式安装python3.7.5。

1. 使用wget下载python3.7.5源码包，可以下载到安装环境的任意目录，命令为：
   ```
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```
2. 进入下载后的目录，解压源码包，命令为：
   ```
   tar -zxvf Python-3.7.5.tgz
   ```
3. 进入解压后的文件夹，创建安装目录，执行配置、编译和安装命令：
   ```
   cd Python-3.7.5
   ./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
   make
   sudo make install
   ```
   其中“--prefix”参数用于指定python安装路径，用户根据实际情况进行修改，“--enable-shared”参数用于编译出libpython3.7m.so.1.0动态库，“--enable-loadable-sqlite-extensions”参数用于加载sqlite-devel依赖。

   本手册以--prefix=/usr/local/python3.7.5路径为例进行说明。执行配置、编译和安装命令后，安装包在/usr/local/python3.7.5路径，libpython3.7m.so.1.0动态库在/usr/local/python3.7.5/lib/libpython3.7m.so.1.0路径。
4. 设置python3.7.5环境变量。
   ```
   #用于设置python3.7.5库文件路径
   export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
   #如果用户环境存在多个python3版本，则指定使用python3.7.5版本
   export PATH=/usr/local/python3.7.5/bin:$PATH
   ```
   通过以上export方式设置环境变量，该种方式设置的环境变量只在当前窗口有效。您也可以通过将以上命令写入~/.bashrc文件中，然后执行**source ~/.bashrc**命令，使上述环境变量永久生效。注意如果后续您有使用环境上其他python版本的需求，则不建议将以上命令写入到~/.bashrc文件中。
5. 安装完成之后，执行如下命令查看安装版本，如果返回相关版本信息，则说明安装成功。
   ```
   python3 --version
   pip3 --version
   ```

 版权所有 © 华为技术有限公司

步骤3 安装前请先使用**pip3 list**命令检查是否安装相关依赖，若已经安装，则请跳过该步骤；若未安装，则安装命令如下（如果只有部分软件未安装，则如下命令修改为只安装还未安装的软件即可）。

- 请在安装前配置好pip源，具体可参考**A.10 配置pip源**。
- 安装前，建议执行命令**pip3 install --upgrade pip**进行升级，避免因pip版本过低导致安装失败。
- 如下命令如果使用非root用户安装，需要在安装命令后加上**--user**，例如：**pip3 install attrs --user**，安装命令可在任意路径下执行。

```
pip3 install attrs
pip3 install numpy
pip3 install decorator
pip3 install sympy
pip3 install cffi
pip3 install pyyaml
pip3 install pathlib2
pip3 install psutil
pip3 install protobuf
pip3 install scipy
pip3 install requests
pip3 install absl-py
```

- 如果安装numpy报错，请参考**B.5 pip3 install numpy报错**解决。
- 如果安装scipy报错，请参考**B.4 pip3 install scipy报错**解决。

步骤4 如果仅需支持离线推理，请跳过此步骤。

CentOS7.6系统使用软件源默认安装的gcc版本为4.8.5，因此需要安装7.3.0版本gcc，安装过程请参见**A.7 安装7.3.0版本gcc**。

----结束

说明

依赖安装完成后，请用户恢复为原umask值（参考**检查root用户的umask**，删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

8.4.5 安装步骤（SLES 12.5）

检查源

安装过程需要下载相关依赖，请确保安装环境能够连接网络。

请在root用户下执行如下命令检查源是否可用。

```
zypper ref
```

如果命令执行报错或者后续安装依赖时等待时间过长甚至报错，可参考以下操作配置本地源。

请以root用户执行如下操作。

1. 挂载系统镜像iso文件（以下命令为示例）。
   ```
   mount -o loop SLE-12-SP5-Server-DVD-x86_64-GM-DVD1.iso /mnt
   ```
2. 使用ar参数添加一个mnt路径下挂好的本地源，并命名为suse（用户可自定义）。
   ```
   zypper ar -f /mnt suse
   ```
3. 清空zypper缓存，刷新源，使配置的源生效。
   ```
   zypper clean
   zypper ref
   ```

 版权所有 © 华为技术有限公司

检查 root 用户的 umask

1. 以root用户登录安装环境。
2. 检查root用户的umask值。

   ```
   umask
   ```

3. 如果umask不等于0022，请执行如下操作配置，在该文件的最后一行添加umask 0022后保存。
   a. 在任意目录下执行如下命令，打开**.bashrc**文件：

      ```
      vi ~/.bashrc
      ```

      在文件最后一行后面添加umask 0022内容。
   b. 执行**:wq!**命令保存文件并退出。
   c. 执行**source ~/.bashrc**命令使其立即生效。

说明

依赖安装完成后，请用户恢复为原umask值（删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

配置安装用户权限

可使用root或非root用户（该非root用户需与软件包安装用户保持一致）安装依赖，如果使用非root用户安装，可能需要用到提权命令，请用户自行获取所需的sudo权限。使用完成后请取消涉及高危命令的权限，否则有sudo提权风险。

安装依赖

步骤1 检查系统是否安装python依赖以及gcc等软件。

分别使用如下命令检查是否安装gcc，make以及python依赖软件等。

```
rpm -qa | grep gcc
rpm -qa | grep make
rpm -qa | grep unzip
rpm -qa | grep zlib-devel
rpm -qa | grep openssl-devel
rpm -qa | grep pciutils
rpm -qa | grep net-tools
rpm -qa | grep gdbm-devel
rpm -qa | grep libffi-devel
```

若分别返回如下信息则说明已经安装，进入下一步（以下回显仅为示例，版本要求请以**6.3.2 依赖列表**为准）。

```
gcc48-4.8.5-31.20.1.x86_64
libgcc_s1-8.2.1+r264010-1.3.3.x86_64
gcc-4.8-6.189.x86_64
libgcc_s1-32bit-8.2.1+r264010-1.3.3.x86_64
gcc-c++-4.8-6.189.x86_64
gcc48-c++-4.8.5-31.20.1.x86_64
cmake-3.5.2-20.6.1.x86_64
makedumpfile-1.6.5-1.19.x86_64
automake-1.13.4-6.2.noarch
make-4.0-4.1.x86_64
unzip-6.00-33.8.1.x86_64
zlib-devel-1.2.11-3.21.1.x86_64
zlib-devel-32bit-1.2.11-3.21.1.x86_64
zlib-devel-static-1.2.11-3.21.1.x86_64
zlib-devel-1.2.11-9.42.x86_64
zlib-devel-static-32bit-1.2.11-3.21.1.x86_64
libopenssl-devel-1.0.2p-1.13.noarch
```

 版权所有 © 华为技术有限公司

```
pciutils-3.2.1-11.3.1.x86_64
pciutils-ids-2018.02.08-12.3.1.noarch
net-tools-1.60-765.5.4.x86_64
gdbm-devel-1.10-9.70.x86_64
libffi-devel-3.2.1.git259-10.8.x86_64
```

否则请执行如下安装命令（如果只有部分软件未安装，则如下命令修改为还未安装的软件即可）：

说明

- 如果使用root用户安装依赖，请将**步骤1至步骤2**命令中的sudo删除。
- 如果python及其依赖是使用非root用户安装，则需要执行**su - *username***命令切换到非root用户继续执行**步骤1至步骤3**。

```
sudo zypper install -y gcc gcc-c++ make cmake unzip zlib-devel openssl-devel pciutils net-tools gdbm-devel
```

由于本地源中缺少libffi-devel依赖，可从**opensuse镜像源**中下载libffi-devel-3.2.1.git259-10.8.x86_64.rpm、libffi7-3.2.1.git259-10.8.x86_64.rpm、libffi7-32bit-3.2.1.git259-10.8.x86_64.rpm（软件包会定时更新，请以实际rpm包名为准）并一同上传至服务器某一目录下（如“/home/test”）。

进入rpm包所在路径（如“/home/test”），执行如下命令安装所需依赖：

```
sudo rpm -ivh *.rpm --nodeps
```

步骤2 检查系统是否安装满足版本要求的python开发环境（具体要求请参见**6.3.2 依赖列表**，此步骤以环境上需要使用python 3.7.*x*为例进行说明）。

执行命令**python3 --version**，如果返回信息满足python版本要求，则直接进入下一步。

否则可参考如下方式安装python3.7.5。

1. 使用wget下载python3.7.5源码包，可以下载到安装环境的任意目录，命令为：
   ```
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```
2. 进入下载后的目录，解压源码包，命令为：
   ```
   tar -zxvf Python-3.7.5.tgz
   ```
3. 进入解压后的文件夹，创建安装目录，执行配置、编译和安装命令：
   ```
   cd Python-3.7.5
   ./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
   make
   sudo make install
   ```
   其中“--prefix”参数用于指定python安装路径，用户根据实际情况进行修改，“--enable-shared”参数用于编译出libpython3.7m.so.1.0动态库。

   本手册以--prefix=/usr/local/python3.7.5路径为例进行说明。执行配置、编译和安装命令后，安装包在/usr/local/python3.7.5路径，libpython3.7m.so.1.0动态库在/usr/local/python3.7.5/lib/libpython3.7m.so.1.0路径。
4. 设置python3.7.5环境变量。
   ```
   #用于设置python3.7.5库文件路径
   export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
   #如果用户环境存在多个python3版本，则指定使用python3.7.5版本
   export PATH=/usr/local/python3.7.5/bin:$PATH
   ```
   通过以上export方式设置环境变量，该种方式设置的环境变量只在当前窗口有效。您也可以通过将以上命令写入~/.bashrc文件中，然后执行**source ~/.bashrc**命令，使上述环境变量永久生效。注意如果后续您有使用环境上其他python版本的需求，则不建议将以上命令写入到~/.bashrc文件中。
5. 安装完成之后，执行如下命令查看安装版本，如果返回相关版本信息，则说明安装成功。

 版权所有 © 华为技术有限公司

```
python3 --version
pip3 --version
```

步骤3 安装前请先使用**pip3 list**命令检查是否安装相关依赖，若已经安装，则请跳过该步骤；若未安装，则安装命令如下（如果只有部分软件未安装，则如下命令修改为只安装还未安装的软件即可）。

- 请在安装前配置好pip源，具体可参考**A.10 配置pip源**。
- 安装前，建议执行命令**pip3 install --upgrade pip**进行升级，避免因pip版本过低导致安装失败。
- 如下命令如果使用非root用户安装，需要在安装命令后加上**--user**，例如：**pip3 install attrs --user**，安装命令可在任意路径下执行。

```
pip3 install attrs
pip3 install numpy
pip3 install decorator
pip3 install sympy
pip3 install cffi
pip3 install pyyaml
pip3 install pathlib2
pip3 install psutil
pip3 install protobuf
pip3 install scipy
pip3 install requests
pip3 install gnureadline
pip3 install absl-py
```

----结束

说明

依赖安装完成后，请用户恢复为原umask值（参考**检查root用户的umask**，删除.bashrc文件中**umask 0022**一行）。基于安全考虑，建议用户将umask值改为0027。

8.5 安装深度学习引擎包

前提条件

- 在安装软件前请确保安装环境已安装NPU驱动。
- 通过**8.2 准备软件包**章节获取深度学习引擎包Ascend-cann-nnae_*{version}*_linux-*{arch}*.run。
- 如果用户获取的软件包格式为***.deb**或***.rpm**，可参考**A.2 安装和卸载CANN软件包（适用于.deb格式）**或**A.3 安装和卸载CANN软件包（适用于.rpm格式）**进行安装。

安装步骤

获取并安装深度学习引擎包，详细安装步骤如下。

步骤1 以软件包的安装用户登录安装环境。

若**8.4 安装依赖**中安装依赖的用户为root用户，则软件包的安装用户可自行指定；若**8.4 安装依赖**中安装依赖的用户为非root用户，请确保软件包的安装用户与该用户保持一致。

步骤2 将深度学习引擎包上传到安装环境任意路径（如“/home”）。

步骤3 进入软件包所在路径。

 版权所有 © 华为技术有限公司

步骤4 执行如下命令增加对软件包的可执行权限。

```
chmod +x 软件包名.run
```

软件包名.**run**表示软件包名，例如深度学习引擎包Ascend-cann-nnae_*{version}*_linux-*{arch}*.run。请根据实际包名进行替换。

步骤5 执行如下命令校验软件包安装文件的一致性和完整性。

```
./软件包名.run --check
```

步骤6 执行如下命令安装软件（以下命令支持**--install-path=**<*path*>等参数，具体参数说明请参见**C.1 参数说明**）。

```
./软件包名.run --install
```

说明

- 深度学习引擎包支持不同用户在同一运行环境安装，但安装版本必须保持一致，不同用户所属的属组也必须和驱动运行用户所属属组相同；如果不同，请用户自行添加到驱动运行用户属组。
- 如果以root用户安装，**不允许安装在非root用户目录下**。
- 如果用户未指定安装路径，则软件会安装到默认路径下，默认安装路径如下。
  - root用户：“/usr/local/Ascend”
  - 非root用户：“*${HOME}*/Ascend”
    其中${HOME}为当前用户目录。
- 软件包安装详细日志路径如下。
  - root用户：“/var/log/ascend_seclog/ascend_nnae_install.log”
  - 非root用户：“${HOME}/var/log/ascend_seclog/ascend_nnae_install.log”
    其中${HOME}为当前用户目录。

安装完成后，若显示如下信息，则说明软件安装成功：

```
xxx install success
```

*xxx*表示安装的实际软件包名。

----结束

说明

在包含动态shape网络或ACLNN单算子直调的场景下须安装二进制算子包，具体操作请参考**A.1 安装、升级和卸载二进制算子包**。

8.6 安装实用工具包

若用户需要在昇腾AI设备上使用ascend-dmi工具，请安装实用工具包toolbox。

建议使用root用户请参考《**MindX ToolBox用户指南**》的安装实用工具包toolbox。

8.7 安装框架插件包

在线推理或训练场景下若使用深度学习框架TensorFlow，需要安装框架插件包Ascend-cann-tfplugin_*xxx*.run。

请参照**8.5 安装深度学习引擎包**的步骤安装框架插件包。其中软件包安装详细日志路径如下。

 版权所有 © 华为技术有限公司

- root用户：“/var/log/ascend_seclog/ascend_tfplugin_install.log”
- 非root用户：“‘${HOME}/var/log/ascend_seclog/ascend_tfplugin_install.log”

  其中${HOME}为当前用户目录。

8.8 配置环境变量

nnae等软件提供进程级环境变量设置脚本，供用户在进程中引用，以自动完成环境变量设置。用户进程结束后自动失效。示例如下（以root用户默认安装路径为例）：

```
# 安装nnae包时配置
. /usr/local/Ascend/nnae/set_env.sh

# 安装tfplugin包时配置
. /usr/local/Ascend/tfplugin/set_env.sh

# 安装toolbox包时配置
. /usr/local/Ascend/toolbox/set_env.sh
```

用户也可以通过修改~/.bashrc文件方式设置永久环境变量，操作如下：

1. 以运行用户在任意目录下执行**vi ~/.bashrc**命令，打开**.bashrc**文件，在文件最后一行后面添加上述内容。
2. 执行**:wq!**命令保存文件并退出。
3. 执行**source ~/.bashrc**命令使其立即生效。

8.9 安装深度学习框架

8.9.1 安装 TensorFlow

若用户仅进行离线推理，请跳过此章节。

安装前准备

- 对于x86架构，跳过安装前准备。
- 对于aarch64架构。

  由于TensorFlow依赖h5py，而h5py依赖HDF5，需要先编译安装HDF5，否则使用pip安装h5py会报错，以下步骤以root用户操作。

  - 编译安装HDF5。

    i. 访问**下载链接**下载HDF5源码包，并上传到安装环境的任意目录。

    ii. 进入源码包所在目录，执行如下命令解压源码包。

    ```
    tar -zxvf hdf5-1.10.5.tar.gz
    ```

    iii. 进入解压后的文件夹，执行配置、编译和安装命令：

    ```
    cd hdf5-1.10.5/
    ./configure --prefix=/usr/include/hdf5
    make
    make install
    ```

    iv. 配置环境变量并建立动态链接库软连接。

    1) 配置环境变量。

    ```
    export CPATH="/usr/include/hdf5/include/:/usr/include/hdf5/lib/"
    ```

 版权所有 © 华为技术有限公司

2) root用户建立动态链接库软连接命令如下，非root用户需要在以下命令前添加sudo。

```
ln -s /usr/include/hdf5/lib/libhdf5.so  /usr/lib/libhdf5.so
ln -s /usr/include/hdf5/lib/libhdf5_hl.so  /usr/lib/libhdf5_hl.so
```

- 安装h5py。

  root用户下执行如下命令安装h5py依赖包。

  ```
  pip3 install Cython==0.29
  ```

  root用户下执行如下命令安装h5py。

  ```
  pip3 install h5py==2.8.0
  ```

安装 TensorFlow

说明

- TensorFlow1.15配套的Python版本是：Python3.7.x（ 3.7.5~3.7.11 ）。
- TensorFlow2.6.5配套的Python版本是：Python3.7.x（ 3.7.5~3.7.11 ）、Python3.8.x（ 3.8.0~3.8.11 ）、Python3.9.x（ 3.9.0~3.9.2 ）。
- TensorFlow2.6.5存在漏洞，请参考**相关漏洞及其修复方案**处理。

需要安装TensorFlow才可以进行算子开发验证、训练业务开发。

- 对于x86架构：直接从pip源下载即可，系统要求等具体请参考**TensorFlow官网**。需要注意TensorFlow官网提供的指导描述有误，从pip源下载CPU版本需要显式指定tensorflow-cpu，如果不指定CPU，默认下载的是GPU版本。安装命令参考如下：

  如下命令如果使用非root用户安装，需要在安装命令后加上**--user**，例如：**pip3 install tensorflow-cpu==1.15 --user**

  - 安装TensorFlow1.15

    ```
    pip3 install tensorflow-cpu==1.15
    ```

  - 安装TensorFlow2.6.5

    ```
    pip3 install tensorflow-cpu==2.6.5
    ```

- 对于aarch64架构：由于pip源未提供对应的版本，所以需要用户使用官网要求的linux_gcc7.3.0编译器编译tensorflow1.15.0或tensorflow2.6.5，编译步骤参考官网**TensorFlow官网**。特别注意点如下。

  在下载完tensorflow tag v1.15.0或tensorflow tag v2.6.5源码后需要执行如下步骤。

步骤1 下载“nsync-1.22.0.tar.gz”源码包。

1. 进入源码目录，TensorFlow1.15场景下打开“tensorflow/workspace.bzl”文件，TensorFlow2.6.5场景下打开“tensorflow/workspace2.bzl”文件，找到其中name为nsync的“tf_http_archive”定义。

   ```
   tf_http_archive(
       name = "nsync",
       sha256 = "caf32e6b3d478b78cff6c2ba009c3400f8251f646804bcb65465666a9cea93c4",
       strip_prefix = "nsync-1.22.0",
       system_build_file = clean_dep("//third_party/systemlibs:nsync.BUILD"),
       urls = [          "https://storage.googleapis.com/mirror.tensorflow.org/github.com/google/nsync/archive/1.22.0.tar.gz",
           "https://github.com/google/nsync/archive/1.22.0.tar.gz",
       ],
   )
   ```

2. 从urls中的任一路径下载nsync-1.22.0.tar.gz的源码包，保存到任意路径。

步骤2 修改“nsync-1.22.0.tar.gz”源码包。

 版权所有 © 华为技术有限公司

1. 切换到nsync-1.22.0.tar.gz所在路径，解压缩该源码包。解压缩后存在“nsync-1.22.0”文件夹。
2. 编辑“nsync-1.22.0/platform/c++11/atomic.h”。

   在NSYNC_CPP_START_内容后添加如下加粗字体内容。

```
#include "nsync_cpp.h"
#include "nsync_atomic.h"

NSYNC_CPP_START_

#define ATM_CB_() __sync_synchronize()

static INLINE int atm_cas_nomb_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
    int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p),
&o, n, std::memory_order_relaxed, std::memory_order_relaxed));
    ATM_CB_();
    return result;
}
static INLINE int atm_cas_acq_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
    int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p),
&o, n, std::memory_order_acquire, std::memory_order_relaxed));
    ATM_CB_();
    return result;
}
static INLINE int atm_cas_rel_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
    int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p),
&o, n, std::memory_order_release, std::memory_order_relaxed));
    ATM_CB_();
    return result;
}
static INLINE int atm_cas_relacq_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
    int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p),
&o, n, std::memory_order_acq_rel, std::memory_order_relaxed));
    ATM_CB_();
    return result;
}
```

步骤3 重新压缩“nsync-1.22.0.tar.gz”源码包。

将上个步骤中解压出的内容压缩为一个新的“nsync-1.22.0.tar.gz”源码包，保存（比如，保存在“/tmp/nsync-1.22.0.tar.gz”）。

步骤4 重新生成“nsync-1.22.0.tar.gz”源码包的sha256sum校验码。

执行如下命令后得到sha256sum校验码（一串数字和字母的组合）。

```
sha256sum /tmp/nsync-1.22.0.tar.gz
```

步骤5 修改sha256sum校验码和urls。

进入tensorflow tag源码目录，TensorFlow1.15场景下打开“tensorflow/workspace.bzl”文件，TensorFlow2.6.5场景下打开“tensorflow/workspace2.bzl”文件，找到其中name为nsync的“tf_http_archive”定义，其中“sha256=”后面的数字填写**步骤4**得到的校验码，“urls=”后面的列表第二行，填写存放“nsync-1.22.0.tar.gz”的file://索引。

```
  tf_http_archive(
      name = "nsync",
      sha256 = "caf32e6b3d478b78cff6c2ba009c3400f8251f646804bcb65465666a9cea93c4",
      strip_prefix = "nsync-1.22.0",
      system_build_file = clean_dep("//third_party/systemlibs:nsync.BUILD"),
      urls = [
          "https://storage.googleapis.com/mirror.tensorflow.org/github.com/google/nsync/archive/
1.22.0.tar.gz",
          "file:///tmp/nsync-1.22.0.tar.gz ",
          "https://github.com/google/nsync/archive/1.22.0.tar.gz",
      ],
  )
```

 版权所有 © 华为技术有限公司

步骤6 继续参考官网的“配置build”（**TensorFlow官网**）章节执行编译。

执行完**./configure**之后，需要修改 .tf_configure.bazelrc 配置文件。

添加如下一行build编译选项：

```
build:opt --cxxopt=-D_GLIBCXX_USE_CXX11_ABI=0
```

删除以下两行：

```
build:opt --copt=-march=native
build:opt --host_copt=-march=native
```

步骤7 继续执行官方的编译指导步骤（**TensorFlow官网**）即可。

步骤8 安装编译好的TensorFlow。

以上步骤执行完后会打包TensorFlow到指定目录，进入指定目录后执行如下命令安装：

如下命令如果使用非root用户安装，需要在安装命令后加上**--user**，例如：**pip3 install tensorflow-1.15.0-*.whl --user**

- TensorFlow1.15：
  ```
  pip3 install tensorflow-1.15.0-*.whl
  ```
- TensorFlow2.6.5：
  ```
  pip3 install tensorflow-2.6.5-*.whl
  ```

步骤9 执行如下命令验证安装效果。

```
python3 -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
```

如果返回了张量则表示安装成功。

----结束

8.9.2 安装 PyTorch

安装前请根据**表8-5**完成相应配套版本CANN软件的安装。若用户需要详细了解CANN软件和其安装流程，可从**1 安装须知**章节开始了解手册内容。若用户仅进行离线推理，请跳过此章节。

昇腾开发PyTorch Adapter插件用于适配PyTorch框架，为使用PyTorch框架的开发者提供昇腾AI处理器的超强算力，本章节指导用户安装PyTorch框架和PyTorch Adapter插件。

安装场景

说明

- PyTorch 1.8.1/1.11.0配套的Python版本是：Python3.7.x（3.7.5~3.7.11）、Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。
- PyTorch 2.0.1配套的Python版本是：Python3.8.x（3.8.0~3.8.11）、Python3.9.x（3.9.0~3.9.2）。

根据不同的PyTorch使用场景，用户可以选择以下三种方式安装PyTorch：

- 获取镜像下载安装PyTorch：若用户希望在容器中安装使用PyTorch，可以**点击链接**前往AscendHub昇腾PyTorch镜像仓库，根据**表8-5**选择对应版本的镜像下载使用。

 版权所有 © 华为技术有限公司

- 二进制whl包安装：推荐用户使用编好的二进制whl包安装PyTorch。
  请前往**安装PyTorch环境依赖**继续安装流程。
- 编译安装：请前往**A.5 编译安装PyTorch**进行安装。

如果用户不确定自己的软件包版本，可参考**A.11 查询软件包版本信息**进行查询。

表 8-5 Ascend 配套软件

AscendPyTorch版本	CANN版本	支持PyTorch版本	AscendHub镜像版本/名称
5.0.rc2	CANN 6.3.RC2	1.8.1.post2	23.0.RC2-1.8.1
		1.11.0.post1	23.0.RC2-1.11.0
		2.0.1.rc1	-

安装 PyTorch 环境依赖

执行如下命令安装。如果使用非root用户安装，需要在命令后加**--user**，例如：**pip3 install pyyaml --user**。

```
pip3 install pyyaml
pip3 install wheel
pip3 install typing_extensions
```

安装 PyTorch

用户可选择编译安装方式安装PyTorch。请参考**A.5 编译安装PyTorch**。

步骤1 安装官方torch包。

- x86_64

  说明

  在x86_64架构下，官方torch包使用MKL加速库。如果需要使用其他blas和lapack加速库（如openblas）来提升性能，请使用源码编译安装方式安装官方torch包，步骤请参考**A.5 编译安装PyTorch**章节。

  ```
  # 安装1.8.1版本
  pip3 install torch==1.8.1+cpu
  # 安装1.11.0版本
  pip3 install torch==1.11.0+cpu
  # 安装2.0.1版本
  pip3 install torch==2.0.1+cpu
  ```

  若执行以上命令安装cpu版本PyTorch报错，请点击下方PyTorch官方链接下载whl包。

  PyTorch 1.8.1版本：**下载链接**。

  PyTorch 1.11.0版本：**下载链接**。

  PyTorch 2.0.1版本：**下载链接**。

  执行如下安装命令：

  ```
  # 用户请根据自己实际情况修改命令中的安装包名
  pip3 install torch-1.8.1+cpu-cp37-cp37m-linux_x86_64.whl
  ```

- aarch64

 版权所有 © 华为技术有限公司

a. 获取安装包。

- PyTorch1.8.1/1.11.0：

  进入安装目录，执行如下命令获取鲲鹏文件共享中心上对应版本的whl包。

```
# 安装1.8.1版本
wget https://repo.huaweicloud.com/kunpeng/archive/Ascend/PyTorch/torch-1.8.1-cp37-cp37m-linux_aarch64.whl
# 安装1.11.0版本
wget https://repo.huaweicloud.com/kunpeng/archive/Ascend/PyTorch/torch-1.11.0-cp37-cp37m-linux_aarch64.whl
```

- PyTorch2.0.1：**下载链接**。

  或使用wget命令下载。

```
wget https://download.pytorch.org/whl/torch-2.0.1-cp38-cp38-manylinux2014_aarch64.whl
```

b. 执行如下命令安装。如果使用非root用户安装，需要在命令后加**--user**。

```
# 安装1.8.1版本
pip3 install torch-1.8.1-cp37-cp37m-linux_aarch64.whl
# 安装1.11.0版本
pip3 install torch-1.11.0-cp37-cp37m-linux_aarch64.whl
# 安装2.0.1版本
pip3 install torch-2.0.1-cp38-cp38-manylinux2014_aarch64.whl
```

步骤2 安装PyTorch插件torch_npu。以下命令以在aarch64架构下安装为例。

1. 进入安装目录，执行如下命令获取PyTorch插件的whl包。

```
# 若用户在x86架构下安装插件，请将命令中文件包名中的“aarch64”改为“x86_64”。
# 安装1.8.1版本
wget https://gitee.com/ascend/pytorch/releases/download/v5.0.rc2-pytorch1.8.1/torch_npu-1.8.1.post2-cp37-cp37m-linux_aarch64.whl
# 安装1.11.0版本
wget https://gitee.com/ascend/pytorch/releases/download/v5.0.rc2-pytorch1.11.0/torch_npu-1.11.0.post1-cp37-cp37m-linux_aarch64.whl
# 安装2.0.1版本
wget https://gitee.com/ascend/pytorch/releases/download/v5.0.rc2-pytorch2.0.1/torch_npu-2.0.1rc1-cp38-cp38-linux_aarch64.whl
```

说明

- 如果下载whl包时出现**ERROR: cannot verify gitee.com's certificate**报错，可在下载命令后加上**--no-check-certificate**参数避免此问题。

  样例代码如下所示。

```
wget https://gitee.com/ascend/pytorch/releases/download/v5.0.rc2-pytorch1.11.0/torch_npu-1.11.0.post1-cp37-cp37m-linux_aarch64.whl --no-check-certificate
```

- PyTorch 1.11版本在aarch64架构上可能出现CPU精度问题，需在Docker环境编译torch_npu，请参见**B.12 PyTorch框架在ARM CPU上算子计算结果异常**。

2. 执行如下命令安装。如果使用非root用户安装，需要在命令后加**--user**。

```
# 若用户在x86架构下安装插件，请将命令中文件包名中的“aarch64”改为“x86_64”。
# 安装1.8.1版本
pip3 install torch_npu-1.8.1.post2-cp37-cp37m-linux_aarch64.whl
# 安装1.11.0版本
pip3 install torch_npu-1.11.0.post1-cp37-cp37m-linux_aarch64.whl
# 安装2.0.1版本
pip3 install torch_npu-2.0.1rc1-cp38-cp38m-linux_aarch64.whl
```

步骤3 执行如下命令，若返回**True**则说明安装成功。

```
python3 -c "import torch;import torch_npu;print(torch_npu.npu.is_available())"
```

步骤4 安装对应框架版本的torchvision。

```
#PyTorch 1.8.1需安装0.9.1版本，PyTorch 1.11.0需安装0.12.0版本，PyTorch 2.0.1版本需安装0.15.2版本
pip3 install torchvision==0.9.1
```

----结束

 版权所有 © 华为技术有限公司

安装 APEX 混合精度模块

混合精度训练是在训练时混合使用单精度（ float32 ）与半精度(float16)数据类型，将两者结合在一起，并使用相同的超参数实现了与float32几乎相同的精度。在迁移完成、训练开始之前，基于NPU芯片的架构特性，用户需要开启混合精度，可以提升模型的性能。APEX混合精度模块是一个集优化性能、精度收敛于一身的综合优化库，可以提供不同场景下的混合精度训练支持。混合精度的介绍可参考《**PyTorch 模型迁移和训练指南**》中的“自动混合精度（ AMP ）”章节，APEX模块的使用介绍可参考《**PyTorch 模型迁移和训练指南**》中的“参考信息>模型套件和第三方库>APEX”章节。

推荐用户通过编译源码包安装APEX模块。

步骤1 安装依赖。

选择编译安装方式安装时需要安装系统依赖。目前支持CentOS与Ubuntu操作系统。

- CentOS
  ```
  yum install -y patch libjpeg-turbo-devel dos2unix openblas git
  yum install -y gcc==7.3.0 cmake==3.12.0 #gcc7.3.0版本及以上，cmake3.12.0版本及以上。若用户要安装1.11.0版本PyTorch，则gcc需为7.5.0版本以上。
  ```
- Ubuntu
  ```
  apt-get install -y patch build-essential libbz2-dev libreadline-dev wget curl llvm libncurses5-dev libncursesw5-dev xz-utils tk-dev liblzma-dev m4 dos2unix libopenblas-dev git
  apt-get install -y gcc==7.3.0 cmake==3.12.0 #gcc7.3.0版本及以上，cmake3.12.0版本及以上。若用户要安装1.11.0版本PyTorch，则gcc需为7.5.0版本以上。
  ```

步骤2 获取昇腾适配的APEX源码。

```
git clone -b 分支名称 https://gitee.com/ascend/apex.git
```

分支名称请根据实际使用的PyTorch版本选择，例如使用PyTorch1.11.0-5.0.rc2，则apex分支也使用5.0.rc2。

步骤3 进入昇腾适配的APEX源码目录，执行命令编译生成二进制安装包。

```
cd apex
bash scripts/build.sh
```

说明

请确保NPU版本的PyTorch可以正常使用，否则会影响APEX的编译。

步骤4 执行如下命令安装。如果使用非root用户安装，需要在命令后加**--user**。

```
# 若用户在x86架构下安装，请将命令中文件包名中的“aarch64”改为“x86_64”。
pip3 install apex/dist/apex-0.1_ascend-cp37-cp37m-linux_aarch64.whl
```

----结束

8.9.3 安装昇思 MindSpore

若用户仅进行离线推理，请跳过此章节。

用户要使用MindSpore框架，请登录**MindSpore官网**获取安装MindSpore框架的方法。

8.10 安装 Python 版本的 proto

如果训练脚本依赖protobuf的Python版本进行序列化结构的数据存储（ 例如TensorFlow的序列化相关接口 ），则需要安装Python版本的proto。

 版权所有 © 华为技术有限公司

步骤1 检查系统中是否存在“/usr/local/python3.7.5/lib/python3.7/site-packages/google/protobuf/pyext/_message.cpython-37m-*<arch>*-linux-gnu.so”（以参考**6.3 安装依赖**章节安装的python3.7.5为例）这个动态库，如果没有，需要按照如下步骤安装。其中*<arch>*为系统架构类型。

说明

“/usr/local/python3.7.5/lib/python3.7/site-packages”是pip安装第三方库的路径，可以使用**pip3 -V**检查。

如果系统显示：/usr/local/python3.7.5/lib/python3.7/site-packages/pip，则pip安装第三方库的路径为/usr/local/python3.7.5/lib/python3.7/site-packages。

步骤2 执行如下命令卸载protobuf。

```
pip3 uninstall protobuf
```

步骤3 下载protobuf软件包。

从**https://github.com/protocolbuffers/protobuf/releases/download/v3.11.3/protobuf-python-3.11.3.tar.gz**路径下下载3.11.3版本protobuf-python-3.11.3.tar.gz软件包（或者其他版本，保证和当前环境上安装的tensorflow兼容），并以root用户上传到所在Linux服务器任意目录并执行命令解压。

```
tar zxvf protobuf-python-3.11.3.tar.gz
```

步骤4 以root用户安装protobuf。

进入protobuf软件包目录。

1. 安装protobuf的依赖。

   当操作系统为Ubuntu时，安装命令如下：

   ```
   apt-get install autoconf automake libtool curl make g++ unzip libffi-dev -y
   ```

   当操作系统为CentOS/BClinux时，安装命令如下：

   ```
   yum install autoconf automake libtool curl make gcc-c++ unzip libffi-devel -y
   ```

2. 为autogen.sh脚本添加可执行权限并执行此脚本。

   ```
   chmod +x autogen.sh
   ./autogen.sh
   ```

3. 配置安装路径（默认安装路径为"**/usr/local**"）。

   ```
   ./configure
   ```

   如果想指定安装路径，可参考以下命令。

   ```
   ./configure --prefix=/protobuf
   ```

   “/protobuf”为用户指定的安装路径。

4. 执行protobuf安装命令。

   ```
   make -j15        # 通过grep -w processor /proc/cpuinfo|wc -l查看cpu数，示例为15，用户可自行设置相应参数。
   make install
   ```

5. 刷新共享库。

   ```
   ldconfig
   ```

   protobuf安装完成后，会在**步骤4.3**中配置的路径下面的include目录中生成google/protobuf文件夹，存放protobuf相关头文件；在**步骤4.3**中配置路径下面的bin目录中生成protoc可执行文件，用于进行*.proto文件的编译，生成protobuf的C++头文件及实现文件。

6. 检查是否安装完成。

   ```
   ln -s /protobuf/bin/protoc /usr/bin/protoc
   protoc --version
   ```

   其中**/protobuf**为**步骤4.3**中用户配置的安装路径。如果用户未配置安装路径，则直接执行**protoc --version**检查是否安装成功。

 版权所有 © 华为技术有限公司

步骤5 安装protobuf的python版本运行库。

1. 进入protobuf软件包目录的python子目录，编译python版本的运行库。
```
python3 setup.py build --cpp_implementation
```
说明

这里需要编译二进制版本的运行库，如果使用**python3 setup.py build**命令无法生成二进制版本的运行库，在序列化结构的处理时性能会非常慢。

2. 安装动态库。
```
cd .. && make install
```
进入python子目录，安装python版本的运行库。
```
python3 setup.py install --cpp_implementation
```
3. 检查是否安装成功。

检查系统中是否存在“/usr/local/python3.7.5/lib/python3.7/site-packages/protobuf-3.11.3-py3.7-linux-aarch64.egg/google/protobuf/pyext/_message.cpython-37m-*<arch>*-linux-gnu.so”这个动态库。其中*<arch>*为系统架构类型。

说明

“/usr/local/python3.7.5/lib/python3.7/site-packages”是pip安装第三方库的路径，可以使用**pip3 -V**检查。

如果系统显示：/usr/local/python3.7.5/lib/python3.7/site-packages/pip，则pip安装第三方库的路径为/usr/local/python3.7.5/lib/python3.7/site-packages。

4. 若用户在步骤4.3中指定了安装路径，则需在运行脚本中增加环境变量的设置：
```
export LD_LIBRARY_PATH=/protobuf/lib:${LD_LIBRARY_PATH}
```
"**/protobuf**"为步骤4.3中用户配置的安装路径。

5. 建立软连接。

当用户自行配置安装路径时，需要建立软连接，否则导入tensorflow会报错。命令如下：
```
ln -s /protobuf/lib/libprotobuf.so.22.0.3 /usr/lib/libprotobuf.so.22
```
其中"/protobuf"为步骤4.3中用户配置的安装路径。

----结束

8.11 配置 device 的网卡 IP

当进行分布式训练时，需要通过昇腾软件中的HCCN Tool工具配置device的网卡IP，用于多个device间通信以实现网络模型参数的同步更新。本章节只介绍使用HCCN Tool工具配置网络的命令，如果用户需要使用HCCN Tool工具的其他功能（如检查网口Link状态），请参见《**Ascend 910 23.0.RC2 HCCN Tool 接口参考（AI加速卡）**》或《**Ascend 910B 23.0.RC2 HCCN Tool 接口参考**》。

Atlas 800 训练服务器、Atlas 900 AI 集群场景

说明

判定是SMP模式还是AMP模式，请登录BMC后台执行命令“ipmcget -d npuworkmode”进行查询。

- SMP（对称多处理器）模式：

  以root用户登录到AI Server配置每个device的网卡IP。配置要求：

 版权所有 © 华为技术有限公司

  - AI Server中的第0/4，1/5，2/6，3/7号网卡需处于同一网段，第0/1/2/3号网卡在不同网段，第4/5/6/7号网卡在不同网段。
  - 对于集群场景，各AI Server对应的位置的device需处于同一网段，例如AI Server1和AI Server2的0号网卡需处于同一网段，AI Server1和AI Server2的1号网卡需处于同一网段。如下命令所用IP地址为示例，配置时以实际规划IP为准。

```
hccn_tool -i 0 -ip -s address 192.168.100.101 netmask 255.255.255.0
hccn_tool -i 1 -ip -s address 192.168.101.101 netmask 255.255.255.0
hccn_tool -i 2 -ip -s address 192.168.102.101 netmask 255.255.255.0
hccn_tool -i 3 -ip -s address 192.168.103.101 netmask 255.255.255.0
hccn_tool -i 4 -ip -s address 192.168.100.100 netmask 255.255.255.0
hccn_tool -i 5 -ip -s address 192.168.101.100 netmask 255.255.255.0
hccn_tool -i 6 -ip -s address 192.168.102.100 netmask 255.255.255.0
hccn_tool -i 7 -ip -s address 192.168.103.100 netmask 255.255.255.0
```

- AMP（非对称多处理器）模式：

  AMP模式下暂不需要配置device的网卡IP。

Atlas 300T 训练卡、Atlas 300T A2 训练卡场景

Atlas 300T 训练卡和Atlas 300T A2 训练卡每台服务器可以配置1或2张标卡，每张标卡对应1个Device OS，每张标卡需要配置1个地址，不同标卡配置相同网段IP地址即可。

以root用户登录到AI Server配置每个device的网卡IP，以Atlas 300T 训练卡为例，配置操作如下：

步骤1 先使用命令**npu-smi info查看**待配置device的ID，如图8-2中的NPU值，下文以NPU值为1和4为例，实际操作中以查询结果为准：

图 8-2 查看 device ID

```
root@debian:/home/model_train/Resnet50# npu-smi info
+------------------------------------------------------------------------------------------------+
| npu-smi                                 Version:                                               |
+------------------+----------------+-------------------------------------------------------+
| NPU     Name     | Health         | Power(W)       Temp(C)                                |
| Chip             | Bus-Id         | AICore(%)      Memory-Usage(MB)     HBM-Usage(MB)     |
+==================+================+=======================================================+
| 1       910      | OK             | 61.5           47                                     |
| 0                | 0000:3B:00.0   | 0              1404 / 15600         0    / 32255     |
+==================+================+=======================================================+
| 4       910      | OK             | 61.1           47                                     |
| 0                | 0000:86:00.0   | 0              1404 / 15600         0    / 32255     |
+==================+================+=======================================================+
```

步骤2 执行如下命令配置device的网卡IP，如下命令所用IP地址为示例，配置时以实际规划IP为准。

```
hccn_tool -i 1 -ip -s address 192.168.0.2 netmask 255.255.255.0
hccn_tool -i 4 -ip -s address 192.168.0.3 netmask 255.255.255.0
```

----结束

说明

需要确认在服务器上安装有npu-smi工具，执行**npu-smi -h**查询。

8.12 安装后检查

通过“ascend-dmi”工具可检查设备健康信息与软硬件间的兼容性，配置完环境变量即可在任意目录使用工具。用户可以**root**用户或非**root**用户使用工具。若以非**root**用

户使用工具，需要执行以下步骤加入软件包运行用户属组（若在安装软件包时使用了**--install-for-all**，则无需执行此操作）。例如软件包默认运行用户属组为HwHiAiUser（查询软件运行用户属组可执行命令**source /etc/ascend_install.info; echo ${UserGroup}**），操作如下：

1. 以**root**用户登录服务器。
2. 执行**usermod -a -G HwHiAiUser *{username}***命令，加入属组。***{username}***为非root用户名，请用户自行替换。

安装后检查具体步骤如下。

步骤1 检查设备健康状态。

执行如下命令查询健康信息。

```
ascend-dmi --dg
```

设备健康状态检查详情请参见《**MindX ToolBox用户指南**》的“ascend-dmi工具>故障诊断”章节。

步骤2 检查环境软硬件兼容性。

- 若用户安装软件包时，采用的是默认路径。

  执行如下命令检查软硬件兼容性。

  ```
  ascend-dmi -c
  ```

- 若软件包安装在用户指定目录下，用户使用兼容性检测功能时，需提供软件包安装目录。例如软件包安装在“/home/*xxx*/Ascend”目录下，命令如下：

  ```
  ascend-dmi -c -p /home/xxx/Ascend
  ```

软硬件兼容性检查详情请参见《**MindX ToolBox用户指南**》的“ascend-dmi工具>软硬件版本兼容性测试”章节。

----结束

 版权所有 © 华为技术有限公司

9 安装运行环境（在容器安装）

9.1 安装前必读

本章节指导用户进行容器部署，容器部署支持以下三种方式：

1. （推荐）直接从**AscendHub**上拉取容器镜像（如ascend-infer、ascend-tensorflow）。
2. 参考**ascend-docker-image**自行构建容器镜像，再参考《**MindX DL Ascend Docker Runtime用户指南**》启动容器镜像，从而完成容器部署。
3. 将**9.2 宿主机目录挂载至容器**所示宿主机上的设备、动态库和信息文件手动挂载至容器内，在容器内再自行安装CANN软件，具体操作步骤请参考**9.3 容器部署**。

9.2 宿主机目录挂载至容器

在容器部署过程中，用户无需在容器内安装驱动，只需根据不同产品类型将如下目录挂载至容器内，并在容器内安装CANN软件，完成容器部署。

可参考如下示例命令启动容器，具体挂载信息请根据产品类型和实际路径进行修改。

```
docker run -it \
--ipc=host \
--device=/dev/davinci0 \
--device=/dev/davinci_manager \
--device=/dev/devmm_svm \
--device=/dev/hisi_hdc \
-v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi \
-v /usr/local/Ascend/driver/lib64/common:/usr/local/Ascend/driver/lib64/common \
-v /usr/local/Ascend/driver/lib64/driver:/usr/local/Ascend/driver/lib64/driver \
-v /etc/ascend_install.info:/etc/ascend_install.info \
-v /etc/vnpu.cfg:/etc/vnpu.cfg \
-v /usr/local/Ascend/driver/version.info:/usr/local/Ascend/driver/version.info \
docker_image_id /bin/bash
```

- --ipc=host配置为和host宿主机共享内存。

版权所有 © 华为技术有限公司

- “/etc/vnpu.cfg”为宿主机算力切分持久化配置文件，仅在物理机特权容器的算力切分场景下使用（仅有部分产品支持，请以产品对应的挂载信息列表为准）。
- *docker_image_id* 为镜像ID，用户需自行替换，可以使用**docker images**命令查看镜像信息。

请根据以下产品类型，将对应的目录挂载至容器内。

Atlas 300I Pro、Atlas 300V Pro、Atlas 300V、Atlas 300I Duo

表 9-1 参数说明

参数	参数说明
--device	表示映射的设备，可以挂载一个或者多个设备。 需要挂载的设备如下： ● /dev/davinci*X*：NPU设备，X是ID号，如：davinci0。 ● /dev/davinci_manager：davinci相关的管理设备。 ● /dev/devmm_svm：内存管理相关设备。 ● /dev/hisi_hdc：hdc相关管理设备。
-v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi	将宿主机npu-smi工具“/usr/local/bin/npu-smi”挂载到容器中，请根据实际情况修改。
-v /usr/local/Ascend/driver/lib64/common:/usr/local/Ascend/driver/lib64/common	将宿主机目录“/usr/local/Ascend/driver/lib64/common”和“/usr/local/Ascend/driver/lib64/driver”挂载到容器中。请根据驱动所在实际路径修改。
-v /usr/local/Ascend/driver/lib64/driver:/usr/local/Ascend/driver/lib64/driver	
-v /etc/ascend_install.info:/etc/ascend_install.info	将宿主机安装信息文件“/etc/ascend_install.info”挂载到容器中。
-v /etc/vnpu.cfg:/etc/vnpu.cfg	将宿主机算力切分持久化配置文件“/etc/vnpu.cfg”挂载到容器中。且仅在物理机特权容器的算力切分场景下使用。
-v /usr/local/Ascend/driver/version.info:/usr/local/Ascend/driver/version.info	将宿主机版本信息文件“/usr/local/Ascend/driver/version.info”挂载到容器中，请根据实际情况修改。
--ipc=host	可能存在容器中共享内存不足的情况，启动容器时需要添加该参数以配置和host宿主机共享内存。

 版权所有 © 华为技术有限公司

A300T-9000、Atlas 300T Pro、A800-9000、A800-9010、A900-9000

表 9-2 参数说明

参数	参数说明
--device	表示映射的设备，可以挂载一个或者多个设备。 需要挂载的设备如下： ● /dev/davinci*X*：NPU设备，X是ID号，如：davinci0。 ● /dev/davinci_manager：davinci相关的管理设备。 ● /dev/devmm_svm：内存管理相关设备。 ● /dev/hisi_hdc：hdc相关管理设备。
-v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi	将宿主机npu-smi工具“/usr/local/bin/npu-smi”挂载到容器中，请根据实际情况修改。
-v /usr/local/Ascend/driver/lib64/common:/usr/local/Ascend/driver/lib64/common	将宿主机目录“/usr/local/Ascend/driver/lib64/common”和“/usr/local/Ascend/driver/lib64/driver”挂载到容器中。请根据驱动所在实际路径修改。
-v /usr/local/Ascend/driver/lib64/driver:/usr/local/Ascend/driver/lib64/driver	
-v /etc/ascend_install.info:/etc/ascend_install.info	将宿主机安装信息文件“/etc/ascend_install.info”挂载到容器中。
-v /etc/vnpu.cfg:/etc/vnpu.cfg	将宿主机算力切分持久化配置文件“/etc/vnpu.cfg”挂载到容器中。且仅在物理机特权容器的算力切分场景下使用。
-v /usr/local/Ascend/driver/version.info:/usr/local/Ascend/driver/version.info	将宿主机版本信息文件“/usr/local/Ascend/driver/version.info”挂载到容器中，请根据实际情况修改。
--ipc=host	可能存在容器中共享内存不足的情况，启动容器时需要添加该参数以配置和host宿主机共享内存。
--pids-limit 409600	当host宿主机系统为CentOS和BC-linux时，docker内的线程数最大为4092，无法满足训练要求，启动容器时需要添加该参数以配置CentOS/BC-linux下docker的最大线程。

 版权所有 © 华为技术有限公司

Atlas 200I SoC A1

参数	参数说明
--device	表示映射的设备可以挂载一个或者多个设备。 需要挂载的设备如下： ● /dev/davinci0：NPU设备。 ● /dev/svm0：内存管理的相关设备。 ● /dev/sys：dvpp相关的设备。 ● /dev/vdec：dvpp相关的设备。 ● /dev/venc：dvpp相关的设备。 ● /dev/vpc：dvpp相关的设备。 ● /dev/davinci_manager：davinci相关的管理设备。 ● /dev/spi_smbus：设备带外spi通信相关设备。 ● /dev/upgrade：获取昇腾系统相关配置、固件的相关设备。 ● /dev/user_config：管理用户配置相关设备。 ● /dev/ts_aisle：aicpudrv驱动对应设备。 ● /dev/memory_bandwidth：内存带宽相关设备。
-v /etc/sys_version.conf:/etc/sys_version.conf:ro	将宿主机驱动版本配置文件“/etc/sys_version.conf”以只读方式挂载到容器中。
-v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi:ro	将宿主机npu-smi工具“/usr/local/bin/npu-smi”以只读方式挂载到容器中，请根据实际情况修改。
-v /var/dmp_daemon:/var/dmp_daemon:ro	将宿主机dcmi服务进程文件“/var/dmp_daemon”以只读方式挂载到容器中。
-v /var/slogd:/var/slogd:ro	将宿主机日志进程文件“/var/slogd”以只读方式挂载到容器中。
-v /var/log/npu/conf/slog/slog.conf:/var/log/npu/conf/slog/slog.conf:ro	将宿主机日志守护进程配置文件“/var/log/npu/conf/slog/slog.conf”以只读方式挂载到容器中。
-v /etc/hdcBasic.cfg:/etc/hdcBasic.cfg:ro	将宿主机hdc配置文件“/etc/hdcBasic.cfg”以只读方式挂载到容器中。
-v /usr/local/Ascend/driver/tools:/usr/local/Ascend/driver/tools	将宿主机驱动相关工具目录“/usr/local/Ascend/driver/tools”挂载到容器中。

 版权所有 © 华为技术有限公司

参数	参数说明
-v /usr/local/Ascend/driver/lib64:/usr/local/Ascend/driver/lib64	将宿主机驱动依赖动态库目录“/usr/local/Ascend/driver/lib64”挂载到容器中。
-v /usr/lib64/aicpu_kernels:/usr/lib64/aicpu_kernels:ro	将宿主机aicpu lib库目录“/usr/lib64/aicpu_kernels”以只读方式挂载到容器中。
-v /usr/lib64/libtensorflow.so:/usr/lib64/libtensorflow.so:ro	将宿主机tensorflow的aicpu算子库文件“/usr/lib64/libtensorflow.so”以只读方式挂载到容器中。
-v /sys/fs/cgroup/memory:/sys/fs/cgroup/memory:ro	将宿主机查询内存占用率所需依赖目录“/sys/fs/cgroup/memory”以只读方式挂载到容器中。
-v /etc/ascend_install.info:/etc/ascend_install.info	将宿主机安装信息文件“/etc/ascend_install.info”挂载到容器中。
-v /usr/local/Ascend/driver/version.info:/usr/local/Ascend/driver/version.info	将宿主机版本信息文件“/usr/local/Ascend/driver/version.info”挂载到容器中，请根据实际情况修改。
--ipc=host	可能存在容器中共享内存不足的情况，启动容器时需要添加该参数以配置和host宿主机共享内存。

Atlas 300T A2、Atlas 800T A2、Atlas 900 A2 PoD

表 9-3 参数说明

参数	参数说明
--device	表示映射的设备，可以挂载一个或者多个设备。 需要挂载的设备如下： ● /dev/davinci*X*：NPU设备，X是ID号，如：davinci0。 ● /dev/davinci_manager：davinci相关的管理设备。 ● /dev/devmm_svm：内存管理相关设备。 ● /dev/hisi_hdc：hdc相关管理设备。
-v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi	将宿主机npu-smi工具“/usr/local/bin/npu-smi”挂载到容器中，请根据实际情况修改。

 版权所有 © 华为技术有限公司

参数	参数说明
-v /usr/local/Ascend/driver/lib64/common:/usr/local/Ascend/driver/lib64/common	将宿主机目录“/usr/local/Ascend/driver/lib64/common”和“/usr/local/Ascend/driver/lib64/driver”挂载到容器中。请根据driver的驱动.so所在路径修改。
-v /usr/local/Ascend/driver/lib64/driver:/usr/local/Ascend/driver/lib64/driver	
-v /etc/ascend_install.info:/etc/ascend_install.info	将宿主机安装信息文件“/etc/ascend_install.info”挂载到容器中。
-v /usr/local/Ascend/driver/version.info:/usr/local/Ascend/driver/version.info	将宿主机版本信息文件“/usr/local/Ascend/driver/version.info”挂载到容器中，请根据实际情况修改。
--ipc=host	可能存在容器中共享内存不足的情况，启动容器时需要添加该参数以配置和host宿主机共享内存。

9.3 容器部署

前提条件

- 宿主机已经安装过驱动和固件，详情请参见**4 安装驱动和固件**。
- 用户在宿主机自行安装docker（版本要求大于等于18.03）。

操作步骤

步骤1 参考**表9-4**将所需目录挂载至容器内。

 版权所有 © 华为技术有限公司

表 9-4 宿主机目录挂载至容器的操作指导

产品型号	操作指导
A800-3000+Atlas 300I Pro A800-3000+Atlas 300V Pro A800-3010+Atlas 300I Pro A800-3010+Atlas 300V Pro A800-3000 + Atlas 300V A800-3010 + Atlas 300V A500 Pro-3000+Atlas 300I Pro A500 Pro-3000+Atlas 300V Pro Atlas 300I Duo	《**Ascend 310P 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》的“容器内安装”章节。
A800-9000 A800-9010	《**Ascend 910 23.0.RC2 NPU驱动和固件安装指南（AI服务器）**》的“容器内安装”章节。
A800-3000+A300T-9000 A800-3000+Atlas 300T Pro A800-3010+A300T-9000 A800-3010+Atlas 300T Pro	《**Ascend 910 23.0.RC2 NPU驱动和固件安装指南（AI加速卡）**》的“容器内安装”章节。
Atlas 200I SoC A1	《**Atlas 200I SoC A1核心板 23.0.RC2 NPU驱动和固件安装指南**》的“容器内安装”章节。
Atlas 900 A2 PoD Atlas 800T A2 Atlas 800-3000 + Atlas 300T A2	《**Ascend 910B 23.0.RC2 NPU驱动和固件安装指南**》的“容器内安装”章节。

说明

- 训练场景下，当host宿主机系统为CentOS和BC-linux时，docker内的线程数最大为4092，无法满足训练要求，启动容器时需要添加**--pids-limit 409600**参数，以配置CentOS/BC-linux下docker的最大线程。
- 可能存在容器中共享内存不足的情况，启动容器时需要添加**--ipc=host**参数，以配置和host宿主机共享内存。

步骤2 使用**exit**命令退出容器，在宿主机CANN软件包所在路径，执行如下命令将软件包复制到容器内部。

```
docker cp /home/HwHiAiUser/Ascend-cann-nnrt_{version}_linux-{arch}.run container_id:/home/HwHiAiUser/software
```

所有路径请根据实际情况进行修改：

1. “*/home/HwHiAiUser/*”为宿主机上软件包的存放路径。
2. Ascend-cann-nnrt_*{version}*_linux-*{arch}*.run请替换为具体CANN软件包名。

 版权所有 © 华为技术有限公司

3. *container_id*为容器ID，可以使用**docker ps -a**命令查看所使用容器的ID。
4. “*/home/HwHiAiUser/software*”为容器内软件包的存放路径，如果没有该路径，请先手动创建。

步骤3 使用如下命令重新进入容器。

```
docker start container_id
docker attach container_id
```

*container_id*为具体容器ID或容器名，可以使用**docker ps -a**命令查看所使用容器的ID。

步骤4 进入CANN软件包所在目录，参考宿主机的安装方式（**7 安装运行环境（nnrt软件，在物理机/虚拟机安装）**或**8 安装运行环境（nnae软件，在物理机/虚拟机安装）**）自行安装所需CANN软件。

----结束

 版权所有 © 华为技术有限公司

10 升级

10.1 升级前必读

升级影响

- 升级过程禁止进行其他维护操作动作。
- 软件版本升级过程中会导致业务中断。
- 升级软件包后，不会影响正常业务。

注意事项

软件版本升级时的注意事项如**表10-1**所示。

表 10-1 升级时注意事项

序号	描述
1	在进行升级操作之前，请仔细阅读本文档，确定已经理解全部内容。如果您对文档有任何意见或建议，请联系华为技术支持解决。
2	为了减少对业务的影响，请提前切走业务或在业务量低时进行升级操作。
3	升级后，请确保所有软件的版本保持一致。
4	支持多版本情况下升级，但默认只升级安装目录下latest软链接指向的版本（即当前版本）。

说明

运行环境如果安装的是nnrt软件包，如果需要支持在线推理，请先卸载nnrt，再安装nnae。

 版权所有 © 华为技术有限公司

版本要求

驱动版本、固件版本需与CANN软件版本保持配套关系。版本配套关系请参见**CANN**中的版本配套表。

表 10-2 CANN 软件包升级路径说明

原版本	当前版本	是否支持升级
CANN 3.2.1	CANN 6.3.RC2	是。
CANN 3.3.0	CANN 6.3.RC2	否。请先执行卸载，再进行安装操作。
CANN 5.0.2	CANN 6.3.RC2	否。请先执行卸载，再进行安装操作。
CANN 5.0.3	CANN 6.3.RC2	是。
CANN 5.0.3.6	CANN 6.3.RC2	是。
CANN 5.0.4	CANN 6.3.RC2	是。
CANN 5.0.4.6	CANN 6.3.RC2	是。
CANN 5.1.RC1	CANN 6.3.RC2	是。
CANN 5.1.RC2	CANN 6.3.RC2	是。
CANN 6.0.RC1	CANN 6.3.RC2	是。
CANN 6.0.1	CANN 6.3.RC2	是。
CANN 6.3.RC1	CANN 6.3.RC2	是。

10.2 升级操作

升级流程

- 可单独升级CANN软件。
- 如果也要升级固件和驱动，请按照“固件->驱动->CANN软件”的顺序进行升级。

升级驱动和固件

请根据**表10-3**参考对应文档进行升级操作。

表 10-3 驱动和固件升级指导

产品型号	参考文档
Atlas 200I SoC A1	请参见《**Ascend 310P 23.0.RC2 NPU驱动和固件升级指南（AI加速卡）**》。

 版权所有 © 华为技术有限公司

产品型号	参考文档
Atlas 300I Pro	请参见《Ascend 310P 23.0.RC2 NPU驱动和固件升级指南（AI加速卡）》。
Atlas 300V Pro	请参见《Ascend 310P 23.0.RC2 NPU驱动和固件升级指南（AI加速卡）》。
Atlas 300V	请参见《Ascend 310P 23.0.RC2 NPU驱动和固件升级指南（AI加速卡）》。
Atlas 300I Duo	请参见《Ascend 310P 23.0.RC2 NPU驱动和固件升级指南（AI加速卡）》。
A300T-9000	请参见《Ascend 910 23.0.RC2 NPU驱动和固件升级指南（AI加速卡）》。
Atlas 300T Pro	请参见《Ascend 910 23.0.RC2 NPU驱动和固件升级指南（AI加速卡）》。
A500 Pro-3000	请参见所配置标卡对应的文档。
A800-3000	请参见所配置标卡对应的文档。
A800-3010	请参见所配置标卡对应的文档。
A800-9000	请参见《Ascend 910 23.0.RC2 NPU驱动和固件升级指南（AI服务器）》。
A800-9010	请参见《Ascend 910 23.0.RC2 NPU驱动和固件升级指南（AI服务器）》。
A900-9000	请参见《Ascend 910 23.0.RC2 NPU驱动和固件升级指南（AI服务器）》。

升级 CANN 软件

说明

升级开发套件包前请确保安装目录可用空间大于7G，如不满足请清理空间或更换安装目录。

CANN软件升级操作如下：

步骤1 以软件包的安装用户登录软件包的安装环境。

步骤2 进入软件包所在路径。

步骤3 增加对软件包的可执行权限。

```
chmod +x 软件包名.run
```

步骤4 执行如下命令校验软件包安装文件的一致性和完整性。

```
./软件包名.run --check
```

步骤5 软件包升级。

- 从CANN 3.2.1升级到当前版本。
  - 若用户安装时指定了安装路径，执行命令如下：

    ```
    ./软件包名.run --upgrade --install-path=<path>
    ```

 版权所有 © 华为技术有限公司

其中*<path>*为用户指定的软件包安装目录，软件包名请根据实际包名进行替换。

- 若用户安装时未指定安装路径，执行命令如下：
  ```
  ./软件包名.run --upgrade
  ```

- 从CANN 3.3.0起低版本升级到高版本，可自动从安装配置文件中导入原有配置信息，无需输入任何安装参数。执行命令如下：
  ```
  ./软件包名.run --upgrade
  ```

说明

- 若您获取的是***.deb**包，升级同安装，具体请参考**A.2 安装和卸载CANN软件包（适用于.deb格式）**。
- 若您获取的是***.rpm**包，升级命令参考如下（其中***.rpm**请根据实际软件包全名替换）：
  - root用户：
    ```
    rpm -Uvh *.rpm
    ```
  - 非root用户（请自行获取所需的sudo权限）：
    ```
    sudo -E rpm -Uvh *.rpm
    ```

  如果执行以上命令升级失败，并确认需要升级，则可使用“--force”强制升级，如：**rpm -Uvh *.rpm --force**

----结束

 版权所有 © 华为技术有限公司

11 卸载

如果用户只是卸载CANN软件包（如nnrt、toolkit等），那卸载没有先后顺序，但是如果也要卸载驱动和固件，则需要卸载其他软件包以后再卸载驱动和固件。

通过*.run格式安装CANN软件后，可参考以下两种方式进行卸载。如果安装的软件包格式为***.deb**或***.rpm**，可参考**A.2 安装和卸载CANN软件包（适用于.deb格式）**或**A.3 安装和卸载CANN软件包（适用于.rpm格式）**进行卸载。

方法一 脚本卸载

用户可以通过卸载脚本完成卸载。

步骤1 进入软件的卸载脚本所在路径，一般放置在“script”目录下（“script”所在路径请以实际为准）。

以toolkit和nnae软件包为例：

- 进入toolkit软件的卸载脚本所在路径（nnrt目录结构同toolkit）
  cd *<path>*/**ascend-toolkit/***<version>*/*{arch}*-**linux/script**
- 进入nnae软件的卸载脚本所在路径（tfplugin目录结构同nnae）
  cd *<path>*/**nnae/***<version>*/**script**

*其中<path>*为软件包的安装路径，*<version>*为软件包版本，*{arch}*-**linux**为CPU架构，请用户根据实际情况替换。

步骤2 执行**./uninstall.sh**命令运行脚本，完成卸载。

卸载完成后，若显示如下信息，则说明软件卸载成功：
[INFO] *xxx* uninstall success

*xxx*表示卸载的实际软件包名。

----结束

方法二 软件包卸载

如果用户想要对已安装的软件包进行卸载，可以执行如下步骤：

步骤1 以软件包的安装用户登录软件包的安装环境。

步骤2 进入软件包所在路径。

 版权所有 © 华为技术有限公司

步骤3 执行以下命令卸载软件包。

```
./软件包名.run --uninstall
```

卸载完成后，若显示如下信息，则说明软件卸载成功：

```
[INFO] xxx uninstall success
```

*xxx*表示卸载的实际软件包名。

----结束

 版权所有 © 华为技术有限公司

12 后续任务

- 可从**ModelZoo**获取离线模型或训练脚本。
- 如需进行容器部署，有以下两种方式：
  - （推荐）直接从**AscendHub**上拉取推理或训练镜像。
  - 自行构建容器镜像，详情请参考《**MindX DL Ascend Docker Runtime用户指南**》。
- 如需进行推理应用开发，可参考《**CANN 6.3.RC2 应用软件开发指南 (C&C++)**》或《**CANN 6.3.RC2 应用软件开发指南 (Python)**》在开发环境上开发应用程序。
- 如需进行模型迁移和训练，可参考《**CANN 6.3.RC2 TensorFlow 1.15网络模型迁移和训练指南**》或《**CANN TensorFlow 2.6网络模型迁移和训练指南**》和《**PyTorch 模型迁移和训练指南**》。

版权所有 © 华为技术有限公司

A 常用操作

A.1 安装、升级和卸载二进制算子包

包含动态shape网络和ACLNN单算子直调场景下须安装二进制算子包，二进制算子包支持安装、升级和卸载功能。

约束

二进制算子包依赖CANN软件包toolkit或nnae，执行安装和升级操作时，当前环境需已安装配套版本的toolkit或nnae，并使用同一用户安装。

下载软件包

软件安装前，请参考**表A-1**获取所需软件包和对应的数字签名文件，各软件包版本号需要保持一致。

下载本软件即表示您同意**华为企业软件许可协议**的条款和条件。

表 A-1 CANN 软件包

名称	软件包	说明	获取链接
二进制算子包	Ascend-cann-kernels-*{chip_type}*_*{version}*_linux.run	在包含动态shape网络或ACLNN单算子直调的场景下使用。	**获取链接**

说明

*{chip_type}*表示芯片类型，*{version}*表示软件版本号，请根据当前昇腾AI处理器的类型选择对应的软件包。

软件数字签名验证

为了防止软件包在传递过程或存储期间被恶意篡改，下载软件包时需下载对应的数字签名文件用于完整性验证。

 版权所有 © 华为技术有限公司

在软件包下载之后，请参考《 OpenPGP签名验证指南 》，对从Support网站下载的软件包进行PGP数字签名校验。如果校验失败，请不要使用该软件包，先联系华为技术支持工程师解决。

使用软件包安装/升级之前，也需要按上述过程先验证软件包的数字签名，确保软件包未被篡改。

运营商客户请访问：**http://support.huawei.com/carrier/digitalSignatureAction**

企业客户请访问：**https://support.huawei.com/enterprise/zh/tool/pgp-verify-TL1000000054**

相关操作（适用于.run 格式）

二进制算子包支持安装、升级和卸载等操作，用户根据实际需要选择相应参数，可参见**表A-2**。

以安装二进制算子包为例，命令参考如下（请注意将命令中的*软件包名*.run替换为实际包名）：

```
./软件包名.run --install
```

安装后的路径（以跟随toolkit安装为例）：“*软件包安装路径*/ascend-toolkit/latest/opp/built-in/op_impl/ai_core/tbe/kernel”。

说明

若环境中同时安装了配套版本的toolkit或nnae软件包，安装或升级时需通过以下两种方式识别具体的安装目录。

- 方式一：执行命令时需要添加参数 **--type=***<package_type>*，识别到具体的安装目录。
- 方式二：先配置需依赖的toolkit或nnae的环境变量，再执行安装或升级。示例如下（以root用户默认安装路径为例）：
  ```
  # 安装toolkit时配置
  . /usr/local/Ascend/ascend-toolkit/set_env.sh
  # 安装nnae时配置
  . /usr/local/Ascend/nnae/set_env.sh
  ```

二进制算子包可通过.run格式的软件包单独卸载，且支持通过软件包toolkit或nnae统一卸载，具体操作请参考**11 卸载**。

表 A-2 算子包支持的参数说明

参数	说明
--help | -h	查询帮助信息。
--version	查询版本信息。
--info	查询软件包构建信息。
--list	查询软件包文件列表。
--check	检查软件包的一致性和完整性。
--quiet	静默安装，跳过交互式信息。
--nox11	不使用x11模式运行。

 版权所有 © 华为技术有限公司

参数	说明
--noexec	解压软件包到当前目录，但不执行安装脚本。配套--extract=<path>使用，格式为： --noexec --extract=<path>。
--extract=<path>	解压软件包中文件到指定目录。
--tar arg1 [arg2 ...]	对软件包执行tar命令，使用tar后面的参数作为命令的参数。例如执行**--tar xvf**命令，解压run安装包的内容到当前目录。
--install	安装软件包。后面可以指定安装路径--install-path=<path>，也可以不指定安装路径，直接安装到默认路径下。
--devel	按照开发模式安装软件包，即只安装开发环境需安装的文件。
--install-for-all	安装或升级时，允许其他用户具有安装群组的权限。 当安装或者升级携带该参数时，软件包中创建的目录及文件，其他用户权限=安装群组权限。 该参数需要与--install、--devel、--upgrade等其中一个参数配合使用，例如**./***软件包名***.run** --install --install-for-all **说明** 使用该参数将会存在安全风险：其他所有用户都有权限访问安装目录，请谨慎使用。
--install-path=<path>	指定安装路径，当环境上存在全局配置文件“ascend_cann_install.info”时，支持使用该参数，但指定路径必须与全局配置文件中保存的安装路径保持一致。如用户想更换安装路径，需先卸载原路径下的CANN软件包并确保全局配置文件“ascend_cann_install.info”已被删除。 可在如下目录查看是否存在该文件： ● root用户：/etc/Ascend ● 非root用户：*${HOME}*/Ascend 若不指定，将安装到默认路径下： ● 若使用root用户安装，默认安装路径为：/usr/local/Ascend。 ● 若使用非root用户安装，则默认安装路径为：*${HOME}*/Ascend。 若通过该参数指定了安装目录，运行用户需要对指定的安装路径有可读写权限。
--uninstall	卸载已安装的软件。
--upgrade	升级已安装的软件。

 版权所有 © 华为技术有限公司

参数	说明
--type=<package_type>	指定已安装的nnae或toolkit软件包类型，用于在执行安装（--install）时指定跟随安装的软件包（“nnae”、“toolkit”），以识别到具体的安装目录。该参数需要配合“--install”一起使用。
--force	强制安装、升级命令。 该参数需要配合“--install”或“--upgrade”一起使用。

相关操作（适用于.deb 和.rpm 格式）

二进制算子包.deb格式安装、卸载请参考**A.2 安装和卸载CANN软件包（适用于.deb格式）**，.rpm格式安装请参考**A.3 安装和卸载CANN软件包（适用于.rpm格式）**。

A.2 安装和卸载 CANN 软件包（适用于.deb 格式）

安装

若您获取的是***.deb**包，安装过程参考如下：

1. 以安装用户登录服务器。

   说明

   - 如果使用非root用户安装，需要用到提权命令，请用户自行获取所需的sudo权限。使用完成后请取消涉及高危命令的权限，否则有sudo提权风险。
   - 若从root用户切换至非root用户，执行命令如下：

     su - *username*

2. 安装deb包（其中***.deb**请根据实际软件包全名替换）。
   - 若使用root用户安装，执行命令如下：
     ```
     dpkg -i *.deb
     ```
   - 若使用非root用户安装，执行命令如下：
     ```
     sudo -E dpkg -i *.deb
     ```

   说明

   - .deb格式的CANN软件包遵循deb通用规则，安装后其他用户均可使用。如果安装驱动时未携带“--install-for-all”，并且CANN软件包运行用户为非root，则该CANN软件包运行用户所属的属组必须和驱动运行用户所属属组相同；如果不同，请用户自行添加到驱动运行用户属组。
   - 安装完成后可执行命令**apt list | grep nnrt**（以nnrt软件包为例）查询软件包安装信息。
   - .deb格式的CANN软件包只支持默认路径安装，默认安装路径为“/usr/local/Ascend”。

卸载

如果用户只是卸载CANN软件包（如nnrt、toolkit等），那卸载没有先后顺序，但是如果也要卸载驱动和固件，则需要卸载其他软件包以后再卸载驱动和固件。

 版权所有 © 华为技术有限公司

若您获取的是***.deb**包，如卸载nnrt软件，卸载命令参考如下：

- dpkg方式：
  - root用户：
    ```
    dpkg -P ascend-cann-nnrt
    ```
  - 非root用户（请自行获取所需的sudo权限）：
    ```
    sudo -E dpkg -P ascend-cann-nnrt
    ```
- apt方式：
  - root用户：
    ```
    apt remove ascend-cann-nnrt
    ```
  - 非root用户（请自行获取所需的sudo权限）：
    ```
    sudo -E apt remove ascend-cann-nnrt
    ```

A.3 安装和卸载 CANN 软件包（适用于.rpm 格式）

若您获取的是***.rpm**的CANN软件包，安装过程参考如下：

1. 以安装用户登录服务器。

   说明

   - 如果使用非root用户安装，需要用到提权命令，请用户自行获取所需的sudo权限。使用完成后请取消涉及高危命令的权限，否则有sudo提权风险。
   - 安装rpm包前请将设置python3（以实际安装的python版本为准，请参考**6.3 安装依赖**章节）环境变量的命令写入“/root/.bashrc”文件中。
   - 若从root用户切换至非root用户，执行命令如下：

     su - *username*

2. 安装rpm包（其中***.rpm**请根据实际软件包全名替换）。
   - 若使用root用户安装，执行命令如下：
     ```
     rpm -ivh *.rpm
     ```
   - 若使用非root用户安装，执行命令如下：
     ```
     sudo -E rpm -ivh *.rpm
     ```

   说明

   - .rpm格式的CANN软件包遵循rpm通用规则，安装后其他用户均可使用。如果安装驱动时未携带“--install-for-all”，并且CANN软件包运行用户为非root，则该CANN软件包运行用户所属的属组必须和驱动运行用户所属属组相同；如果不同，请用户自行添加到驱动运行用户属组。
   - 如果用户之前已安装过同一CANN软件包，请先卸载该软件包，再执行安装操作。
   - 安装完成后可执行命令**yum list | grep nnrt**（以nnrt软件包为例）查询软件包安装信息。
   - .rpm格式的CANN软件包只支持默认路径安装，默认安装路径为“/usr/local/Ascend”。

卸载

如果用户只是卸载CANN软件包（如nnrt、toolkit等），那卸载没有先后顺序，但是如果也要卸载驱动和固件，则需要卸载其他软件包以后再卸载驱动和固件。

若您获取的是***.rpm**的CANN软件包，如卸载nnrt软件，卸载命令参考如下：

说明

以下命令中的*{arch}*请根据实际架构替换（aarch64或x86_64）。

 版权所有 © 华为技术有限公司

- rpm方式：
  - root用户：
    rpm -e Ascend-cann-nnrt.*{arch}*
  - 非root用户（请自行获取所需的sudo权限）：
    sudo -E rpm -e Ascend-cann-nnrt.*{arch}*
- yum方式：
  - root用户：
    yum remove Ascend-cann-nnrt.*{arch}*
  - 非root用户（请自行获取所需的sudo权限）：
    sudo -E yum remove Ascend-cann-nnrt.*{arch}*

A.4 安装、回退和卸载 CANN 补丁包

CANN软件包支持补丁升级，即支持独立升级单个或多个库文件的功能。因此发布CANN补丁包。

约束

- 补丁仅能支持对应的基线版本或相关的补丁版本进行升级。
- 基于同一基线版本的补丁，需保证后续安装的补丁版本大于之前安装的补丁版本。
- 仅支持回退一个补丁版本。

安装、回退影响

- 安装、回退过程禁止进行其他维护操作动作。
- 补丁包安装、回退过程中会导致业务中断。
- 补丁包安装、回退后，不会影响正常业务。

CANN 补丁包相关操作（适用于.run 格式）

CANN补丁包支持安装、回退等操作，用户根据实际需要选择相应参数。参数说明请参见**表A-3**。

以安装冷补丁为例，命令参考如下（请注意将命令中的*软件包名*.run替换为实际包名）：

./*软件包名*.run --install

如需指定补丁包安装路径，可添加“--install-path”参数指定安装路径（如“/usr/local/Ascend”）。

表 A-3 补丁包支持参数说明

参数	说明
--help | -h	查询帮助信息。
--version	查询软件包版本。
--info	查询软件包构建信息。

 版权所有 © 华为技术有限公司

参数	说明
--list	查询软件包文件列表。
--check	检查软件包的一致性和完整性。
--quiet	静默安装，跳过交互式信息。
--nox11	安装过程中不弹出图形终端窗口。
--noexec	解压软件包到当前目录，但不执行安装脚本。配套--extract=<path>使用，格式为： --noexec --extract=<path>
--extract=<path>	解压软件包中文件到指定目录。
--tar arg1 [arg2 ...]	对软件包执行tar命令，使用tar后面的参数作为命令的参数。例如执行**--tar xvf**命令，解压run安装包的内容到当前目录。
--install	安装补丁包。
--rollback	回退到前一版本。
--uninstall	清除原版本的备份库文件，不支持回退。
--install-path=<path>	指定补丁包安装路径。 若通过该参数指定了安装路径，运行用户需要对指定的安装路径有可读写权限。 若不指定： ● 优先读取全局配置文件“ascend_cann_install.info”中的安装路径。 可在如下目录查看是否存在该文件： – root用户：/etc/Ascend – 非root用户：*${HOME}*/Ascend ● 如果当前环境不存在“ascend_cann_install.info”，则将安装到默认路径下： – 若使用root用户安装，默认安装路径为：/usr/local/Ascend – 若使用非root用户安装，默认安装路径为：*${HOME}*/Ascend

CANN 补丁包安装（适用于.rpm 格式）

说明

- 如果使用非root用户执行相关操作，需要用到提权命令，请用户自行获取所需的sudo权限。使用完成后请取消涉及高危命令的权限，否则有sudo提权风险。
- 如果使用非root用户执行相关操作，请注意在以下执行的安装命令前加上**sudo -E**。例如以非root用户安装冷补丁：**sudo -E rpm -Uvh *.rpm**。

CANN补丁包安装命令示例如下（其中***.rpm**请根据实际软件包全名替换）：

```
rpm -Uvh *.rpm
```

A.5 编译安装 PyTorch

安装依赖

选择编译安装方式安装时需要安装系统依赖。目前支持CentOS与Ubuntu操作系统。

- CentOS
  ```
  yum install -y patch libjpeg-turbo-devel dos2unix openblas git
  yum install -y gcc==7.3.0 cmake==3.12.0 #gcc7.3.0版本及以上，cmake3.12.0版本及以上。若用户要安装1.11.0版本PyTorch，则gcc需为7.5.0版本以上。
  ```
- Ubuntu
  ```
  apt-get install -y patch build-essential libbz2-dev libreadline-dev wget curl llvm libncurses5-dev libncursesw5-dev xz-utils tk-dev liblzma-dev m4 dos2unix libopenblas-dev git
  apt-get install -y gcc==7.3.0 cmake==3.12.0 #gcc7.3.0版本及以上，cmake3.12.0版本及以上。若用户要安装1.11.0版本PyTorch，则gcc需为7.5.0版本以上。
  ```

安装 PyTorch

以下操作步骤以安装PyTorch 1.8.1版本为例。

步骤1 安装官方torch包。

- 快速安装。仅在x86_64架构下支持。
  ```
  pip3 install torch==1.8.1+cpu
  ```
  若执行以上命令安装cpu版本PyTorch报错，请点击下方PyTorch官方链接下载whl包安装。
  - PyTorch 1.8.1版本：
    - x86_64架构：**下载链接**。
    - aarch64架构：**下载链接**。
  - PyTorch 1.11.0版本：
    - x86_64架构：**下载链接**。
    - aarch64架构：**下载链接**。
  - PyTorch 2.0.1版本：
    - x86_64架构：**下载链接**。
    - aarch64架构：**下载链接**。

  执行如下安装命令：
  ```
  # 用户请根据自己实际情况修改命令中的安装包名
  pip3 install torch-1.8.1+cpu-cp37-cp37m-linux_x86_64.whl
  ```
- 编译安装。在x86_64和aarch64架构下均可使用。
  a. 下载PyTorch v1.8.1源码包，1.11.0版本请替换版本号为v1.11.0，2.0.1版本请替换版本号为v2.0.1。
  ```
  git clone -b v1.8.1 https://github.com/pytorch/pytorch.git --depth=1 pytorch_v1.8.1
  ```
  b. 进入源码包获取被动依赖代码。
  ```
  cd pytorch_v1.8.1
  git submodule sync
  git submodule update --init --recursive
  ```

 版权所有 © 华为技术有限公司

c. 配置环境变量。

```
export USE_XNNPACK=0
```

d. 执行编译安装。

```
python3 setup.py install
```

步骤2 编译生成PyTorch插件的二进制安装包。请参考**表A-4**下载对应PyTorch版本分支代码。

```
# 下载对应PyTorch版本分支代码，进入插件根目录，以v1.8.1-5.0.rc2为例，其他版本请替换对应版本号
git clone -b v1.8.1-5.0.rc2 https://gitee.com/ascend/pytorch.git
cd pytorch
# 指定Python版本编包方式，以Python3.7为例，其他Python版本请使用 --python=3.8或--python3.9
bash ci/build.sh --python=3.7
```

步骤3 安装pytorch/dist目录下生成的插件torch_npu包，如果使用非root用户安装，需要在命令后加**--user**。

```
# 请用户根据实际情况更改命令中的torch_npu包名
pip3 install --upgrade dist/torch_npu-1.8.1.post2-cp38-cp38-linux_aarch64.whl
```

步骤4 安装对应框架版本的torchvision。

```
#PyTorch 1.8.1需安装0.9.1版本，PyTorch 1.11.0需安装0.12.0版本，PyTorch 2.0.1版本需安装0.15.2版本
pip3 install torchvision==0.9.1
```

步骤5 配置环境变量，验证是否安装成功。

1. 新建环境变量shell脚本env.sh，写入以下代码：

```
# 配置CANN相关环境变量
CANN_INSTALL_PATH_CONF='/etc/Ascend/ascend_cann_install.info'
if [ -f $CANN_INSTALL_PATH_CONF ]; then
    DEFAULT_CANN_INSTALL_PATH=$(cat $CANN_INSTALL_PATH_CONF | grep Install_Path | cut -d "=" -f 2)
else
    DEFAULT_CANN_INSTALL_PATH="/usr/local/Ascend/"
fi
CANN_INSTALL_PATH=${1:-${DEFAULT_CANN_INSTALL_PATH}}
if [ -d ${CANN_INSTALL_PATH}/ascend-toolkit/latest ];then
    source ${CANN_INSTALL_PATH}/ascend-toolkit/set_env.sh
else
    source ${CANN_INSTALL_PATH}/nnae/set_env.sh
fi

#将Host日志输出到串口,0-关闭/1-开启
export ASCEND_SLOG_PRINT_TO_STDOUT=0

ulimit -SHn 512000
path_lib=$(python3.7 -c """
import sys
import re
result=''
for index in range(len(sys.path)):
    match_sit = re.search('-packages', sys.path[index])
    if match_sit is not None:
        match_lib = re.search('lib', sys.path[index])

        if match_lib is not None:
            end=match_lib.span()[1]
            result += sys.path[index][0:end] + ':'

        result+=sys.path[index] + '/torch/lib:'
print(result)"""
)

echo ${path_lib}

export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib/:${path_lib}:$LD_LIBRARY_PATH
```

执行脚本，配置环境变量。

```
source env.sh
```

 版权所有 © 华为技术有限公司

2. 执行如下命令，验证PyTorch是否安装成功。
```
python3 -c "import torch;import torch_npu; a = torch.randn(3, 4).npu(); print(a + a);"
```
显示如下回显证明PyTorch框架与插件安装成功。
```
tensor([[-0.6066,  6.3385,  0.0379,  3.3356],
        [ 2.9243,  3.3134, -1.5465,  0.1916],
        [-2.1807,  0.2008, -1.1431,  2.1523]], device='npu:0')
```

----结束

表 A-4 Ascend 配套 PyTorch 代码分支

AscendPyTorch版本	CANN版本	支持PyTorch版本	代码分支名称
5.0.rc2	CANN 6.3.RC2	1.8.1.post2	v1.8.1-5.0.rc2
		1.11.0.post1	v1.11.0-5.0.rc2
		2.0.1.rc1	v2.0.1-5.0.rc2

A.6 安装 3.5.2 版本 cmake

步骤1 使用wget下载cmake源码包，可以下载到安装服务器任意目录，命令为：
```
wget https://cmake.org/files/v3.5/cmake-3.5.2.tar.gz --no-check-certificate
```

步骤2 进入下载后的目录，解压源码包，命令为：
```
tar -zxvf cmake-3.5.2.tar.gz
```

步骤3 进入解压后的文件夹，执行配置，编译和安装命令：
```
cd cmake-3.5.2
./bootstrap --prefix=/usr
make
sudo make install
```

步骤4 安装完成后重新执行**cmake --version**查看版本号。

----结束

A.7 安装 7.3.0 版本 gcc

以下步骤请在root用户下执行。

步骤1 下载gcc-7.3.0.tar.gz，下载地址为**https://mirrors.tuna.tsinghua.edu.cn/gnu/gcc/gcc-7.3.0/gcc-7.3.0.tar.gz**。

步骤2 安装gcc时候会占用大量临时空间，所以先执行下面的命令清空/tmp目录：
```
rm -rf /tmp/*
```

步骤3 安装依赖（以CentOS和Ubuntu系统为例）。

- CentOS执行如下命令安装。
```
yum install bzip2
```
- Ubuntu执行如下命令安装。
```
apt-get install bzip2
```

 版权所有 © 华为技术有限公司

步骤4 编译安装gcc。

1. 进入gcc-7.3.0.tar.gz源码包所在目录，解压源码包，命令为：
```
tar -zxvf gcc-7.3.0.tar.gz
```
2. 进入解压后的文件夹，执行如下命令下载gcc依赖包：
```
cd gcc-7.3.0
./contrib/download_prerequisites
```
如果执行上述命令报错，需要执行如下命令在“gcc-7.3.0/”文件夹下下载依赖包：
```
wget http://gcc.gnu.org/pub/gcc/infrastructure/gmp-6.1.0.tar.bz2
wget http://gcc.gnu.org/pub/gcc/infrastructure/mpfr-3.1.4.tar.bz2
wget http://gcc.gnu.org/pub/gcc/infrastructure/mpc-1.0.3.tar.gz
wget http://gcc.gnu.org/pub/gcc/infrastructure/isl-0.16.1.tar.bz2
```
下载好上述依赖包后，重新执行以下命令：
```
./contrib/download_prerequisites
```
如果上述命令校验失败，需要确保依赖包为一次性下载成功，无重复下载现象。
3. 执行配置、编译和安装命令：
```
./configure --enable-languages=c,c++ --disable-multilib --with-system-zlib --prefix=/usr/local/gcc7.3.0
make -j15    # 通过grep -w processor /proc/cpuinfo|wc -l查看cpu数，示例为15，用户可自行设置相应参数。
make install
```

说明

其中“--prefix”参数用于指定gcc7.3.0安装路径，用户可自行配置，但注意不要配置为“/usr/local”及“/usr”，因为会与系统使用软件源默认安装的gcc相冲突，导致系统原始gcc编译环境被破坏。示例指定为“/usr/local/gcc7.3.0”。

步骤5 配置环境变量（请在实际需要时再进行配置）。

例如用户在启动在线推理或训练进程前需执行如下命令配置环境变量。

```
export LD_LIBRARY_PATH=/usr/local/gcc7.3.0/lib64:${LD_LIBRARY_PATH}
```

其中“/usr/local/gcc7.3.0”为**步骤4.3**中配置的gcc7.3.0安装路径，请根据实际情况替换。

----结束

A.8 配置网卡 IP 地址

安装完操作系统后，需在当前iBMC远程管理界面中配置网卡IP地址才能远程连接服务器，配置方法如下（以CentOS、Ubuntu系统为例）。

CentOS/EulerOS 操作系统配置网卡 IP 地址

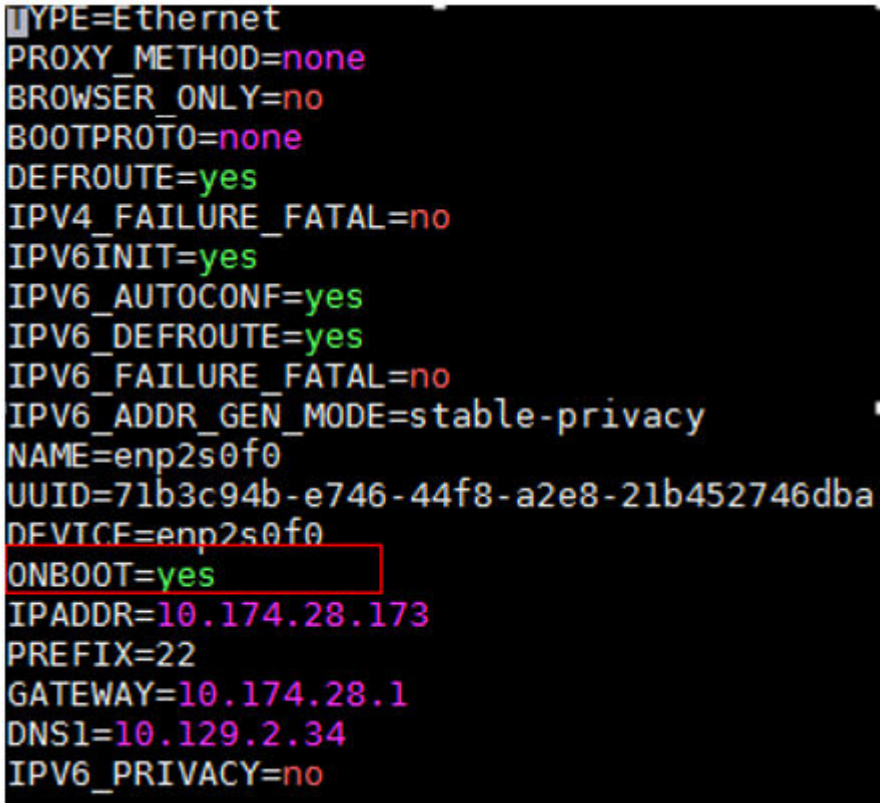
步骤1 以root用户进入OS界面。

步骤2 进入需配置IP地址的网卡的配置文件，此处以网卡名为enp2s0f0举例，具体网卡和路径请以实际环境为准。
```
vi /etc/sysconfig/network-scripts/ifcfg-enp2s0f0
```

步骤3 配置对应网卡的业务IP地址、子网掩码和网关，具体以实际为准，如**图A-1**所示。

 版权所有 © 华为技术有限公司

图 A-1 设置网络参数



说明

- BOOTPROTO表示设备的IP类型，如果是静态IP可设置成static或none；如果是动态IP，请配置成dhcp，自动获取IP。
- 更改ONBOOT参数为yes，设置自动启动网络连接。
- PREFIX表示网络位，值为22对应的子网掩码为255.255.252.0。

步骤4 重启网络服务。

```
service network restart
```

步骤5 查看IP地址是否配置成功。

```
ifconfig
```

步骤6 执行以下命令查看服务状态。

```
systemctl status NetworkManager
```

若显示为disabled，则执行以下命令将NetworkManager服务设置为自启动。

```
systemctl enable NetworkManager
```

再次查看服务状态，若为enabled则配置成功。

----结束

Ubuntu 操作系统配置网卡 IP 地址

步骤1 以root用户进入OS界面。

步骤2 进入网卡的配置文件，此处以网卡名为enp125s0f0举例，具体网卡请以实际环境为准。

 版权所有 © 华为技术有限公司

vi /etc/netplan/01-netcfg.yaml

步骤3 配置业务IP地址、子网掩码和网关，如图A-2所示。

图 A-2 配置 IP 地址

```
root@Taishan-2280-V2:/etc/netplan# cat 01-netcfg.yaml
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    enp125s0f0:
      dhcp4: no
      addresses: [10.174.216.162/23, ]
      gateway4: 10.174.216.1

root@Taishan-2280-V2:/etc/netplan#
```

步骤4 应用配置文件。

netplan apply

步骤5 查看IP地址是否配置成功。

ifconfig

----结束

A.9 设置用户有效期

为保证用户的安全性，应设置用户的有效期，使用系统命令chage来设置用户的有效期。

命令为：

```
chage [-m mindays] [-M maxdays] [-d lastday] [-I inactive] [-E expiredate] [-W warndays] user
```

相关参数请参见表A-5。

表 A-5 设置用户有效期

参数	参数说明
-m	口令可更改的最小天数。设置为“0”表示任何时候都可以更改口令。
-M	口令保持有效的最大天数。设置为“-1”表示可删除这项口令的检测。设置为“99999”，表示无限期。
-d	上一次更改的日期。
-I	停滞时期。过期指定天数后，设定密码为失效状态。
-E	用户到期的日期。超过该日期，此用户将不可用。
-W	用户口令到期前，提前收到警告信息的天数。

版权所有 © 华为技术有限公司

参数	参数说明
-l	列出当前的设置。由非特权用户来确定口令或帐户何时过期。

说明

- 表A-5只列举出常用的参数，用户可通过**chage --help**命令查询详细的参数说明。
- 日期格式为YYYY-MM-DD，如chage -E 2017-12-01 *test*表示用户*test*的口令在2017年12月1日过期。
- User必须填写，填写时请替换为具体用户，默认为root用户。

举例说明：修改用户*test*的有效期为90天。

```
chage -M 90 test
```

A.10 配置 pip 源

配置pip源，配置方法如下：

步骤1 使用软件包的安装用户，执行如下命令：

```
cd ~/.pip
```

如果提示目录不存在，则执行如下命令创建：

```
mkdir ~/.pip
cd ~/.pip
```

步骤2 编辑pip.conf文件。

使用**vi pip.conf**命令打开pip.conf文件，写入如下内容：

```
[global]
#以华为源为例，请根据实际情况进行替换。
index-url = https://mirrors.huaweicloud.com/repository/pypi/simple
trusted-host = mirrors.huaweicloud.com
timeout = 120
```

步骤3 执行**:wq!**命令保存文件。

----结束

A.11 查询软件包版本信息

以下操作适用于查询CANN软件包（如toolkit等）、toolbox软件包的版本信息。

步骤1 以软件包的安装用户登录软件包的安装环境。

步骤2 进入软件包安装信息文件目录。（以下以Ascend-cann-toolkit软件包为例，其他软件包以实际目录为准）

```
cd /usr/local/Ascend/ascend-toolkit/latest/{arch}-linux
```

其中/usr/local/Ascend为root用户默认安装路径，请用户根据实际安装路径替换。*{arch}*表示CPU架构（aarch64或x86_64）。

步骤3 执行以下命令获取版本信息。

 版权所有 © 华为技术有限公司

```
cat ascend_toolkit_install.info
```

----结束

 版权所有 © 华为技术有限公司

B FAQ

B.1 安装驱动时提示“The user and group are not same with last installation”

问题描述

安装驱动时提示：

```
The user and group are not same with last installation,do not support overwriting installation
```

可能原因

当前指定的驱动运行用户与“/etc/ascend_install.info”文件中记录的运行用户不一致，导致校验失败，安装退出。原因是之前卸载驱动和CANN软件时，aicpu没有被卸载，导致“/etc/ascend_install.info”文件未清空。

解决措施

先卸载aicpu，再安装驱动。卸载aicpu操作如下：

步骤1 以root用户登录安装环境。

步骤2 进入卸载脚本所在目录。

cd /usr/local/Ascend/opp/aicpu/script

步骤3 执行**./uninstall.sh**命令运行脚本，完成卸载。

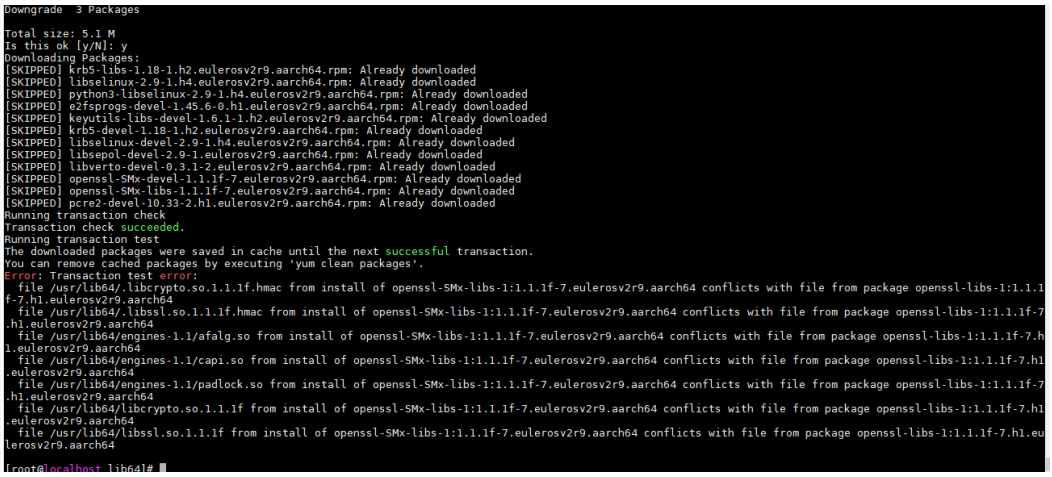
----结束

B.2 openssl-devel 安装时报错提示 so 文件冲突

问题描述

运行yum install -y openssl-devel命令进行安装时，出现so文件冲突错误，如下图所示：

 版权所有 © 华为技术有限公司



可能原因

原生包openssl-libs-1.1.1f-7.h1.eulerosv2r9 和即将安装的openssl-SMx-libs-1.1.1f-7.eulerosv2r9 包冲突。

注意：不要去强制卸载openssl-libs,因为ssl认证的服务依赖这个包，比如scp或者ssh等服务依赖于这个包的libcrypto.so.1.1库。另外这个包丢失，会导致rpm包失效。

解决措施

步骤1 新创建一个文件夹并进入。

```
mkdir openDown
cd openDown
```

步骤2 下载openssl的rpm包，通过以下命令下载到当前文件夹。

```
yum install --downloadonly --downloaddir=. openssl-devel
```

步骤3 rpm命令强制安装。

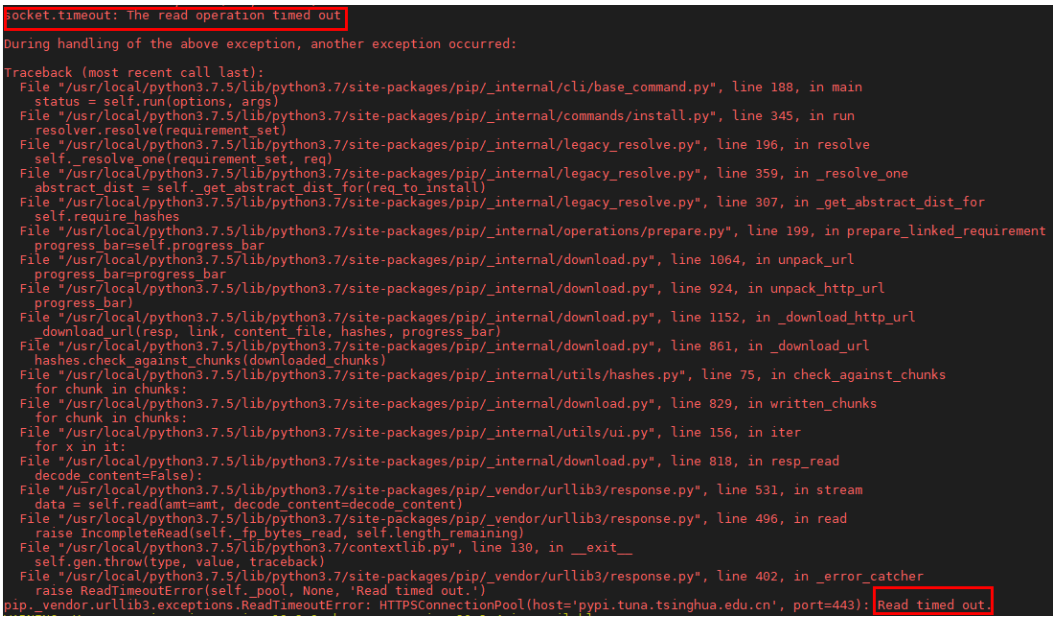
```
rpm -Uvh *.rpm --force
```

----结束

B.3 pip3 安装软件包时，出现 read time out 报错

问题描述

使用pip3下载相关依赖报错，比如执行pip3 install scipy进行安装出现read time out错误，如下图所示：



可能原因

网络不稳定的情况下，下载时间长而且下载速度慢，尤其是下载 numpy和scipy这种较大的包的时候socket数据通信会经常出现这个问题。

解决措施

用 - i 参数来指定常用的其它源，如下命令所示：

```
pip3 install scipy -i xxxxxx
```

xxxxxx 为客户根据需求自行填写可用的源。

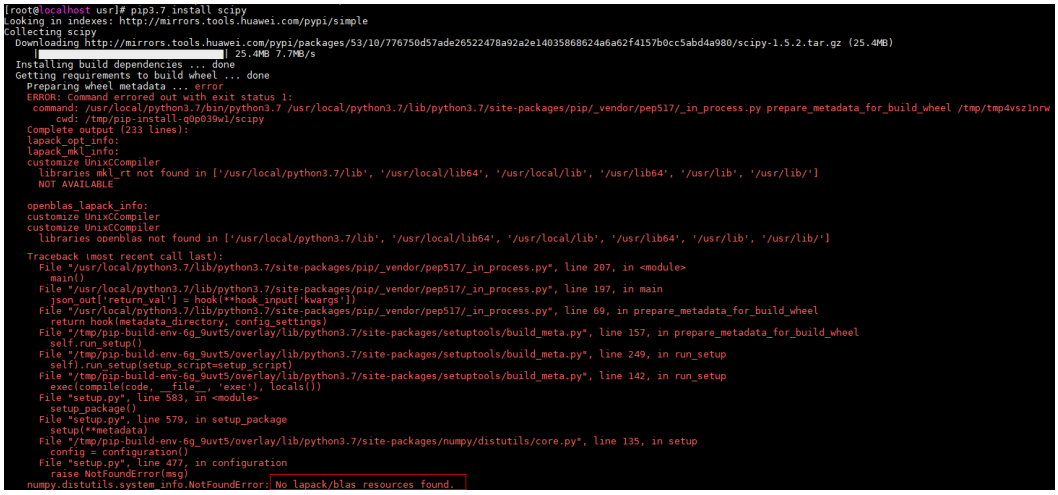
B.4 pip3 install scipy 报错

问题描述

安装scipy时，提示如下错误信息。

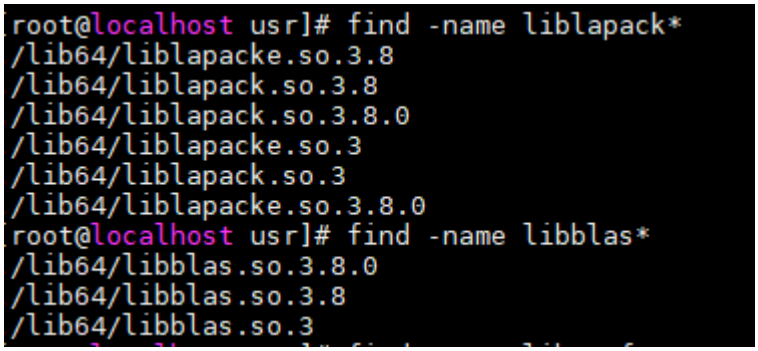
 版权所有 © 华为技术有限公司

图 B-1 错误信息



可能原因

报错信息提示在/usr/lib64目录下找不到lapack和blas的库，但是实际在该目录下能找到，应该是识别不到so.3的库。



解决措施

创建liblapack.so.3和libblas.so.3对应的so的软连接。

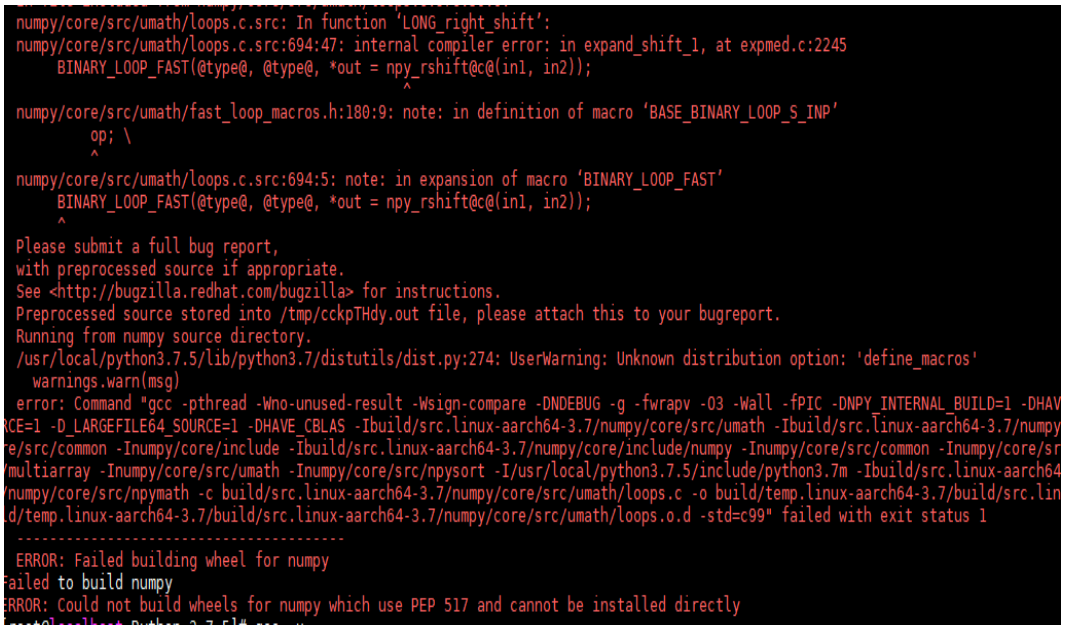
使用软件包的安装用户，执行如下命令：

```
cd /usr/lib64
ln -s libblas.so.3 libblas.so
ln -s liblapack.so.3 liblapack.so
```

B.5 pip3 install numpy 报错

问题描述

安装依赖时，使用**pip3 install numpy**命令安装时报错“Could not build wheels for numpy which use PEP 517 and cannot be install directly”，提示信息如下：



可能原因

centos等系统默认安装的gcc版本较低，导致numpy安装失败。

解决措施

执行如下命令安装：

```
export CFLAGS=-std=c99
pip3 install numpy==1.19.2
```

B.6 pip3 install 报错“subprocess.CalledProcessError: Command '('lsb_release', '-a')' return non-zero exit status 1”

问题描述

安装依赖时，使用**pip3 install xxx**命令安装相关软件时报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”，提示信息如下：

 版权所有 © 华为技术有限公司

```
root@debian:/home/work_env/Python-3.7.5# pip3.7.5 install attrs
ERROR: Exception:
Traceback (most recent call last):
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_internal/cli/base_command.py", line 188, in main
    status = self.run(options, args)
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_internal/commands/install.py", line 286, in run
    with self._build_session(options) as session:
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_internal/cli/base_command.py", line 108, in _build_session
    index_urls=self._get_index_urls(options),
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_internal/download.py", line 559, in __init__
    self.headers["User-Agent"] = user_agent()
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_internal/download.py", line 144, in user_agent
    zip(["name", "version", "id"], distro.linux_distribution()),
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 122, in linux_distribution
    return _distro.linux_distribution(full_distribution_name)
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 677, in linux_distribution
    self.version(),
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 737, in version
    self.lsb_release_attr('release'),
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 899, in lsb_release_attr
    return self._lsb_release_info.get(attribute, '')
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 552, in __get__
    ret = obj.__dict__[self._fname] = self._f(obj)
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 1012, in _lsb_release_info
    stdout = subprocess.check_output(cmd, stderr=devnull)
  File "/usr/local/python3.7.5/lib/python3.7/subprocess.py", line 411, in check_output
    **kwargs).stdout
  File "/usr/local/python3.7.5/lib/python3.7/subprocess.py", line 512, in run
    output=stdout, stderr=stderr)
subprocess.CalledProcessError: Command '('lsb_release', '-a')' returned non-zero exit status 1.
```

可能原因

用户自行编译安装的python3.7.5在执行subprocess模块时，在执行lsb_release -a 时提示找不到lsb_release.py模块，用户自行编译安装的python3.7.5的lib路径是“/usr/local/python3.7.5/lib/python3.7/”，该路径下没有lsb_release.py模块，因此会报错。

解决措施

步骤1 查找缺失文件'lsb_release.py'，使用如下命令：

```
find / -name lsb_release
```

执行上述命令后，得到如下路径，以下仅为示例，用户实际可能有差别：

```
/usr/bin/lsb_release
```

步骤2 将**步骤1**找到的“/usr/bin/lsb_release”文件备份：

```
mv /usr/bin/lsb_release /usr/bin/lsb_release.bak
```

步骤3 然后执行**pip3 list**查看是否解决。

----结束

B.7 Python 版本不一致导致的问题

问题描述

普通用户下，如果用户的其他软件依赖的Python版本和我们要求的Python版本不一致，可能会出现问题。

解决措施

建议使用Python的venv。Python虚拟环境用于将软件包安装与系统隔离开来。

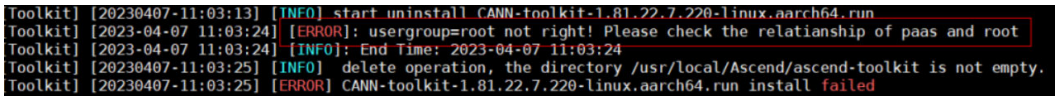
 版权所有 © 华为技术有限公司

B.8 安装 run 包失败，日志中报错“usergroup=root not right！”

问题描述

安装run包失败，日志中报错“usergroup=root not right！”。

图 B-2 报错截图



可能原因

使用非root用户登录环境，切换至root用户执行安装操作时，切换后系统环境变量**USER**值仍为登录环境的非root用户。

解决措施

执行如下命令，手动将**USER**环境变量设置为实际执行安装的用户后，再进行安装：

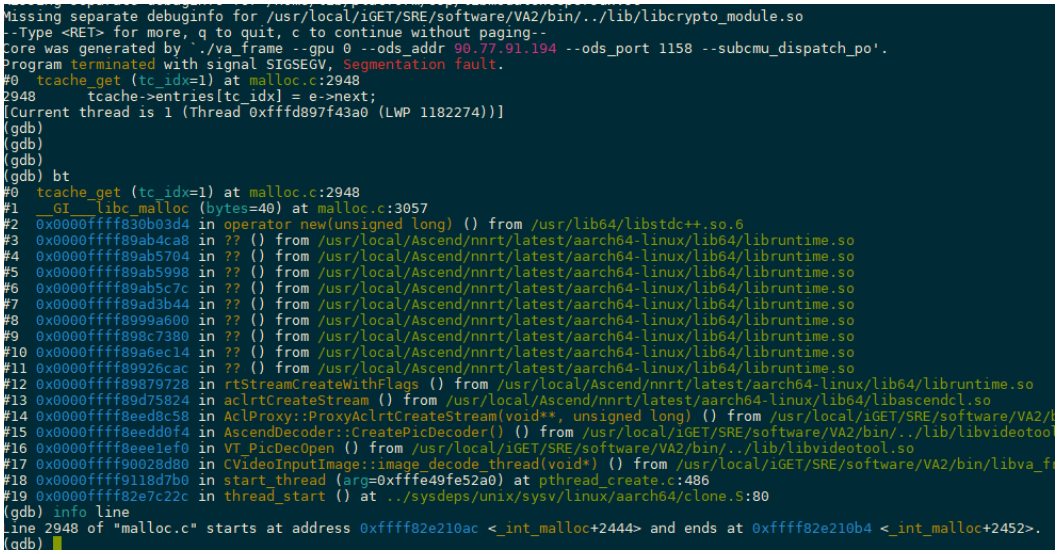
```
export USER=root
```

B.9 使用 glibc 2.28 版本运行业务调用 malloc 接口出现 Core Dump 问题

问题描述

在glibc 2.28版本的docker环境中运行视频解析、目标识别相关业务时出现Core Dump，通过排查堆栈信息，实际为调用glibc的malloc接口时出现异常，堆栈信息如下图所示。

图 B-3 堆栈信息



 版权所有 © 华为技术有限公司

解决措施

因glibc 2.28版本存在bug引发该问题，glibc 2.29版本现已修复，用户可以在执行业务之前，通过设置环境变量的方式修改malloc接口，配置命令如下：

```
export GLIBC_TUNABLES=glibc.malloc.tcache_count=0
```

B.10 openEuler 22.03 上运行业务时，出现 firewalld 相关 soft lockup 或 ksoftirqd 占用 CPU 过高

问题描述

在openEuler 22.03上同时使用默认的18.09版本docker与1.0.2版本firewalld时，运行业务出现firewalld相关soft lockup或ksoftirqd占用CPU过高的情况。

查询结果如下所示：

图 B-4 查询结果 1

```
top - 10:04:39 up 16:40,  5 users,  load average: 3.58, 3.35, 3.11
Tasks: 2020 total,   2 running, 2018 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.1 sy,  0.0 ni, 99.4 id,  0.0 wa,  0.0 hi,  0.5 si,  0.0 st
MiB Mem : 770865.1 total, 691639.8 free,   8959.9 used,  70265.4 buff/cache
MiB Swap:   4096.0 total,   4096.0 free,      0.0 used. 756585.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    588 root      20   0       0      0      0 R  99.3   0.0 363:18.21 ksoftirqd/114
```

图 B-5 查询结果 2

```
Jun 15 03:15:53 localhost kernel: [35477.382201] watchdog: BUG: soft lockup - CPU#166 stuck for 67s! [firewalld:1376895]
```

可能原因

18.09版本docker基于iptables写入规则，1.0.2版本firewalld基于nftables写入规则，两者无法匹配。

解决措施

编辑“/etc/firewalld/firewalld.conf”文件，将**FirewallBackend**字段修改为**iptables**。

B.11 安装原生 PyTorch 框架时使用华为源下载 typing 依赖导致 Python 环境错误

问题描述

在pip设置为华为源时，安装requirements.txt中的typing依赖后，会导致Python环境错误。

解决措施

在pip设置为华为源时，需打开requirements.txt文件，删除typing依赖，再执行命令。

 版权所有 © 华为技术有限公司

```
vi requirements.txt #进入requirements.txt删除typing依赖

pip3 install -r requirements.txt #重新安装依赖列表
```

B.12 PyTorch 框架在 ARM CPU 上算子计算结果异常

问题描述

当前使用的PyTorch官方原生框架在ARM CPU上运行时，算子计算结果会出现异常，此问题为原生框架社区的已知问题，详细内容可参考PyTorch官方社区**ISSUE**。

解决措施

可通过以下方式解决：

- 修改算子输入数据类型，使用float64数据类型进行运算。
- 升级编译ARM版本PyTorch使用的gcc编译器至9.4版本及以上，并使用相同编译器重新编译torch_npu、APEX、MMCV等其他配套软件（避免因编译器版本不匹配导致兼容性问题）。
- 在Docker环境编译torch_npu。
  a. 拉取容器镜像。
     ```
     docker pull quay.io/pypa/manylinux2014_aarch64:latest
     ```
  b. 执行命令查看镜像信息。
     ```
     docker images
     ```
     回显如下：
     ```
     REPOSITORY                                TAG                 IMAGE ID       CREATED        SIZE
     quay.io/pypa/manylinux2014_aarch64        latest              fe2afc7b4b0d   9 days ago     2.09GB
     ```
  c. 启动镜像并在后台运行。
     ```
     docker run -dit  [IMAGE ID] bash
     ```
     [IMAGE ID]为**b**查询出的镜像ID。
  d. 执行命令查看运行的容器。
     ```
     docker ps
     ```
     回显如下：
     ```
     CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
     6c2ead81ede5   fe2afc7b4b0d   "manylinux-entrypoin…"   6 seconds ago   Up 4 seconds             dreamy_einstein
     ```
  e. 执行命令进入容器。
     ```
     docker exec -it [CONTAINER ID] bash
     ```
     [CONTAINER ID]为**d**查询到的容器ID。
  f. 编译torch_npu。
     i. 创建软链接。
        ```
        cd /usr/local/bin/
        ln -s /opt/_internal/cpython-3.7.17/bin/pip3.7 pip3.7
        ln -s /opt/_internal/cpython-3.8.17/bin/pip3.8 pip3.8
        ln -s /opt/_internal/cpython-3.9.17/bin/pip3.9 pip3.9
        ln -s python3.7 python3
        ```
     ii. 安装PyTorch 1.11.0以及依赖。
        ```
        pip3.7 install torch==1.11.0
        pip3.7 install pyyaml
        ```

 版权所有 © 华为技术有限公司

说明

若镜像内无法通过pip命令下载torch包，可使用wget命令下载对应whl包进行安装。

```
wget https://download.pytorch.org/whl/torch-1.11.0-cp37-cp37m-manylinux2014_aarch64.whl
```

安装whl包：

```
pip3.7 install torch-1.11.0-cp37-cp37m-manylinux2014_aarch64.whl
```

iii. 进入任意目录，下载torch_npu源码，此处以“/home/”为例。

```
cd /home/
git clone https://gitee.com/ascend/pytorch.git -b v1.11.0 --depth=1 v1.11.0
cd v1.11.0/
```

iv. 编译torch_npu。

```
bash ci/build.sh --python=3.7
```

g. 退出容器，在服务器取回编译完成的包并安装。

```
exit
docker cp [CONTAINER ID]:/home/v1.11.0/dist/torch_npu-1.11.0.post1-cp37-cp37m-linux_aarch64.whl ./
pip3 install torch_npu-1.11.0.post1-cp37-cp37m-linux_aarch64.whl
```

B.13 原生 PyTorch 编译过程报错 no module named yaml/ typing_extensions.

问题描述

PyTorch编译依赖yaml库和typing_extensions库，需要手动安装。

解决措施

安装依赖成功后，需要先执行**make clean**再执行**bash build.sh**进行编译，否则可能因缓存出现未知编译错误。

```
pip3 install pyyaml
pip3 install typing_extensions
```

B.14 PyTorch 运行遇到找不到 te 问题

问题描述

PyTorch运行过程中，提示缺少te依赖。

解决措施

在开发环境下：

```
cd /urs/local/Ascend/ascend-toolkit/latest/{arch}-linux/lib64  #{arch}为架构名称。

pip3 install --upgrade topi-0.4.0-py3-none-any.whl

pip3 install --upgrade te-0.4.0-py3-none-any.whl
```

在运行环境下：

```
cd /urs/local/Ascend/nnae/latest/{arch}-linux/lib64  #{arch}为架构名称。
```

 版权所有 © 华为技术有限公司

```
pip3 install --upgrade topi-0.4.0-py3-none-any.whl

pip3 install --upgrade te-0.4.0-py3-none-any.whl
```

B.15 PyTorch 编译过程提示 CMAKE 相关错误

问题描述

PyTorch编译时cmake依赖时提示找不到包、编译cmake报错版本过低，可使用安装脚本或源码编译安装。

解决措施

方法一：下载安装脚本安装cmake。（参考**cmake官网**）

X86_64环境脚本安装：cmake-3.12.0-Linux-x86_64.sh

aarch64环境脚本安装：cmake-3.12.0-Linux-aarch64.sh

步骤1 执行命令。

```
./cmake-3.12.0-Linux-{arch}.sh #{arch}为架构名称
```

步骤2 设置软连接。

```
ln -s /usr/local/cmake/bin/cmake /usr/bin/cmake
```

步骤3 执行如下命令验证是否安装成功。

```
cmake --version
```

如显示“cmake version 3.12.0”则表示安装成功。

----结束

方法二：使用源码编译安装。

步骤1 获取cmake软件包。

```
wget https://cmake.org/files/v3.12/cmake-3.12.0.tar.gz --no-check-certificate
```

步骤2 解压并进入软件包目录。

```
tar -xf cmake-3.12.0.tar.gz
cd cmake-3.12.0/
```

步骤3 执行配置、编译和安装命令。

```
./configure --prefix=/usr/local/cmake
make && make install
```

步骤4 设置软连接。

```
ln -s /usr/local/cmake/bin/cmake /usr/bin/cmake
```

步骤5 执行如下命令验证是否安装成功。

```
cmake --version
```

如显示“cmake version 3.12.0”则表示安装成功。

----结束

 版权所有 © 华为技术有限公司

B.16 PyTorch 运行提示找不到 libblas.so

问题描述

环境缺少openblas库，需要安装openblas库。

解决措施

在CentOS、EulerOS环境下：

```
yum -y install openblas
```

在Ubuntu环境下：

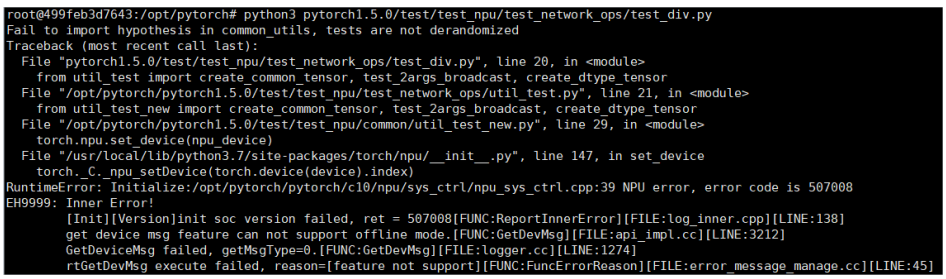
```
apt install libopenblas-dev
```

B.17 PyTorch 在容器中运行未挂载 device 问题

问题描述

在容器中运行脚本出现NPU相关ERROR。由于启动容器实例时，未挂载device参数，导致无法正常启动实例。

图 B-6 错误提示



解决措施

请用户参考以下命令，重启容器。

```
docker run -it --ipc=host \
--device=/dev/davinci*X* \
--device=/dev/davinci_manager \
--device=/dev/devmm_svm \
--device=/dev/hisi_hdc \
-v /usr/local/Ascend/driver \
-v /usr/local/dcmi \
-v /usr/local/bin/npu-smi \
${镜像名称}:{tag} \
/bin/bash
```

参数说明：

- /dev/davinci*X*：NPU设备，*X*是芯片物理ID号，例如davinci0。
- /dev/davinci_manager：管理设备。

 版权所有 © 华为技术有限公司

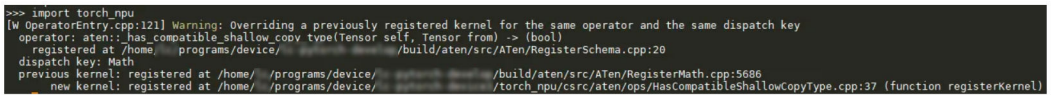
- /dev/devmm_svm：管理设备。
- /dev/hisi_hdc：管理设备。
- /usr/local/Ascend/driver：驱动目录。
- /usr/local/dcmi：DCMI目录。
- /usr/local/bin/npu-smi：npu-smi工具。
- ${镜像名称}:{tag}：镜像名称与版本号。

B.18 PyTorch 运行 import torch_npu 显示 _has_compatible_shallow_copy_type 重复注册 warning 问题

问题描述

warning如下图所示，由Tensor.set_data浅拷贝操作触发。主要原因是PyTorch插件化并解耦后，“_has_compatible_shallow_copy_type”缺乏对NPU Tensor的浅拷贝判断支持，因此需要重新注册“_has_compatible_shallow_copy_type”。

图 B-7 错误提示



解决措施

该warning不影响模型的精度和性能，可以忽略。待NPU设备号合入社区或者后续PyTorch版本“_has_compatible_shallow_copy_type”注册方式发生变动，该warning会被解决。

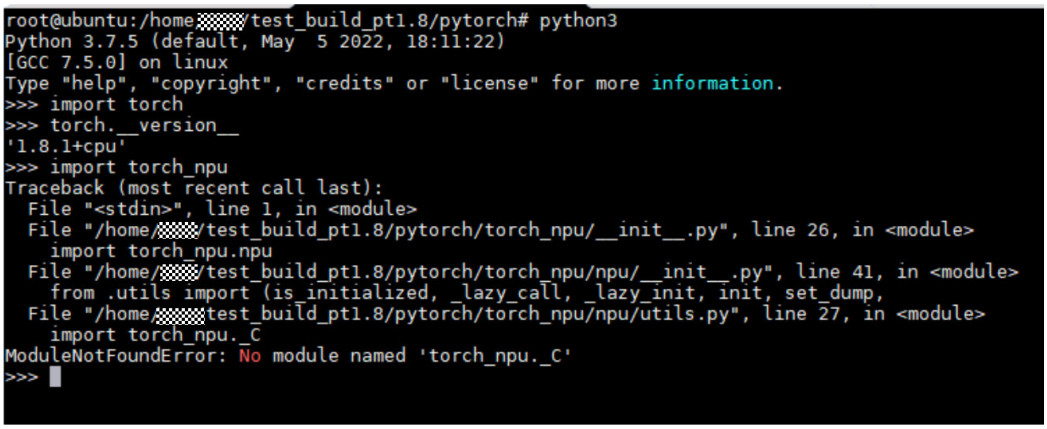
B.19 PyTorch 运行 import torch_npu 显示 No module named 'torch_npu._C'

问题描述

在编译torch_npu的目录进入Python引用torch_npu报错问题。

 版权所有 © 华为技术有限公司

图 B-8 报错提示



解决措施

请切换至其他目录验证torch_npu的引入。在编译目录执行验证会在当前目录下查找torch_npu，故会报错。

B.20 PyTorch 编译时出现 Breakpad error: field 'regs' has incomplete type 'google_breakpad::user_regs_struct'报错

问题描述

PyTorch编译时出现Breakpad error: field 'regs' has incomplete type 'google_breakpad::user_regs_struct'报错。

解决措施

编译原生PyTorch时，未配置相关环境变量，导致编译不成功。

步骤1 执行命令配置环境变量。

```
export BUILD_BREAKPAD=0
```

步骤2 执行命令清除当前编译内容。

```
make clean
```

步骤3 重新编译。

----结束

 版权所有 © 华为技术有限公司

C 附录

C.1 参数说明

软件包支持根据命令行完成一键式安装，各个命令之间可以配合使用，用户根据安装需要选择对应参数完成安装，所有参数都是可选参数。

安装命令格式： **./*软件包名*.run** [options]

详细参数请参见**表C-1**。

须知

如果通过**./*软件包名*.run --help**命令查询出的参数未解释在如下表格，则说明该参数预留或适用于其他芯片版本，用户无需关注。

表 C-1 安装包支持的参数说明

参数	说明
--help | -h	查询帮助信息。
--version	查询版本信息。
--info	查询软件包构建信息。
--list	查询软件包文件列表。
--check	检查软件包的一致性和完整性。
--quiet	静默安装，跳过交互式信息。
--nox11	不使用x11模式运行。
--noexec	解压软件包到当前目录，但不执行安装脚本。配套--extract=<path>使用，格式为： --noexec --extract=<path>。

 版权所有 © 华为技术有限公司

参数	说明
--extract=<path>	解压软件包中文件到指定目录。
--tar arg1 [arg2 ...]	对软件包执行tar命令，使用tar后面的参数作为命令的参数。例如执行**--tar xvf**命令，解压run安装包的内容到当前目录。
--install	安装软件包。后面可以指定安装路径--install-path=<path>，也可以不指定安装路径，直接安装到默认路径下。
--install-for-all	安装或升级时，允许其他用户具有安装群组的权限。 当安装或者升级携带该参数时，软件包中创建的目录及文件，其他用户权限=安装群组权限。 该参数需要与--install、--devel、--upgrade等其中一个参数配合使用，例如**./***软件包名***.run** --install --install-for-all **说明** 使用该参数将会存在安全风险：其他所有用户都有权限访问安装目录，请谨慎使用。
--install-path=<path>	指定安装路径，当环境上存在全局配置文件“ascend_cann_install.info”时，支持使用该参数，但指定路径必须与全局配置文件中保存的安装路径保持一致。如用户想更换安装路径，需先卸载原路径下的CANN软件包并确保全局配置文件“ascend_cann_install.info”已被删除。 可在如下目录查看是否存在该文件： ● root用户：/etc/Ascend ● 非root用户：*${HOME}*/Ascend 若不指定，将安装到默认路径下： ● 若使用root用户安装，默认安装路径为：/usr/local/Ascend。 ● 若使用非root用户安装，则默认安装路径为：*${HOME}*/Ascend。 若通过该参数指定了安装目录，运行用户需要对指定的安装路径有可读写权限。
--force	强制安装、升级命令。当安装、升级检测不通过导致安装或升级失败时，则会提示用户使用该参数跳过安装检测，使用后方可安装或升级成功。 该参数需要配合“--install”或“--upgrade”一起使用。
--full	仅软件包toolkit支持使用该参数。 支持在无驱动场景下实现全量安装。
--uninstall	卸载已安装的软件。
--upgrade	升级已安装的软件。

 版权所有 © 华为技术有限公司

参数	说明
--feature-list=<feature>	仅软件包toolkit、nnae支持使用该参数。 指定升级特性。仅支持升级使用，参数仅支持输入“Acclibs”（独立升级算子包），如“--feature-list=Acclibs”。 该参数需要配合“--upgrade”一起使用。 **说明** 推荐用户全量升级。如果版本配套表中有算子包独立升级的兼容性列表，则参考该兼容性列表，否则推荐用户全量升级。
--devel	按照开发模式安装软件包，即只安装开发环境需安装的文件。
--chip=<chip_type>	指定芯片型号，以便在安装过程中选择匹配的软件（如AICPU算子包）。芯片型号参数chip_type可选范围如下（以软件包toolkit为例）： ● Ascend310，表示芯片型号为Ascend310 PCIe芯片。 ● Ascend310P，表示芯片型号为Ascend310P PCIe芯片。 ● Ascend910，表示芯片型号为Ascend910 PCIe芯片。 ● Ascend310-minirc，表示芯片型号为Ascend310 SoC芯片（与Ascend310相同，但以RC模式启动，作为主控CPU） ● Ascend，表示芯片型号为Ascend310B SoC芯片和Ascend910B PCIe芯片。 **须知** 该参数仅对软件包toolkit、nnrt、nnae有效。未指定此参数时，默认安装软件包内的全部AICPU算子包。
--alternative	仅软件包toolkit支持使用该参数。 查询可选安装特性列表。用于可选安装场景，在安装前查询软件包支持的可选安装特性列表，以便拷贝到--whitelist输入参数中。 toolkit可选安装特性如下： ● atc：用于支持模型编译、模型转换。 ● devtools：主要为调测、仿真工具。选择安装该特性时请配套安装nnae。 ● nnrt：用于支持离线推理场景。 ● nnae：用于支持离线推理、在线推理、训练以及IR构图场景。
--whitelist=<feature_type>	仅软件包toolkit支持使用该参数。 可选安装白名单，用于在执行安装（--install）时指定部分可选安装特性。不支持在升级场景下使用。 该参数需要配合“--install”一起使用。

 版权所有 © 华为技术有限公司

C.2 相关信息记录路径

CANN软件包在安装过程中会生成相关配置、日志信息等，文件存放路径如表C-2所示。

其中*{arch}*表示CPU架构，*${HOME}*为当前用户目录。

表 C-2 信息记录路径

信息说明	路径
软件包安装详细日志路径	以软件包toolkit为例（以软件包默认安装路径进行说明，请根据实际替换）： ● root用户：“/var/log/ascend_seclog/ascend_toolkit_install.log” ● 非root用户：“*${HOME}*/var/log/ascend_seclog/ascend_toolkit_install.log”
安装后软件包版本、CPU架构和安装路径等信息的记录路径	以软件包toolkit（nnrt目录结构同toolkit）和nnae（tfplugin目录结构同nnae）为例（以软件包默认安装路径进行说明，请根据实际替换）： ● root用户： toolkit：“/usr/local/Ascend/ascend-toolkit/latest/*{arch}*-linux/ascend_toolkit_install.info” nnae：“/usr/local/Ascend/nnae/latest/ascend_nnae_install.info” ● 非root用户： toolkit：“*${HOME}*/Ascend/ascend-toolkit/latest/*{arch}*-linux/ascend_toolkit_install.info” nnae：“*${HOME}*/Ascend/nnae/latest/ascend_nnae_install.info”
软件包安装路径的记录路径	● root用户：“/etc/Ascend/ascend_cann_install.info” ● 非root用户：“*${HOME}*/Ascend/ascend_cann_install.info”

 版权所有 © 华为技术有限公司

信息说明	路径
软件包安装时指定的安装参数（如--install-for-all、--whitelist等）的记录路径	以软件包toolkit（nnrt目录结构同toolkit）和nnae（tfplugin目录结构同nnae）为例（以软件包默认安装路径进行说明，请根据实际替换）： ● root用户： toolkit：“/usr/local/Ascend/ascend-toolkit/latest/*{arch}*-linux/install.conf” nnae：“/usr/local/Ascend/nnae/latest/install.conf” ● 非root用户： toolkit：“*${HOME}*/Ascend/ascend-toolkit/latest/*{arch}*-linux/install.conf” nnae：“*${HOME}*/Ascend/nnae/latest/install.conf”

C.3 AICPU Kernel 加载说明

表 C-3 AICPU Kernel 加载说明

场景	说明	约束
物理机	AICPU Kernel安装后跟随业务进程初始化，自动加载到Device。	● 不支持一个昇腾AI处理器同时被多个用户使用时，各用户加载不同版本AICPU Kernel。 ● 不支持在用户运行环境内，指定某个昇腾AI处理器加载特定版本的AICPU Kernel。
容器	● 每个容器可以加载一份AICPU Kernel。 ● 单芯片多容器场景下，支持各容器加载不同版本的AICPU Kernel。 ● 容器内多用户场景下，支持其中一个用户以install-for-all方式安装AICPU Kernel，其他用户通过设置相应环境变量使用同份AICPU Kernel。	● 不支持容器内多用户场景下，各用户安装不同版本的AICPU Kernel。 ● 不支持容器内指定某个昇腾AI处理器加载特定版本的AICPU Kernel（AICPU Kernel容器内全局生效）。

C.4 口令复杂度要求

口令至少满足如下要求：

1. 口令长度至少8个字符。

 版权所有 © 华为技术有限公司

2. 口令必须包含如下至少两种字符的组合：
   - 一个小写字母
   - 一个大写字母
   - 一个数字
   - 一个特殊字符：`~!@#$%^&*()-_=+\|[{}];:'",<.>/?和空格
3. 口令不能和账号一样。

 版权所有 © 华为技术有限公司