CANN
V100R020C10

软件安装指南 (开发&运行场景, 通过命令行方式)

文档版本 01

发布日期 2021-02-04

华为技术有限公司

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默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 安装须知

- 本文档提供命令行方式安装开发或运行环境。
  - **开发环境：主要用于代码开发、编译、调测等开发活动。**
    - **在昇腾AI设备上安装开发环境，同时可以作为运行环境，运行应用程序或进行模型训练。**
    - **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。**
  - **运行环境：在昇腾AI设备上运行用户开发的应用程序，或进行模型训练。**
- 对于如下昇腾AI设备，支持安装开发或运行环境。
  - 已配置Atlas 200 AI加速模块（EP场景）的服务器
  - 已配置Atlas 300I 推理卡的服务器
  - 已配置Atlas 300T 训练卡的服务器
  - Atlas 500 Pro 智能边缘服务器
  - Atlas 800 推理服务器
  - Atlas 800 训练服务器
  - Atlas 900 AI集群
- 对于Atlas 200 AI加速模块（RC场景）、Atlas 500 智能小站，仅能用于运行环境，需要另外准备一台服务器用于安装纯开发环境，详情参考：
  - **2.1.2 Atlas 200 AI加速模块（RC场景）**
  - **2.1.4 Atlas 500 智能小站（型号 3000）**
- 对于非昇腾AI设备，支持安装纯开发环境，详情请参考**3 安装开发环境（推理）**。

  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。

2 准备硬件环境

2.1 推理

2.1.1 Atlas 200 AI 加速模块（EP 场景）

适配操作系统

Atlas 200 AI加速模块（型号 3000）作为PCIe从设备（EP模式）时，适配操作系统如**表2-1**所示。

表 2-1 适配操作系统

操作系统	版本	获取方式
Ubuntu	操作系统版本：Ubuntu 18.04.1 处理器架构：x86_64	从Ubuntu官网获取。 请从**http://old-releases.ubuntu.com/releases/18.04.1/**网站下载如下推荐的版本：**ubuntu-18.04.1-server-amd64.iso**。

安装驱动和固件

- 请参考**表2-2**下载驱动和固件包。

表 2-2 Atlas 200 AI 加速模块（型号 3000）软件包

软件包类型	软件包名称	host操作系统版本	说明	获取方式
驱动包	A200-3000-npu-driver_*{version}*_ubuntu18.04-x86_64.run	Ubuntu 18.04.1	驱动安装包	**获取链接**
固件包	A200-3000-npu-firmware_*{version}*.run		固件安装包	

其中*{version}*表示软件版本号。

- 安装驱动和固件，详情请参见《**Atlas 200 AI加速模块 1.0.7及以上 软件安装与维护指南（EP场景，型号 3000）**》。

2.1.2 Atlas 200 AI 加速模块（RC 场景）

Atlas 200 AI加速模块（RC场景）仅能用于运行环境，需要另外准备一台X86 CPU的服务器用作搭建开发环境，用户需要提前安装好Ubuntu 18.04操作系统。

开发环境安装流程如下：

1. **准备开发套件包**（请同时获取两种架构的开发套件包Ascend-cann-toolkit）
2. **3.3 安装OS依赖**
3. **3.4 安装开发套件包**
4. **3.5 配置交叉编译环境**

Atlas 200 AI加速模块（RC场景）运行环境安装请参考《**Atlas 200 AI加速模块 1.0.8 软件安装与维护指南（RC场景，型号 3000）**》。

2.1.3 Atlas 300I 推理卡（型号 3010）

适配操作系统

Atlas 300I 推理卡（型号 3010）适配操作系统如**表2-3**所示。

表 2-3 适配操作系统

操作系统	版本	获取方式
Ubuntu	操作系统版本：Ubuntu 18.04.1 处理器架构：x86_64	从Ubuntu官网获取。 请从**http://old-releases.ubuntu.com/releases/18.04.1/**网站下载如下推荐的版本：**ubuntu-18.04.1-server-amd64.iso**。

操作系统	版本	获取方式
CentOS	操作系统版本：CentOS 7.6 处理器架构：x86_64	从CentOS官网获取。 请从**http://vault.centos.org/7.6.1810/isos/x86_64/**网站下载对应版本软件进行安装，可以下载如下推荐的版本：**CentOS-7-x86_64-DVD-1810.iso**。

安装驱动、固件和升级 MCU

- 请参考**表2-4**下载驱动和固件包。

  表 2-4 Atlas 300I 推理卡（型号 3010）软件包

软件包类型	软件包名称	说明	获取方式
驱动包	A300-3010-npu-driver_*{version}*_*{os}*-x86_64.run	• OS版本若为CentOS7.6，根据现场实际规划或部署的gcc版本取包，可以在操作系统上使用**gcc --version**命令查询gcc版本。 • npu-smi工具包集成在驱动包内，安装驱动过程中会自动安装npu-smi。	**获取链接**
固件包	A300-3010-npu-firmware_*{version}*.run	昇腾固件安装包	
MCU包	• A300-3010-mcu_*{version}*.bin • A300-3010-mcu_*{version}*.hpm	MCU是Atlas 300 推理卡的带外管理模块，具备单板监控、故障上报等功能。 • 通过npu-smi工具升级请获取*.bin的MCU包。 • 通过iBMC升级请获取*.hpm的MCU包。	

  *其中{version}*表示软件版本号，*{os}*表示操作系统版本。

- 安装驱动和固件，详情请参见《**Atlas 300I 推理卡 用户指南（型号 3010）**》的“安装与维护”章节。

  物理机部署：在服务器上安装驱动和固件。

  虚拟机部署：在宿主机上安装驱动和固件，在虚拟机上安装驱动。

  升级MCU请参见《**Atlas 300I 推理卡 1.0.7 升级指导书（型号 3000, 3010）**》的“升级组件>升级MCU”章节。

2.1.4 Atlas 500 智能小站（型号 3000）

Atlas 500 智能小站出厂预安装Euler OS、驱动&固件，仅能用于运行环境。需要另外准备一台X86 CPU的服务器用作搭建开发环境，用户需要提前安装好Ubuntu 18.04操作系统。

开发环境安装流程如下：

1. **准备开发套件包**（请同时获取两种架构的开发套件包Ascend-cann-toolkit）
2. **3.3 安装OS依赖**
3. **3.4 安装开发套件包**
4. **3.5 配置交叉编译环境**

Atlas 500 智能小站的初始配置、容器镜像制作请参考**《Atlas 500 智能小站 用户指南（型号 3000, 3010）》**。

2.1.5 Atlas 500 Pro 智能边缘服务器（型号 3000）

安装与配置

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）》**，安装操作系统完成后，配置业务网口IP地址，请参见**C.1 配置网卡IP地址**。

Atlas 500 Pro 智能边缘服务器（型号 3000）适配操作系统如**表2-5**所示。

表 2-5 适配操作系统

操作系统	版本	获取方式
Ubuntu	操作系统版本：Ubuntu 18.04.1 操作系统架构：aarch64	从Ubuntu官网获取。 请从**http://old-releases.ubuntu.com/releases/18.04.1/**网站下载如下推荐的版本：**ubuntu-18.04.1-server-arm64.iso。**
EulerOS	操作系统版本：EulerOS 2.8 操作系统架构：aarch64	-
CentOS	操作系统版本：CentOS 7.6 操作系统架构：aarch64	从CentOS官网获取。 请从**https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/**网站下载如下推荐的版本：**CentOS-7-aarch64-Everything-1810.iso。**

安装驱动、固件和升级 MCU

Atlas 500 Pro 智能边缘服务器（型号 3000）支持配置Atlas 300I 推理卡（型号 3000）。

- 请参考**表2-6**下载驱动和固件包。

  表 2-6 Atlas 300I 推理卡（型号 3000）软件包

软件包类型	软件包名称	说明	获取方式
驱动包	A300-3000-npu-driver_*{version}*_*{os}*-aarch64.run	• OS版本若为CentOS7.6，根据现场实际规划或部署的gcc版本取包，可以在操作系统上使用**gcc --version**命令查询gcc版本。 • npu-smi工具包集成在驱动包内，安装驱动过程中会自动安装npu-smi。	**获取链接**
固件包	A300-3000-npu-firmware_*{version}*.run	昇腾固件安装包	
MCU包	• A300-3000-mcu_*{version}*.bin • A300-3000-mcu_*{version}*.hpm	MCU是Atlas 300 推理卡的带外管理模块，具备单板监控、故障上报等功能。 • 通过npu-smi工具升级请获取*.bin的MCU包。 • 通过iBMC升级请获取*.hpm的MCU包。	

  *其中{version}*表示软件版本号，*{os}*表示操作系统版本。

- 安装驱动和固件，详情请参见《**Atlas 300I 推理卡 用户指南（型号 3000）**》的“安装与维护”章节。

  物理机部署：在服务器上安装驱动和固件。

  虚拟机部署：在宿主机上安装驱动和固件，在虚拟机上安装驱动。

  升级MCU请参见《**Atlas 300I 推理卡 1.0.7 升级指导书（型号 3000, 3010）**》的“升级组件>升级MCU”章节。

2.1.6 Atlas 800 推理服务器（型号 3000）

安装与配置

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）**》，安装操作系统完成后，配置业务网口IP地址，请参见**C.1 配置网卡IP地址**。

Atlas 800 推理服务器（型号 3000）适配操作系统如**表2-7**所示。

表 2-7 适配操作系统

操作系统	版本	获取方式
Ubuntu	操作系统版本：Ubuntu 18.04.1 处理器架构：aarch64	从Ubuntu官网获取。 请从**http://old-releases.ubuntu.com/releases/18.04.1/**网站下载如下推荐的版本：**ubuntu-18.04.1-server-arm64.iso。**
CentOS	操作系统版本：CentOS 7.6 处理器架构：aarch64	从CentOS官网获取。 请从**https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/**网站下载如下推荐的版本：**CentOS-7-aarch64-Everything-1810.iso。**
EulerOS	操作系统版本：EulerOS 2.8 处理器架构：aarch64	-

安装驱动、固件和升级 MCU

Atlas 800 推理服务器（型号 3000）支持配置Atlas 300I 推理卡（型号 3000）。

- 请参考**表2-8**下载驱动和固件包。

表 2-8 Atlas 300I 推理卡（型号 3000）软件包

软件包类型	软件包名称	说明	获取方式
驱动包	A300-3000-npu-driver_*{version}*_*{os}*-aarch64.run	• OS版本若为CentOS7.6，根据现场实际规划或部署的gcc版本取包，可以在操作系统上使用**gcc --version**命令查询gcc版本。 • npu-smi工具包集成在驱动包内，安装驱动过程中会自动安装npu-smi。	**获取链接**
固件包	A300-3000-npu-firmware_*{version}*.run	昇腾固件安装包	

软件包类型	软件包名称	说明	获取方式
MCU包	• A300-3000-mcu_*{version}*.bin • A300-3000-mcu_*{version}*.hpm	MCU是Atlas 300 推理卡的带外管理模块，具备单板监控、故障上报等功能。 • 通过npu-smi工具升级请获取*.bin的MCU包。 • 通过iBMC升级请获取*.hpm的MCU包。	

*其中{version}*表示软件版本号，*{os}*表示操作系统版本。

- 安装驱动和固件，详情请参见《**Atlas 300I 推理卡 用户指南（型号 3000）**》的“安装与维护”章节。

  物理机部署：在服务器上安装驱动和固件。

  虚拟机部署：在宿主机上安装驱动和固件，在虚拟机上安装驱动。

  升级MCU请参见《**Atlas 300I 推理卡 1.0.7 升级指导书（型号 3000, 3010）**》的“升级组件>升级MCU”章节。

2.1.7 Atlas 800 推理服务器（型号 3010）

安装与配置

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）**》，安装操作系统完成后，配置业务网口IP地址，请参见**C.1 配置网卡IP地址**。

Atlas 800 推理服务器（型号 3010）适配操作系统如**表2-9**所示。

表 2-9 适配操作系统

操作系统	版本	获取方式
Ubuntu	操作系统版本：Ubuntu 18.04.1 操作系统架构：x86_64	从Ubuntu官网获取。 请从**http://old-releases.ubuntu.com/releases/18.04.1/**网站下载如下推荐的版本：**ubuntu-18.04.1-server-amd64.iso**。
CentOS	操作系统版本：CentOS 7.6 操作系统架构：x86_64	从CentOS官网获取。 请从**http://vault.centos.org/7.6.1810/isos/x86_64/**网站下载对应版本软件进行安装，可以下载如下推荐的版本：**CentOS-7-x86_64-DVD-1810.iso**。

安装驱动、固件和升级 MCU

Atlas 800 推理服务器（型号 3010）支持配置Atlas 300I 推理卡（型号 3010）。

- 请参考**表2-10**下载驱动和固件包。

  表 2-10 Atlas 300I 推理卡（型号 3010）软件包

软件包类型	软件包名称	说明	获取方式
驱动包	A300-3010-npu-driver_*{version}*_*{os}*-x86_64.run	• OS版本若为CentOS7.6，根据现场实际规划或部署的gcc版本取包，可以在操作系统上使用**gcc --version**命令查询gcc版本。 • npu-smi工具包集成在驱动包内，安装驱动过程中会自动安装npu-smi。	**获取链接**
固件包	A300-3010-npu-firmware_*{version}*.run	昇腾固件安装包	
MCU包	• A300-3010-mcu_*{version}*.bin • A300-3010-mcu_*{version}*.hpm	MCU是Atlas 300 推理卡的带外管理模块，具备单板监控、故障上报等功能。 • 通过npu-smi工具升级请获取*.bin的MCU包。 • 通过iBMC升级请获取*.hpm的MCU包。	

  *其中{version}*表示软件版本号，*{os}*表示操作系统版本。

- 安装驱动和固件，详情请参见《**Atlas 300I 推理卡 用户指南（型号 3010）**》的“安装与维护”章节。

  物理机部署：在服务器上安装驱动和固件。

  虚拟机部署：在宿主机上安装驱动和固件，在虚拟机上安装驱动。

  升级MCU请参见《**Atlas 300I 推理卡 1.0.7 升级指导书（型号 3000, 3010）**》的“升级组件>升级MCU”章节。

2.2 训练

2.2.1 Atlas 300T 训练卡（型号 9000）

适配操作系统

已安装Atlas 300T 训练卡（型号 9000）的训练服务器适配的操作系统如表2-11所示。

表 2-11 适配的操作系统

操作系统	版本	获取方式
Ubuntu	操作系统版本：Ubuntu 18.04 处理器架构：x86_64	从Ubuntu官网获取。 请从**http://old-releases.ubuntu.com/releases/18.04.1/**网站下载如下推荐的版本：**ubuntu-18.04.1-server-amd64.iso**。
CentOS	操作系统版本：CentOS 7.6 处理器架构：x86_64	从CentOS官网获取。 请从**http://vault.centos.org/7.6.1810/isos/x86_64/**网站下载对应版本软件进行安装，可以下载如下推荐的版本：**CentOS-7-x86_64-DVD-1810.iso**。

安装驱动、固件和升级 MCU

- 请参考表2-12下载驱动和固件包。

表 2-12 Atlas 300T 训练卡（型号 9000）软件包

名称	软件包	说明	获取方式
昇腾芯片驱动安装包	A300t-9000-npu-driver_*{version}*_*{os}*-*{arch}*.run	OS版本若为CentOS7.6，请获取gcc7.3.0的软件包。	**获取链接**
昇腾芯片固件安装包	A300t-9000-npu-firmware_*{version}*.run	-	

名称	软件包	说明	获取方式
MCU固件包	• A300t-9000-mcu_*{version}*.bin • A300t-9000-mcu_*{version}*.hpm	MCU是Atlas 300T训练卡的带外管理模块，具备单板监控、故障上报等功能。 • 通过npu-smi工具升级请获取*.bin的MCU包。 • 通过iBMC升级请获取*.hpm的MCU包。	

*其中{version}*表示软件版本号，*{os}*表示操作系统版本，*{arch}*为CPU架构。

- 安装驱动和固件，详情请参见《**Atlas 300T 训练卡 驱动和固件安装升级指南 (型号9000)**》的“安装与维护”章节。

  物理机部署：在服务器上安装驱动和固件。

  虚拟机部署：在宿主机上安装驱动和固件，在虚拟机上安装驱动。

  升级MCU请参见《**Atlas 300T 训练卡 驱动和固件安装升级指南 (型号9000)**》的“升级>升级组件>升级MCU”章节。

2.2.2 Atlas 800 训练服务器（型号 9000）

安装与配置

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)**》。

Atlas 800 训练服务器（型号 9000）适配操作系统如**表2-13**所示。

表 2-13 适配的操作系统

操作系统	版本	获取方式
Ubuntu	操作系统版本：Ubuntu 18.04.1 处理器架构：aarch64	从Ubuntu官网获取。 请从**http://old-releases.ubuntu.com/releases/18.04.1/**网站下载如下推荐的版本：**ubuntu-18.04.1-server-arm64.iso**。

操作系统	版本	获取方式
CentOS	操作系统版本：CentOS 7.6 处理器架构：aarch64	从CentOS官网获取。 请从**https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/**网站下载如下推荐的版本： **CentOS-7-aarch64-Everything-1810.iso**。
EulerOS	操作系统版本：EulerOS 2.8 处理器架构：aarch64	-

安装驱动和固件

- 请参考**表2-14**下载驱动和固件包。

  表 2-14 Atlas 800 训练服务器（型号 9000）软件包

名称	软件包	说明	获取方式
昇腾芯片驱动安装包	A800-9000-npu-driver_*{version}*_*{os}*-aarch64.run	OS版本若为CentOS7.6，请获取gcc7.3.0的软件包。	**获取链接**
昇腾芯片固件安装包	A800-9000-npu-firmware_*{version}*.run	-	

  *其中{version}*表示软件版本号，*{os}*表示操作系统版本。

- 安装驱动和固件，详情请参见《**Atlas 800 训练服务器 驱动和固件安装升级指南 (型号9000)**》。

  物理机部署：在服务器上安装驱动和固件。

  虚拟机部署：在宿主机上安装驱动和固件，在虚拟机上安装驱动。

2.2.3 Atlas 800 训练服务器（型号 9010）

安装与配置

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)**》。

Atlas 800 训练服务器（型号 9010）适配操作系统如**表2-15**所示。

表 2-15 适配的操作系统

操作系统	版本	获取方式
Ubuntu	操作系统版本： Ubuntu 18.04.1 处理器架构：x86_64	从Ubuntu官网获取。 请从**http://old-releases.ubuntu.com/releases/18.04.1/**网站下载如下推荐的版本：**ubuntu-18.04.1-server-amd64.iso**。
CentOS	操作系统版本： CentOS 7.6 处理器架构：x86_64	从CentOS官网获取。 请从**http://vault.centos.org/7.6.1810/isos/x86_64/**网站下载对应版本软件进行安装，可以下载如下推荐的版本：**CentOS-7-x86_64-DVD-1810.iso**。

安装驱动和固件

- 请参考**表2-16**下载驱动和固件包。

  表 2-16 Atlas 800 训练服务器（型号 9010）软件包

名称	软件包	说明	获取方式
昇腾芯片驱动安装包	A800-9010-npu-driver_*{version}*_*{os}*-x86_64.run	OS版本若为CentOS7.6，请获取gcc7.3.0的软件包。	**获取链接**
昇腾芯片固件安装包	A800-9010-npu-firmware_*{version}*.run	-	

  *其中{version}*表示软件版本号，*{os}*表示操作系统版本。

- 安装驱动和固件，详情请参见《**Atlas 800 训练服务器 驱动和固件安装升级指南 (型号9010)**》。

  物理机部署：在服务器上安装驱动和固件。

  虚拟机部署：在宿主机上安装驱动和固件，在虚拟机上安装驱动。

2.2.4 Atlas 900 AI 集群（型号 9000）

安装与配置

Atlas 900 AI集群安装上架、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, 直流)》**
- **《Atlas 900 PoD 用户指南 (型号9000, 交流)》**
- **《Atlas 900 计算节点 用户指南 (风冷)》**

- **《Atlas 900 计算节点 用户指南 (液冷)》**

Atlas 900 AI集群适配操作系统如**表2-17**所示。

表 2-17 适配的操作系统

操作系统	版本	获取方式
Ubuntu	操作系统版本：Ubuntu 18.04.1 处理器架构：aarch64	从Ubuntu官网获取。 请从**http://old-releases.ubuntu.com/releases/18.04.1/**网站下载如下推荐的版本：**ubuntu-18.04.1-server-arm64.iso**。
CentOS	操作系统版本：CentOS 7.6 处理器架构：aarch64	从CentOS官网获取。 请从**https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/**网站下载如下推荐的版本：**CentOS-7-aarch64-Everything-1810.iso**。
EulerOS	操作系统版本：EulerOS 2.8 处理器架构：aarch64	-

安装驱动和固件

- 请参考**表2-18**下载驱动和固件包。

  表 2-18 Atlas 900 AI 集群（型号 9000）软件包

名称	软件包	说明	获取方式
昇腾芯片驱动安装包	A900-9000-npu-driver_*{version}*_*{os}*-aarch64.run	OS版本若为CentOS7.6，请获取gcc7.3.0的软件包。	**获取链接**
昇腾芯片固件安装包	A900-9000-npu-firmware_*{version}*.run	-	

  *其中{version}*表示软件版本号，*{os}*表示操作系统版本。

- 安装驱动和固件，详情请参见**《Atlas 900 计算节点 驱动和固件安装升级指南》**。

  物理机部署：在服务器上安装驱动和固件。

  虚拟机部署：在宿主机上安装驱动和固件，在虚拟机上安装驱动。

3 安装开发环境（推理）

3.1 准备软件包

下载软件包

软件安装前，请获取软件包和数字签名文件。

表 3-1 开发套件包

软件包类型	软件包名称	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、开发辅助工具如ATC模型转换工具。 • 请根据CPU架构（x86_64、aarch64）获取对应的软件包。 • 对于运行环境为aarch64而开发环境为x86_64的场景，需同时获取两种架构的开发套件包。	**获取链接**

说明

*{version}*表示软件版本号，*{arch}*表示CPU架构。

检查软件包的完整性

说明

若您获取的是社区版本，可不进行此操作。

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。

3.2 准备安装及运行用户

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。

创建安装及运行用户

- 安装用户：实际安装软件包的用户。
  - 若使用root用户安装，因要求使用非root用户运行，所以安装前需要先准备运行用户。
  - 若使用非root用户安装，则安装及运行用户必须相同。
- 运行用户：实际运行推理业务或执行训练的用户。
- **使用root用户安装**
  - （推荐）如果创建的运行用户是HwHiAiUser，其为CANN软件包的默认运行用户，在安装时无需指定该运行用户。
  - 如果创建的运行用户是非HwHiAiUser，安装CANN软件包时需要指定该运行用户。
- **使用非root用户安装**
  - 已有非root用户，则无需再次创建。

- 新建非root用户，请参见如下方法创建。

须知

- CANN软件包如果使用非root用户安装，用户所属的属组必须和Driver运行用户所属属组相同；如果不同，请用户自行添加到Driver运行用户属组。
- 创建的运行用户不建议为root用户属组，原因是：权限控制可能存在安全风险。

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**

   以创建运行用户HwHiAiUser为例：

   ```
   groupadd HwHiAiUser
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser -s /bin/bash
   ```

2. 设置非root用户密码。
   passwd *username*

   示例如下：

   ```
   passwd HwHiAiUser
   ```

说明

- 创建完HwHiAiUser用户后， 请勿关闭该用户的登录认证功能。
- 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C.7 设置用户有效期。

3.3 安装 OS 依赖

3.3.1 CentOS aarch64 系统

环境要求

开发套件包、Python、开发辅助工具、应用程序编译需要安装以下软件或依赖。

表 3-2 依赖信息

类别	版本要求
Python	3.7.5
cmake	3.5.1+
protobuf	3.11.3+
numpy	1.13.3+
gcc	软件源默认安装gcc版本为4.8.5。
gcc-c++	

类别	版本要求
make unzip zlib-devel libffi-devel openssl-devel sqlite-devel blas-devel lapack-devel openblas-devel pciutils net-tools attrs psutil decorator scipy sympy cffi	无版本要求，安装的版本以操作系统自带的源为准。

配置安装用户权限（可选）

当用户使用非root用户安装时，需要操作该章节，否则请忽略。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo yum**权限，请以root用户执行如下操作。

1. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
2. 在该文件“root ALL=(ALL) ALL”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   请将“username”替换成实际安装用户。

   说明

   请确保“ /etc/sudoers”文件存在“ #includedir /etc/sudoers.d”，如果没有该信息，请手动添加至文件末尾。
3. 添加完成后，执行**:wq!**保存文件。
4. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```


配置源

开发套件包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/CentOS-Base.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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

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
rpm -qa |grep unzip
rpm -qa |grep zlib-devel
rpm -qa |grep libffi-devel
rpm -qa |grep openssl-devel
rpm -qa |grep sqlite-devel
rpm -qa |grep blas-devel
rpm -qa |grep lapack-devel
rpm -qa |grep openblas-devel
rpm -qa |grep pciutils
rpm -qa |grep net-tools
```

若分别返回类似如下信息（版本号以现场操作系统实际显示为准）则说明已经安装，进入下一步。

```
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 4.1
cmake version 3.5.1
unzip-6.0-21.el7.aarch64
zlib-devel-1.2.7-18.el7.aarch64
libffi-devel-3.0.13-18.el7.aarch64
openssl-devel-1.0.2k-19.el7.aarch64
sqlite-devel-3.7.17-8.el7_7.1.aarch64
blas-devel-3.4.2-8.el7.aarch64
lapack-devel-3.4.2-8.el7.aarch64
openblas-devel-0.3.3-2.el7.aarch64
pciutils-3.5.1-3.el7.aarch64
net-tools-2.0-0.24.20131004git.el7.aarch64
```

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

⚠ 注意

- 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python3.7.5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 sqlite-devel blas-devel lapack-devel openblas-devel pciutils net-tools
```

上述步骤中，若安装openblas-devel时报“No package libopenblas available”的错误，需要安装企业版Linux的扩展包后重新安装。扩展包安装命令如下：

```
sudo yum install epel-release
```

如果通过上述方式安装的cmake版本低于3.5.1，则请参见**C.3 安装3.5.2版本cmake**解决。

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。

   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```

   当出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。

   ```
   cp: overwrite 'libpython3.7m.so.1.0'?y
   ```

   如果环境上没有/usr/lib64，则复制到/usr/lib目录：

   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```

   libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。

5. 执行如下命令设置软链接：

   ```
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```

   执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。

   ```
   sudo rm -rf  /usr/bin/python3.7.5
   sudo rm -rf  /usr/bin/pip3.7.5
   sudo rm -rf  /usr/bin/python3.7
   sudo rm -rf  /usr/bin/pip3.7
   ```

6. 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。


```
python3.7.5 --version
pip3.7.5  --version
```

步骤3 安装 Python3开发环境。

安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install attrs
pip3.7.5 install psutil
pip3.7.5 install decorator
pip3.7.5 install numpy==1.17.2
pip3.7.5 install protobuf==3.11.3
pip3.7.5 install scipy
pip3.7.5 install sympy
pip3.7.5 install cffi
```

- 如果安装numpy报错，请参考**D.4 pip3.7.5 install numpy报错**解决。
- 如果安装scipy报错，请参考**D.3 pip3.7.5 install scipy报错**解决。

----结束

3.3.2 Ubuntu aarch64 系统

环境要求

开发套件包、Python、开发辅助工具、应用程序编译需要安装以下软件或依赖。

表 3-3 依赖信息

类别	版本要求
Python	3.7.5
cmake	3.5.1+
protobuf	3.11.3+
g++	7.3.0及以上
gcc	7.3.0及以上

类别	版本要求
make zlib1g zlib1g-dev libsqlite3-dev openssl libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3 libopenblas-dev attrs psutil decorator numpy scipy sympy cffi	无版本要求，安装的版本以操作系统自带的源为准。

配置安装用户权限（可选）

当用户使用非root用户安装时，需要操作该章节，否则请忽略。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo apt-get**权限，请以root用户执行如下操作。

1. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
2. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5, /usr/bin/unzip
   ```
   请将“username”替换成实际安装用户。

   说明

   请确保“ /etc/sudoers”文件存在“ #includedir /etc/sudoers.d”，如果没有该信息，请手动添加至文件末尾。
3. 添加完成后，执行**:wq!**保存文件。

4. 执行以下命令取消“/etc/sudoers”文件的写权限：
```
chmod u-w /etc/sudoers
```

检查源

开发套件包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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

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
dpkg -l libopenblas-dev| grep libopenblas-dev| grep ii
```

若分别返回如下信息则说明已经安装，进入下一步。

```
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 amd64       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
libopenblas-dev:arm64 0.2.20+ds-4  arm64    Optimized BLAS (linear algebra) library (development files)
```

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

⚠ 注意

- 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python3.7.5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 libsqlite3-dev openssl libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3 libopenblas-dev
```

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   当出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。
   ```
   cp: overwrite 'libpython3.7m.so.1.0'?y
   ```
   如果环境上没有/usr/lib64，则复制到/usr/lib目录：
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```
   libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。
5. 执行如下命令设置软链接：
   ```
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。
   ```
   sudo rm -rf  /usr/bin/python3.7.5
   sudo rm -rf  /usr/bin/pip3.7.5
   ```



```
sudo rm -rf  /usr/bin/python3.7
sudo rm -rf  /usr/bin/pip3.7
```

6. 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。

```
python3.7.5 --version
pip3.7.5  --version
```

步骤3 安装 Python3开发环境。

安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install attrs
pip3.7.5 install psutil
pip3.7.5 install decorator
pip3.7.5 install numpy
pip3.7.5 install protobuf==3.11.3
pip3.7.5 install scipy
pip3.7.5 install sympy
pip3.7.5 install cffi
```

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**D.5 pip3.7.5 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

3.3.3 EulerOS aarch64 系统

环境要求

开发套件包、Python、开发辅助工具、应用程序编译需要安装以下软件或依赖。

表 3-4 依赖信息

类别	版本要求
Python	3.7.5
cmake	3.5.1+
protobuf	3.11.3+
gcc	7.3.0及以上
gcc-c++	7.3.0及以上

类别	版本要求
make unzip zlib-devel libffi-devel openssl-devel sqlite-devel pciutils net-tools lapack lapack-devel blas blas-devel gcc-gfortran attrs psutil decorator numpy scipy sympy cffi	无版本要求，安装的版本以操作系统自带的源为准。

配置安装用户权限（可选）

当用户使用非root用户安装时，需要操作该章节，否则请忽略。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo yum**权限，请以root用户执行如下操作。

1. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
2. 在该文件“root ALL=(ALL) ALL”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   请将“username”替换成实际安装用户。

   说明

   请确保“ /etc/sudoers”文件存在“ #includedir /etc/sudoers.d”，如果没有该信息，请手动添加至文件末尾。
3. 添加完成后，执行**:wq!**保存文件。
4. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```


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 repolist

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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

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
rpm -qa |grep unzip
rpm -qa |grep zlib-devel
rpm -qa |grep libffi-devel
rpm -qa |grep openssl-devel
rpm -qa |grep sqlite-devel
rpm -qa |grep pciutils
rpm -qa |grep net-tools
rpm -qa |grep lapack
rpm -qa |grep lapack-devel
rpm -qa |grep blas
rpm -qa |grep blas-devel
rpm -qa |grep gcc-gfortran
```

若分别返回如下信息则说明已经安装，进入下一步。

```
gcc (GCC) 7.3.0
g++ (GCC) 7.3.0
GNU Make 4.2.1
cmake version 3.12.1
unzip-6.0-40.eulerosv2r8.aarch64
zlib-devel-1.2.11-14.eulerosv2r8.aarch64
libffi-devel-3.1-18.h3.eulerosv2r8.aarch64
openssl-devel-1.1.1-3.h7.eulerosv2r8.aarch64
sqlite-devel-3.24.0-2.h4.eulerosv2r8.aarch64
pciutils-3.5.1-3.el7.aarch64
net-tools-2.0-0.24.20131004git.el7.aarch64
lapack-3.8.0-10.eulerosv2r8.aarch64
lapack-devel-3.8.0-10.eulerosv2r8.aarch64
blas-3.8.0-10.eulerosv2r8.aarch64
blas-devel-3.8.0-10.eulerosv2r8.aarch64
gcc-gfortran-7.3.0-20190804.h22.eulerosv2r8.aarch64
```

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

⚠ 注意

- 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python3.7.5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 sqlite-devel pciutils net-tools lapack lapack-devel blas blas-devel gcc-gfortran
```

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python --version**、**pip --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 执行如下命令复制libpython3.7m.so.1.0动态库：
   a. 由于yum工具强依赖于系统自带的libpython3.7m.so.1.0文件，因此需要备份系统自带的libpython3.7m.so.1.0文件。请执行如下命令：
      ```
      cd /usr/lib64
      cp libpython3.7m.so.1.0 libpython3.7m.so.1.0.bak
      ```
      如果环境上没有/usr/lib64，则以实际路径为准。
   b. 将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。
      ```
      sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
      ```
      如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。
      ```
      cp: overwrite 'libpython3.7m.so.1.0'?
      ```
      如果环境上没有/usr/lib64，则复制到/usr/lib目录：
      ```
      sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
      ```
      libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。
   c. 当需要使用yum工具时，需要执行如下命令将a备份的文件libpython3.7m.so.1.0.bak恢复。


```
mv libpython3.7m.so.1.0.bak libpython3.7m.so.1.0
```

5. 执行如下命令设置软链接：

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。

```
sudo rm -rf  /usr/bin/python3.7
sudo rm -rf  /usr/bin/pip3.7
sudo rm -rf  /usr/bin/python3.7.5
sudo rm -rf  /usr/bin/pip3.7.5
```

6. 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。

```
python3.7 --version
pip3.7 --version
python3.7.5 --version
pip3.7.5 --version
```

步骤3 安装 Python3开发环境。

安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install attrs
pip3.7.5 install psutil
pip3.7.5 install decorator
pip3.7.5 install numpy
pip3.7.5 install protobuf==3.11.3
pip3.7.5 install scipy
pip3.7.5 install sympy
pip3.7.5 install cffi
```

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**D.5 pip3.7.5 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

3.3.4 CentOS x86_64 系统

环境要求

开发套件包、Python、开发辅助工具、应用程序编译需要安装以下软件或依赖。

表 3-5 依赖信息

类别	版本要求
Python	3.7.5
cmake	3.5.1+

类别	版本要求
protobuf	3.11.3+
numpy	1.13.3+
gcc	软件源默认安装gcc版本为4.8.5。
gcc-c++	
make unzip zlib-devel libffi-devel openssl-devel sqlite-devel blas-devel lapack-devel openblas-devel pciutils net-tools attrs psutil decorator scipy sympy cffi	无版本要求，安装的版本以操作系统自带的源为准。

配置安装用户权限（可选）

当用户使用非root用户安装时，需要操作该章节，否则请忽略。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo yum**权限，请以root用户执行如下操作。

1. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
2. 在该文件“root ALL=(ALL) ALL”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   请将“username”替换成实际安装用户。

   说明

   请确保“ /etc/sudoers”文件存在“ #includedir /etc/sudoers.d”，如果没有该信息，请手动添加至文件末尾。

3. 添加完成后，执行**:wq!**保存文件。
4. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```

配置源

开发套件包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/CentOS-Base.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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

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
  rpm -qa |grep unzip
  rpm -qa |grep zlib-devel
  rpm -qa |grep libffi-devel
  rpm -qa |grep openssl-devel
  rpm -qa |grep sqlite-devel
  rpm -qa |grep blas-devel
  rpm -qa |grep lapack-devel
  rpm -qa |grep openblas-devel
  rpm -qa |grep pciutils
  rpm -qa |grep net-tools
  ```

  若分别返回如下信息则说明已经安装，进入下一步。

  ```
  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 4.1
  cmake version 3.5.1
  unzip-6.0-21.el7.x86_64
  zlib-devel-1.2.7-18.el7.x86_64
  libffi-devel-3.0.13-18.el7.x86_64
  openssl-devel-1.0.2k-19.el7.x86_64
  sqlite-devel-3.7.17-8.el7_7.1.aarch64
  blas-devel-3.4.2-8.el7.x86_64
  lapack-devel-3.4.2-8.el7.x86_64
  openblas-devel-0.3.3-2.el7.x86_64
  pciutils-3.5.1-3.el7.aarch64
  net-tools-2.0-0.24.20131004git.el7.aarch64
  ```

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

> **注意**
>
> - 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python3.7.5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 sqlite-devel blas-devel lapack-devel openblas-devel pciutils net-tools
```

上述步骤中，若安装openblas-devel时报“No package libopenblas available”的错误，需要安装企业版Linux的扩展包后重新安装。扩展包安装命令如下：

```
sudo yum install epel-release
```

如果通过上述方式安装的cmake版本低于3.5.1，则请参见**C.3 安装3.5.2版本cmake**解决。

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。

   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```

   当出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。

   ```
   cp: overwrite 'libpython3.7m.so.1.0'?y
   ```

   如果环境上没有/usr/lib64，则复制到/usr/lib目录：

   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```

   libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。

5. 执行如下命令设置软链接：


```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。

```
sudo rm -rf  /usr/bin/python3.7.5
sudo rm -rf  /usr/bin/pip3.7.5
sudo rm -rf  /usr/bin/python3.7
sudo rm -rf  /usr/bin/pip3.7
```

6. 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。

```
python3.7.5 --version
pip3.7.5  --version
```

步骤3 安装 Python3开发环境。

安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install attrs
pip3.7.5 install psutil
pip3.7.5 install decorator
pip3.7.5 install numpy==1.17.2
pip3.7.5 install protobuf==3.11.3
pip3.7.5 install scipy
pip3.7.5 install sympy
pip3.7.5 install cffi
```

- 如果安装numpy报错，请参考**D.4 pip3.7.5 install numpy报错**解决。
- 如果安装scipy报错，请参考**D.3 pip3.7.5 install scipy报错**解决。

----结束

3.3.5 Ubuntu x86_64 系统

环境要求

开发套件包、Python、开发辅助工具、应用程序编译需要安装以下软件或依赖。

表 3-6 依赖信息

类别	版本要求
Python	3.7.5
cmake	3.5.1+
protobuf	3.11.3+
g++	7.3.0及以上
gcc	7.3.0及以上

类别	版本要求
make zlib1g zlib1g-dev libsqlite3-dev openssl libssl-dev libffi-dev unzip pciutils net-tools attrs psutil decorator numpy scipy sympy cffi	无版本要求，安装的版本以操作系统自带的源为准。

配置安装用户权限（可选）

当用户使用非root用户安装时，需要操作该章节，否则请忽略。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo apt-get**权限，请以root用户执行如下操作。

1. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
2. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5, /usr/bin/unzip
   ```
   请将“username”替换成实际安装用户。

   说明

   请确保“ /etc/sudoers”文件存在“ #includedir /etc/sudoers.d”，如果没有该信息，请手动添加至文件末尾。
3. 添加完成后，执行**:wq!**保存文件。
4. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```

检查源

开发套件包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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

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
```

若分别返回如下信息则说明已经安装，进入下一步。

```
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:amd64   1:1.2.11.dfsg-0ubuntu2 amd64       compression library - runtime
zlib1g-dev:amd64 1:1.2.11.dfsg-0ubuntu2 amd64       compression library - development
libsqlite3-dev:amd64 3.22.0-1ubuntu0.3 amd64       SQLite 3 development files
openssl        1.1.1-1ubuntu2.1~18.04.5 amd64       Secure Sockets Layer toolkit - cryptographic utility
libssl-dev:amd64 1.1.1-1ubuntu2.1~18.04.5 amd64       Secure Sockets Layer toolkit - development files
libffi-dev:amd64 3.2.1-8      amd64       Foreign Function Interface library (development files)
unzip          6.0-21ubuntu1 amd64       De-archiver for .zip files
pciutils       1:3.5.2-1ubuntu1.1 amd64       Linux PCI Utilities
net-tools      1.60+git20161116.90da8a0-1ubuntu1 amd64       NET-3 networking toolkit
```

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

注意

- 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python3.7.5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 libsqlite3-dev openssl libssl-dev libffi-dev
unzip pciutils net-tools
```

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   当出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。
   ```
   cp: overwrite 'libpython3.7m.so.1.0'?y
   ```
   如果环境上没有/usr/lib64，则复制到/usr/lib目录：
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```
   libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。
5. 执行如下命令设置软链接：
   ```
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。
   ```
   sudo rm -rf  /usr/bin/python3.7.5
   sudo rm -rf  /usr/bin/pip3.7.5
   sudo rm -rf  /usr/bin/python3.7
   sudo rm -rf  /usr/bin/pip3.7
   ```
6. 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。
   ```
   python3.7.5 --version
   pip3.7.5  --version
   ```

步骤3 安装 Python3开发环境。

安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。

- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install attrs
pip3.7.5 install psutil
pip3.7.5 install decorator
pip3.7.5 install numpy
pip3.7.5 install protobuf==3.11.3
pip3.7.5 install scipy
pip3.7.5 install sympy
pip3.7.5 install cffi
```

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**D.5 pip3.7.5 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

3.4 安装开发套件包

前提条件

- 请参见**3.3 安装OS依赖**完成安装前准备。
- 通过**3.1 准备软件包**章节获取开发套件包Ascend-cann-toolkit_*xxx*.run。

安装步骤

步骤1 以软件包的安装用户登录安装环境。

若**3.3 安装OS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**3.3 安装OS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 *.run

说明

其中***.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。

./*.run --check

步骤6 （可选）创建软件安装路径。

- 如果用户想指定安装路径，则需要先创建安装路径。以安装路径“/home/work”为例， 用户先执行**mkdir -p /home/work**命令创建安装路径，再选择该路径进行软件安装。
- 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。

步骤7 执行以下命令安装软件。（以下命令支持**--install-for-all**和**--install-path=***<path>*等，具体参数说明请参见**E.1 参数说明**）

- 若使用非root用户安装，请执行以下命令。

  ./*.run --install

- 若使用root用户安装，请执行以下命令。
  - 若使用默认运行用户HwHiAiUser：

    ./*.run --install

  - 若用户需自行指定运行用户：

    ./*.run --install-username=*username* **--install-usergroup=***usergroup* **--install**

    其中**--install-username**和**--install-usergroup**用于指定运行用户。

说明

- 若软件包安装后提示重启生效，请执行**reboot**命令。
- 开发套件包支持不同用户在同一开发环境安装多个版本，不同用户所属的属组必须和Driver运行用户所属属组相同；如果不同，请用户自行添加到Driver运行用户属组。
- 如果以root用户安装，**建议不要安装在非root用户目录下**，否则存在被非root用户替换root用户文件以达到提权目的的安全风险。
- 开发套件包Ascend-cann-toolkit_*{version}*_linux-*{arch}*.run在**X86系统**中安装的时候不支持--quiet选项。
- 开发套件包Ascend-cann-toolkit_*{version}*_linux-*{arch}*.run在**X86系统**中安装的时候，如果出现如下情况，即询问是否进行热复位，则用户需输入n，安装完毕后请重启OS，才能生效。当前版本只支持n选项。

  ```
  The installation of aicpu_kernels needs to restart the device to take effect, do you want to hot_reset the device? [y/n] n
  ```

更多安装模式请参见**E.1 参数说明**。

安装完成后，若显示如下信息，则说明软件安装成功：

```
[INFO] xxx install success
[INFO] process end
```

*xxx*表示安装的实际软件包名。

步骤8 若上一步使用非root用户安装开发套件包，则需执行此步骤。请切换成root用户（也可使用sudo）执行以下安装命令。

1. 进入AICPU算子包所在路径。

   cd *${install_path}*/ascend-toolkit/latest/*{arch}*-linux/aicpu

2. 安装AICPU算子包。

   ./Ascend310-aicpu_kernels-*{version}*.run --full

   AICPU不支持指定安装路径，共用Driver的安装路径。

----结束

（参考）日志文件和软件包信息

表 3-7 日志文件和软件包信息路径

项目	路径
安装详细日志路径	• V100R020C10版本：“*${install_path}*/ascend-toolkit/latest/*{arch}*-linux/ascend_install.log” • V100R020C10SPC100及以上版本： root用户：“/var/log/ascend_seclog/ascend_toolkit_install.log” 非root用户：“*${HOME}*/var/log/ascend_seclog/ascend_toolkit_install.log”
安装后软件包版本、CPU架构、GCC版本和安装路径等信息的记录路径	“*${install_path}*/ascend-toolkit/latest/*{arch}*-linux/ascend_toolkit_install.info”

表3-7中各变量的含义如表3-8所示：

表 3-8 变量含义

参数	含义
{arch}-linux	*{arch}*为CPU架构
${install_path}	软件包的安装路径。

3.5 配置交叉编译环境

对于Atlas 200 AI加速模块（RC场景）和Atlas 500 智能小站，是准备一台X86服务器（或者PC）用作搭建开发环境。因此需要在开发环境安装交叉编译工具，具体如表3-9所示：

表 3-9 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，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）： **sudo apt-get install g++-aarch64-linux-gnu**

4 安装开发环境（训练）

4.1 准备软件包

下载软件包

软件安装前，请获取所需软件包和对应数字签名文件，各软件包版本号需要保持一致。

表 4-1 CANN 软件包

名称	软件包	作用	获取链接
开发套件包	Ascend-cann-toolkit_*{version}*_linux-*{arch}*.run	• 主要用于用户开发自定义算子。 • 请根据CPU架构（x86_64、aarch64）获取对应的软件包。	**获取链接**

名称	软件包	作用	获取链接
框架插件包	• V100R020C10版本：Ascend-fwk-tfplugin_*{version}*_linux-*{arch}*.run • V100R020C10SPC100及以上版本：Ascend-cann-tfplugin_*{version}*_linux-*{arch}*.run	包含框架适配插件Plugin，用于对接上层框架，如Tensorflow的适配插件。	获取链接

说明

*{version}*表示软件版本号，*{arch}*表示CPU架构。

检查软件包的完整性

说明

若您获取的是社区版本，可不进行此操作。

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。

4.2 准备安装及运行用户

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。

创建安装及运行用户

- 安装用户：实际安装软件包的用户。
  - 若使用root用户安装，因要求使用非root用户运行，所以安装前需要先准备运行用户。
  - 若使用非root用户安装，则安装及运行用户必须相同。
- 运行用户：实际运行推理业务或执行训练的用户。
- **使用root用户安装**
  - （推荐）如果创建的运行用户是HwHiAiUser，其为CANN软件包的默认运行用户，在安装时无需指定该运行用户。
  - 如果创建的运行用户是非HwHiAiUser，安装CANN软件包时需要指定该运行用户。
- **使用非root用户安装**
  - 已有非root用户，则无需再次创建。
  - 新建非root用户，请参见如下方法创建。

须知

- CANN软件包如果使用非root用户安装，用户所属的属组必须和Driver运行用户所属属组相同；如果不同，请用户自行添加到Driver运行用户属组。
- 创建的运行用户不建议为root用户属组，原因是：权限控制可能存在安全风险。

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
   ```
   以创建运行用户HwHiAiUser为例：
   ```
   groupadd HwHiAiUser
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser -s /bin/bash
   ```
2. 设置非root用户密码。
   ```
   passwd username
   ```
   示例如下：
   ```
   passwd HwHiAiUser
   ```

说明

- 创建完HwHiAiUser用户后， 请勿关闭该用户的登录认证功能。
- 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C.7 设置用户有效期。

4.3 安装 OS 依赖

4.3.1 Ubuntu aarch64 系统

环境要求

开发环境需要安装以下软件或依赖。

表 4-2 依赖信息

类别	版本要求	说明
Python	3.7.5	-
cmake	3.5.1+	编译环境依赖。
make	-	
gcc	7.3.0及以上	
g++	7.3.0及以上	
zlib1g zlib1g-dev libbz2-dev libsqlite3-dev openssl libssl-dev libxslt1-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3 liblapack-dev liblapack3 libopenblas-dev	无版本要求，安装的版本以操作系统自带的源为准。	
numpy	>=1.13.3	CANN软件包python依赖。
decorator	>=4.4.0	
sympy	1.4	
cffi	1.12.3	
pyyaml pathlib2 protobuf scipy requests psutil	无版本要求，安装的版本以pip源为准。	

检查源

深度学习引擎包、实用工具包和框架插件包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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

配置安装用户权限

当用户使用非root用户安装时，需要操作该章节；当用户使用root用户安装时，仅需执行步骤**1**。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo apt-get**权限，请以root用户执行如下操作。

1. 安装sudo，使用如下命令安装。
   ```
   apt-get install sudo
   ```
2. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
3. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   “username”为执行安装脚本的普通用户名。

   说明

   请确保“/etc/sudoers”文件的最后一行为“#includedir /etc/sudoers.d”，如果没有该信息，请手动添加。
4. 添加完成后，执行**:wq!**保存文件。
5. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```

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 libbz2-dev| grep libbz2-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 libxslt1-dev| grep libxslt1-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
dpkg -l libopenblas-dev| grep libopenblas-dev| grep ii
```

若分别返回如下信息则说明已经安装，进入下一步。

```
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
libbz2-dev:arm64 1.0.6-8.1ubuntu0.2 arm64       high-quality block-sorting file compressor 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
libxslt1-dev:arm64 1.1.29-5ubuntu0.2 arm64       XSLT 1.0 processing library - development kit
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
libopenblas-dev:arm64 0.2.20+ds-4  arm64     Optimized BLAS (linear algebra) library (development files)
```

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

⚠ 注意

- 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python3.7.5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 libbz2-dev openssl libsqlite3-dev libssl-dev libxslt1-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3 libopenblas-dev
```

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。

   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```

   如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。

   ```
   cp: overwrite 'libpython3.7m.so.1.0'?
   ```

   如果环境上没有/usr/lib64，则复制到/usr/lib目录：

   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```

   libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。

5. 执行如下命令设置软链接：

   ```
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```

   执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。

   ```
   sudo rm -rf  /usr/bin/python3.7.5
   sudo rm -rf  /usr/bin/pip3.7.5
   sudo rm -rf  /usr/bin/python3.7
   sudo rm -rf  /usr/bin/pip3.7
   ```

6. 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。

   ```
   python3.7.5 --version
   pip3.7.5  --version
   ```

步骤3 安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install numpy
pip3.7.5 install decorator
pip3.7.5 install sympy==1.4
pip3.7.5 install cffi==1.12.3
pip3.7.5 install pyyaml
pip3.7.5 install pathlib2
pip3.7.5 install psutil
pip3.7.5 install protobuf
pip3.7.5 install scipy
pip3.7.5 install requests
```

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**D.5 pip3.7.5 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

4.3.2 EulerOS aarch64 系统

环境要求

开发环境需要安装以下软件或依赖。

表 4-3 依赖信息

类别	版本要求	说明
Python	3.7.5	-
cmake	3.5.1+	编译环境依赖。
make	-	
gcc	7.3.0及以上	
gcc-c++	7.3.0及以上	
zlib-devel libffi-devel openssl-devel sqlite-devel unzip pciutils net-tools lapack lapack-devel blas blas-devel gcc-gfortran	无版本要求，安装的版本以操作系统自带的源为准。	
numpy	>=1.13.3	CANN软件包python依赖。
decorator	>=4.4.0	
sympy	1.4	
cffi	1.12.3	
pyyaml pathlib2 protobuf scipy requests psutil	无版本要求，安装的版本以pip源为准。	

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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

配置安装用户权限

当用户使用非root用户安装时，需要操作该章节；当用户使用root用户安装时，仅需执行步骤**1**。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo yum**权限，请以root用户执行如下操作。

1. 安装sudo，使用如下命令安装。
   ```
   yum install sudo
   ```
2. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
3. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   “username”为执行安装脚本的普通用户名。

   说明

   请确保“/etc/sudoers”文件的最后一行为“#includedir /etc/sudoers.d”，如果没有该信息，请手动添加。
4. 添加完成后，执行**:wq!**保存文件。
5. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```

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
rpm -qa |grep zlib-devel
rpm -qa |grep libffi-devel
rpm -qa |grep openssl-devel
rpm -qa |grep sqlite-devel
rpm -qa |grep unzip
rpm -qa |grep pciutils
rpm -qa |grep net-tools
rpm -qa |grep lapack
rpm -qa |grep lapack-devel
```



```
rpm -qa |grep blas
rpm -qa |grep blas-devel
rpm -qa |grep gcc-gfortran
```

若分别返回如下信息则说明已经安装，进入下一步。

```
gcc (GCC) 7.3.0
g++ (GCC) 7.3.0
GNU Make 4.2.1
cmake version 3.12.1
zlib-devel-1.2.11-14.eulerosv2r8.aarch64
libffi-devel-3.1-18.h3.eulerosv2r8.aarch64
openssl-devel-1.1.1-3.h7.eulerosv2r8.aarch64
sqlite-devel-3.24.0-2.h4.eulerosv2r8.aarch64
unzip-6.0-40.eulerosv2r8.aarch64
pciutils-3.6.2-1.eulerosv2r8.aarch64
net-tools-2.0-0.52.20160912git.eulerosv2r8.aarch64
lapack-3.8.0-10.eulerosv2r8.aarch64
lapack-devel-3.8.0-10.eulerosv2r8.aarch64
blas-3.8.0-10.eulerosv2r8.aarch64
blas-devel-3.8.0-10.eulerosv2r8.aarch64
gcc-gfortran-7.3.0-20190804.h22.eulerosv2r8.aarch64
```

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

⚠ 注意

- 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python3.7.5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 zlib-devel libffi-devel openssl-devel sqlite-devel unzip pciutils net-tools lapack lapack-devel blas blas-devel gcc-gfortran
```

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python --version**、**pip --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 执行如下命令复制libpython3.7m.so.1.0动态库：

   a. 由于yum工具强依赖于系统自带的libpython3.7m.so.1.0文件，因此需要备份系统自带的libpython3.7m.so.1.0文件。请执行如下命令：

   ```
   cd /usr/lib64
   cp libpython3.7m.so.1.0 libpython3.7m.so.1.0.bak
   ```

   如果环境上没有/usr/lib64，则以实际路径为准。

   b. 将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。

   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```

   如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。

   ```
   cp: overwrite 'libpython3.7m.so.1.0'?
   ```

   如果环境上没有/usr/lib64，则复制到/usr/lib目录：

   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```

   libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。

   c. 当需要使用yum工具时，需要执行如下命令将**a**备份的文件libpython3.7m.so.1.0.bak恢复。

   ```
   mv libpython3.7m.so.1.0.bak libpython3.7m.so.1.0
   ```

5. 执行如下命令设置软链接：

   ```
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```

   执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。

   ```
   sudo rm -rf  /usr/bin/python3.7.5
   sudo rm -rf  /usr/bin/pip3.7.5
   sudo rm -rf  /usr/bin/python3.7
   sudo rm -rf  /usr/bin/pip3.7
   ```

6. 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。

   ```
   python3.7 --version
   pip3.7 --version
   python3.7.5 --version
   pip3.7.5 --version
   ```

步骤3 安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install numpy
pip3.7.5 install decorator
pip3.7.5 install sympy==1.4
pip3.7.5 install cffi==1.12.3
pip3.7.5 install pyyaml
pip3.7.5 install pathlib2
pip3.7.5 install psutil
pip3.7.5 install protobuf
pip3.7.5 install scipy
pip3.7.5 install requests
```


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**D.5 pip3.7.5 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

4.3.3 CentOS aarch64 系统

环境要求

开发环境需要安装以下软件或依赖。

表 4-4 依赖信息

类别	版本限制	说明
Python	3.7.5	-
cmake	3.5.1+	编译环境依赖。
make	-	
gcc	软件源默认安装gcc版本为4.8.5。 **训练场景下**，要求7.3.0版本及以上gcc，安装过程可参考**C.5 安装7.3.0版本gcc**。	
gcc-c++		
unzip zlib-devel libffi-devel openssl-devel pciutils net-tools sqlite-devel blas-devel lapack-devel openblas-devel gcc-gfortran	无版本要求，安装的版本以操作系统自带的源为准。	
numpy	>=1.13.3	CANN软件包python依赖。
decorator	>=4.4.0	
sympy	1.4	
cffi	1.12.3	

类别	版本限制	说明
pyyaml pathlib2 protobuf scipy requests psutil	无版本要求，安装的版本以pip源为准。	

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 repolist
```

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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

配置安装用户权限

当用户使用非root用户安装时，需要操作该章节；当用户使用root用户安装时，仅需执行步骤**1**。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo yum**权限，请以root用户执行如下操作。

1. 安装sudo，使用如下命令安装。
   ```
   yum install sudo
   ```
2. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
3. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   “username”为执行安装脚本的普通用户名。

   说明

   请确保“/etc/sudoers”文件的最后一行为“#includedir /etc/sudoers.d”，如果没有该信息，请手动添加。
4. 添加完成后，执行**:wq!**保存文件。
5. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```


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
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 blas-devel
rpm -qa |grep lapack-devel
rpm -qa |grep openblas-devel
rpm -qa |grep gcc-gfortran
```

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，请以实际情况为准）。

```
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 2.8.12.2
unzip-6.0-21.el7.aarch64
zlib-devel-1.2.7-18.el7.aarch64
libffi-devel-3.0.13-18.el7.aarch64
openssl-devel-1.0.2k-19.el7.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
blas-devel-3.4.2-8.el7.aarch64
lapack-devel-3.4.2-8.el7.aarch64
openblas-devel-0.3.3-2.el7.aarch64
gcc-gfortran-4.8.5-39.el7.aarch64
```

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

注意

- 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python3.7.5环境，在此之后再安装sqlite-devel，则需要重新编译python环境。
- **训练场景下**，要求7.3.0版本及以上gcc，因此需要安装7.3.0版本gcc，安装过程可参考**C.5 安装7.3.0版本gcc**。

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

上述步骤中，安装openblas-devel如报No package libopenblas available。需要安装企业版linux的扩展包，安装命令如下：

```
sudo yum install epel-release
```


如果通过上述方式安装的cmake版本低于3.5.1，则请参见**C.3 安装3.5.2版本cmake**解决。

步骤2 检查系统是否安装python开发环境。

深度学习引擎包、实用工具包和框架插件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。
   ```
   cp: overwrite 'libpython3.7m.so.1.0'?
   ```
   如果环境上没有/usr/lib64，则复制到/usr/lib目录：
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```
   libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。
5. 执行如下命令设置软链接：
   ```
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。
   ```
   sudo rm -rf  /usr/bin/python3.7.5
   sudo rm -rf  /usr/bin/pip3.7.5
   sudo rm -rf  /usr/bin/python3.7
   sudo rm -rf  /usr/bin/pip3.7
   ```
6. 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。
   ```
   python3.7.5 --version
   pip3.7.5  --version
   ```


步骤3 安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install numpy==1.17.2
pip3.7.5 install decorator
pip3.7.5 install sympy==1.4
pip3.7.5 install cffi==1.12.3
pip3.7.5 install pyyaml
pip3.7.5 install pathlib2
pip3.7.5 install psutil
pip3.7.5 install protobuf
pip3.7.5 install scipy
pip3.7.5 install requests
```

- 如果安装numpy报错，请参考**D.4 pip3.7.5 install numpy报错**解决。
- 如果安装scipy报错，请参考**D.3 pip3.7.5 install scipy报错**解决。

----结束

4.3.4 Ubuntu x86_64 系统

环境要求

开发环境需要安装以下软件或依赖。

表 4-5 依赖信息

类别	版本限制	说明
Python	3.7.5	-
cmake	3.5.1+	编译环境依赖。
make	-	
gcc	7.3.0及以上	
g++	7.3.0及以上	

类别	版本限制	说明
zlib1g zlib1g-dev libbz2-dev libsqlite3-dev libssl-dev libxslt1-dev libffi-dev unzip pciutils net-tools liblapack-dev liblapack3 libblas-dev gfortran libblas3	无版本要求，安装的版本以操作系统自带的源为准。	
numpy	>=1.13.3	CANN软件包python依赖。
decorator	>=4.4.0	
sympy	1.4	
cffi	1.12.3	
pyyaml pathlib2 protobuf scipy requests psutil	无版本要求，安装的版本以pip源为准。	

检查源

深度学习引擎包、实用工具包和框架插件包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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

配置安装用户权限（可选）

当用户使用非root用户安装时，需要操作该章节，否则请忽略。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo apt-get**权限，请以root用户执行如下操作。

1. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
2. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5, /usr/bin/unzip
   ```
   请将“username”替换成实际安装用户。

   说明

   请确保“ /etc/sudoers”文件存在“ #includedir /etc/sudoers.d”，如果没有该信息，请手动添加至文件末尾。
3. 添加完成后，执行**:wq!**保存文件。
4. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```

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 libbz2-dev| grep libbz2-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 libxslt1-dev| grep libxslt1-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
```

若分别返回如下信息则说明已经安装，进入下一步。

```
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:amd64   1:1.2.11.dfsg-0ubuntu2 amd64       compression library - runtime
zlib1g-dev:amd64  1:1.2.11.dfsg-0ubuntu2  amd64       compression library - development
libbz2-dev:amd64    1.0.6-8.1ubuntu0.2 amd64       high-quality block-sorting file compressor library - development
libsqlite3-dev:amd64    3.22.0-1ubuntu0.3 amd64       SQLite 3 development files
openssl        1.1.1-1ubuntu2.1~18.04.6 amd64       Secure Sockets Layer toolkit - cryptographic utility
libssl-dev:amd64    1.1.1-1ubuntu2.1~18.04.6 amd64       Secure Sockets Layer toolkit - development files
libxslt1-dev:amd64 1.1.29-5ubuntu0.2 amd64       XSLT 1.0 processing library - development kit
libffi-dev:amd64 3.2.1-8      amd64       Foreign Function Interface library (development files)
unzip          6.0-21ubuntu1 amd64       De-archiver for .zip files
```



```
pciutils      1:3.5.2-1ubuntu1 amd64       Linux PCI Utilities
net-tools     1.60+git20161116.90da8a0-1ubuntu1 amd64       NET-3 networking toolkit
```

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

⚠ 注意

- 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python3.7.5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 gcc g++ make cmake zlib1g zlib1g-dev libbz2-dev libsqlite3-dev libssl-dev libxslt1-dev libffi-dev unzip pciutils net-tools -y
```

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。
   ```
   cp: overwrite 'libpython3.7m.so.1.0'?
   ```
   如果环境上没有/usr/lib64，则复制到/usr/lib目录：
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```
   libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。
5. 执行如下命令设置软链接：
   ```
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   ```



```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。

```
sudo rm -rf  /usr/bin/python3.7.5
sudo rm -rf  /usr/bin/pip3.7.5
sudo rm -rf  /usr/bin/python3.7
sudo rm -rf  /usr/bin/pip3.7
```

6. 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。

```
python3.7.5 --version
pip3.7.5  --version
```

步骤3 安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install numpy
pip3.7.5 install decorator
pip3.7.5 install sympy==1.4
pip3.7.5 install cffi==1.12.3
pip3.7.5 install pyyaml
pip3.7.5 install pathlib2
pip3.7.5 install psutil
pip3.7.5 install protobuf
pip3.7.5 install scipy
pip3.7.5 install requests
```

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**D.5 pip3.7.5 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

4.3.5 CentOS x86_64 系统

环境要求

开发环境需要安装以下软件或依赖。

表 4-6 依赖信息

类别	版本限制	说明
Python	3.7.5	-
cmake	3.5.1+	编译环境依赖。
make	-	
gcc	软件源默认安装gcc版本为4.8.5。	

类别	版本限制	说明
gcc-c++	**训练场景下**，要求7.3.0版本及以上gcc，安装过程可参考**C.5 安装7.3.0版本gcc**。	
unzip zlib-devel libffi-devel openssl-devel pciutils net-tools sqlite-devel blas-devel lapack-devel openblas-devel gcc-gfortran	无版本要求，安装的版本以操作系统自带的源为准。	
numpy	>=1.13.3	CANN软件包python依赖。
decorator	>=4.4.0	
sympy	1.4	
cffi	1.12.3	
pyyaml pathlib2 protobuf scipy requests psutil	无版本要求，安装的版本以pip源为准。	

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 repolist
```

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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

配置安装用户权限（可选）

当用户使用非root用户安装时，需要操作该章节，否则请忽略。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo yum**权限，请以root用户执行如下操作。

1. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
2. 在该文件“root ALL=(ALL) ALL”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   请将“username”替换成实际安装用户。

   说明

   请确保“ /etc/sudoers”文件存在“ #includedir /etc/sudoers.d”，如果没有该信息，请手动添加至文件末尾。
3. 添加完成后，执行**:wq!**保存文件。
4. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```

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
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 blas-devel
rpm -qa |grep lapack-devel
rpm -qa |grep openblas-devel
rpm -qa |grep gcc-gfortran
```

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，请以实际情况为准）。

```
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 2.8.12.2
unzip-6.0-21.el7.x86_64
zlib-devel-1.2.7-18.el7.x86_64
libffi-devel-3.0.13-18.el7.x86_64
openssl-devel-1.0.2k-19.el7.x86_64
pciutils-3.5.1-3.el7.x86_64
net-tools-2.0-0.25.20131004git.el7.x86_64
sqlite-devel-3.7.17-8.el7_7.1.x86_64
blas-devel-3.4.2-8.el7.x86_64
lapack-devel-3.4.2-8.el7.x86_64
openblas-devel-0.3.3-2.el7.x86_64
gcc-gfortran-4.8.5-39.el7.x86_64
```

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

⚠ 注意

- 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python3.7.5环境，在此之后再安装sqlite-devel，则需要重新编译python环境。
- **训练场景下**，要求7.3.0版本及以上gcc，因此需要安装7.3.0版本gcc，安装过程可参考**C.5 安装7.3.0版本gcc**。

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

上述步骤中，安装openblas-devel如报No package libopenblas available。需要安装企业版linux的扩展包，安装命令如下：

```
sudo yum install epel-release
```

如果通过上述方式安装的cmake版本低于3.5.1，则请参见**C.3 安装3.5.2版本cmake**解决。

步骤2 检查系统是否安装python开发环境。

深度学习引擎包、实用工具包和框架插件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。
   ```
   cp: overwrite 'libpython3.7m.so.1.0'?
   ```
   如果环境上没有/usr/lib64，则复制到/usr/lib目录：
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```


libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。

5. 执行如下命令设置软链接：

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。

```
sudo rm -rf  /usr/bin/python3.7.5
sudo rm -rf  /usr/bin/pip3.7.5
sudo rm -rf  /usr/bin/python3.7
sudo rm -rf  /usr/bin/pip3.7
```

6. 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。

```
python3.7.5 --version
pip3.7.5  --version
```

步骤3 安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install numpy==1.17.2
pip3.7.5 install decorator
pip3.7.5 install sympy==1.4
pip3.7.5 install cffi==1.12.3
pip3.7.5 install pyyaml
pip3.7.5 install pathlib2
pip3.7.5 install psutil
pip3.7.5 install protobuf
pip3.7.5 install scipy
pip3.7.5 install requests
```

- 如果安装numpy报错，请参考**D.4 pip3.7.5 install numpy报错**解决。
- 如果安装scipy报错，请参考**D.3 pip3.7.5 install scipy报错**解决。

----结束

4.4 安装开发套件和框架插件包

前提条件

- 请参见**4.3 安装OS依赖**完成安装前准备。
- 通过**4.1 准备软件包**章节获取开发套件包Ascend-cann-toolkit和框架插件包tfplugin。

安装步骤

步骤1 以软件包的安装用户登录安装环境。

请确保软件包的安装用户与**4.3 安装OS依赖**中安装依赖的用户保持一致。

步骤2 将获取到的开发套件包和框架插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 *.run

说明

其中***.run**表示软件包名（如开发套件包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。

./*.run --check

步骤6 （可选）创建软件安装路径。

- 如果用户想指定安装路径，则需要先创建安装路径。以安装路径“/home/work”为例， 用户先执行**mkdir -p /home/work**命令创建安装路径，再选择该路径进行软件安装。
- 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。

步骤7 执行以下命令安装软件。（以下命令支持**--install-for-all**和**--install-path=***<path>*等，具体参数说明请参见**E.1 参数说明**）

- 若使用非root用户安装，请执行以下命令。

  ./*.run --install
- 若使用root用户安装，请执行以下命令。
  - 若使用默认运行用户HwHiAiUser：

    ./*.run --install
  - 若用户需自行指定运行用户：

    ./*.run --install-username=*username* **--install-usergroup=***usergroup* **--install**

    其中**--install-username**和**--install-usergroup**用于指定运行用户。

⚠ 注意

- 若软件包安装后提示重启生效，请执行**reboot**命令。
- 开发套件包Ascend-cann-toolkit_*{version}*_linux-*{arch}*.run支持不同用户在同一开发环境安装多个版本，不同用户所属的属组必须和Driver运行用户所属属组相同；如果不同，请用户自行添加到Driver运行用户属组。
- 如果以root用户安装，**建议不要安装在非root用户目录下**，否则存在被非root用户替换root用户文件以达到提权目的的安全风险。
- 开发套件包Ascend-cann-toolkit_*{version}*_linux-*{arch}*.run在**X86系统**中安装的时候不支持--quiet选项。
- 开发套件包Ascend-cann-toolkit_*{version}*_linux-*{arch}*.run在**X86系统**中安装的时候，如果出现如下情况，即询问是否进行热复位，则用户需输入n，安装完毕后重启os生效，当前版本只支持n选项。

  ```
  The installation of aicpu_kernels needs to restart the device to take effect, do you want to hot_reset the device? [y/n]
  ```

更多安装模式请参见**E.1 参数说明**。

安装完成后，若显示如下信息，则说明软件安装成功：

```
[INFO] xxx install success
[INFO] process end
```

*xxx*表示安装的实际软件包名。

步骤8 若上一步使用非root用户安装开发套件包，则需执行此步骤。请切换成root用户（也可使用sudo）执行以下安装命令。

1. 进入AICPU算子包所在路径。

   cd *${install_path}*/**ascend-toolkit/latest/***{arch}*-**linux/aicpu**

2. 安装AICPU算子包。

   ./Ascend910-aicpu_kernels-*{version}*.**run --full**

   AICPU不支持指定安装路径，共用Driver的安装路径。

----结束

（参考）日志文件和软件包信息

下面以开发套件包Ascend-cann-toolkit_*xxx*.run为例介绍对应的日志文件和软件包信息路径。

表 4-7 日志文件和软件包信息路径

项目	路径
安装详细日志路径	• V100R020C10版本：“*${install_path}*/ascend-toolkit/latest/*{arch}*-linux/ascend_install.log” • V100R020C10SPC100及以上版本： root用户：“/var/log/ascend_seclog/ascend_toolkit_install.log” 非root用户：“*${HOME}*/var/log/ascend_seclog/ascend_toolkit_install.log”
安装后软件包版本、CPU架构、GCC版本和安装路径等信息的记录路径	“*${install_path}*/ascend-toolkit/latest/*{arch}*-linux/ascend_toolkit_install.info”

表4-7中各变量的含义如表4-8所示：

表 4-8 变量含义

参数	含义
{arch-linux}	形态目录，以软件包的CPU架构命名。
${install_path}	软件包的安装路径。

4.5 安装深度学习框架

本文档以TensorFlow为例介绍安装深度学习框架的步骤，如果用户要使用MindSpore框架，请登录**https://www.mindspore.cn/install**获取安装MindSpore框架的方法。如果用户要使用PyTorch框架，请参见《**FrameworkPTAdapter PyTorch网络模型移植&训练指南**》的“安装PyTorch框架”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。
  - 由于tensorflow依赖grpcio，建议在安装tensorflow之前指定版本安装grpcio，否则可能导致安装失败。

步骤1 编译安装HDF5。

1. 使用wget下载HDF5源码包，可以下载到安装环境的任意目录，命令为：
   ```
   wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz --no-check-certificate
   ```


2. 进入下载后的目录，解压源码包，命令为：
```
tar -zxvf hdf5-1.10.5.tar.gz
```
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
```

步骤2 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
```

步骤3 安装h5py。

root用户下安装h5py依赖包命令如下。
```
pip3.7 install Cython
```
安装h5py命令如下：
```
pip3.7 install h5py==2.8.0
```

步骤4 安装grpcio。

root用户下安装grpcio命令如下。
```
pip3.7 install grpcio==1.32.0
```

----结束

安装 Tensorflow 1.15.0

需要安装Tensorflow 1.15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，具体步骤请参考**https://www.tensorflow.org/install/pip?lang=python3**。需要注意tensorflow官网提供的指导描述有错误，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。即官网的“Tensorflow==1.15：仅支持 CPU 的版本”需要修改成“Tensorflow-cpu==1.15：仅支持 CPU 的版本”。另外，官网描述的安装命令“pip3 install --user --upgrade tensorflow”需要在root更改为“pip3.7 install Tensorflow-cpu==1.15”，非root用户下更改为“pip3.7 install Tensorflow-cpu==1.15 --user”。
- 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gcc7.3.0编译器编译tensorflow1.15.0，编译步骤参考官网https://www.tensorflow.org/install/source。特别注意点如下：

在下载完tensorflow tag v1.15.0源码后执行需要如下步骤。

步骤1 下载“nsync-1.22.0.tar.gz”源码包。

1. 进入tensorflow tag v1.15.0源码目录，打开“tensorflow/workspace.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”文件夹和“pax_global_header”文件。
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。

将解压出的“nsync-1.22.0”文件夹和“pax_global_header”压缩为一个新的“nsync-1.22.0.tar.gz”源码包，保存（假如保存在“/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。

sha256sum /tmp/nsync-1.22.0.tar.gz

执行如上命令后得到sha256sum校验码（一串数字和字母的组合）

步骤5 修改sha256sum校验码和urls。

进入tensorflow tag v1.15.0源码目录，打开“tensorflow/workspace.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=”后面的列表第一行，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 = [
        "file:///tmp/nsync-1.22.0.tar.gz ",        "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",
    ],
  )
```

步骤6 继续从官方的“配置build”（**https://www.tensorflow.org/install/source**）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（**https://www.tensorflow.org/install/source**）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root用户执行如下命令安装Tensorflow1.15：

```
pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
```

非root用户执行如下命令：

```
pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user
```

----结束

4.6 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”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**pip3.7 -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 卸载protobuf。

pip3.7 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服务器任一目录并进行解压（**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EulerOS时，安装命令如下：

   ```
   yum install autoconf automake libtool curl make gcc-c++ unzip libffi-devel -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，可以通过

   ./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

   说明

   若执行**ldconfig**提示“ldconfig: /lib64/libpython3.7m.so.1.0 is not a symbolic link”，可尝试以下操作解决：

   mv libpython3.7m.so.1.0 libpython3.7m.so.1.0.py

   ln -s libpython3.7m.so.1.0.py libpython3.7m.so.1.0

   protobuf安装完成后，会在**3. 配置安装路径**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**3. 配置安装路径**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**3. 配置安装路径**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。

   python3.7 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**python3.7 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。

   python3.7 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**pip3.7 -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. 在运行脚本中增加环境变量的设置：

   export LD_LIBRARY_PATH=/protobuf/lib

   "**/protobuf**"为**3. 配置安装路径**中用户配置的安装路径，默认安装路径为"**/usr/local**"。

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**3. 配置安装路径**中用户配置的安装路径。

----结束

4.7 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 HCCN Tool 接口参考》**。

Atlas 800 训练服务器、Atlas 900 AI集群场景

- 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。IP地址需要根据实际情况修改。

```
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：

  不需要限制网段，所有网卡设置成相同网段即可。

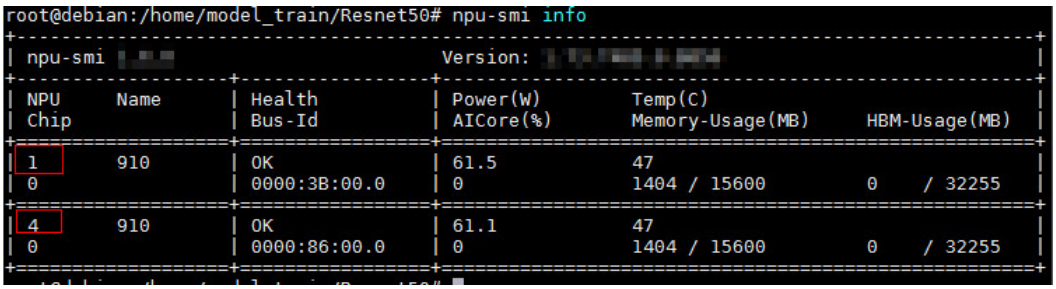
Atlas 300T 训练卡场景

Atlas 300T 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。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4-1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 4-1 查看 device ID



步骤2 执行如下命令配置device的网卡IP，下文所用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。
- 对于集群场景，各AI Server的device处于同一网段即可。

5 安装运行环境（推理）

5.1 安装前必读

运行环境是实际运行用户开发的应用程序的环境，运行环境要求配置昇腾AI设备（如推理卡或AI加速模块），应用程序依赖离线推理引擎包、实用工具包、驱动和固件进行离线加速推理。

应用程序运行环境包含物理机和容器两种场景。

运行场景	安装流程
物理机场景	1. **5.2 准备软件包** 2. **5.3.1 安装前准备** 3. **5.3.2 安装推理软件**
容器场景	1. **5.2 准备软件包** 2. **5.4.1 在宿主机安装实用工具包** 3. **5.4.2 构建推理容器镜像** 4. **5.4.3 部署推理容器**

说明

若配置的昇腾AI设备为Atlas 200 AI加速模块（EP场景），则应用程序运行环境仅存在物理机场景。

5.2 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，各软件包版本号需要保持一致。

表 5-1 CANN 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	主要包含ACL库ACLlib，用于应用程序的模型推理。 请根据CPU架构（x86_64、aarch64）获取对应得软件包	**获取链接**
实用工具包	• V100R020C10版本：Ascend-mindstudio-toolbox_*{version}*_linux-*{arch}*.run • V100R020C10SPC100及以上版本：Ascend-cann-toolbox_*{version}*_linux-*{arch}*.run	主要包含AICPU算子，用于推理模型调用算子。	**获取链接**

说明

*{version}*表示软件版本号，*{arch}*表示CPU架构。

检查软件包的完整性

说明

若您获取的是社区版本，可不进行此操作。

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。

5.3 在物理机安装

5.3.1 安装前准备

创建安装及运行用户

- 安装用户：实际安装软件包的用户。
  - 若使用root用户安装，因要求使用非root用户运行，所以安装前需要先准备运行用户。
  - 若使用非root用户安装，则安装及运行用户必须相同。
- 运行用户：实际运行推理业务或执行训练的用户。
- **使用root用户安装**
  - （推荐）如果创建的运行用户是HwHiAiUser，其为CANN软件包的默认运行用户，在安装时无需指定该运行用户。
  - 如果创建的运行用户是非HwHiAiUser，安装CANN软件包时需要指定该运行用户。
- **使用非root用户安装**
  - 已有非root用户，则无需再次创建。
  - 新建非root用户，请参见如下方法创建。

须知

- CANN软件包如果使用非root用户安装，用户所属的属组必须和Driver运行用户所属属组相同；如果不同，请用户自行添加到Driver运行用户属组。
- 创建的运行用户不建议为root用户属组，原因是：权限控制可能存在安全风险。

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
   ```
   以创建运行用户HwHiAiUser为例：
   ```
   groupadd HwHiAiUser
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser -s /bin/bash
   ```
2. 设置非root用户密码。
   ```
   passwd username
   ```
   示例如下：
   ```
   passwd HwHiAiUser
   ```

说明

- 创建完HwHiAiUser用户后， 请勿关闭该用户的登录认证功能。
- 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C.7 设置用户有效期。

5.3.2 安装推理软件

前提条件

- 请参见5.3.1 安装前准备完成安装前准备。

- 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**5.2 准备软件包**章节获取离线推理引擎包nnrt和实用工具包toolbox。

安装步骤

获取并安装离线推理引擎包和实用工具包，两者安装无顺序要求，且安装步骤相同，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和实用工具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 *.run

说明

其中***.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。

./*.run --check

步骤6 （可选）创建软件安装路径。

- 如果用户想指定安装路径，则需要先创建安装路径。以安装路径“/home/work”为例， 用户先执行**mkdir -p /home/work**命令创建安装路径，再选择该路径进行软件安装。
- 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。

步骤7 执行以下命令安装软件。（以下命令支持**--install-for-all**和**--install-path=***<path>*等，具体参数说明请参见**E.1 参数说明**）

- 若使用非root用户安装，请执行以下命令。

  ./*.run --install
- 若使用root用户安装，请执行以下命令。
  - 若使用默认运行用户HwHiAiUser：

    ./*.run --install
  - 若用户需自行指定运行用户：

    ./*.run --install-username=*username* **--install-usergroup=***usergroup* **--install**

    其中**--install-username**和**--install-usergroup**用于指定运行用户。

说明

- 若软件包安装后提示重启生效，请执行**reboot**命令。
- 离线推理引擎包和实用工具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Driver运行用户所属属组相同；如果不同，请用户自行添加到Driver运行用户属组。
- 如果以root用户安装，**建议不要安装在非root用户目录下**，否则存在被非root用户替换root用户文件以达到提权目的的安全风险。
- 实用工具包toolbox在**X86系统**中安装的时候不支持--quiet选项
- 实用工具包toolbox在**X86系统**中安装的时候，如果出现如下情况，即询问是否进行热复位，则用户需输入n，安装完毕后重启os生效，当前版本只支持n选项。
  ```
  The installation of aicpu_kernels needs to restart the device to take effect, do you want to hot_reset the device? [y/n] n
  ```

更多安装模式请参见**E.1 参数说明**。

安装完成后，若显示如下信息，则说明软件安装成功：

```
[INFO] xxx install success
[INFO] process end
```

*xxx*表示安装的实际软件包名。

步骤8 若上一步使用非root用户安装实用工具包，则需执行此步骤。

1. 进入实用工具包所在路径，执行以下命令解压实用工具包。

   ./Ascend-*xxx***-toolbox**_*{version}*_**linux-***{arch}***.run --noexec --extract=***{path}*

   其中*{path}*为解压路径，用户可自行指定。
2. 进入AICPU算子包所在路径。

   cd *{path}***/Tuscany/common**
3. 增加对AICPU算子包的可执行权限。

   chmod +x Ascend310-aicpu_kernels-*{version}***.run**
4. 安装AICPU算子包。

   请切换成root用户（也可使用sudo）执行以下安装命令。

   ./Ascend310-aicpu_kernels-*{version}***.run --full**

   AICPU不支持指定安装路径，共用Driver的安装路径。

----结束

（参考）日志文件和软件包信息

下面以离线推理引擎包Ascend-cann-nnrt_*xxx*.run为例介绍对应的日志文件和软件包信息路径。

表 5-2 日志文件和软件包信息路径

项目	路径
安装详细日志路径	• V100R020C10版本：“${install_path}/nnrt/latest/{arch-linux}/ascend_install.log” • V100R020C10SPC100及以上版本： root用户：“/var/log/ascend_seclog/ascend_nnrt_install.log” 非root用户：“${HOME}/var/log/ascend_seclog/ascend_nnrt_install.log”
安装后软件包版本、CPU架构、GCC版本和安装路径等信息的记录路径	“${install_path}/nnrt/latest/{arch-linux}/ascend_nnrt_install.info”

表5-2中各变量的含义如表5-3所示：

表 5-3 变量含义

参数	含义
{arch-linux}	形态目录，以软件包的CPU架构命名。
${install_path}	软件包的安装路径。

5.3.3 安装后处理及检查

本章节内容暂不支持在Atlas 200 AI加速模块（型号 3000）作为PCIe从设备（EP模式）的服务器操作。

配置环境变量

步骤1 以**root**用户登录服务器。

步骤2 执行**vi ~/.bashrc**命令。

步骤3 将表5-4中的环境变量加入“.bashrc”文件。

表 5-4 环境变量

使用场景	环境变量
推理(运行环境)	install_path=/usr/local/Ascend #请用户根据实际路径修改 export PATH=${install_path}/toolbox/latest/Ascend-DMI/bin:${PATH} export LD_LIBRARY_PATH=/usr/local/dcmi:${install_path}/toolbox/latest/Ascend-DMI/lib64:${install_path}/nnrt/latest/acllib/lib64:/usr/local/Ascend/driver/lib64:${LD_LIBRARY_PATH} export PYTHONPATH=${install_path}/nnrt/latest/pyACL/python/site-packages/acl:$PYTHONPATH

步骤4 执行命令**source ~/.bashrc**，使环境变量生效。

----结束

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，操作如下：

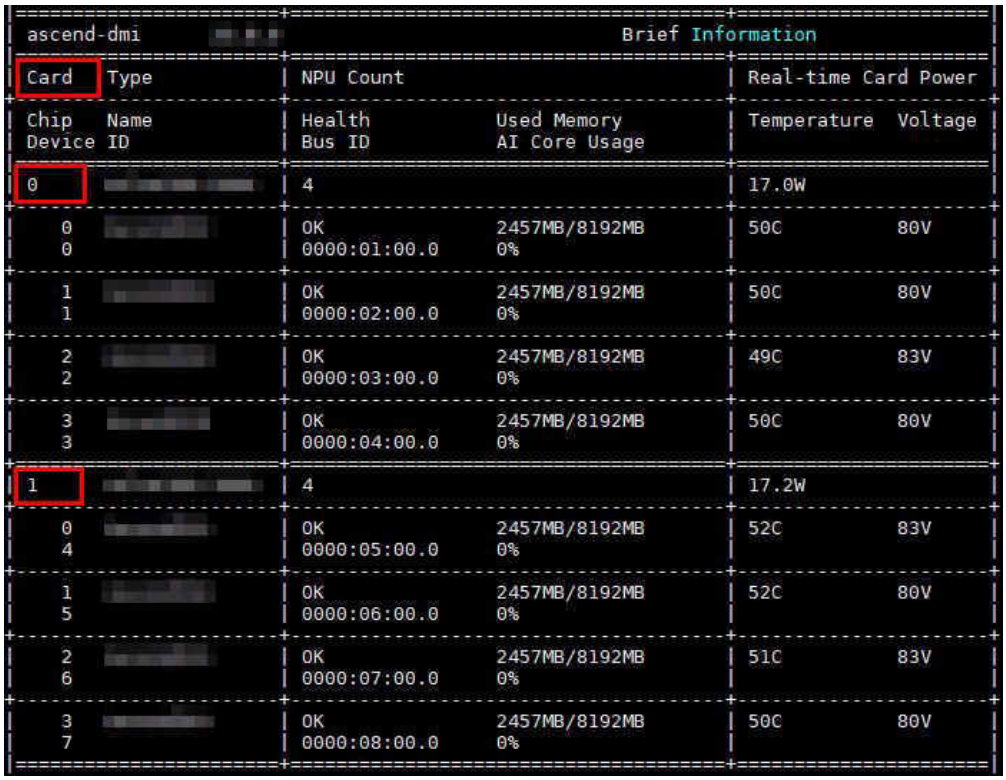
1. 以**root**用户登录服务器。
2. 执行**usermod -a -G HwHiAiUser *{username}***命令，加入属组。***{username}***为非root用户名，请用户自行替换。

安装后检查具体步骤如下。

步骤1 检查设备健康状态。

1. 执行**ascend-dmi info**在显示界面表格中的Card参数处获得环境中已安装的卡的号码，如图5-1中红框所示。

图 5-1 已安装卡号码



2. 执行**ascend-dmi -dg -c *{card-number}* -l 1**命令查询健康状态，***{card-number}***为环境中已安装的卡的号码，请用户根据实际安装情况替换。

   设备健康状态检查具体请参见**C.8 故障诊断**。

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**ascend-dmi -c**命令检查软硬件兼容性。

- 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，指令为：

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

软硬件兼容性检查具体请参见**C.9 软硬件版本兼容性测试**。

----结束

5.4 在容器安装

5.4.1 在宿主机安装实用工具包

前提条件

- 容器场景，需用户自行安装docker（版本要求大于等于18.03）。
- 已参照**2 准备硬件环境**完成驱动和固件的安装。

安装实用工具包

请使用root用户在宿主机上执行以下步骤。

步骤1 请参照5.3.2 安装推理软件中的步骤安装实用工具包toolbox。

步骤2 Ascend-docker-runtime已集成至实用工具包中，安装实用工具包后需重启docker，以确保Ascend-docker-runtime能够正常使用。

请执行以下命令，重启docker。

```
sudo systemctl restart docker
```

----结束

5.4.2 构建推理容器镜像

宿主机与容器操作系统兼容性关系

宿主机操作系统与容器操作系统的兼容性关系如表5-5所示（推荐Ubuntu 18.04作为容器OS）。

表 5-5 兼容性关系

架构	宿主机OS版本	宿主机kernel版本	容器OS版本
x86_64	Ubuntu 18.04.1	4.15.0-29-generic	Ubuntu 18.04
			CentOS 7.6 **说明** 需要在容器中安装driver。
	CentOS 7.6	3.10.0-957.el7.x86_64	CentOS 7.6
			Ubuntu 18.04
aarch64	Ubuntu 18.04.1	4.15.0-29-generic	Ubuntu 18.04
			CentOS 7.6 **说明** 需要在容器中安装driver。
	CentOS 7.6	4.14.0-115.el7a.0.1.aarch64	CentOS 7.6
			Ubuntu 18.04

说明

- 表5-5中若未注明“需要在容器中安装driver”，则不需要在容器中安装driver。
- 容器镜像OS场景只支持客户运行态业务，不支持开发态。
- 容器镜像OS目前支持的Ubuntu操作系统版本为18.04.1和18.04.4。

前提条件

- 请按照表5-6所示，获取软件包与打包镜像所需Dockerfile文件与脚本文件。
  软件包名称中*{version}*表示软件包版本，*{arch}*表示CPU架构。
- 用户环境拉取镜像需要联网，若未联网，请参见C.2 配置系统网络代理。
- 容器OS镜像可从Docker Hub上拉取，如拉取容器镜像Ubuntu 18.04：
  ```
  docker pull ubuntu:18.04
  ```
  由于不能从Docker Hub上获取到aarch64架构的CentOS 7.6镜像，若需使用，可从AscendHub上拉取。拉取成功后需要执行以下命令进行重命名：
  ```
  docker tag swr.cn-south-1.myhuaweicloud.com/public-ascendhub/centos:7.6.1810 arm64v8/centos:7
  ```

表 5-6 所需软件

软件包	说明	获取方法
Ascend-cann-nnrt_*{version}*_linux-*{arch}*.run	离线推理引擎包。	获取链接
A300-30*x*0-npu-driver_*{version}*_centos7.6-gcc4.8.5-*{arch}*.run	驱动安装包。根据表5-5，若需在容器中安装driver，请获取该软件包。	获取链接
Dockerfile	制作镜像需要。	用户根据业务自行准备
业务推理程序压缩包	编译后的业务程序压缩包，支持tgz格式。	
install.sh	业务推理程序的安装脚本。	
run.sh	业务推理程序的运行脚本。	

操作步骤

步骤1 以**root**用户登录服务器。

步骤2 创建软件包上传路径（以“/home/test”为例）。

```
mkdir -p /home/test
```

步骤3 将准备的软件包上传至服务器“/home/test”目录下。

- Ascend-cann-nnrt_*{version}*_linux-*{arch}*.run
- A300-30*x*0-npu-driver_*{version}*_centos7.6-gcc4.8.5-*{arch}*.run（若不需要在容器中安装driver请忽略）
- 业务推理程序压缩包

步骤4 准备Dockerfile文件。

toolbox软件包已为用户提供了多种场景下的昇腾基础镜像Dockerfile文件，请用户根据实际情况结合自身理解进行二次开发。

可将昇腾基础镜像Dockerfile文件等拷贝到“/home/test”目录下。命令示例如下：

cp /usr/local/Ascend/toolbox/latest/Ascend-Docker-Images/infer/*xxx***/Dockerfile** /home/test

其中*xxx*表示容器镜像OS（ubuntu、centos_x64或centos_arm64）。注意Dockerfile文件仅提供基础示例，需用户自行修改编写。例如需要指定用户ID、在容器中装driver等，需在基础镜像Dockerfile文件中添加相应内容，以下为在两种场景中的Dockerfile文件编写示例。

- 不需要在容器中安装driver，内容以Ubuntu 18.04操作系统为例，详细介绍请参见**编写示例**。

```
#拉取容器镜像ubuntu:18.04
FROM ubuntu:18.04

#指定离线推理引擎包与用户ID
ARG NNRT_PKG
ARG ASCEND_BASE=/usr/local/Ascend
ARG HWHIAIUSER_UID
ARG HWHIAIUSER_GID

#环境变量
ENV LD_LIBRARY_PATH=\
$LD_LIBRARY_PATH:\
$ASCEND_BASE/driver/lib64:\
$ASCEND_BASE/add-ons:\
$ASCEND_BASE/nnrt/latest/acllib/lib64

#指定工作容器主目录与拷贝安装包
WORKDIR /root
COPY $NNRT_PKG .

#安装离线推理引擎包
RUN umask 0022 && \
    if [ "$(uname -m)" = "aarch64" ] && [ ! -d "/lib64" ]; \
    then \
        mkdir /lib64 && ln -sf /lib/ld-linux-aarch64.so.1 /lib64/ld-linux-aarch64.so.1; \
    fi && \
    groupadd HwHiAiUser && \
    useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser && \
    groupmod -g $HWHIAIUSER_GID HwHiAiUser && \
    usermod -u $HWHIAIUSER_UID HwHiAiUser && \
    chmod +x ${NNRT_PKG} && \
    ./${NNRT_PKG} --quiet --install && \
    rm ${NNRT_PKG}

#拷贝业务程序压缩包与脚本文件
ARG DIST_PKG
COPY $DIST_PKG .
COPY install.sh .
COPY run.sh /usr/local/bin/

#执行安装脚本
RUN chmod +x /usr/local/bin/run.sh && \
sh install.sh && \
rm $DIST_PKG && \
rm install.sh

#容器启动时默认执行的程序
CMD run.sh
```

- 需要在容器中安装driver，内容以CentOS aarch64 7.6为例，详细介绍请参见**编写示例**。

```
FROM arm64v8/centos:7

#指定离线推理引擎包、驱动包与用户ID

ARG NNRT_PKG
```



```
ARG DRIVER_PKG
ARG ASCEND_BASE=/usr/local/Ascend
ARG HWHIAIUSER_UID
ARG HWHIAIUSER_GID

#环境变量
ENV LD_LIBRARY_PATH=\
$LD_LIBRARY_PATH:\
$ASCEND_BASE/driver/lib64:\
$ASCEND_BASE/add-ons:\
$ASCEND_BASE/nnrt/latest/acllib/lib64

#指定工作容器主目录与拷贝安装包
WORKDIR /root
COPY $NNRT_PKG .
COPY $DRIVER_PKG .

#安装离线推理引擎包、驱动包
RUN umask 0022 && \
    groupadd HwHiAiUser && \
    useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser && \
    groupmod -g $HWHIAIUSER_GID HwHiAiUser && \
    usermod -u $HWHIAIUSER_UID HwHiAiUser && \
    chmod +x ${DRIVER_PKG} && \
    chmod +x ${NNRT_PKG} && \
    ./${DRIVER_PKG} --docker && \
    ./${NNRT_PKG} --quiet --install && \
    rm ${DRIVER_PKG} && \
    rm ${NNRT_PKG}

#拷贝业务程序压缩包与脚本文件
ARG DIST_PKG
COPY $DIST_PKG .
COPY install.sh .
COPY run.sh /usr/local/bin/

#执行安装脚本
RUN chmod +x /usr/local/bin/run.sh && \
sh install.sh && \
rm $DIST_PKG && \
rm install.sh

#容器启动时默认执行的程序
CMD run.sh
```

说明

- 本文档所给示例都是基于宿主机的驱动运行用户为HwHiAiUser，若用户指定其他用户作为驱动的运行用户，请根据实际情况进行修改。
- 在创建Dockerfile文件后，执行以下命令修改Dockerfile文件权限。

  chmod 600 Dockerfile

步骤5 准备“install.sh”脚本与“run.sh”脚本文件。

1. 进入软件包上传路径（“/home/test”），执行以下命令创建脚本文件（以install.sh为例）。

   vi install.sh

2. 写入内容可参见**编写示例**。执行**:wq**命令保存。

步骤6 进入软件包所在目录，执行以下命令（此命令根据本文档编写示例给出，用户需根据实际情况修改），构建容器镜像。

docker build -t *image-name* **--build-arg NNRT_PKG=***nnrt-name* **--build-arg DRIVER_PKG=***drivepackage-name* **--build-arg HWHIAIUSER_UID=***uid* **--build-arg HWHIAIUSER_GID=***gid* **--build-arg DIST_PKG=***distpackage-name* **.**

其中**--build-arg DRIVER_PKG=***drivepackage-name*为指定驱动包，若不需要在容器中安装driver，请自行删除。注意不要遗漏命令结尾的“.”，命令解释如表5-7所示。

表 5-7 命令参数说明

参数	说明
image-name	镜像名称与标签，请用户根据实际情况更换。
--build-arg	指定dockerfile文件内的参数。
NNRT_PKG	*nnrt-name*为离线推理引擎包名称，注意不要遗漏文件后缀，请用户自行更换。
DRIVER_PKG	*drivepackage-name*为驱动安装包名称，注意不要遗漏文件后缀，请用户自行更换。
HWHIAIUSER_UID	*uid*为HwHiAiUser的uid，请用户根据host侧HwHiAiUser用户的uid进行替换。
HWHIAIUSER_GID	*gid*为HwHiAiUser的gid，请用户根据host侧HwHiAiUser用户的gid进行替换。
DIST_PKG	*distpackage-name*为业务推理程序压缩包名称，注意不要遗漏文件后缀，请用户自行更换。

说明

可执行如下命令查询host侧HwHiAiUser用户的uid、gid。

id HwHiAiUser

当出现“Successfully built xxx”表示镜像构建成功。

步骤7 构建完成后，执行以下命令查看镜像信息。

docker images

显示示例：

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
workload-image      v1.0                1372d2961ed2        About an hour ago   249MB
```

----结束

编写示例

Dockerfile编写有以下注意事项：

拉取基础镜像的语句替换：

```
FROM ubuntu:18.04
```

为获取镜像“ubuntu：18.04”。根据容器需要的操作系统的不同，用户需要参考表5-8替换。

表 5-8 示例

容器OS版本	镜像示例
CentOS 7.6 aarch64	FROM arm64v8/centos:7
CentOS 7.6 x86_64	FROM centos:7.6.1810
Ubuntu 18.04 x86_64 Ubuntu 18.04 aarch64	FROM ubuntu:18.04

install.sh编写示例

```
#!/bin/bash

umask 0022
#创建日志相关文件
mkdir -p /usr/slog
mkdir -p /var/log/npu/slog/slogd
chown -Rf HwHiAiUser:HwHiAiUser  /usr/slog
chown -Rf HwHiAiUser:HwHiAiUser  /var/log/npu/slog

#进入容器工作目录、解压业务推理程序压缩包
cd /root
tar xf dist.tar
```

run.sh编写示例

```
#!/bin/bash

#验证npu-smi工具的安装
npu-smi info

#启动slogd守护进程
su HwHiAiUser --command "/usr/local/Ascend/driver/tools/slogd &"
sleep 1
ps -ef | grep -v grep | grep "tools/slogd"

#进入业务推理程序的可执行文件所在目录
cd /root/dist
#运行可执行文件
./main
```

5.4.3 部署推理容器

前提条件

- 容器镜像已构建成功。
- 确保打包环境中已安装Docker程序。
- 在拉起容器前请确保宿主机已安装推理卡驱动、固件以及实用工具包。

操作步骤

根据表5-9启动容器镜像，验证npu-smi工具的安装、slogd守护进程的启动与业务程序的运行。

表 5-9 执行命令

场景	执行命令
不在容器中安装driver	**docker run -it -e ASCEND_VISIBLE_DEVICES=***xxx image-name*
在容器中安装driver	**docker run -it -e ASCEND_VISIBLE_DEVICES=***xxx* **-e ASCEND_RUNTIME_OPTIONS=***NODRV* **-v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi** *image-name*

表 5-10 参数解释

参数	参数说明
-e ASCEND_VISIBLE_DEVICES=*xxx*	使用ASCEND_VISIBLE_DEVICES环境变量指定被挂载至容器中的NPU设备，使用设备序号指定设备，支持单个和范围指定且支持混用。例如： 1. -e ASCEND_VISIBLE_DEVICES=0 表示将0号设备（/dev/davinci0）挂载入容器中。 2. -e ASCEND_VISIBLE_DEVICES=1,3 表示将1、3号设备挂载入容器中。 3. -e ASCEND_VISIBLE_DEVICES=0-2 表示将0号至2号设备（包含0号和2号）挂载入容器中，效果同 -e ASCEND_VISIBLE_DEVICES=0,1,2 4. ASCEND_VISIBLE_DEVICES=0-2,4 表示将0号至2号以及4号设备挂载入容器，效果同 -e ASCEND_VISIBILE_DEVICES=0,1,2,4
image-name	镜像名称与标签，请用户根据实际情况更换。
-e ASCEND_RUNTIME_OPTIONS=*NODRV*	容器中将仅挂载NPU设备和管理设备（如/dev/davinci0、/dev/davinci_manager、/dev/hisi_hdc、/dev/devmm_svm），为在容器中安装驱动的场景提供支持。
-v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi	将npu-smi工具挂载到容器，请根据实际情况修改。

若回显以下类似内容，说明npu-smi工具可用以及slogd守护进程启动。

```
+------------------------------------------------------------------------------+
| npu-smi 20.0.0                   Version: 1.73.T5.0.B050                     |
+------------------+----------------+------------------------------------------+
| NPU     Name     | Health         | Power(W)          Temp(C)                |
| Chip    Device   | Bus-Id         | AICore(%)         Memory-Usage(MB)       |
+==================+================+==========================================+
```



```
| 2048    310       | OK            | 12.8             41                |
| 0       0         | 0000:81:00.0  | 0                2457 / 8192       |
+===================+================+=======================================+
HwHiAiU+     24      1  0 13:08 ?        00:00:00 /usr/local/Ascend/driver/tools/slogd
```

对于Atlas 500 Pro 智能边缘服务器（型号 3000），如果用户需要在FusionDirector平台批量部署容器镜像，可以参考《**MindX Edge 应用部署与模型更新指南**》。

说明

- 以上命令示例会默认执行业务程序，若用户需要直接进入容器，请在以上命令的末尾添加 /bin/bash，示例如下。

  docker run -it -e ASCEND_VISIBLE_DEVICES=*xxx image-name* **/bin/bash**

- 以上命令仅列出最少挂载内容，如有需要，请自行添加。
- AscendDocker Runtime默认挂载的内容如**E.2 AscendDocker Runtime默认挂载内容**所示。若用户在宿主机安装驱动时采用的是指定路径方式，需自行挂载**表E-2**中的目录和文件。示例如下：

  docker run -it -e ASCEND_VISIBLE_DEVICES=*xxx* **-v /var/log/npu/conf/slog/slog.conf:/var/log/npu/conf/slog/slog.conf -v** ***${install_path}*****/driver:***${install_path}***/driver -v** ***${install_path}*****/add-ons:***${install_path}***/add-ons -v /usr/local/dcmi:/usr/local/dcmi -v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi** *image-name*

  其中${install_path}为驱动安装路径。

6 安装运行环境（训练）

6.1 安装前必读

运行环境是实际训练模型的环境，运行环境要求配置昇腾AI设备（如训练服务器或训练卡）。

运行环境包含物理机和容器两种场景。

运行场景	安装流程
物理机场景	1. **6.2 准备软件包** 2. **6.3.2 安装OS依赖** 3. **6.3.3 安装训练软件** 4. **6.3.4 安装深度学习框架** 5. **6.3.5 安装python版本的proto** 6. **6.3.6 配置device的网卡IP**
容器场景	1. **6.2 准备软件包** 2. **6.4.1 在宿主机安装实用工具包** 3. **6.4.2 构建容器镜像** 4. **6.4.3 部署训练容器**

6.2 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，各软件包版本号需要保持一致。

表 6-1 CANN 软件包

名称	软件包	说明	获取链接
训练软件包	Ascend-cann-nnae_*{version}*_linux-*{arch}*.run	包含FWK库Fwklib和算子库OPP组件，用于训练模型。 请根据CPU架构（x86_64、aarch64）获取对应软件包	**获取链接**
实用工具包	• V100R020C10版本：Ascend-mindstudio-toolbox_*{version}*_linux-*{arch}*.run • V100R020C10SPC100及以上版本：Ascend-cann-toolbox_*{version}*_linux-*{arch}*.run	主要包含AICPU算子，用于训练模型调用算子。 请根据CPU架构（x86_64、aarch64）获取对应软件包	**获取链接**
框架插件包	• V100R020C10版本：Ascend-fwk-tfplugin_*{version}*_linux-*{arch}*.run • V100R020C10SPC100及以上版本：Ascend-cann-tfplugin_*{version}*_linux-*{arch}*.run	包含框架适配插件Plugin，用于对接上层框架，如Tensorflow的适配插件。	**获取链接**

说明

*{version}*表示软件版本号，*{arch}*表示CPU架构。

检查软件包的完整性

说明

若您获取的是社区版本，可不进行此操作。

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。

6.3 在物理机安装

6.3.1 准备安装及运行用户

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。

创建安装及运行用户

- 安装用户：实际安装软件包的用户。
  - 若使用root用户安装，因要求使用非root用户运行，所以安装前需要先准备运行用户。
  - 若使用非root用户安装，则安装及运行用户必须相同。
- 运行用户：实际运行推理业务或执行训练的用户。
- **使用root用户安装**
  - （推荐）如果创建的运行用户是HwHiAiUser，其为CANN软件包的默认运行用户，在安装时无需指定该运行用户。
  - 如果创建的运行用户是非HwHiAiUser，安装CANN软件包时需要指定该运行用户。
- **使用非root用户安装**
  - 已有非root用户，则无需再次创建。
  - 新建非root用户，请参见如下方法创建。

须知

- CANN软件包如果使用非root用户安装，用户所属的属组必须和Driver运行用户所属属组相同；如果不同，请用户自行添加到Driver运行用户属组。
- 创建的运行用户不建议为root用户属组，原因是：权限控制可能存在安全风险。

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
   ```
   以创建运行用户HwHiAiUser为例：
   ```
   groupadd HwHiAiUser
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser -s /bin/bash
   ```
2. 设置非root用户密码。
   ```
   passwd username
   ```
   示例如下：
   ```
   passwd HwHiAiUser
   ```

说明

- 创建完HwHiAiUser用户后， 请勿关闭该用户的登录认证功能。
- 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C.7 设置用户有效期。

6.3.2 安装 OS 依赖

6.3.2.1 Ubuntu aarch64 系统

环境要求

运行环境需要安装以下软件或依赖。

表 6-2 依赖信息

类别	版本要求	说明
Python	3.7.5	-
cmake	3.5.1+	编译环境依赖。
make	-	
gcc	7.3.0及以上	
g++	7.3.0及以上	

类别	版本要求	说明
zlib1g zlib1g-dev libbz2-dev libsqlite3-dev openssl libssl-dev libxslt1-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3 liblapack-dev liblapack3 libopenblas-dev	无版本要求，安装的版本以操作系统自带的源为准。	
numpy	>=1.13.3	CANN软件包python依赖。
decorator	>=4.4.0	
sympy	1.4	
cffi	1.12.3	
pyyaml pathlib2 protobuf scipy requests psutil	无版本要求，安装的版本以pip源为准。	

检查源

深度学习引擎包、实用工具包和框架插件包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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

配置安装用户权限

当用户使用非root用户安装时，需要操作该章节；当用户使用root用户安装时，仅需执行步骤1。

深度学习引擎包、实用工具包和框架插件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo apt-get**权限，请以root用户执行如下操作。

1. 安装sudo，使用如下命令安装。
   ```
   apt-get install sudo
   ```
2. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
3. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   “username”为执行安装脚本的普通用户名。

   说明

   请确保“/etc/sudoers”文件的最后一行为“#includedir /etc/sudoers.d”，如果没有该信息，请手动添加。
4. 添加完成后，执行**:wq!**保存文件。
5. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```

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 libbz2-dev| grep libbz2-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 libxslt1-dev| grep libxslt1-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
dpkg -l libopenblas-dev| grep libopenblas-dev| grep ii
```

若分别返回如下信息则说明已经安装，进入下一步。

```
gcc (Ubuntu/Linaro 7.5.0-3ubuntu1~18.04) 7.5.0
g++ (Ubuntu/Linaro 7.5.0-3ubuntu1~18.04) 7.5.0
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
libbz2-dev:arm64 1.0.6-8.1ubuntu0.2 arm64       high-quality block-sorting file compressor 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
libxslt1-dev:arm64 1.1.29-5ubuntu0.2 arm64        XSLT 1.0 processing library - development kit
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
libopenblas-dev:arm64 0.2.20+ds-4  arm64    Optimized BLAS (linear algebra) library (development files)
```

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

⚠ 注意

- 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python3.7.5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 libbz2-dev openssl libsqlite3-dev libssl-dev
libxslt1-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3 libopenblas-dev
```

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```


如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。

```
cp: overwrite 'libpython3.7m.so.1.0'?
```

如果环境上没有/usr/lib64，则复制到/usr/lib目录：

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
```

libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。

5. 执行如下命令设置软链接：

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。

```
sudo rm -rf  /usr/bin/python3.7
sudo rm -rf  /usr/bin/python3.7.5
sudo rm -rf  /usr/bin/pip3.7
sudo rm -rf  /usr/bin/pip3.7.5
```

6. 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。

```
python3.7.5 --version
pip3.7.5  --version
```

步骤3 安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install numpy
pip3.7.5 install decorator
pip3.7.5 install sympy==1.4
pip3.7.5 install cffi==1.12.3
pip3.7.5 install pyyaml
pip3.7.5 install pathlib2
pip3.7.5 install psutil
pip3.7.5 install protobuf
pip3.7.5 install scipy
pip3.7.5 install requests
```

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**D.5 pip3.7.5 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

6.3.2.2 EulerOS aarch64 系统

环境要求

运行环境需要安装以下软件或依赖。

表 6-3 依赖信息

类别	版本要求	说明
Python	3.7.5	-
cmake	3.5.1+	编译环境依赖。
make	-	
gcc	7.3.0及以上	
gcc-c++	7.3.0及以上	
zlib-devel libffi-devel openssl-devel sqlite-devel unzip pciutils net-tools lapack lapack-devel blas blas-devel gcc-gfortran	无版本要求，安装的版本以操作系统自带的源为准。	
numpy	>=1.13.3	CANN软件包python依赖。
decorator	>=4.4.0	
sympy	1.4	
cffi	1.12.3	
pyyaml pathlib2 protobuf scipy requests psutil	无版本要求，安装的版本以pip源为准。	

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 repolist
```

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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

配置安装用户权限

当用户使用非root用户安装时，需要操作该章节；当用户使用root用户安装时，仅需执行步骤**1**。

深度学习引擎包、实用工具包和框架插件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo yum**权限，请以root用户执行如下操作。

1. 安装sudo，使用如下命令安装。
   ```
   yum install sudo
   ```
2. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
3. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   “username”为执行安装脚本的普通用户名。

   说明

   请确保“/etc/sudoers”文件的最后一行为“#includedir /etc/sudoers.d”，如果没有该信息，请手动添加。
4. 添加完成后，执行**:wq!**保存文件。
5. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```

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
rpm -qa |grep zlib-devel
rpm -qa |grep libffi-devel
rpm -qa |grep openssl-devel
rpm -qa |grep sqlite-devel
rpm -qa |grep unzip
rpm -qa |grep pciutils
rpm -qa |grep net-tools
rpm -qa |grep lapack
rpm -qa |grep lapack-devel
rpm -qa |grep blas
rpm -qa |grep blas-devel
rpm -qa |grep gcc-gfortran
```

若分别返回如下信息则说明已经安装，进入下一步。

```
gcc (GCC) 7.3.0
g++ (GCC) 7.3.0
GNU Make 4.2.1
cmake version 3.12.1
zlib-devel-1.2.11-14.eulerosv2r8.aarch64
```



```
libffi-devel-3.1-18.h3.eulerosv2r8.aarch64
openssl-devel-1.1.1-3.h7.eulerosv2r8.aarch64
sqlite-devel-3.24.0-2.h4.eulerosv2r8.aarch64
unzip-6.0-40.eulerosv2r8.aarch64
pciutils-3.6.2-1.eulerosv2r8.aarch64
net-tools-2.0-0.52.20160912git.eulerosv2r8.aarch64
lapack-3.8.0-10.eulerosv2r8.aarch64
lapack-devel-3.8.0-10.eulerosv2r8.aarch64
blas-3.8.0-10.eulerosv2r8.aarch64
blas-devel-3.8.0-10.eulerosv2r8.aarch64
gcc-gfortran-7.3.0-20190804.h22.eulerosv2r8.aarch64
```

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

⚠ 注意

- 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python3.7.5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 zlib-devel libffi-devel openssl-devel sqlite-devel unzip pciutils net-tools lapack lapack-devel blas blas-devel gcc-gfortran
```

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python --version**、**pip --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 执行如下命令复制libpython3.7m.so.1.0动态库：
   a. 由于yum工具强依赖于系统自带的libpython3.7m.so.1.0文件，因此需要备份系统自带的libpython3.7m.so.1.0文件。请执行如下命令：
      ```
      cd /usr/lib64
      cp libpython3.7m.so.1.0 libpython3.7m.so.1.0.bak
      ```


如果环境上没有/usr/lib64，则以实际路径为准。

b. 将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。

```
cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
```

如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。

```
cp: overwrite 'libpython3.7m.so.1.0'?
```

如果环境上没有/usr/lib64，则复制到/usr/lib目录：

```
cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
```

libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。

c. 当需要使用yum工具时，需要执行如下命令将a备份的文件libpython3.7m.so.1.0.bak恢复。

```
mv libpython3.7m.so.1.0.bak libpython3.7m.so.1.0
```

5. 执行如下命令设置软链接：

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。

```
sudo rm -rf  /usr/bin/python3.7
sudo rm -rf  /usr/bin/python3.7.5
sudo rm -rf  /usr/bin/pip3.7
sudo rm -rf  /usr/bin/pip3.7.5
```

6. 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。

```
python3.7 --version
python3.7.5 --version
pip3.7 --version
pip3.7.5 --version
```

步骤3 安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install numpy
pip3.7.5 install decorator
pip3.7.5 install sympy==1.4
pip3.7.5 install cffi==1.12.3
pip3.7.5 install pyyaml
pip3.7.5 install pathlib2
pip3.7.5 install psutil
pip3.7.5 install protobuf
pip3.7.5 install scipy
pip3.7.5 install requests
```

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**D.5 pip3.7.5 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

6.3.2.3 CentOS aarch64 系统

环境要求

运行环境需要安装以下软件或依赖。

表 6-4 依赖信息

类别	版本限制	说明
Python	3.7.5	-
cmake	3.5.1+	编译环境依赖。
make	-	
gcc	软件源默认安装gcc版本为4.8.5。 **训练场景下**，要求7.3.0版本及以上gcc，安装过程可参考**C.5 安装7.3.0版本gcc**。	
gcc-c++		
unzip zlib-devel libffi-devel openssl-devel pciutils net-tools sqlite-devel blas-devel lapack-devel openblas-devel gcc-gfortran	无版本要求，安装的版本以操作系统自带的源为准。	
numpy	>=1.13.3	CANN软件包python依赖。
decorator	>=4.4.0	
sympy	1.4	
cffi	1.12.3	
pyyaml pathlib2 protobuf scipy requests psutil	无版本要求，安装的版本以pip源为准。	

检查源

深度学习引擎包、实用工具包和框架插件包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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

配置安装用户权限

当用户使用非root用户安装时，需要操作该章节；当用户使用root用户安装时，仅需执行步骤**1**。

CANN软件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo yum**权限，请以root用户执行如下操作。

1. 安装sudo，使用如下命令安装。
   ```
   yum install sudo
   ```
2. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
3. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   “username”为执行安装脚本的普通用户名。

   说明

   请确保“/etc/sudoers”文件的最后一行为“#includedir /etc/sudoers.d”，如果没有该信息，请手动添加。
4. 添加完成后，执行**:wq!**保存文件。
5. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```

配置最大线程数

1. 以root用户登录安装环境。
2. 配置环境变量，修改线程最大数，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。

```
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 blas-devel
rpm -qa |grep lapack-devel
rpm -qa |grep openblas-devel
rpm -qa |grep gcc-gfortran
```

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，请以实际情况为准）。

```
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 2.8.12.2
unzip-6.0-21.el7.aarch64
zlib-devel-1.2.7-18.el7.aarch64
libffi-devel-3.0.13-18.el7.aarch64
openssl-devel-1.0.2k-19.el7.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
blas-devel-3.4.2-8.el7.aarch64
lapack-devel-3.4.2-8.el7.aarch64
openblas-devel-0.3.3-2.el7.aarch64
gcc-gfortran-4.8.5-39.el7.aarch64
```

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

⚠ 注意

- 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python3.7.5环境，在此之后再安装sqlite-devel，则需要重新编译python环境。
- **训练场景下**，要求7.3.0版本及以上gcc，因此需要安装7.3.0版本gcc，安装过程可参考**C.5 安装7.3.0版本gcc**。

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

上述步骤中，安装openblas-devel如报No package libopenblas available。需要安装企业版linux的扩展包，安装命令如下：

```
sudo yum install epel-release
```

如果通过上述方式安装的cmake版本低于3.5.1，则请参见**C.3 安装3.5.2版本cmake**解决。

步骤2 检查系统是否安装python开发环境。

深度学习引擎包、实用工具包和框架插件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。
   ```
   cp: overwrite 'libpython3.7m.so.1.0'?
   ```
   如果环境上没有/usr/lib64，则复制到/usr/lib目录：
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```
   libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。
5. 执行如下命令设置软链接：
   ```
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。
   ```
   sudo rm -rf  /usr/bin/python3.7.5
   sudo rm -rf  /usr/bin/pip3.7.5
   sudo rm -rf  /usr/bin/python3.7
   sudo rm -rf  /usr/bin/pip3.7
   ```
6. 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。
   ```
   python3.7.5 --version
   pip3.7.5  --version
   ```

步骤3 安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install numpy==1.17.2
pip3.7.5 install decorator
pip3.7.5 install sympy==1.4
```



```
pip3.7.5 install cffi==1.12.3
pip3.7.5 install pyyaml
pip3.7.5 install pathlib2
pip3.7.5 install psutil
pip3.7.5 install protobuf
pip3.7.5 install scipy
pip3.7.5 install requests
```

- 如果安装numpy报错，请参考**D.4 pip3.7.5 install numpy报错**解决。
- 如果安装scipy报错，请参考**D.3 pip3.7.5 install scipy报错**解决。

----结束

6.3.2.4 Ubuntu x86_64 系统

环境要求

运行环境需要安装以下软件或依赖。

表 6-5 依赖信息

类别	版本限制	说明
Python	3.7.5	-
cmake	3.5.1+	编译环境依赖。
make	-	
gcc	7.3.0及以上	
g++	7.3.0及以上	
zlib1g zlib1g-dev libbz2-dev libsqlite3-dev libssl-dev libxslt1-dev libffi-dev unzip pciutils net-tools liblapack-dev liblapack3 libblas-dev gfortran libblas3	无版本要求，安装的版本以操作系统自带的源为准。	
numpy	>=1.13.3	CANN软件包python依赖。
decorator	>=4.4.0	

类别	版本限制	说明
sympy	1.4	
cffi	1.12.3	
pyyaml pathlib2 protobuf scipy requests psutil	无版本要求，安装的版本以pip源为准。	

检查源

深度学习引擎包、实用工具包和框架插件包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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

配置安装用户权限

当用户使用非root用户安装时，需要操作该章节；当用户使用root用户安装时，仅需执行步骤**1**。

深度学习引擎包、实用工具包和框架插件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo apt-get**权限，请以root用户执行如下操作。

1. 安装sudo，使用如下命令安装。
   ```
   apt-get install sudo
   ```
2. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
3. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   “username”为执行安装脚本的普通用户名。

   说明

   请确保“/etc/sudoers”文件的最后一行为“#includedir /etc/sudoers.d”，如果没有该信息，请手动添加。
4. 添加完成后，执行**:wq!**保存文件。

5. 执行以下命令取消“/etc/sudoers”文件的写权限：
```
chmod u-w /etc/sudoers
```

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 libbz2-dev| grep libbz2-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 libxslt1-dev| grep libxslt1-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
```

若分别返回如下信息则说明已经安装，进入下一步。
```
gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
g++ (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
GNU Make 4.1
cmake version 3.10.2
zlib1g:amd64   1:1.2.11.dfsg-0ubuntu2 amd64       compression library - runtime
zlib1g-dev:amd64  1:1.2.11.dfsg-0ubuntu2  amd64       compression library - development
libbz2-dev:amd64    1.0.6-8.1ubuntu0.2 amd64       high-quality block-sorting file compressor library - development
libsqlite3-dev:amd64    3.22.0-1ubuntu0.3 amd64       SQLite 3 development files
openssl        1.1.1-1ubuntu2.1~18.04.6 amd64       Secure Sockets Layer toolkit - cryptographic utility
libssl-dev:amd64    1.1.1-1ubuntu2.1~18.04.6 amd64       Secure Sockets Layer toolkit - development files
libxslt1-dev:amd64 1.1.29-5ubuntu0.2 amd64       XSLT 1.0 processing library - development kit
libffi-dev:amd64 3.2.1-8      amd64       Foreign Function Interface library (development files)
unzip          6.0-21ubuntu1 amd64       De-archiver for .zip files
pciutils       1:3.5.2-1ubuntu1 amd64       Linux PCI Utilities
net-tools      1.60+git20161116.90da8a0-1ubuntu1 amd64       NET-3 networking toolkit
```

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

⚠ 注意

- 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python3.7.5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 gcc g++ make cmake zlib1g zlib1g-dev libbz2-dev libsqlite3-dev libssl-dev libxslt1-dev libffi-dev unzip pciutils net-tools -y
```

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。


```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。
   ```
   cp: overwrite 'libpython3.7m.so.1.0'?
   ```
   如果环境上没有/usr/lib64，则复制到/usr/lib目录：
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```
   libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。
5. 执行如下命令设置软链接：
   ```
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。
   ```
   sudo rm -rf  /usr/bin/python3.7.5
   sudo rm -rf  /usr/bin/pip3.7.5
   sudo rm -rf  /usr/bin/python3.7
   sudo rm -rf  /usr/bin/pip3.7
   ```
6. 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。
   ```
   python3.7.5 --version
   pip3.7.5  --version
   ```

步骤3 安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install numpy
pip3.7.5 install decorator
pip3.7.5 install sympy==1.4
```



```
pip3.7.5 install cffi==1.12.3
pip3.7.5 install pyyaml
pip3.7.5 install pathlib2
pip3.7.5 install psutil
pip3.7.5 install protobuf
pip3.7.5 install scipy
pip3.7.5 install requests
```

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**D.5 pip3.7.5 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”**。

----结束

6.3.2.5 CentOS x86_64 系统

环境要求

运行环境需要安装以下软件或依赖。

表 6-6 依赖信息

类别	版本限制	说明
Python	3.7.5	-
cmake	3.5.1+	编译环境依赖。
make	-	
gcc	软件源默认安装gcc版本为4.8.5。 **训练场景下**，要求7.3.0版本及以上gcc，安装过程可参考**C.5 安装7.3.0版本gcc**。	
gcc-c++		
unzip zlib-devel libffi-devel openssl-devel pciutils net-tools sqlite-devel blas-devel lapack-devel openblas-devel gcc-gfortran	无版本要求，安装的版本以操作系统自带的源为准。	
numpy	>=1.13.3	CANN软件包python依赖。
decorator	>=4.4.0	
sympy	1.4	

类别	版本限制	说明
cffi	1.12.3	
pyyaml pathlib2 protobuf scipy requests psutil	无版本要求，安装的版本以pip源为准。	

检查源

深度学习引擎包、实用工具包和框架插件包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**）。如需配置网络代理，请参见**C.2 配置系统网络代理**。

配置安装用户权限

当用户使用非root用户安装时，需要操作该章节；当用户使用root用户安装时，仅需执行步骤**1**。

深度学习引擎包、实用工具包和框架插件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo yum**权限，请以root用户执行如下操作。

1. 安装sudo，使用如下命令安装。
   ```
   yum install sudo
   ```
2. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```
3. 在该文件“# User privilege specification”下面增加如下内容：
   ```
   username ALL=(ALL:ALL)  NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   “username”为执行安装脚本的普通用户名。

   说明

   请确保“/etc/sudoers”文件的最后一行为“#includedir /etc/sudoers.d”，如果没有该信息，请手动添加。
4. 添加完成后，执行**:wq!**保存文件。
5. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```


配置最大线程数

1. 以root用户登录安装环境。
2. 配置环境变量，修改线程最大数，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。

```
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 blas-devel
rpm -qa |grep lapack-devel
rpm -qa |grep openblas-devel
rpm -qa |grep gcc-gfortran
```

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，请以实际情况为准）。

```
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 2.8.12.2
unzip-6.0-21.el7.x86_64
zlib-devel-1.2.7-18.el7.x86_64
libffi-devel-3.0.13-18.el7.x86_64
openssl-devel-1.0.2k-19.el7.x86_64
pciutils-3.5.1-3.el7.x86_64
net-tools-2.0-0.25.20131004git.el7.x86_64
sqlite-devel-3.7.17-8.el7_7.1.x86_64
blas-devel-3.4.2-8.el7.x86_64
lapack-devel-3.4.2-8.el7.x86_64
openblas-devel-0.3.3-2.el7.x86_64
gcc-gfortran-4.8.5-39.el7.x86_64
```

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

⚠ 注意

- 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python3.7.5环境，在此之后再安装sqlite-devel，则需要重新编译python环境。
- **训练场景下**，要求7.3.0版本及以上gcc，因此需要安装7.3.0版本gcc，安装过程可参考**C.5 安装7.3.0版本gcc**。

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

上述步骤中，安装openblas-devel如报No package libopenblas available。需要安装企业版linux的扩展包，安装命令如下：

```
sudo yum install epel-release
```

如果通过上述方式安装的cmake版本低于3.5.1，则请参见**C.3 安装3.5.2版本cmake**解决。

步骤2 检查系统是否安装python开发环境。

深度学习引擎包、实用工具包和框架插件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装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. 查询/usr/lib64或/usr/lib下是否有libpython3.7m.so.1.0，若有则跳过此步骤或将系统自带的libpython3.7m.so.1.0文件备份后执行如下命令：

   将编译后的文件libpython3.7m.so.1.0复制到/usr/lib64。
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   如果出现如下显示，则输入y，表示覆盖系统自带的libpython3.7m.so.1.0文件。
   ```
   cp: overwrite 'libpython3.7m.so.1.0'?
   ```
   如果环境上没有/usr/lib64，则复制到/usr/lib目录：
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```


libpython3.7m.so.1.0文件所在路径请根据实际情况进行替换。

5. 执行如下命令设置软链接：

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

执行上述软链接时如果提示链接已经存在，则可以先执行如下命令删除原有链接然后重新执行。

```
sudo rm -rf  /usr/bin/python3.7.5
sudo rm -rf  /usr/bin/pip3.7.5
sudo rm -rf  /usr/bin/python3.7
sudo rm -rf  /usr/bin/pip3.7
```

6. 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。

```
python3.7.5 --version
pip3.7.5  --version
```

步骤3 安装前请先使用**pip3.7.5 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）。

- 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：**pip3.7.5 install numpy --user**，安装命令可在任意路径下执行。
- 为确保顺利安装，请在安装前配置好pip源，具体可参考**C.4 配置pip源**。

```
pip3.7.5 install numpy==1.17.2
pip3.7.5 install decorator
pip3.7.5 install sympy==1.4
pip3.7.5 install cffi==1.12.3
pip3.7.5 install pyyaml
pip3.7.5 install pathlib2
pip3.7.5 install psutil
pip3.7.5 install protobuf
pip3.7.5 install scipy
pip3.7.5 install requests
```

- 如果安装numpy报错，请参考**D.4 pip3.7.5 install numpy报错**解决。
- 如果安装scipy报错，请参考**D.3 pip3.7.5 install scipy报错**解决。

----结束

6.3.3 安装训练软件

前提条件

- 在安装软件前请确保安装环境已安装昇腾芯片驱动。
- 通过**6.2 准备软件包**章节获取深度学习引擎包nnae、实用工具包toolbox和框架插件包tfplugin。
- 请确保训练软件包的安装用户与**6.3.2 安装OS依赖**中安装依赖的用户保持一致。

安装步骤

获取并安装深度学习引擎包、实用工具包和框架插件包，各个软件包安装无顺序要求，且安装步骤相同，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 将深度学习引擎包、实用工具包和框架插件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。

步骤4 增加对软件包的可执行权限。

chmod +x *.run

说明

其中***.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。

./*.run --check

步骤6 （可选）创建软件安装路径。

- 如果用户想指定安装路径，则需要先创建安装路径。以安装路径“/home/work”为例， 用户先执行**mkdir -p /home/work**命令创建安装路径，再选择该路径进行软件安装。
- 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。

步骤7 执行以下命令安装软件。（以下命令支持**--install-for-all**和**--install-path=***<path>*等，具体参数说明请参见**E.1 参数说明**）

- 若使用非root用户安装，请执行以下命令。
  ./*.run --install
- 若使用root用户安装，请执行以下命令。
  - 若使用默认运行用户HwHiAiUser：
    ./*.run --install
  - 若用户需自行指定运行用户：
    ./*.run --install-username=*username* **--install-usergroup=***usergroup* **--install**
    其中**--install-username**和**--install-usergroup**用于指定运行用户。

注意

- 若软件包安装后提示重启生效，请执行**reboot**命令。
- 如果以root用户安装，**建议不要安装在非root用户目录下**，否则存在被非root用户替换root用户文件以达到提权目的的安全风险。
- 实用工具包toolbox在**X86系统**中安装的时候不支持--quiet选项
- 实用工具包toolbox在**X86系统**中安装的时候，如果出现如下情况，即询问是否进行热复位，则用户需输入n，安装完毕后重启os生效，当前版本只支持n选项。
  ```
  The installation of aicpu_kernels needs to restart the device to take effect, do you want to hot_reset the device? [y/n]
  ```

更多安装模式请参见**E.1 参数说明**。

安装完成后，若显示如下信息，则说明软件安装成功：


```
[INFO] xxx install success
[INFO] process end
```

*xxx*表示安装的实际软件包名。

步骤8 若上一步使用非root用户安装工具集，则需执行此步骤。

1. 进入实用工具包所在路径，执行以下命令解压实用工具包。
   ./Ascend-*xxx*-toolbox_*{version}*_linux-*{arch}*.run --noexec --extract=*{path}*
   其中*{path}*为解压路径，用户可自行指定。
2. 进入AICPU算子包所在路径。
   cd *{path}*/Tuscany/common
3. 增加对AICPU算子包的可执行权限。
   chmod +x Ascend910-aicpu_kernels-*{version}*.run
4. 安装AICPU算子包。
   请切换成root用户（也可使用sudo）执行以下安装命令。
   ./Ascend910-aicpu_kernels-*{version}*.run --full
   AICPU不支持指定安装路径，共用Driver的安装路径。

----结束

（参考）日志文件和软件包信息

下面以深度学习引擎包Ascend-cann-nnae_*{version}*_linux-*{arch}*.run为例介绍其对应的日志文件和软件包信息路径。

表 6-7 日志文件和软件包信息路径

项目	路径
安装详细日志路径	• V100R020C10版本：“${install_path}/nnae/latest/ascend_install.log” • V100R020C10SPC100及以上版本： root用户：“/var/log/ascend_seclog/ascend_nnae_install.log” 非root用户：“${HOME}/var/log/ascend_seclog/ascend_nnae_install.log”
安装后软件包版本、CPU架构、GCC版本和安装路径等信息的记录路径	“${install_path}/nnae/latest/ascend_nnae_install.info”

说明

${install_path}为软件包的安装路径。

6.3.4 安装深度学习框架

本文档以TensorFlow为例介绍安装深度学习框架的步骤，如果用户要使用MindSpore框架，请登录**https://www.mindspore.cn/install**获取安装MindSpore框架的方法。

如果用户要使用PyTorch框架，请参见《**FrameworkPTAdapter PyTorch网络模型移植&训练指南**》的“安装PyTorch框架”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。
  - 由于tensorflow依赖grpcio，建议在安装tensorflow之前指定版本安装grpcio，否则可能导致安装失败。

步骤1 编译安装HDF5。

1. 使用wget下载HDF5源码包，可以下载到安装环境的任意目录，命令为：
   ```
   wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz --no-check-certificate
   ```
2. 进入下载后的目录，解压源码包，命令为：
   ```
   tar -zxvf hdf5-1.10.5.tar.gz
   ```
   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
   ```

步骤2 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
   ```

步骤3 安装h5py。

root用户下安装h5py依赖包命令如下。
```
pip3.7 install Cython
```

安装h5py命令如下：

```
pip3.7 install h5py==2.8.0
```

步骤4 安装grpcio。

root用户下安装grpcio命令如下。

```
pip3.7 install grpcio==1.32.0
```

----结束

安装 Tensorflow 1.15.0

需要安装Tensorflow 1.15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，具体步骤请参考**https://www.tensorflow.org/install/pip?lang=python3**。需要注意tensorflow官网提供的指导描述有错误，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。即官网的“Tensorflow==1.15：仅支持 CPU

的版本”需要修改成“Tensorflow-cpu==1.15：仅支持 CPU 的版本”。另外，官网描述的安装命令“pip3 install --user --upgrade tensorflow”需要在root更改为“pip3.7 install Tensorflow-cpu==1.15”，非root用户下更改为“pip3.7 install Tensorflow-cpu==1.15 --user”。

- 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gcc7.3.0编译器编译tensorflow1.15.0，编译步骤参考官网https://www.tensorflow.org/install/source。特别注意点如下：

在下载完tensorflow tag v1.15.0源码后执行需要如下步骤。

步骤1 下载“nsync-1.22.0.tar.gz”源码包。

1. 进入tensorflow tag v1.15.0源码目录，打开“tensorflow/workspace.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”文件夹和“pax_global_header”文件。
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。

将解压出的“nsync-1.22.0”文件夹和“pax_global_header”压缩为一个新的“nsync-1.22.0.tar.gz”源码包，保存（假如保存在“/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。

sha256sum /tmp/nsync-1.22.0.tar.gz

执行如上命令后得到sha256sum校验码（一串数字和字母的组合）

步骤5 修改sha256sum校验码和urls。

进入tensorflow tag v1.15.0源码目录，打开“tensorflow/workspace.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=”后面的列表第一行，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 = [
          "file:///tmp/nsync-1.22.0.tar.gz ",          "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",
      ],
  )
```

步骤6 继续从官方的“配置build”（**https://www.tensorflow.org/install/source**）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（**https://www.tensorflow.org/install/source**）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root用户执行如下命令安装Tensorflow1.15：

```
pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
```

非root用户执行如下命令：

```
pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user
```

----结束

6.3.5 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”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**pip3.7 -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 卸载protobuf。

pip3.7 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服务器任一目录并进行解压（**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EulerOS时，安装命令如下：

   ```
   yum install autoconf automake libtool curl make gcc-c++ unzip libffi-devel -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，可以通过

   ./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

说明

若执行**ldconfig**提示“ldconfig: /lib64/libpython3.7m.so.1.0 is not a symbolic link”，可尝试以下操作解决：

mv libpython3.7m.so.1.0 libpython3.7m.so.1.0.py

ln -s libpython3.7m.so.1.0.py libpython3.7m.so.1.0

protobuf安装完成后，会在**3. 配置安装路径**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**3. 配置安装路径**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**3. 配置安装路径**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。

   python3.7 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**python3.7 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。

   python3.7 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。

   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**pip3.7 -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。

   arch为系统架构类型，还可以是x86_64。

4. 在运行脚本中增加环境变量的设置：

   export LD_LIBRARY_PATH=/protobuf/lib

   "**/protobuf**"为**3. 配置安装路径**中用户配置的安装路径，默认安装路径为"**/usr/local**"。

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**3. 配置安装路径**中用户配置的安装路径。

----结束

6.3.6 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 HCCN Tool 接口参考》**。

Atlas 800 训练服务器、Atlas 900 AI集群场景

- 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。IP地址需要根据实际情况修改。

```
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：

  不需要限制网段，所有网卡设置成相同网段即可。

Atlas 300T 训练卡场景

Atlas 300T 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。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

```
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，下文所用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。
- 对于集群场景，各AI Server的device处于同一网段即可。

6.3.7 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，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 添加环境变量。

1. 以**root**用户登录服务器。
2. 执行**vi ~/.bashrc**命令。
3. 将表6-8中的环境变量加入“.bashrc”文件。

表 6-8 环境变量

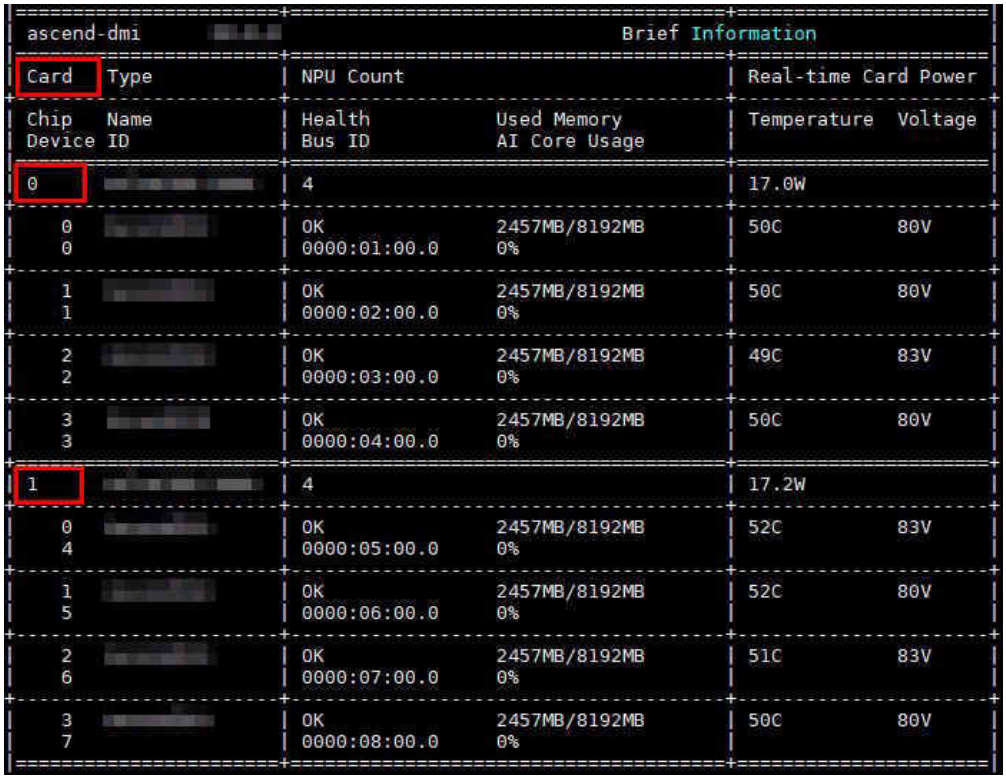
使用场景	环境变量
训练（运行环境）	install_path=/usr/local/Ascend #请用户根据实际安装路径修改 export PATH=${install_path}/toolbox/latest/Ascend-DMI/bin:${PATH} export LD_LIBRARY_PATH=/usr/local/dcmi:${install_path}/toolbox/latest/Ascend-DMI/lib64:${install_path}/nnae/latest/fwkacllib/lib64:/usr/local/Ascend/driver/lib64/common:/usr/local/Ascend/driver/lib64/driver:${LD_LIBRARY_PATH}

4. 执行命令**source ~/.bashrc**，使环境变量生效。

步骤2 检查设备健康状态。

1. 执行**ascend-dmi info**在显示界面表格中的Card参数处获得环境中已安装的卡的号码，如图6-2中红框所示。

图 6-2 已安装卡号码



2. 执行**ascend-dmi -dg -c {card-number} -l 1**命令查询健康状态，**{card-number}**为环境中已安装的卡的号码，请用户根据实际安装情况替换。

   设备健康状态检查具体请参见**C.8 故障诊断**。

步骤3 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**ascend-dmi -c**命令检查软硬件兼容性。

- 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，指令为：

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

软硬件兼容性检查具体请参见**C.9 软硬件版本兼容性测试**。

----结束

6.4 在容器安装

6.4.1 在宿主机安装实用工具包

前提条件

- 容器场景，需用户自行安装docker（版本要求大于等于18.03）。
- 已参照**2 准备硬件环境**完成驱动和固件的安装。

安装实用工具包

请使用root用户在宿主机上执行以下步骤。

步骤1 请参照**6.3.3 安装训练软件**中的步骤安装实用工具包toolbox。

步骤2 Ascend-docker-runtime已集成至实用工具包中，安装实用工具包后需重启docker，以确保Ascend-docker-runtime能够正常使用。

请执行以下命令，重启docker。

```
sudo systemctl restart docker
```

----结束

6.4.2 构建容器镜像

宿主机与容器操作系统兼容性关系

宿主机操作系统与容器操作系统的兼容性关系如**表6-9**（推荐Ubuntu 18.04作为容器OS）。

表 6-9 兼容性关系

宿主机OS	容器镜像OS
Ubuntu 18.04 x86_64	Ubuntu 18.04 x86_64
	CentOS 7.6 x86_64
Ubuntu 18.04 aarch64	Ubuntu 18.04 aarch64
	CentOS 7.6 aarch64
CentOS 7.6 x86_64	Ubuntu 18.04 x86_64
	CentOS 7.6 x86_64
CentOS 7.6 aarch64	Ubuntu 18.04 aarch64
	CentOS 7.6 aarch64
Debian 9.9 x86_64	Ubuntu 18.04 x86_64
	CentOS 7.6 x86_64
	Debian 9.9 x86_64

说明

- 容器镜像OS场景只支持客户运行态业务，不支持开发态。
- 容器镜像OS目前支持的Ubuntu操作系统版本为18.04.1和18.04.4。

前提条件

- 请按照**表6-10**所示，获取对应操作系统的软件包与打包镜像所需Dockerfile文件与脚本文件。

深度学习加速引擎包及框架插件包名称中*{version}*表示版本号，*{arch}*表示操作系统。

- 用户环境拉取镜像需要联网，若未联网，请参见**C.2 配置系统网络代理**。
- 容器OS镜像可从Docker Hub上拉取，如拉取容器镜像Ubuntu 18.04：
  ```
  docker pull ubuntu:18.04
  ```
  由于不能从Docker Hub上获取到aarch64架构的CentOS 7.6镜像，若需使用，可从**AscendHub**上拉取。拉取成功后需要执行以下命令进行重命名：
  ```
  docker tag swr.cn-south-1.myhuaweicloud.com/public-ascendhub/centos:7.6.1810 arm64v8/centos:7
  ```

表 6-10 所需软件

软件包	说明	获取方法
Ascend-cann-nnae_*{version}*_linux-*{arch}*.run	深度学习加速引擎包	**获取链接**
Ascend-xxx-tfplugin_*{version}*_linux-*{arch}*.run	框架插件包	
tensorflow-1.15.0-cp37-cp37m-*{version}*.whl	tensorflow框架whl包	aarch64架构请参见**6.3.4 安装深度学习框架**自行编译tensorflow安装包。x86_64架构请忽略。
Dockerfile	制作镜像需要	用户根据业务自行准备。
ascend_install.info	软件包安装日志文件	从host拷贝“/etc/ascend_install.info”文件。 以实际路径为准。
version.info	driver包版本信息文件	从host拷贝“/usr/local/Ascend/driver/version.info”文件。 以实际路径为准。
libstdc++.so.6.0.24	动态库文件	仅当容器镜像OS为CentOS时需要准备libstdc++.so.6.0.24文件。 可以通过find命令查询libstdc++.so.6.0.24文件所在路径，然后从host拷贝。
prebuild.sh	执行训练运行环境安装准备工作，例如配置代理等。	用户根据业务自行准备。
install_ascend_pkgs.sh	昇腾软件包安装脚本	
postbuild.sh	清除不需要保留在容器中的安装包、脚本、代理配置等	

操作步骤

步骤1 以**root**用户登录服务器。

步骤2 创建软件包上传路径（以“/home/test”为例）。

mkdir -p /home/test

步骤3 将准备的软件包、深度学习框架（x86_64架构请忽略）、host昇腾软件包安装信息文件及driver包版本信息文件上传至服务器“/home/test”目录下。

- Ascend-cann-nnae_*{version}*_linux-*{arch}*.run
- Ascend-xxx-tfplugin_*{version}*_linux-*{arch}*.run
- tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl（x86_64架构请忽略）
- ascend_install.info
- version.info
- libstdc++.so.6.0.24（仅当容器镜像OS为CentOS时需要）

步骤4 准备Dockerfile、prebuild.sh、install_ascend_pkgs.sh和postbuild.sh文件。

toolbox软件包已为用户提供了多种场景下的昇腾基础镜像Dockerfile文件，请用户根据实际情况结合自身理解进行二次开发。

可将昇腾基础镜像Dockerfile文件等拷贝到“/home/test”目录下。命令示例如下：

cp /usr/local/Ascend/toolbox/latest/Ascend-Docker-Images/train/tf_*{os}_{arch}*/*文件名* /home/test

其中*{os}*表示容器镜像OS（ubuntu、centos或debian），*{arch}*表示CPU架构（arm64或x64），文件名为Dockerfile、prebuild.sh、install_ascend_pkgs.sh或postbuild.sh。注意各文件仅提供基础示例或提示，需用户自行修改编写。写入内容可参见**编写示例**。

说明

- 本文档所给示例都是基于宿主机的驱动运行用户为HwHiAiUser，若用户指定其他用户作为驱动的运行用户，请根据实际情况进行修改。
- 在创建Dockerfile等文件后，可自行修改Dockerfile等文件权限。
- 为获取镜像“ubuntu:18.04”，用户也可以通过执行**docker pull ubuntu:18.04**命令从Docker Hub拉取。
- 由于Dockerfile文件里有安装第三方软件包，如果第三方软件出现漏洞，请用户自行修复。

步骤5 进入软件包所在目录，执行以下命令，构建容器镜像。

docker build -t *image-name* **--build-arg http_proxy=http://***user:password@proxyserverip:port*

--build-arg https_proxy=http://*user:password@proxyserverip:port* **.**

其中user为用户在内部网络中的用户名，password为用户密码，proxyserverip为代理服务器的ip地址，port为端口。

命令解释如**表6-11**所示。

表 6-11 命令参数说明

参数	说明
image-name	镜像名称与标签，请用户根据实际情况更换。
--build-arg	指定dockerfile文件内的参数
test_train:v0.1	镜像名称与标签，请用户根据实际情况更换。
http_proxy、https_proxy	网络代理，用户自行配置。如果用户服务器能直接连接外部网络，则无需配置此选项。

当出现“Successfully built xxx”表示镜像构建成功，注意不要遗漏命令结尾的“.”。

步骤6 构建完成后，执行以下命令查看镜像信息。

docker images

显示示例：

```
REPOSITORY      TAG         IMAGE ID       CREATED          SIZE
test_train      v1.0        d82746acd7f0   27 minutes ago   749MB
```

----结束

编写示例

1. prebuild.sh编写示例

   a. Ubuntu操作系统

   ```
   #!/bin/bash
   #----------------------------------------------------------------------------
   # 请在此处使用bash语法编写脚本代码，执行安装准备工作，例如配置代理等
   # 本脚本将会在正式构建过程启动前被执行
   # 以下内容仅为示例，用户需根据服务器网络情况自行配置
   # 注：本脚本运行结束后不会被自动清除，若无需保留在镜像中请在postbuild.sh脚本中清除
   #----------------------------------------------------------------------------
   # dns代理配置，修改“/etc/resolv.conf”文件，在文件中加入如下粗体内容，用户需根据实际情况进行配置。
   tee /etc/resolv.conf <<- EOF
   nameserver 10.129.2.34
   EOF
   ```

   b. CentOS操作系统

   ```
   #!/bin/bash
   #----------------------------------------------------------------------------
   # 请在此处使用bash语法编写脚本代码，执行安装准备工作，例如配置代理等
   # 本脚本将会在正式构建过程启动前被执行
   #
   # 注：本脚本运行结束后不会被自动清除，若无需保留在镜像中请在postbuild.sh脚本中清除
   #----------------------------------------------------------------------------
   # https://bbs.huaweicloud.com/forum/thread-50114-1-1.html
   # 修改如下文件，解决centos镜像中安装numpy报错的问题。
   tee ./npy_math_internal.h.src.patch <<-EOF
   480c480
   < NPY_INPLACE @type@ npy_@kind@@c@(@type@ x, @type@ y)
   ---
   > NPY_INPLACE __attribute__((optimize("-O0"))) @type@ npy_@kind@@c@(@type@ x,
   ```



```
@type@ y)
EOF
```

2. install_ascend_pkgs.sh编写示例

```
#!/bin/bash
#----------------------------------------------------------------------------
# 请在此处使用bash语法编写脚本代码，安装昇腾软件包
#
# 注：本脚本运行结束后不会被自动清除，若无需保留在镜像中请在postbuild.sh脚本中清除
#----------------------------------------------------------------------------
umask 0022
# 构建之前把host上的/etc/ascend_install.info拷贝一份到当前目录
cp ascend_install.info /etc/
# 构建之前把host的/usr/local/Ascend/driver/version.info拷贝一份到当前目录
mkdir -p /usr/local/Ascend/driver/
mv version.info /usr/local/Ascend/driver/

# Ascend-cann-nnae_{version}_linux-aarch64.run
chmod +x Ascend-cann-nnae_{version}_linux-aarch64.run
./Ascend-cann-nnae_{version}_linux-aarch64.run --install-path=/usr/local/Ascend/ --install --quiet

# Ascend-xxx-tfplugin_{version}_linux-aarch64.run
chmod +x Ascend-xxx-tfplugin_{version}_linux-aarch64.run
./Ascend-xxx-tfplugin_{version}_linux-aarch64.run --install-path=/usr/local/Ascend/ --install --quiet

# 只为了安装nnae包，所以需要清理，容器启动时通过ascend docker挂载进来
rm -rf /usr/local/Ascend/driver/
```

3. postbuild.sh编写示例（ubuntu）

```
#!/bin/bash
#----------------------------------------------------------------------------
# 请在此处使用bash语法编写脚本代码，清除不需要保留在容器中的安装包、脚本、代理配置等
# 本脚本将会在正式构建过程结束后被执行
#
# 注：本脚本运行结束后会被自动清除，不会残留在镜像中；脚本所在位置和Working Dir位置为/root
#----------------------------------------------------------------------------
rm -f ascend_install.info
rm -f prebuild.sh
rm -f install_ascend_pkgs.sh
rm -f Dockerfile*
rm -f Ascend-cann-nnae_20.1.0.B001_linux-aarch64.run
rm -f Ascend-xxx-tfplugin_20.1.0.B001_linux-aarch64.run
rm -f tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl#若之前修改过dns设置，可在此处恢复为修改前状态
tee /etc/resolv.conf <<- EOF


options edns0

nameserver 8.8.8.8
nameserver 8.8.4.4
EOF
```

4. Dockerfile编写示例：

   Ubuntu aarch64系统Dockerfile示例。

```
FROM ubuntu:18.04

ARG TF_PKG=tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
ARG HOST_ASCEND_BASE=/usr/local/Ascend
ARG NNAE_PATH=/usr/local/Ascend/nnae/latest
ARG TF_PLUGIN_PATH=/usr/local/Ascend/tfplugin/latest
ARG INSTALL_ASCEND_PKGS_SH=install_ascend_pkgs.sh
ARG PREBUILD_SH=prebuild.sh
ARG POSTBUILD_SH=postbuild.sh
WORKDIR /tmp
COPY . ./

# 触发prebuild.sh
RUN bash -c "test -f $PREBUILD_SH && bash $PREBUILD_SH || true"
```



```
# 系统包
RUN apt update
RUN apt install --no-install-recommends python3.7 python3.7-dev -y
RUN apt install --no-install-recommends curl g++ pkg-config unzip -y
RUN apt install --no-install-recommends libblas3 liblapack3 liblapack-dev libblas-dev gfortran libhdf5-
dev libffi-dev \
                                    libicu60 libxml2 -y

# pip3.7
RUN curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py && \
    cd /tmp && \
    apt-get download python3-distutils && \
    dpkg-deb -x python3-distutils_*.deb / && \
    rm python3-distutils_*.deb && \
    cd - && \
    python3.7 get-pip.py && \
    rm get-pip.py

# HwHiAiUser
RUN groupadd HwHiAiUser && \
    useradd -g HwHiAiUser -m -d /home/HwHiAiUser HwHiAiUser

# python包
RUN pip3.7 install numpy && \
    pip3.7 install decorator && \
    pip3.7 install sympy==1.4 && \
    pip3.7 install cffi==1.12.3 && \
    pip3.7 install pyyaml && \
    pip3.7 install pathlib2 && \


    pip3.7 install protobuf && \
    pip3.7 install scipy && \
    pip3.7 install requests

# Ascend包
RUN bash $INSTALL_ASCEND_PKGS_SH

# TF安装
ENV LD_LIBRARY_PATH=\
/usr/lib/aarch64-linux-gnu/hdf5/serial:\
$HOST_ASCEND_BASE/add-ons:\
$NNAE_PATH/fwkacllib/lib64:\
$HOST_ASCEND_BASE/driver/lib64/common:\
$HOST_ASCEND_BASE/driver/lib64/driver:$LD_LIBRARY_PATH

RUN pip3.7 install $TF_PKG

# 环境变量
ENV GLOG_v=2
ENV TBE_IMPL_PATH=$NNAE_PATH/opp/op_impl/built-in/ai_core/tbe
ENV TF_PLUGIN_PKG=$TF_PLUGIN_PATH/tfplugin/python/site-packages
ENV FWK_PYTHON_PATH=$NNAE_PATH/fwkacllib/python/site-packages
ENV PATH=$NNAE_PATH/fwkacllib/ccec_compiler/bin:$PATH
ENV ASCEND_OPP_PATH=$NNAE_PATH/opp
ENV PYTHONPATH=\
$FWK_PYTHON_PATH:\
$FWK_PYTHON_PATH/auto_tune.egg:\
$FWK_PYTHON_PATH/schedule_search.egg:\
$TF_PLUGIN_PKG:\
$TBE_IMPL_PATH:\
$PYTHONPATH

# 创建/lib64/ld-linux-aarch64.so.1
RUN umask 0022 && \
    if [ ! -d "/lib64" ]; \
    then \
        mkdir /lib64 && ln -sf /lib/ld-linux-aarch64.so.1 /lib64/ld-linux-aarch64.so.1; \
    fi
```



```
# 触发postbuild.sh
RUN bash -c "test -f $POSTBUILD_SH && bash $POSTBUILD_SH || true" && \
rm $POSTBUILD_SH
```

6.4.3 部署训练容器

前提条件

- 确保打包环境中已安装Docker程序。
- 在拉起容器前请确保宿主机已安装昇腾芯片驱动、固件以及实用工具包。
- 如果是多台服务器集群训练，那么部署训练容器前请确保宿主机已设置NPU板卡IP地址，参见**6.3.6 配置device的网卡IP**。

部署镜像文件

注意

当宿主机环境为CentOS系统时，由于CentOS的安全模块selinux默认开启，会导致挂载到容器的本地目录没有执行权限，因此需要临时关闭selinux，命令为：su -c "setenforce 0"。完成相关业务后再重新开启selinux，命令为： su -c "setenforce 0"。

步骤1 执行如下命令查询host侧HwHiAiUser用户的uid、gid。

id HwHiAiUser

得到信息如下。

```
uid=1001(HwHiAiUser) gid=1001(HwHiAiUser) groups=1001(HwHiAiUser)
```

其中1001为示例显示，用户需根据实际情况替换。

步骤2 如果用户需要在容器内使用profiling工具，需要在host侧使用HwHiAiUser用户（该用户需与容器内运行用户保持一致）创建“/home/HwHiAiUser/profiling”目录。执行命令如下：

su HwHiAiUser

mkdir -p /home/HwHiAiUser/profiling

步骤3 切换回root用户，在运行环境中执行如下命令基于新的镜像运行一个容器。

docker run -it -e ASCEND_VISIBLE_DEVICES=*xxx* **-v /var/log/npu/slog/container/***docker名称***:/var/log/npu/slog -v /var/log/npu/profiling/***docker名称***:/var/log/npu/profiling -v /home/HwHiAiUser/profiling:/var/log/npu/conf/profiling --pids-limit 409600** *image-name* **/bin/bash**

表 6-12 参数解释

参数	参数说明
-e ASCEND_VISIBLE_DEVICES=*xxx*	使用ASCEND_VISIBLE_DEVICES环境变量指定被挂载至容器中的NPU设备，使用设备序号指定设备，支持单个和范围指定且支持混用。例如： 1. -e ASCEND_VISIBLE_DEVICES=0 表示将0号设备（/dev/davinci0）挂载入容器中。 2. -e ASCEND_VISIBLE_DEVICES=1,3 表示将1、3号设备挂载入容器中。 3. -e ASCEND_VISIBLE_DEVICES=0-2 表示将0号至2号设备（包含0号和2号）挂载入容器中，效果同 -e ASCEND_VISIBLE_DEVICES=0,1,2 4. ASCEND_VISIBLE_DEVICES=0-2,4 表示将0号至2号以及4号设备挂载入容器，效果同 -e ASCEND_VISIBILE_DEVICES=0,1,2,4
-v /var/log/npu/slog/container/*docker名称:*/var/log/npu/slog	将宿主机日志路径（例如“/var/log/npu/slog/container/docker名称”）挂载到容器中。该目录可以由客户自定义，该目录的所有者为容器运行用户。
-v /var/log/npu/profiling/*docker名称:*/var/log/npu/profiling	将宿主机profiling路径“/var/log/npu/profiling/docker名称”挂载到容器中。请根据实际情况修改。
-v /home/HwHiAiUser/profiling:/var/log/npu/conf/profiling	将宿主机profiling配置文件“/home/HwHiAiUser/profiling”挂载到容器中。不需要修改。
--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。
image-name	镜像名称与标签，请用户根据实际情况更换。

执行该命令后，如果显示容器ID（本例为“3330d6524117”），则表示已经运行并进入该容器。

```
root@3330d6524117:/#
```


说明

- AscendDocker Runtime默认挂载的内容如**E.2 AscendDocker Runtime默认挂载内容**所示。若用户在宿主机安装驱动时采用的是指定路径方式，需自行挂载**表E-2**中的目录和文件。执行命令如下：

  docker run -it -e ASCEND_VISIBLE_DEVICES=*xxx* **--pids-limit 409600 -v /var/log/npu/conf/slog/slog.conf:/var/log/npu/conf/slog/slog.conf -v** ${*install_path*}**/driver/lib64:**${*install_path*}**/driver/lib64 -v** ${*install_path*}**/driver/tools:**${*install_path*}**/driver/tools -v** ${*install_path*}**/add-ons:**${*install_path*}**/add-ons -v /usr/local/dcmi:/usr/local/dcmi -v /var/log/npu/slog/container/***docker名称:***/var/log/npu/slog -v /var/log/npu/profiling/***docker名称:***/var/log/npu/profiling -v /home/HwHiAiUser/profiling:/var/log/npu/conf/profiling -v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi** *image-name* **/bin/bash**

  其中*${install_path}*为昇腾芯片驱动安装路径。

- 可使用ASCEND_RUNTIME_OPTIONS环境变量调整被挂载的内容，若值设为NODRV则不挂载非设备的目录和文件，例：

  docker run --rm -it -e ASCEND_VISIBLE_DEVICES=*xxx* **-e ASCEND_RUNTIME_OPTIONS=***NODRV docker_image* **/bin/bash**

  容器中将仅挂载NPU设备和管理设备（如/dev/davinci0、/dev/davinci_manager、/dev/hisi_hdc、/dev/devmm_svm），为在容器中安装驱动的场景提供支持。

步骤4 修改容器内HwHiAiUser用户的uid、gid。

OLD_HWHIAIUSER_UID=`id -u HwHiAiUser`

OLD_HWHIAIUSER_GID=`id -g HwHiAiUser`

usermod -u *UID* **HwHiAiUser**

groupmod -g *GID* **HwHiAiUser**

chown -R --quiet --from=$OLD_HWHIAIUSER_UID:$OLD_HWHIAIUSER_GID HwHiAiUser:HwHiAiUser /

其中*UID*、*GID*为步骤**1**中查询到的uid、gid，用户根据实际情况替换。

步骤5 创建日志相关文件，命令如下：

mkdir -p /var/log/npu/slog/slogd

mkdir -p /usr/slog

步骤6 修改"/var/log/npu/slog"和"/usr/slog"文件的所有者为HwHiAiUser，命令如下。

chown -R HwHiAiUser:HwHiAiUser /var/log/npu/slog

chown -R HwHiAiUser:HwHiAiUser /usr/slog

步骤7 执行如下命令启动slogd守护进程。

su HwHiAiUser --command "${*install_path*}**/driver/tools/docker/slogd &"**

其中*${install_path}*是昇腾芯片驱动安装路径，默认安装路径为“/usr/local/Ascend”。

步骤8 执行如下命令确认slogd守护进程是否执行成功。

ps -ef | grep -v grep | grep "tools/docker/slogd"

若返回如下信息，则说明进程已启动，镜像文件已经部署成功。

```
HwHiAiU+     13      1  0 09:15 ?        00:00:00 /usr/local/Ascend/driver/tools/docker/slogd
```


步骤9 为保障profiling数据传输，请执行如下命令进行profiling运行前初始化和启动ada守护进程。

su HwHiAiUser --command "${*install_path*}**/driver/tools/profiling_docker_init.sh &"**

步骤10 执行如下命令确认ada守护进程是否执行成功。

ps -ef | grep -v grep | grep ada

若返回如下信息，则说明进程已启动，镜像文件已经部署成功。

```
HwHiAiU+    22124 21940  0 20:00 ?        00:00:00 /usr/local/Ascend/driver/tools/ada --docker
```

----结束

说明

若用户在部署训练容器时需要运行相关业务，建议使用非root用户。

A 卸载

- 如果用户只是卸载CANN软件包（如nnrt、nnae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。
- Ascend-cann-toolkit或toolbox软件包中的AICPU算子包需要单独卸载。

方法一 脚本卸载

用户可以通过卸载脚本完成卸载，以卸载开发套件包为例。

步骤1 进入软件的卸载脚本所在目录，一般放置在“script”目录下。

cd *<path>*/**ascend-toolkit/***<version>*/*xxx*-**linux/script**

*其中<path>*为工具的安装目录，若用户安装软件包时指定目录，请用户替换为指定的目录。若未指定则为默认目录。

*<version>*为软件包版本，*xxx*-**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。

步骤3 卸载AICPU算子包。

1. 以root用户登录安装环境。
2. 进入卸载脚本所在目录。
   cd /usr/local/Ascend/opp/aicpu/script
3. 执行**./uninstall.sh**命令运行脚本，完成卸载。

----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。

步骤3 软件包卸载。

- 若用户安装软件包时指定路径，执行**./***软件包名*.**run --uninstall --install-path=***<path>*命令完成卸载。
  *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。

- 若用户安装软件包时未指定路径，执行**./***软件包名*.**run --uninstall**命令对软件包进行卸载。

  说明

  - 若用户安装软件包时使用**--install-username**和**--install-usergroup**指定了运行用户，则卸载时也需指定。命令示例如下：

    ./*软件包名*.**run --uninstall --install-username=***username* **--install-usergroup=***usergroup*
  - 若环境上同时存在多个版本的软件包，则执行卸载命令时需指定软件包安装目录（需要指定到版本号路径）。
  - 如需卸载AICPU算子包，请参见**步骤3**进行卸载。

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
[INFO] process end
```

*xxx*表示卸载的实际软件包名。

----结束

B 升级

B.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需要复位系统，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B-1所示。

表 B-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	若环境中已安装多个版本的CANN软件（如开发套件包Ascend-cann-toolkit等），请先执行卸载操作，确保环境中仅存在一个版本的CANN软件。

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Atlas Data Center Solution或Atlas Intelligent Edge Solution中的版本配套表。

- 不支持从Atlas Data Center Solution V100R020C00*xxx*、Atlas Intelligent Edge Solution V100R020C00*xxx*升级至当前版本，请先执行卸载，再进行安装操作。

升级流程

请按照“固件->驱动->MCU（推理卡或训练卡）->CANN软件”的顺序进行升级。

B.2 升级操作

升级驱动和固件

请根据表B-2参考对应文档进行升级操作。

表 B-2 驱动和固件升级指导

产品型号	参考文档
Atlas 200 AI加速模块（型号 3000）	请参见《**Atlas 200 AI加速模块 1.0.7及以上 软件安装与维护指南（EP场景，型号 3000）**》的“升级”章节。
Atlas 300I 推理卡（型号 3010）	请参见《**Atlas 300I 推理卡 1.0.7 升级指导书（型号 3000, 3010）**》。
Atlas 500 Pro 智能边缘服务器（型号 3000）	支持配置Atlas 300I 推理卡（型号 3000），请参见《**Atlas 300I 推理卡 1.0.7 升级指导书（型号 3000, 3010）**》。
Atlas 800 推理服务器（型号 3000）	支持配置Atlas 300I 推理卡（型号 3000），请参见《**Atlas 300I 推理卡 1.0.7 升级指导书（型号 3000, 3010）**》。
Atlas 800 推理服务器（型号 3010）	支持配置Atlas 300I 推理卡（型号 3010），请参见《**Atlas 300I 推理卡 1.0.7 升级指导书（型号 3000, 3010）**》。
Atlas 300T 训练卡（型号 9000）	请参见《**Atlas 300T 训练卡 驱动和固件安装升级指南 (型号9000)**》的“升级”章节。
Atlas 800 训练服务器（型号 9000）	请参见《**Atlas 800 训练服务器 驱动和固件安装升级指南 (型号9000)**》的“升级”章节。
Atlas 800 训练服务器（型号 9010）	请参见《**Atlas 800 训练服务器 驱动和固件安装升级指南 (型号9010)**》的“升级”章节。
Atlas 900 AI集群（型号 9000）	请参见《**Atlas 900 计算节点 驱动和固件安装升级指南**》的“升级”章节。

升级 CANN 软件

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。

- 若用户安装软件包时指定路径，请执行以下命令。
  ./*软件包名*.**run --upgrade --install-path=**<*path*>
  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。
- 若用户安装软件包时未指定路径，请执行以下命令。
  ./*软件包名*.**run --upgrade**

⚠ 注意

开发套件包Ascend-cann-toolkit和实用工具包toolbox在**X86系统**中进行升级的时候，如果出现如下情况，即询问是否进行热复位，则用户需输入n，安装完毕后重启os生效，当前版本只支持n选项。

```
The installation of aicpu_kernels needs to restart the device to take effect, do you want to hot_reset the device? [y/n]
```

----结束

C 常用操作

C.1 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。

CentOS/EulerOS 操作系统配置网卡 IP 地址

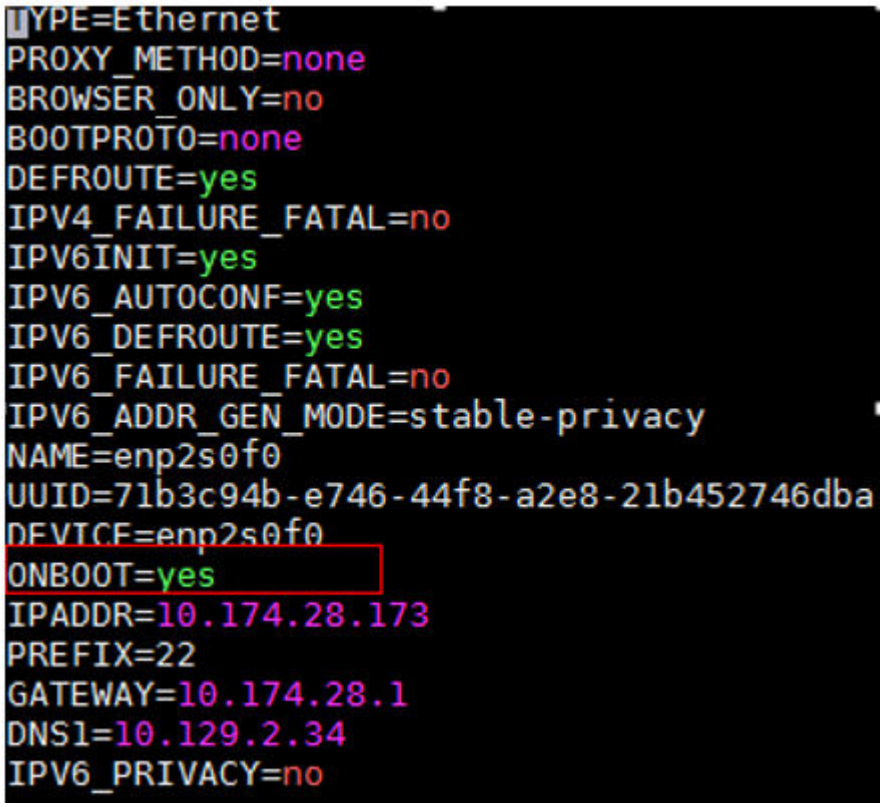
步骤1 以root用户进入OS界面。

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C-1所示。

图 C-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

----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。

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C-2所示。

图 C-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 重启网络服务。

service network restart

步骤5 查看IP地址是否配置成功。

ifconfig

----结束

C.2 配置系统网络代理

以下步骤是配置网络代理的通用方法，不一定能适配所有的网络环境，网络代理的配置方式以现场实际的网络环境为准。

前提条件

- 确保服务器网线已连接，代理服务器能连接外网。
- 配置代理是建立在服务器处在内部网络，无法直接连接外部网络的条件下。

配置系统网络代理

步骤1 使用root权限登录用户环境。

步骤2 使用如下命令编辑“/etc/profile”文件：

```
vi /etc/profile
```

在文件最后添加如下内容后保存退出：

```
export http_proxy="http://user:password@proxyserverip:port"
export https_proxy="http://user:password@proxyserverip:port"
```

其中user为用户在内部网络中的用户名，password为用户密码（特殊字符需转义），proxyserverip为代理服务器的ip地址，port为端口。

步骤3 执行如下命令使配置生效。

```
source /etc/profile
```

步骤4 执行如下命令测试是否连接外网。

```
wget www.baidu.com
```

如果能下载到网页html文件则表明服务器已成功连接外网。

----结束

C.3 安装 3.5.2 版本 cmake

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

C.4 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
```

在.pip目录下创建pip.conf 文件，命令为：

```
touch pip.conf
```

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

C.5 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：

```
sudo rm -rf /tmp/*
```

步骤3 安装依赖。

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
```

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
   ```

> **⚠ 注意**
>
> 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。

用户需要使用gcc7.3.0的编译环境时，需配置以下环境变量。

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

其中${install_path}为**3.**中配置的gcc7.3.0安装路径，本示例为“/usr/local/gcc7.3.0/”。

----结束

C.6 查询 CANN 软件包版本信息

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）

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

其中/usr/local/Ascend为root用户默认安装路径（安装路径说明可参见**（参考）日志文件和软件包信息**），请用户根据实际情况替换。*{arch}*表示CPU架构（arm64或x86_64）。

步骤3 执行以下命令获取版本信息。

cat ascend_toolkit_install.info

----结束

C.7 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C-1**。

表 C-1 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。
-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C-1**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2017年12月31日。

```
chage -E 2017-12-31 test
```

C.8 故障诊断

测试项功能

故障诊断会获取芯片健康信息，同时对芯片进行算力、功耗、带宽测试并输出测试结果，用以判断当前产品的健康状态。

测试项参数查询

用户可任选以下指令之一查看故障诊断命令的可用参数。

ascend-dmi -dg -h

ascend-dmi -dg --help

各参数解释如表C-2所示。

表 C-2 参数说明

参数	说明	是否必填
[-dg, diagnosis]	使用该参数进行整卡的故障诊断测试。	是
[-c, --card]	指定Card进行诊断，不指定则默认诊断所有Card的信息。 用户可以执行**ascend-dmi info**命令，在显示界面表格中的Card参数处获得已安装的卡的号码。	否
[-l, --level]	指定诊断等级，不指定则默认诊断Level 0等级信息。 目前支持以下两种等级： ● 0：诊断芯片的健康信息。 ● 1：诊断卡的功耗、带宽、算力等信息。	否
不填写-c与-l参数	诊断所有card的Level 0等级信息。	否
[-fmt, --format]	指定输出格式，可以为normal或json。若未指定则默认为normal。	否

说明

为保证返回检测结果的正确性和准确性，故障诊断需要单独执行。

使用实例

使用实例中命令的回显在推理服务器与训练服务器类似，截图取自推理服务器。

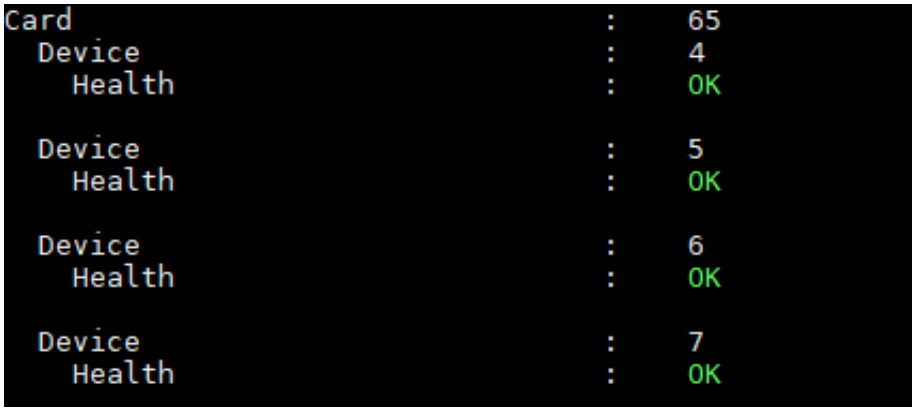
- 以指定卡号和诊断等级Level 0为例。

  ascend-dmi -dg -c 65 -l 0

  65为环境中已安装的卡号码的示例，用户可以执行**ascend-dmi info**命令，在显示界面表格中的Card参数处获得已安装的卡的号码，请用户自行替换。

  若返回如图C-3所示信息，表示工具运行正常，图中参数介绍如表C-3所示。

图 C-3 故障诊断示例（Level 0）



- 以指定卡号和诊断等级Level 1为例。

  ascend-dmi -dg -c 65 -l 1

  65为环境中已安装的卡号码的示例，用户可以执行**ascend-dmi info**命令，在显示界面表格中的Card参数处获得已安装的卡的号码，请用户自行替换。

  若返回如**图C-4**所示信息，表示工具运行正常，图中参数介绍如**表C-3**所示。

图 C-4 故障诊断示例（ Level 1 ）

```
Card                                :    65
  Device                            :    4
    Health                          :    OK
    Flops Test
      Duration(ms)                  :    3766
      Computing Power(TFLOPS)       :    11.137291
    Bandwidth Test                  :    Device To Device
      Bandwidth(MB/s)               :    45397.267701
    Bandwidth Test                  :    Device To Host
      Bandwidth(MB/s)               :    2936.311216
    Bandwidth Test                  :    Host To Device
      Bandwidth(MB/s)               :    2557.242812

  Device                            :    5
    Health                          :    OK
    Flops Test
      Duration(ms)                  :    3766
      Computing Power(TFLOPS)       :    11.137291
    Bandwidth Test                  :    Device To Device
      Bandwidth(MB/s)               :    45483.661086
    Bandwidth Test                  :    Device To Host
      Bandwidth(MB/s)               :    2921.620144
    Bandwidth Test                  :    Host To Device
      Bandwidth(MB/s)               :    2571.231763

  Device                            :    6
    Health                          :    OK
    Flops Test
      Duration(ms)                  :    3766
      Computing Power(TFLOPS)       :    11.137291
    Bandwidth Test                  :    Device To Device
      Bandwidth(MB/s)               :    45477.239995
    Bandwidth Test                  :    Device To Host
      Bandwidth(MB/s)               :    2928.073737
    Bandwidth Test                  :    Host To Device
      Bandwidth(MB/s)               :    2525.945231

  Device                            :    7
    Health                          :    OK
    Flops Test
      Duration(ms)                  :    3766
      Computing Power(TFLOPS)       :    11.137291
    Bandwidth Test                  :    Device To Device
      Bandwidth(MB/s)               :    45415.762753
    Bandwidth Test                  :    Device To Host
      Bandwidth(MB/s)               :    2884.494700
    Bandwidth Test                  :    Host To Device
      Bandwidth(MB/s)               :    2512.327214

  Power Test
    Max Power(W)                    :    62.600002
    Average Power(W)                :    60.729111
    Max AI Core(%)                  :    100
    Average AI Core(%)              :    89
    Max Temp(C)                     :    83
    Average Temp(C)                 :    80
    Max Voltage(V)                  :    75
    Average Voltage(V)              :    75
```


表 C-3 实现界面参数说明

参数	说明
Card	卡号
Device	芯片编号
Health	芯片的健康程度
Flops Test	算力工具测试
Duration(ms)	测试时间
Computing Power(TFLOPS)	算力
Bandwidth Test	带宽测试类型
Bandwidth(MB/s)	带宽值
Power Test	功耗测试
Max Power(W)	最大功耗
Average Power(W)	平均功耗
Max AI Core(%)	AI Core最大利用率
Average AI Core(%)	AI Core平均利用率
Max Temp(C)	最大温度
Average Temp(C)	平均温度
Max Voltage(V)	最大电压
Average Voltage(V)	平均电压

C.9 软硬件版本兼容性测试

测试项功能

软硬件兼容性工具会获取硬件信息、架构、驱动版本、固件版本以及软件版本，并检测软硬件间的兼容性。

使用限制

固件版本只有在用root用户执行ascend-dmi工具的时候才可以查询。

测试项参数查询

用户可任选以下指令之一查看软硬件版本兼容性测试命令的可用参数。

ascend-dmi -c -h

ascend-dmi -c --help

各参数解释如表C-4所示。

表 C-4 参数说明

参数	说明	是否必填
[-c, compatible]	使用该参数进行软硬件版本兼容性检测。	是
[-p, --path]	用户指定检测兼容性的软件包的安装路径。 root用户安装软件包时采用的是默认路径“/usr/local/Ascend”，可不填写此参数。 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，指令为： **ascend-dmi -c -p /home/xxx/Ascend**	否
[-fmt, --format]	指定输出格式，可以为normal或json。若未指定则默认为normal。	否

说明

兼容性工具检测的软件包如下：

- toolkit：开发套件
- nnrt：离线推理引擎
- nnae：训练/在线推理软件包
- tfplugin：框架插件
- toolbox：实用工具
- npu-driver：驱动
- npu-firmware：固件

使用实例

以测试软硬件版本兼容性为例。

ascend-dmi -c

若推理服务器返回如图C-5所示信息或训练服务器返回如图C-6所示信息，表示工具运行正常，图中参数请参见表C-5。

图 C-5 软硬件版本兼容性检测示例（推理服务器）

```
|=============================================================================================================|
|                                          System Information                                                 |
|=============================================================================================================|
| Architecture         | x86_64                                                                               |
+----------------------+--------------------------------------------------------------------------------------+
| Product Type         | Atlas300-3010                                                                        |
|=============================================================================================================|
|                                  Compatibility Check Result: Compatible                                     |
|=============================================================================================================|
| Package              | Version           | Status                    | Dependencies                                 |
+----------------------+-------------------+---------------------------+----------------------------------------------+
| npu-driver           | 20.1.0            | OK                        | NA                                           |
+----------------------+-------------------+---------------------------+----------------------------------------------+
| npu-firmware         | 1.75.22.1.220     | OK                        | NA                                           |
+----------------------+-------------------+---------------------------+----------------------------------------------+
| nnrt                 | 20.1.spc200       | OK                        | NA                                           |
+----------------------+-------------------+---------------------------+----------------------------------------------+
| toolbox              | 20.1.spc200       | OK                        | NA                                           |
|=============================================================================================================|
```

图 C-6 软硬件版本兼容性检测示例（训练服务器）

```
|=============================================================================================================|
|                                          System Information                                                 |
|=============================================================================================================|
| Architecture         | x86_64                                                                               |
+----------------------+--------------------------------------------------------------------------------------+
| Product Type         | Atlas800-9010                                                                        |
|=============================================================================================================|
|                                  Compatibility Check Result: Compatible                                     |
|=============================================================================================================|
| Package              | Version           | Status                    | Dependencies                                 |
+----------------------+-------------------+---------------------------+----------------------------------------------+
| npu-driver           | 20.1.0            | OK                        | NA                                           |
+----------------------+-------------------+---------------------------+----------------------------------------------+
| npu-firmware         | 1.75.22.1.220     | OK                        | NA                                           |
+----------------------+-------------------+---------------------------+----------------------------------------------+
| nnae                 | 20.1.0            | OK                        | NA                                           |
+----------------------+-------------------+---------------------------+----------------------------------------------+
| toolbox              | 20.1.0            | OK                        | NA                                           |
|=============================================================================================================|
```

表 C-5 显示界面参数说明

参数	说明
System Information	系统信息
Architecture	架构
Product Type	硬件类型
Compatibility Check Result	兼容性检测结果
Package	包名
Version	版本
Status	状态
Dependencies	依赖

D FAQ

D.1 run 包安装时提示软件已经安装

问题描述

软件包安装时提示：

```
run package is already installed, install failed
```

解决方案

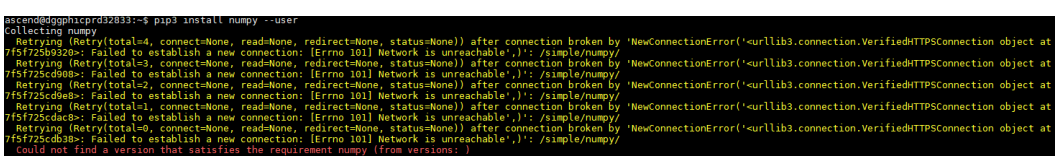
软件包不支持重复安装，需要先卸载后，再安装。卸载方法请参见**A 卸载**。

D.2 使用 pip3.7.5 install 软件时提示" Could not find a version that satisfies the requirement xxx"

问题描述

安装依赖时，使用**pip3.7.5 install xxx**命令安装相关软件时提示无法连接网络，且提示“Could not find a version that satisfies the requirement xxx"，提示信息如所示。

图 D-1 使用 pip3 安装软件提示信息



可能原因

没有配置pip源。

解决方法

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
```

在.pip目录下创建pip.conf 文件，命令为：

```
touch pip.conf
```

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。

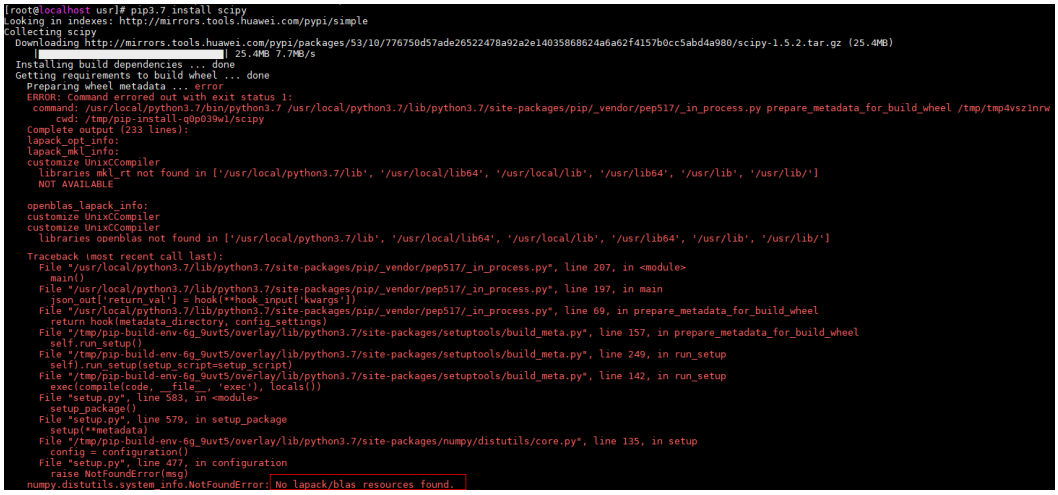
----结束

D.3 pip3.7.5 install scipy 报错

问题描述

安装scipy时，提示如下错误信息。

图 D-2 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。


```
root@localhost usr]# find -name liblapack*
/lib64/liblapacke.so.3.8
/lib64/liblapack.so.3.8
/lib64/liblapack.so.3.8.0
/lib64/liblapacke.so.3
/lib64/liblapack.so.3
/lib64/liblapacke.so.3.8.0
root@localhost usr]# find -name libblas*
/lib64/libblas.so.3.8.0
/lib64/libblas.so.3.8
/lib64/libblas.so.3
```

解决方法

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
```

D.4 pip3.7.5 install numpy 报错

问题描述

安装依赖时，使用**pip3.7.5 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：

```
numpy/core/src/umath/loops.c.src: In function ‘LONG_right_shift’:
numpy/core/src/umath/loops.c.src:694:47: internal compiler error: in expand_shift_1, at expmed.c:2245
     BINARY_LOOP_FAST(@type@, @type@, *out = npy_rshift@c@(in1, in2));
                                               ^
numpy/core/src/umath/fast_loop_macros.h:180:9: note: in definition of macro ‘BASE_BINARY_LOOP_S_INP’
         op; \
         ^
numpy/core/src/umath/loops.c.src:694:5: note: in expansion of macro ‘BINARY_LOOP_FAST’
     BINARY_LOOP_FAST(@type@, @type@, *out = npy_rshift@c@(in1, in2));
     ^
Please submit a full bug report,
with preprocessed source if appropriate.
See <http://bugzilla.redhat.com/bugzilla> for instructions.
Preprocessed source stored into /tmp/cckpTHdy.out file, please attach this to your bugreport.
Running from numpy source directory.
/usr/local/python3.7.5/lib/python3.7/distutils/dist.py:274: UserWarning: Unknown distribution option: 'define_macros'
  warnings.warn(msg)
error: Command "gcc -pthread -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -fPIC -DNPY_INTERNAL_BUILD=1 -DHAV
RCE=1 -D_LARGEFILE64_SOURCE=1 -DHAVE_CBLAS -Ibuild/src.linux-aarch64-3.7/numpy/core/src/umath -Ibuild/src.linux-aarch64-3.7/numpy
re/src/common -Inumpy/core/include -Ibuild/src.linux-aarch64-3.7/numpy/core/include/numpy -Inumpy/core/src/common -Inumpy/core/sr
/multiarray -Inumpy/core/src/umath -Inumpy/core/src/npysort -I/usr/local/python3.7.5/include/python3.7m -Ibuild/src.linux-aarch64
/numpy/core/src/npymath -c build/src.linux-aarch64-3.7/numpy/core/src/umath/loops.c -o build/temp.linux-aarch64-3.7/build/src.lin
ld/temp.linux-aarch64-3.7/build/src.linux-aarch64-3.7/numpy/core/src/umath/loops.o.d -std=c99" failed with exit status 1
  ----------------------------------------
  ERROR: Failed building wheel for numpy
Failed to build numpy
ERROR: Could not build wheels for numpy which use PEP 517 and cannot be installed directly
```

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。

解决方法

执行如下命令安装：

```
export CFLAGS=-std=c99
pip3.7 install numpy==1.17.2
```


D.5 pip3.7.5 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**pip3.7.5 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：

```
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。

解决方法

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**pip3.7.5 list**查看是否解决。

----结束

E 参考信息

E.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： **./*.run** [options]

详细参数请参见**表E-1**。

须知

如果通过**./*.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 E-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。
--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。

参数	说明
--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**./*.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-username=<username>	首次安装：可以指定运行用户名，若不指定，则默认是HwHiAiUser。 覆盖安装：沿用上次运行用户名。 该参数需要配合“--install-usergroup=<usergroup>”一起使用。
--install-usergroup=<usergroup>	首次安装：可以指定运行用户组，若不指定，则默认是HwHiAiUser。 覆盖安装：沿用上次运行用户组。 该参数需要配合“--install-username=<username>”一起使用。
--install-path=<path>	指定安装路径，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。自动进行版本号检查，如果版本号不是从小到大，则不能升级。
--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。

E.2 AscendDocker Runtime 默认挂载内容

除NPU设备和管理设备（/dev/davinci*X*、/dev/davinci_manager、/dev/hisi_hdc、/dev/devmm_svm）外，昇腾Docker Runtime根据实际环境情况默认以只读方式挂载以下目录和文件到容器中。

表 E-2 默认挂载目录和文件

路径	说明
/usr/local/Ascend/driver/lib64	目录，驱动提供的用户态库
/usr/local/Ascend/driver/tools	目录，驱动提供的工具（非所有工具都支持容器场景）
/usr/local/Ascend/driver/include	目录，驱动提供的头文件（dsmi_common_interface.h）
/usr/local/Ascend/add-ons	目录，第三方库
/usr/local/dcmi	目录，DCMI头文件和库
/usr/local/bin/npu-smi	文件，NPU-SMI工具
/var/log/npu/conf/slog/slog.conf	文件，日志服务配置文件

E.3 CCEC 编译器说明

昇腾AI软件栈中提供了CCEC编译器进行TBE算子的编译，将CodeGen生成的类C代码文件（CCE）编译生成昇腾AI处理器海思SoC可执行的文件。图编译时，TBE内部会自动调用CCEC编译器，无需用户手工调用。

CCEC编译器存储在ATCFwkACLlib安装路径下的“ccec_compiler”目录中，包含的主要二进制工具如下所示：

表 E-3 CCEC 编译器说明

工具名称	存储相对路径	功能说明及使用场景	风险分析
clang	ccec_compiler/bin/clang	CCE算子编译器。 编译TBE生成的算子时会自动使用clang进行编译，不建议用户单独调用。	攻击者可能会利用这些已经存在的调试编译工具来编译新的程序，对系统造成二次攻击。
llvm-objdump	ccec_compiler/bin/llvm-objdump	调试工具，可打印算子函数名和地址偏移，用于辅助定位问题。 使用AI Core Error分析工具解析AI Core错误信息时会自动使用此工具，定位TBE算子出错时的函数位置信息，不建议用户单独调用。	

工具名称	存储相对路径	功能说明及使用场景	风险分析
lld	ccec_compiler/bin/lld	单架构CCE算子链接器。 编译TBE算子过程中会自动使用此工具链接算子组成完整的可执行程序，不建议用户单独调用。	

E.4 HCC 编译器说明

Toolkit包中提供了HCC编译器，用于进行Device侧驱动及应用程序的编译。HCC编译器存储在Toolkit安装路径的“toolchain/hcc/”目录中，包含的主要二进制工具说明如下表所示。

表 E-4 HCC 编译器工具说明

工具名称	存储相对路径	功能说明及使用场景	风险分析
gcc	hcc/bin/aarch64-target-linux-gnu-gcc	编译源文件生成可执行文件。 用户编写Device侧代码时会使用该工具编译代码生成可执行文件。	攻击者可能会利用这些已经存在的调试编译工具来编译新的程序，对系统造成二次攻击。
ld	hcc/bin/aarch64-target-linux-gnu-ld	创建动态链接的可执行文件时，将所有符号添加到动态符号表中。 用户编写Device侧代码时在编译源码的过程中会使用该工具链接中间文件生成完整的可执行文件。	
readelf	hcc/bin/aarch64-target-linux-gnu-readelf	显示二进制文件的ELF标头包含的信息。 调试问题时可通过该工具查看算子库二进制文件的ELF头信息及各个段的信息。	

F 修订记录

发布日期	修改说明
2020-10-15	第一次正式发布。

