联机帮助

文档版本 01

发布日期 2020-05-12

华为技术有限公司

版权所有 © 华为技术有限公司 2020。 保留一切权利。

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

商标声明

HUAWEI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 认识 Mind Studio.....1
1.1 关于 Mind Studio.....1
1.2 What's New.....3
1.3 关键概念.....8
2 安装.....11
2.1 简介.....11
2.2 下载软件包.....12
2.3 环境准备.....13
2.4 软件包完整性校验.....18
2.5 安装 Mind Studio.....22
2.6 查询 Mind Studio 版本.....27
2.7 启动 Mind Studio.....28
2.8 停止 Mind Studio.....28
2.9 更新 Mind Studio.....28
2.10 卸载 Mind Studio 与 DDK.....29
2.11 FAQ.....29
2.11.1 安装完 Mind Studio 后，无法远程连接 host 为 CentOS 操作系统的服务器.....30
2.11.2 Mind Studio 或 DDK 安装过程中提示 pip2 或 pip 不可用.....30
2.11.3 软件包完整性校验返回 WARNING 或 FAIL.....31
3 编译环境配置.....34
3.1 配置流程.....34
3.2 获取 lib 库.....38
3.2.1 lib 库通过同步方式获取.....38
3.2.2 lib 库通过 lib 包解压获取.....43
3.3 配置编译环境.....43
4 快速入门.....46
4.1 创建首个 AI 应用.....46
4.2 快速体验首个算子工程.....52
5 工程迁移.....62
6 教程.....74
6.1 应用开发.....74

6.2 自定义算子开发 81
7 模型转换 95
7.1 概述 95
7.2 模型转换 97
7.3 模型可视化 118
8 操作指南 123
8.1 工程管理 123
8.1.1 打开/关闭/移除/还原工程 123
8.1.2 新建/删除文件 129
8.1.3 修改工程属性 130
8.2 设备管理 132
8.2.1 概述 132
8.2.2 增加设备 134
8.2.3 删除设备 137
8.2.4 编辑设备 138
8.2.5 更新设备信息 138
8.2.6 同步 lib 139
8.2.7 升级 Atlas 200 DK 143
8.2.8 修改 Atlas 200 DK IP 地址 145
8.3 DDK 版本管理 147
8.3.1 前提条件 147
8.3.2 安装新版本 DDK 版本 148
9 Profiling 工具 151
9.1 简介 151
9.2 APP 工程 Profiling 152
9.2.1 工程准备 152
9.2.2 启动 Profiling 152
9.2.3 查看 Profiling 数据 154
9.3 参考 171
9.3.1 Profiling 数据文件说明 171
9.3.2 脚本清单 174
10 日志工具 187
10.1 工具简介 187
10.2 日志操作 188
10.2.1 管理 Device 设备 188
10.2.2 System Log 操作 189
10.3 日志参考 190
10.3.1 日志处理机制介绍 190
10.3.2 日志文件介绍 191
10.3.3 日志级别介绍 192
10.3.4 日志记录格式介绍 192

10.3.5 日志配置介绍 193
10.3.6 设置日志级别 195
11 算子比对 200
11.1 简介 200
11.2 数据准备 201
11.3 Lower Bound 比对 203
11.4 Vector 比对 209
11.5 FAQ 214
11.5.1 Dump 数据丢失 214
12 术语和缩略语 215
12.1 术语/缩略语 215
13 附录 223
13.1 软件包说明 223
13.2 caffe.proto 文件算子定义 225
13.3 设置 Mind Studio 支持自动解析 TBE Python 依赖库 226
13.4 开源第三方库说明 230
13.5 修订记录 232

1 认识 Mind Studio

1.1 关于Mind Studio

1.2 What's New

1.3 关键概念

1.1 关于 Mind Studio

Mind Studio是一套基于IntelliJ框架的开发工具链平台，提供了算子开发、调试、调优以及第三方算子开发的功能，同时还提供了网络移植、优化和分析功能，为用户开发应用程序带来了极大的便利。

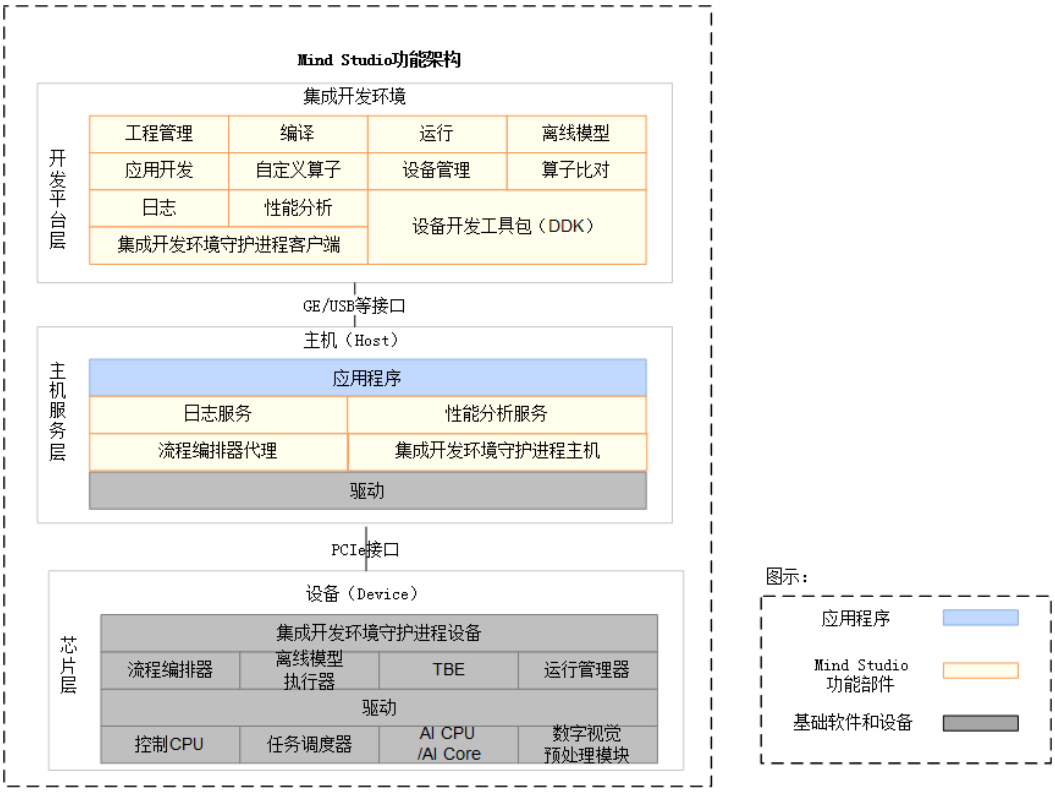
功能简介

- 针对算子开发，Mind Studio提供了全套的算子开发、调优能力。通过Mind Studio提供的工具链也可以进行第三方算子开发，降低了算子开发的门槛，并提高算子开发及调试调优的效率，有效提升了产品竞争力。
- 针对网络模型的开发，Mind Studio集成了离线模型转换工具、模型量化工具、模型精度比对工具、模型运行性能分析工具、日志分析工具，提升了网络模型移植、分析和优化的效率。
- 针对计算引擎开发，Mind Studio预置了典型的分类网络、检测网络等计算引擎代码，降低了开发者的技术门槛，加快了开发者对AI算法引擎的编写及移植效率。
- 针对应用开发，Mind Studio集成了各种工具如分析器（Profiler）和编译器（Compiler）等，为开发者提供了图形化的集成开发环境，通过Mind Studio能够进行工程管理、编译、调试、性能分析等全流程开发，能够很大程度提高开发效率。

功能框架

Mind Studio功能框架如**图1-1**所示，目前含有的工具链包括：工程管理工具、编译工具、流程编排工具、离线模型工具、比对工具、日志管理工具、自定义算子工具、性能分析工具、设备管理工具、设备开发工具包（Device Development Kit，DDK）等多种工具。

图 1-1 工具链功能架构



工具功能

Mind Studio工具中的主要几个功能特性如下：

- 工程管理：为开发人员提供创建工程、打开工程、关闭工程、删除工程、导出工程、新增工程文件目录和属性设置等功能。请参见**8.1 工程管理**。
- 编译：提供算子、计算引擎和应用的开发编译，满足开发者不同场景的开发编译诉求。请参见**工程编译**或**工程编译构建**。
- 运行：支持开发的算子、计算引擎在真实昇腾AI处理器中运行，并可以统一在界面中查看运行相关的信息。请参见**工程运行**或**单算子运行验证**。
- 应用开发：针对业务流程开发人员，Mind Studio工具提供基于matrix流程编排器的应用开发编程方式，编程后的编译、运行、结果显示等一站式服务让流程开发更加智能化，可以让开发者快速上手。请参见**6.1 应用开发**。
- 自定义算子开发：提供了基于TBE的算子编程开发的集成开发环境，让不同平台下的算子移植更加便捷，适配昇腾AI处理器的速度更快。请参见**6.2 自定义算子开发**。
- 离线模型转换：训练好的第三方网络模型可以直接通过离线模型工具导入并转换成离线模型，并可一键式自动生成模型接口，方便开发者基于模型接口进行编程，同时也提供了离线模型的可视化功能。请参见**7 模型转换**。
- 日志管理：Mind Studio为昇腾AI处理器提供了覆盖全系统的日志收集与日志分析解决方案，提升运行时算法问题的定位效率。提供了统一形式的跨平台日志可视化分析能力及运行时诊断能力，提升日志分析系统的易用性。请参见**10 日志工具**。
- 性能分析：Mind Studio以图形界面和命令行两种用户界面呈现方式，实现针对主机和设备上多节点、多模块异构体系的高效、易用、可灵活扩展的系统化性能分

析，以及针对昇腾AI处理器的性能和功耗的同步分析，满足算法优化对系统性能分析的需求。请参见9 Profiling工具。

- 设备管理：Mind Studio提供设备管理工具，实现对连接到主机上的设备的管理功能。请参见8.2 设备管理。
- 算子比对：可以用来比对通过TBE自定义的算子运行结果与Caffe标准算子的运行结果，以便用来确认神经网络运算误差发生的原因。请参见11 算子比对。Mind Studio提供Lower Bound和Vector比对方法。其中：
  - Lower Bound比对，华为自主研发的算法比对。
  - Vector比对，包含余弦相似度、最大绝对误差、累积相对误差、欧几里得相对距离的算子比对。
- 设备开发工具包的安装与管理：为开发者提供基于昇腾AI处理器的相关算法开发工具包DDK（Device Development Kit）,旨在帮助开发者进行快速、高效的人工智能算法开发。开发者可以将设备开发工具包安装到Mind Studio上，使用Mind Studio开发工具进行算法快速开发，也可以使用独立的设备开发工具包进行算法开发。设备开发工具包内部包含了基于昇腾AI处理器开发依赖的头文件和库文件、编译工具链、调优工具等。请参见8.3 DDK版本管理。

1.2 What's New

从开发者使用习惯、成熟度、代码开发体验、工具的可扩展性多个维度综合考虑，基于IntelliJ IDEA架构的Mind Studio打造面向AI全栈开发的集成开发环境，并作为替代基于Eclipse CHE架构的Mind Studio的后续演进版本。

版本配套

Mind Studio当前仅适用于Ubuntu操作系统。版本配套详细信息可以参见环境要求。

版本比较

Mind Studio有如下版本，请参见表1-1，了解各个版本支持哪些功能。

- 基于IntelliJ IDEA架构的Mind Studio：2.1.0及之后版本Mind Studio。
- 基于Eclipse CHE架构的Mind Studio：1.3及之前版本Mind Studio。

表 1-1 特性比较

一级特性	二级特性	功能	基于IntelliJ IDEA架构的Mind Studio	基于Eclipse CHE架构的Mind Studio
自定义算子开发	支持TBE(TE)算子开发	支持基于不同DDK版本和Target的算子工程创建。	2.1.0及后续版本支持	支持
		支持基于Caffe模型和自定义caffe.proto的算子列表展示。	2.1.0及后续版本支持	支持
		支持基于列表选择的算子创建工程、支持基于自定义空算子创建工程。	2.1.0及后续版本支持	支持

一级特性	二级特性	功能	基于IntelliJ IDEA架构的Mind Studio	基于Eclipse CHE架构的Mind Studio
		支持基于当前算子生成插件代码。	2.1.0及后续版本支持	支持
		支持基于当前工程添加Caffe模型解析的算子、添加自定义空算子。	2.1.0及后续版本支持	支持
		支持TBE(TE)算子的编译构建，运行配置创建、删除和修改。	2.1.0及后续版本支持	支持
模型开发	支持离线模型转换	支持caffe、tensorflow模型选择以及模型可视化。	2.1.0及后续版本支持	支持
		支持自动生成输入、输出节点。	2.1.0及后续版本支持	支持
		支持手动从模型网络图中增加输出节点。	2.1.0及后续版本支持	支持
		支持多batch输入、8bit量化配置、加密配置。	2.1.0及后续版本支持	支持
		支持模型转换失败时根据错误报告，创建或替换算子。	2.1.0及后续版本支持	支持
		支持在多个算子实现对应一个算子原型时，用户选择算子指定实现。	2.1.0及后续版本支持	支持
		支持开源Caffe模型导入。	不支持	支持
		支持Caffe和TensorFlow原始网络模型可视化。	2.1.0及后续版本支持	只支持Caffe原始网络模型可视化
		支持离线模型可视化。	2.1.0及后续版本支持	支持
	支持AIPP预处理配置	支持静态AIPP：模型生成后，AIPP参数值被保存在离线模型中。 支持动态AIPP：每次模型推理前需要在推理Engine的代码中设置动态AIPP参数值，然后在模型推理时可使用不同的AIPP参数。	2.1.0及后续版本支持	支持
比对工具	支持LowerBound比对	支持展示融合算子信息。 支持lowerbound比对，以及比对结果文件输出。	2.1.0及后续版本支持	不支持

一级特性	二级特性	功能	基于IntelliJ IDEA架构的Mind Studio	基于Eclipse CHE架构的Mind Studio
	支持向量比对	支持Vector比对、比对结果排序、比对结果文档输出。	2.1.0及后续版本支持	不支持
应用开发	支持APP应用的工程创建，编译和运行	支持创建分类网络、检测网络、自定义网络模板的应用工程。 支持编译目标为Ubuntu x86、CentOS arm、CentOS x86的编译。 支持在Ubuntu x86、CentOS arm、CentOS x86上运行。 支持检测运行环境与编译环境是否一致。	2.1.0及后续版本支持	支持
	支持在APP应用中修改部分配置	支持修改工程Description信息、修改DDK版本信息。	2.1.0及后续版本支持	支持
	导入工程	支持导入从1.3及之前版本Mind Studio导出的mind工程，并支持导入工程的编译构建及运行。	2.1.0及后续版本支持	支持
	其他	支持Mind Studio界面删除工程。	不支持	支持
		支持多网络串联功能、publish功能。	不支持	支持
基础功能	工程管理	支持打开工程时自动检测工程设置中的DDK版本是否已经安装。	2.1.0及后续版本支持	支持
		支持在工程属性中修改DDK版本与Target信息。	不支持	支持
	支持DDK管理	支持通过Mind Studio查看已经安装的DDK版本信息。	2.1.0及后续版本支持	支持
		支持用户选择本地的一个DDK安装包进行安装。	2.1.0及后续版本支持	支持
		支持用户选择本地的另一个DDK版本安装包，升级安装的DDK版本。	2.1.0及后续版本支持	支持
		支持为工程编译提供查询host操作系统、架构的接口。	2.1.0及后续版本支持	支持
	环境检查	支持对运行操作系统、内存、磁盘空间最小要求的检查。	2.1.0及后续版本支持	支持

一级特性	二级特性	功能	基于IntelliJ IDEA架构的Mind Studio	基于Eclipse CHE架构的Mind Studio
		支持在本地没有安装DDK时，根据向导安装DDK。	2.1.0及后续版本支持	支持
	性能Profiling功能	支持运行的配置界面配置Profiling的开关。	2.1.0及后续版本支持	支持
		支持通过单击Run来启动Profiling，启动后不支持主动停止。	2.1.0及后续版本支持	支持
		支持工程文件右键增加查看Profiling结果菜单的功能。	2.1.0及后续版本支持	支持
		支持Profiling的开关配置，包含AICPU、CtrlCPU、AICORE、DDR、LLC、NIC、RTS、HIENGINE、OME等。	2.1.0及后续版本支持	支持
		支持应用运行timeline分析、Hardware分析、AICPUFunction分析、Control CPUFunction分析、TS CPUFunction分析、Summary分析。	2.1.0及后续版本支持	支持
		支持Profiling数据统计表格排序和导出。	2.1.0及后续版本支持	支持
		支持TS CPU、CPU Usage、Time-Consumption Statistics的数据展示。	2.1.0及后续版本支持	支持
		支持内存和CPU占有率的数据展示。	2.1.0及后续版本支持	支持
		支持Host侧的Job数据在Timeline上展示。	2.1.0及后续版本支持	支持
	安装卸载	通过解压IDE的压缩包进行安装。	2.1.0及后续版本支持	支持
		Mind Studio安装成功之后可以进行stop（或start）服务。	不支持	支持
		Mind Studio卸载时有选择是否备份相关内容的选项。	不支持	支持

一级特性	二级特性	功能	基于IntelliJ IDEA架构的Mind Studio	基于Eclipse CHE架构的Mind Studio
	设备管理	支持用户添加新设备，包括IP地址、别名等信息； 支持选中设备的删除、设备信息的修改。 支持所有已管理设备的信息查看，包括显示IP地址、别名等信息。	2.1.0及后续版本支持	支持
		支持设备连通性检测。	2.1.0及后续版本支持	支持
		支持Atlas 200 DK开发者板升级。	2.1.0及后续版本支持	支持
		同步lib：获取编译程序时所依赖的host和device的lib库。	2.1.0及后续版本支持	不支持
	日志	支持连接设备查看系统日志。	2.1.0及后续版本支持	支持
		支持单个日志文件的下载，支持查看单个日志文件的部分内容。	2.1.0及后续版本支持	支持
		支持打开用户本地日志缓存目录。	2.1.0及后续版本支持	支持
		支持日志内容的关键字查找、高亮；支持日志Console中内容的清空。	2.1.0及后续版本支持	支持
	多操作系统打包安装卸载	Mind Studio工具安装脚本支持主机为CentOS x86以及CentOS ARM的操作系统。	不支持	支持
	Mind Studio升级	Mind Studio支持版本在线升级功能。	不支持	支持

一级特性	二级特性	功能	基于IntelliJ IDEA架构的Mind Studio	基于Eclipse CHE架构的Mind Studio
图形化流程编排功能	-	Mind Studio提供图形化设计界面。 Mind Studio提供内置数据集，内置模型等组件： 内置数据集，支持用户自行选择图片数量；内置模型，支持导入第三方模型转换成适配昇腾AI处理器的离线模型后，生成模型节点，可供用户拖拽编排。 Mind Studio提供自定义组件的功能：当系统预置的组件无法满足用户需求时，用户可以通过自定义组件功能添加组件进行流程编排。 Mind Studio支持根据图形以及连线生成调用关系流程文件。	不支持	支持
黑匣子相关特性	-	支持获取设备异常信息的功能。	不支持	支持

1.3 关键概念

昇腾 AI 处理器

昇腾AI处理器是一款华为专门为图像识别、视频处理、推理计算及机器学习等领域设计的高性能、低功耗AI芯片。芯片内置2个AI core，可支持128位宽的LPDDR4x，最高可提供16TOPS（Float16/INT8）的计算能力。

Graph

Graph是matrix框架中的概念，而非深度学习框架中计算图的概念。graph在matrix中是指matrix框架中用于描述整个业务处理流程的图，由多个engine组成，是一个程序处理流程。

Host/Device

在异构计算架构中, 昇腾AI处理器与CPU通过PCIe总线连接在一起来协同工作：

- Host：CPU所在位置称为主机端（Host），是指与昇腾AI处理器所在硬件设备相连接的X86服务器、ARM服务器或者WindowsPC，利用昇腾AI处理器提供的NN（Neural-Network）计算能力完成业务。如图1-3中的主机服务层。
- Device：是指安装了昇腾AI处理器的硬件设备，利用PCIe接口与服务器连接，为服务器提供NN计算能力。如图1-3中的芯片层。

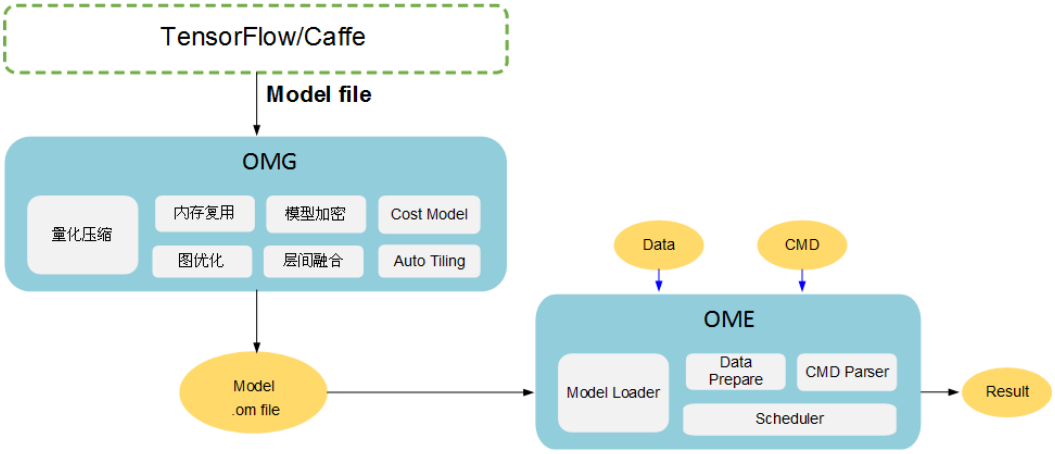
对于Atlas 200 DK，昇腾AI处理器所在硬件设备与与之相连接的Arm服务器合设，统称为Host。

OMG/OME

训练好的第三方模型（Caffe、Tensorflow等）可通过OMG（Offline Model Generator：离线模型生成器）转换成昇腾AI处理器支持的离线模型，应用程序运行时，可通过OME（Offline Model Executor：离线模型执行器）进行离线模型的加载，进行推理。

离线模型功能架构如图1-2所示。

图 1-2 离线模型架构



- OMG：用户使用Caffe/Tensorflow等框架训练好的模型，可通过OMG将其转换成昇腾AI处理器支持的离线模型，模型转换过程中可以实现算子调度的优化、权值数据重排、量化压缩、内存使用优化等，可以脱离设备完成模型的预处理。模型转换详细操作请参见7 模型转换。
- OME：应用程序执行时，OME加载转换好的离线模型文件，分配运行时所需的资源，然后遍历模型文件中的每个算子，创建算子运行所需的描述信息、拷贝权值数据到设备内存，最后创建消息处理线程，在线程中等待输入数据进行处理。

DVPP

数字视觉预处理模块（Digital Vision Pre-Processing，DVPP），作为整个软件流执行过程中的的编解码和图像转换模块，为神经网络发挥着预处理辅助功能。当来自系统内存和网络的视频或图像数据进入昇腾AI处理器的计算资源中运算之前，由于Davinci架构对输入数据有固定的格式要求，如果数据未满足架构规定的输入格式、分辨率等要求，就需要调用数字视觉处理模块进行格式的转换，才可以进行后续的神经网络计算步骤。

AIPP

AIPP（AI Pre Process）是昇腾AI处理器提供的硬件图像预处理模块，包括色域转换，图像归一化（减均值/乘系数）和抠图（指定抠图起始点，抠出神经网络需要大小的图片）等功能。DVPP模块输出的图片多为对齐后的YUV420SP类型，不支持输出RGB图片。因此，业务流需要使用AIPP模块转换对齐后YUV420SP类型图片的格式，并抠出模型需要的输入图片。

Ctrl CPU

主要负责业务逻辑处理，一个昇腾AI处理器中有4个Ctrl CPU。在Mind Studio功能架构中的位置请参见图1-3中的芯片层部分。

AI CPU

主要用于算子任务调度、部分算子的实现，一个昇腾AI处理器中有4个AI CPU。在Mind Studio功能架构中的位置请参见图1-3中的芯片层部分。

AI Core

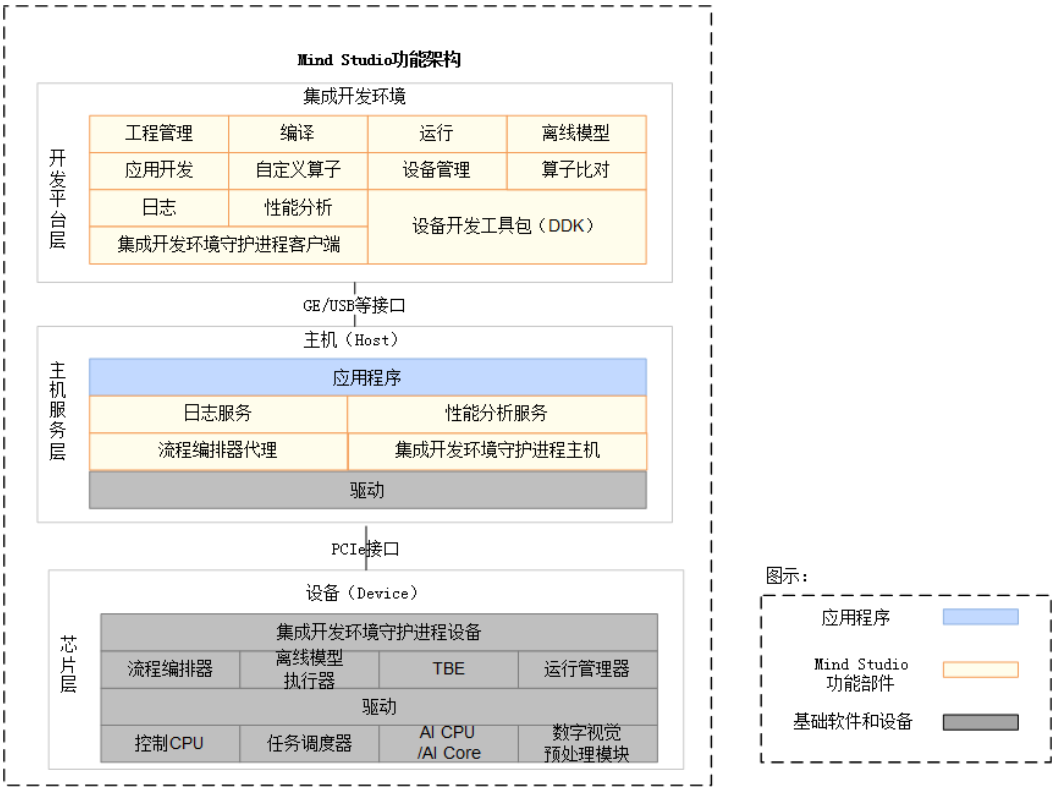
主要负责矩阵运算，一个昇腾AI处理器中有2个AI Core。在Mind Studio功能架构中的位置请参见图1-3中的芯片层部分。

YUV420SP

有损图像颜色编码格式，常用为YUV420SP_UV、YUV420SP_VU两种格式。

Mind Studio 功能框架

图 1-3 工具链功能架构



2 安装

2.1 简介
2.2 下载软件包
2.3 环境准备
2.4 软件包完整性校验
2.5 安装Mind Studio
2.6 查询Mind Studio版本
2.7 启动Mind Studio
2.8 停止Mind Studio
2.9 更新Mind Studio
2.10 卸载Mind Studio与DDK
2.11 FAQ

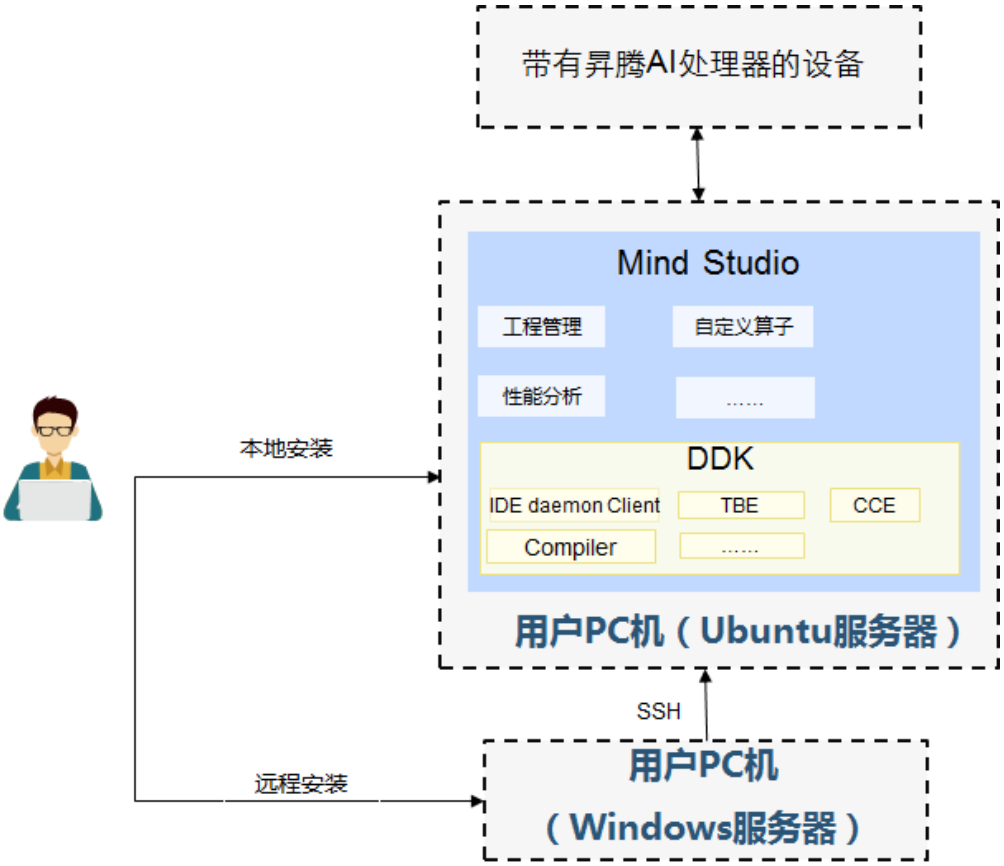
2.1 简介

本手册介绍如何安装Mind Studio以及安装过程中的故障处理方法。

- Mind Studio只能安装在Ubuntu服务器上，可以在Ubuntu服务器上使用原生桌面自带的终端gnome-terminal进行安装，也可以在Windows服务器上通过SSH登录到Ubuntu服务器进行安装，因为Mind Studio是一款GUI程序，所以在Windows服务器上通过SSH登录到Ubuntu服务器进行安装时，需要使用集成了X server的SSH终端（比如mobaxterm）。
- 通过Mind Studio开发人员可以进行普通的工程管理、代码编写、编译、模型转换；如果需要在真实的昇腾AI处理器上运行开发的工程，则需要将Mind Studio连接到主机，并通过主机和设备上的工具后台服务模块进行配合，完成所有开发工程的运行、日志和性能分析等功能。
- DDK（Device Development Kit）设备开发工具包，为开发者提供基于昇腾AI处理器的相关算法开发工具包，旨在帮助开发者进行快速、高效的人工智能算法开发。开发者可以将设备开发工具包安装到Mind Studio上，使用Mind Studio开发工具进行算法快速开发。

Mind Studio与DDK架构图如图2-1所示。

图 2-1 Mind Studio 与 DDK 架构图



2.2 下载软件包

Mind Studio的软件包请从http://139.159.180.132/MindStudio_Release/链接中获取对应的版本，访问用户名：**huawei**，密码：**Huawei12#$**。

软件包说明如下表所示。

表 2-1 Atlas 200 DK 形态软件包说明

安装包	包说明	软件包完整性校验文件
mindstudio.tar.gz	Mind Studio工具软件包。	mindstudio.tar.gz.asc
Ascend_DDK-{software version}-{interface version}-x86_64.ubuntu16.04.tar.gz	DDK软件包。	Ascend_DDK-{software version}-{interface version}-x86_64.ubuntu16.04.tar.gz.asc

表 2-2 Atlas 300 软件包说明

安装包	包说明	软件包完整性校验文件
mindstudio.tar.gz	Mind Studio工具软件包。	mindstudio.tar.gz.asc
Ascend_DDK-{software version}-{interface version}-x86_64.ubuntu16.04.tar.gz	DDK软件包。	Ascend_DDK-{software version}-{interface version}-x86_64.ubuntu16.04.tar.gz.asc
Ascend_lib-{software version}-x86_64.ubuntu16.04-aarch64.miniOS.zip	Atlas 300形态lib包。 如果lib库通过**3.2.1 lib库通过同步方式获取**方式获取，则软件包获取时可以不用获取lib包。	Ascend_lib-1.31.T12.B120-x86_64.ubuntu16.04-aarch64.miniOS.zip.asc

说明

- {software version}：软件版本号，DDK安装包中该参数需要和lib包中该参数版本号保持一致。
- {interface version}：DDK内部版本号。

2.3 环境准备

为保证能够正常使用Mind Studio，请确保环境能满足相应条件。

环境要求

安装Mind Studio的环境，所要求的硬件及操作系统要满足以下条件。

表 2-3 Ubuntu 系统配套版本信息

类别	版本限制	获取方式	注意事项
硬件	● 内存：最小4GB，推荐8GB ● 磁盘空间：最小6GB	-	● 若Linux宿主机内存为4G，在Mind Studio中进行模型转换时，建议Model文件大小不超过350M，如果超过此规格，操作系统可能会因为超过安全内存阈值而工作不稳定。 ● 若Linux宿主机配置升级，比如8G内存，则相应支持的操作对象规格按比例提升。例如，内存由4G升级到8G，则Model文件建议大小不超过700M。
操作系统	版本：16.04 64位操作系统	请从**http://releases.ubuntu.com/releases/**下载对应版本软件进行安装，例如可以下载桌面版：“ubuntu-16.04.xx-desktop-amd64.iso”，或Server版：“ubuntu-16.04.xx-server-amd64.iso”。 **说明** 当前仅支持系统语言为英文。	模型转换时，如果加载了TE自定义插件，可能会出现界面卡住的现象；如果界面卡住，请使用命令**uname -r**或者**cat /proc/version**检查Linux内核版本是否低于4.18，如果低于4.18请参见如下链接升级Linux内核补丁： **https://bugzilla.kernel.org/attachment.cgi?id=277305** 升级Linux内核的方法请参见：**https://bbs.huaweicloud.com/forum/thread-22441-1-1.html**。
Python	python2：2.7+ python3：3.5+	-	Mind Studio安装需要同时安装python2和python3。请参见**安装依赖**。
OpenJDK	openjdk-8	-	请参见**安装依赖**。
cmake	3.5.1+	-	请参见**安装依赖**。
DDK	1.31.T7.B070及之后版本	-	请参见**2.5 安装Mind Studio**。

类别	版本限制	获取方式	注意事项
其他	浏览器	请确保操作系统已经安装浏览器，否则安装完Mind Studio无法查看联机帮助文档。 例如可以用Mind Studio安装用户使用如下命令安装firefox浏览器： `sudo apt-get install firefox` 如果用户使用firefox浏览器打开联机帮助，则会弹出风险告知页面，用户依次单击“Advanced... > Accept the Risk and Continue”即可。	-

准备 Mind Studio 安装用户（可选）

您可以使用任意用户（含root或非root用户）安装Mind Studio。

- 若使用root用户安装，则不需要操作该章节。
- 若使用已存在的非root用户安装，须保证该用户对$HOME目录具有读写以及可执行权限。
- 若使用新的非root用户安装，请参考如下步骤进行创建，如下操作请在root用户下执行。本手册以该种场景为例执行Mind Studio的安装。
  a. 执行以下命令创建Mind Studio安装用户并设置该用户的$HOME目录，并确保用户对$HOME目录具有读写以及可执行权限。
     ```
     useradd -d /home/username -m username
     ```
  b. 执行以下命令设置密码。
     ```
     passwd username
     ```

     说明

     *username*为安装Mind Studio的用户名，该用户的umask值需要等于0027：
     - 若要查看umask的值，则执行命令：**umask**
     - 若要修改umask的值，则执行命令：**umask *新的取值***

配置 Mind Studio 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。

Mind Studio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:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash
   ```
   “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。

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

Mind Studio安装过程需要下载相关依赖，请确保安装Mind Studio的服务器能够连接网络。

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，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。

安装依赖

请使用**su - *username***命令切换到Mind Studio安装用户执行如下操作，安装Mind Studio工具依赖的python2、python3、JDK等组件。

步骤1 检查系统是否安装依赖软件。

1. 检查系统是否安装python开发环境。

   DDK安装依赖python环境，分别使用命令**python2 --version**、**python3 --version**、**pip2 --version**、**pip3 --version**检查是否已经安装，如果返回如下信息则说明已经安装。

```
Python 2.7.12
Python 3.5.2
pip 8.1.1 from /usr/local/lib/python2.7/site-packages (python 2.7)
pip 19.1.1 from /home/ascend/.local/lib/python3.5/site-packages/pip (python 3.5)
```

   否则请执行如下安装命令：

```
sudo apt-get install python python3 python-pip python3-pip
```

2. 检查是否安装Mind Studio显示运行依赖的库文件。

   Mind Studio依赖xterm显示依赖运行的库文件，使用命令**dpkg -l xterm| grep xterm| grep ii**检查是否已经安装，若返回如下信息则说明已经安装。

```
xterm        322-1ubuntu1 amd64        X terminal emulator
```

   否则请执行如下安装命令：

```
sudo apt-get install xterm
```

3. 检查是否安装字体软件。

   Mind Studio中查看联机帮助需要借助相关字体软件，分别使用如下命令检查是否已经安装：

```
dpkg -l fonts-droid-fallback| grep fonts-droid-fallback| grep ii
dpkg -l ttf-wqy-zenhei| grep ttf-wqy-zenhei| grep ii
dpkg -l ttf-wqy-microhei| grep ttf-wqy-microhei| grep ii
dpkg -l fonts-arphic-ukai| grep fonts-arphic-ukai| grep ii
dpkg -l fonts-arphic-uming| grep fonts-arphic-uming| grep ii
```

   若返回如下信息则说明已经安装。

```
fonts-droid-fallback 1:6.0.1r16-1 all        handheld device font with extensive style and language
support (fallback)
ttf-wqy-zenhei 0.9.45-6ubuntu1 all        transitional dummy package
```

```
ttf-wqy-microhei 0.2.0-beta-2 all          transitional dummy package
fonts-arphic-ukai 0.2.20080216.2-4ubuntu2 all          "AR PL UKai" Chinese Unicode TrueType font collection Kaiti style
fonts-arphic-uming 0.2.20080216.2-7ubuntu2 all          "AR PL UMing" Chinese Unicode TrueType font collection Mingti style
```

否则请执行如下安装命令：

```
sudo apt-get install fonts-droid-fallback ttf-wqy-zenhei ttf-wqy-microhei fonts-arphic-ukai fonts-arphic-uming
```

步骤2 安装 Python依赖。

1. Python2依赖安装。

   安装TBE依赖软件（numpy版本仅支持1.11.0~1.16.0，如下以1.16.0为例），执行如下命令分别安装：

   ```
   pip2 install numpy==1.16.0 --user
   pip2 install decorator --user
   ```

2. Python3 (3.5+)依赖安装

   安装TBE依赖软件，执行如下命令分别安装：

   ```
   pip3 install numpy --user
   pip3 install decorator --user
   ```

说明

- 如果依赖安装过程中pip安装异常，请参见“FAQ > Mind Studio或DDK安装过程中提示pip2或pip不可用”解决。
- 上述安装成功后返回信息中的版本号只是样例，请以实际环境中的返回信息为准。

步骤3 安装JDK。

1. 执行如下命令安装JDK。

   ```
   sudo apt-get install -y openjdk-8-jdk
   ```

2. 配置JAVA_HOME环境变量，Mind Studio的安装及运行都依赖该环境变量，设置方法如下：

   a. 在任何目录下执行**vi ~/.bashrc**命令，打开**.bashrc**文件。

   b. 在文件的最后一行后面添加如下内容。

   ```
   export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
   export PATH=$JAVA_HOME/bin:$PATH
   ```

   说明

   “JAVA_HOME”为JDK的安装目录，若用户已经配置了JDK，请根据安装目录的实际情况进行修改。若根据上述步骤安装的JDK，则安装目录不用修改。

   c. 执行**:wq!**命令保存文件并退出。

   d. 执行**source ~/.bashrc**命令使环境变量生效。

   e. 执行 **echo $JAVA_HOME**命令检查环境变量设置，回显信息如下：

   ```
   /usr/lib/jvm/java-8-openjdk-amd64
   ```

   f. 执行**which jconsole**命令检查JDK安装。

   如果输出如下回显信息表示安装成功，如果未输出如下回显信息表示JDK安装失败。

   ```
   /usr/lib/jvm/java-8-openjdk-amd64/bin/jconsole
   ```

----结束

上传软件包

使用Mind Studio的安装用户将如下文件上传至Mind Studio安装服务器任意目录，本示例为上传到$HOME目录：

- mindstudio.tar.gz：Mind Studio软件包。
- Mmindstudio.tar.gz.asc：Mind Studio软件包校验文件。
- Ascend_DDK-{software version}-{interface version}-x86_64.ubuntu16.04.tar.gz：DDK软件包。
- Ascend_DDK-{software version}-{interface version}-x86_64.ubuntu16.04.tar.gz.asc：DDK软件包校验文件。

说明

- DDK包中{software version}和{interface version}详细说明请参见**表13-2**。
- 软件包与软件包校验文件必须放在同一目录下。

2.4 软件包完整性校验

为了防止软件包在传输过程中由于网络原因或存储设备原因出现下载不完整或文件破坏的问题，在执行安装前，建议您对软件包的完整性进行校验。

前提条件

Linux系统已经安装GnuPG工具。检查方法：

- 若已经安装GnuPG工具，在 Shell中输入 **gpg --version**命令，可看到如下所示的回显信息。

```
ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~$ gpg --version
gpg (GnuPG) 1.4.20
Copyright (C) 2015 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: ~/.gnupg
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
```

- 若没有安装GnuPG工具，则在GnuPG的官方网站**https://www.gnupg.org/**，按照网站的指引，完成工具安装。

配置 OpenPGP 公钥

在安装包所在$HOME目录下，使用Mind Studio的安装用户配置公钥，操作步骤如下：

步骤1 获取公钥文件。

说明

为区别Mind Studio和DDK使用的公钥，建议您将下载导入的公钥文件进行重命名或者导入不同的目录中，本例中采用重命名的方式，Mind Studio对应的公钥为：KEYS_mindstudio.txt，DDK对应的公钥为KEYS_DDK.txt。

- Mind Studio公钥获取方式：

  将如下公钥内容复制到KEYS_mindstudio.txt中，然后上传到Mind Studio所在Linux系统中，例如传到"/home/*username*/openpgp/keys"新建目录中。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1

mQENBF1AAUkBCACiCNoruz1vVNCGu8WrCOjVCFJMcVtkQRjRhDzeCnoSkaSCvNDx
hondF9ZEtk14iP+TcUlhjOHMMLP70qUpSylkVW0eeY4g5HkLYm50lGZwtsyVQ6t+
Tko/YgbAqLWXHT6XsPUGw6KpE2KRUU+RDGZvSJc5vxVV8HbA9Dz4k0y59QDYfRNJ
4sSdFWnOk+WSiNaU22/KunsUMF5WL1esxx+pYm69KYMvrkcfTkF70EqRUsEj8Tqo
29ZDL+zeyab06Jkw3Sak65Os9WVQvbmPD3ovpA7WYTMUYBlJHbqAF6mFizXgdMh8
M3AeTDei7b0X51gAVytZgR8jgnd+aehDccd5ABEBAAG0I01pbmQgU3R1ZGlvIDxt
aW5kc3R1ZGlvQGh1YXdlaS5jb20+iQE4BBMBAgAiBQJdQAFJAhsDBgsJCAcDAgYV
CAIJCgsEFgIDAQIeAQIXgAAKCRD1a6ERFnIeLyx3CACZDjOJK6lcgEUVwIWIxHZ+
bm1Yr+wOZyUoe38LS3KhcYiuQd5PhZ/63zlWPSuizFy1N62wVSR96GU0SXP9jiIs
/QurZgAMz23FGQqELmZ5Qzf5HPD18fh5vbOuMzCewN/SiUmFryqhcfOoxwB32pd8
kGutY6E/LuFDxA1CapItDOp1EBICRzq8pmxGZm4MkEjWJlGLLDYdK3Hn7kqWzmqw
K8ymrMPPR1k8vhLktor73NOURZlZJ5eWYRhKySXWdiL9DqPsZhubKQ7zadsf6BZu
zx5crBQzD6thGjuEfSixQQlKL6+8o19BTVIS11LkVNVMFKJkYgrk9XWuJvG1Q1mF
=8vb1
-----END PGP PUBLIC KEY BLOCK-----
```


- DDK公钥获取方式：

  a. 进入**OpenPGP下载页面**，单击下载链接，如**图2-2**所示，界面跳转到文件下载页面。

  图 2-2 单击下载文件

  文件名为“KEYS”的文件为公钥文件，如**图2-3**所示。

  图 2-3 选择 KEYS 文件

  b. 将下载的**KEYS.txt**文件重命名为KEYS_DDK.txt后，上传到Mind Studio所在linux系统中。

  例如传到"/home/*username*/openpgp/keys"新建目录中。

步骤2 导入公钥文件。

分别执行如下命令导入Mind Studio与DDK公钥。

- Mind Studio

  ```
  gpg --import "/home/username/openpgp/keys/KEYS_mindstudio.txt"
  ```

- DDK：

  ```
  gpg --import "/home/username/openpgp/keys/KEYS_DDK.txt"
  ```

  回显信息如下：

  ```
  ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~$ gpg --import "/home/ascend/openpgp/keys/KEYS_mindstudio.txt"
  gpg: key 16721E2F: public key "Mind Studio <mindstudio@huawei.com>" imported
  gpg: Total number processed: 1
  gpg:               imported: 1  (RSA: 1)
  ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~$ gpg --import "/home/ascend/openpgp/keys/KEYS_DDK.txt"
  gpg: key 27A74824: public key "OpenPGP signature key for Huawei software (created on 30th Dec,
  ```

```
2013) <support@huawei.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

说明

其中“/home/*username*/openpgp/keys”是公钥文件“KEYS”所在的绝对路径，username为Mind Studio安装用户名，请修改为实际路径。

步骤3 执行如下命令查看公钥导入结果。

```
gpg --fingerprint
```

回显信息如下：

```
ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~$ gpg --fingerprint
/home/ascend/.gnupg/pubring.gpg
-------------------------------
pub   2048R/16721E2F 2019-07-30
      Key fingerprint = 27D8 8F42 0ADF 4A94 456A  5AD0 F56B A111 1672 1E2F
uid                  Mind Studio <mindstudio@huawei.com>

pub   2048R/27A74824 2013-12-30
      Key fingerprint = B100 0AC3 8C41 525A 19BD  C087 99AD 81DF 27A7 4824
uid                  OpenPGP signature key for Huawei software (created on 30th Dec,2013)
<support@huawei.com>
```

步骤4 验证公钥。

- OpenPGP 公钥的合法性需要根据公钥的 ID、指纹、uid 等信息与发布公钥的主体进行合法性验证。当前对外发布的OpenPGP公钥信息如下：
  - Mind Studio公钥信息
    i. 公钥 ID：16721E2F
    ii. 公钥指纹(Key fingerprint)：27D8 8F42 0ADF 4A94 456A 5AD0 F56B A111 1672 1E2F
    iii. 用户 ID(uid)： Mind Studio <mindstudio@huawei.com>
  - DDK公钥信息
    i. 公钥 ID：27A74824
    ii. 公钥指纹(Key fingerprint)：B100 0AC3 8C41 525A 19BD C087 99AD 81DF 27A7 4824
    iii. 用户 ID(uid)：OpenPGP signature key for Huawei software (created on 30th Dec,2013) <support@huawei.com>

  完成信息核实后，可以对该公钥设置信任级别（Mind Studio和DDK设置信任级别方式相同，下面以Mind Studio为例进行说明）。

- 执行如下命令设置公钥的信任级别。
  - Mind Studio设置公钥信任级别命令：
    ```
    gpg --edit-key "Mind Studio" trust
    ```
  - DDK设置公钥信任级别命令：
    ```
    gpg --edit-key "OpenPGP signature key for Huawei software" trust
    ```

  屏幕回显信息如下，其中“Your decision?”后输入“5”，表示“I trust ultimately”；“Do you really want to set this key to ultimate trust? (y/N)”后输入“y”。

  ```
  ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~$ gpg --edit-key "Mind Studio" trust
  gpg (GnuPG) 1.4.20; Copyright (C) 2015 Free Software Foundation, Inc.
  This is free software: you are free to change and redistribute it.
  There is NO WARRANTY, to the extent permitted by law.
  ```

```
pub  2048R/16721E2F  created: 2019-07-30  expires: never       usage: SC
                     trust: unknown       validity: unknown
[ unknown] (1). Mind Studio <mindstudio@huawei.com>

pub  2048R/16721E2F  created: 2019-07-30  expires: never       usage: SC
                     trust: unknown       validity: unknown
[ unknown] (1). Mind Studio <mindstudio@huawei.com>

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu

Your decision? 5
Do you really want to set this key to ultimate trust? (y/N) y

pub  2048R/16721E2F  created: 2019-07-30  expires: never       usage: SC
                     trust: ultimate      validity: unknown
[ unknown] (1). Mind Studio <mindstudio@huawei.com>
Please note that the shown key validity is not necessarily correct
unless you restart the program.

gpg>
```

步骤5 执行**quit**命令退出。

----结束

软件包完整性校验

在安装包以及校验文件所在目录下，使用Mind Studio的安装用户分别执行如下命令，检测Mind Studio以及DDK软件包是否合法完整。

- Mind Studio：
  ```
  gpg --verify "mindstudio.tar.gz.asc"
  ```
- DDK：
  ```
  gpg --verify "Ascend_DDK-{software version}-{interface version}-<uihost arch.os>.tar.gz.asc"
  ```
  回显信息如下：
  ```
  ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~$ gpg --verify "mindstudio.tar.gz.asc"
  gpg: assuming signed data in `mindstudio.tar.gz'
  gpg: Signature made Tue 10 Dec 2019 09:51:16 PM EST using RSA key ID 16721E2F
  gpg: checking the trustdb
  gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
  gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
  gpg: Good signature from "Mind Studio <mindstudio@huawei.com>"
  ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~$ gpg --verify "Ascend_DDK-{software version}-{interface version}-<uihost arch.os>.tar.gz.asc"
  gpg: assuming signed data in `Ascend_DDK-{software version}-{interface version}-<uihost arch.os>.tar.gz'
  gpg: Signature made Tue 26 Nov 2019 09:15:07 PM EST using RSA key ID 27A74824
  gpg: checking the trustdb
  gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
  gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
  gpg: Good signature from "OpenPGP signature key for Huawei software (created on 30th Dec,2013) <support@huawei.com>"
  ```
- 返回信息中“16721E2F”为Mind Studio公钥ID，“27A74824”为DDK公钥ID。
- 提示信息返回“Good signature”且信息中无 WARNING 或 FAIL，表明此签名为有效签名，软件包完整性校验通过。

- 若提示信息存在 WARNING 或 FAIL，则表明验证不通过，请参见**2.11.3 软件包完整性校验返回WARNING或 FAIL**处理建议解决。

  说明

  - 操作时，请将Ascend_DDK-{software version}-{interface version}-<uihost arch.os>.tar.gz.asc替换为实际安装包对应的校验文件。
  - 软件包和软件包.asc文件必须放在同一个路径，才能进行完整性校验。

2.5 安装 Mind Studio

本节给出Mind Studio的安装场景以及安装方法。

须知

- 当前不支持在同一服务器使用同一用户安装多个Mind Studio，如果系统上同一用户下已存在Mind Studio，请参见**2.10 卸载Mind Studio与DDK**章节，使用该用户将原Mind Studio卸载后再执行安装。
- 请使用Mind Studio的安装用户ssh登录到Ubuntu系统，不支持使用其他用户登录后再切换到Mind Studio的安装用户。

前提条件

已完成**2.3 环境准备**和**2.4 软件包完整性校验**。

操作步骤

步骤1 解压Mind Studio软件包。

使用Mind Studio的安装用户在软件包所在路径执行如下命令，解压mindstudio.tar.gz软件包。

```
tar -zxvf mindstudio.tar.gz
```

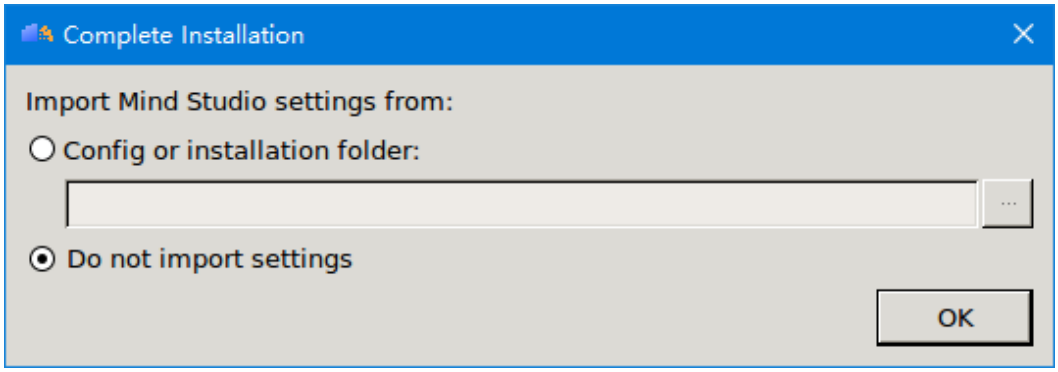
解压后包的内容以及说明请参见**表13-1**。

步骤2 使用Mind Studio的安装用户进入软件包解压后的MindStudio-ubuntu/bin目录，执行如下命令：

```
cd MindStudio-ubuntu/bin
./MindStudio.sh
```

步骤3 进入导入设置界面，如**图2-4**所示界面。

图 2-4 环境设置



- Config or installation folder：表示从自定义配置目录导入Mind Studio设置。
  自定义配置目录所在路径：“$HOME/.MindStudioMS-2.1/”，Mind Studio设置包括工程界面的个性化设置（例如背景色等信息）以及DDK版本信息。
- Do not import settings：不导入设置，若选择该选项，则创建新的配置文件，默认为该选项。

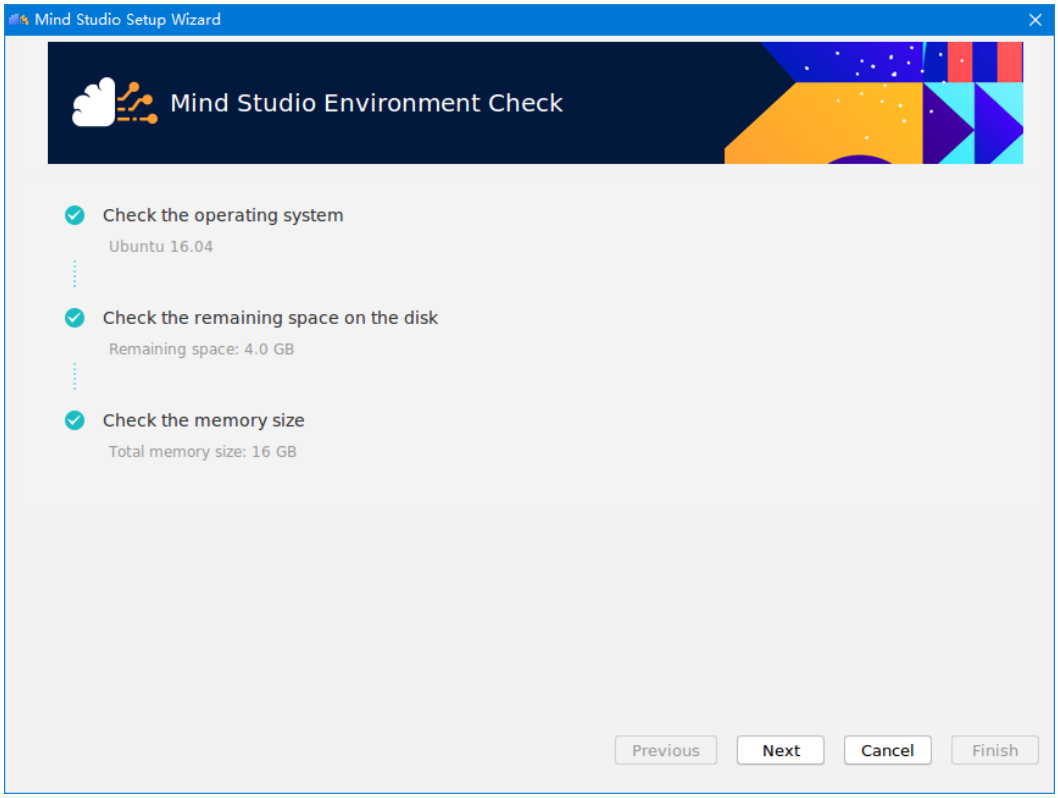
根据需要选择相关选项后，单击“OK”，进入下一步。

说明

- 若为用户首次安装，且Mind Studio安装包解压到$HOME目录，则图2-4界面中会出现“Previous version”参数。
- 若已经安装过低版本Mind Studio，则再次安装高版本时，则图2-4界面中也会出现“Previous version”参数，表示是否从低版本导入Mind Studio设置，您可以根据需要进行选择。
- 若已经安装过相同版本Mind Studio，且卸载前一版本时未删除“$HOME/.MindStudioMS-2.1/”目录，则安装下个版本时会从步骤步骤2直接进入步骤8。
- 若步骤4~步骤5安装过程中单击“Cancel”取消安装，则再次安装时图2-4不会出现。

步骤4 脚本执行后，在用户Ubuntu系统会弹出如图2-5所示硬件环境检查页面。

图 2-5 硬件环境信息检查



检查项包括如下内容：

1. 检查操作系统：

   操作系统为Ubuntu 16.04版本才允许安装，否则退出安装。

2. 检查操作系统磁盘剩余空间：

   需要剩余磁盘空间>=6G才允许安装，否则停止安装；检查通过后，会在检查项下面给出可用的磁盘空间。

3. 检查操作系统内存：

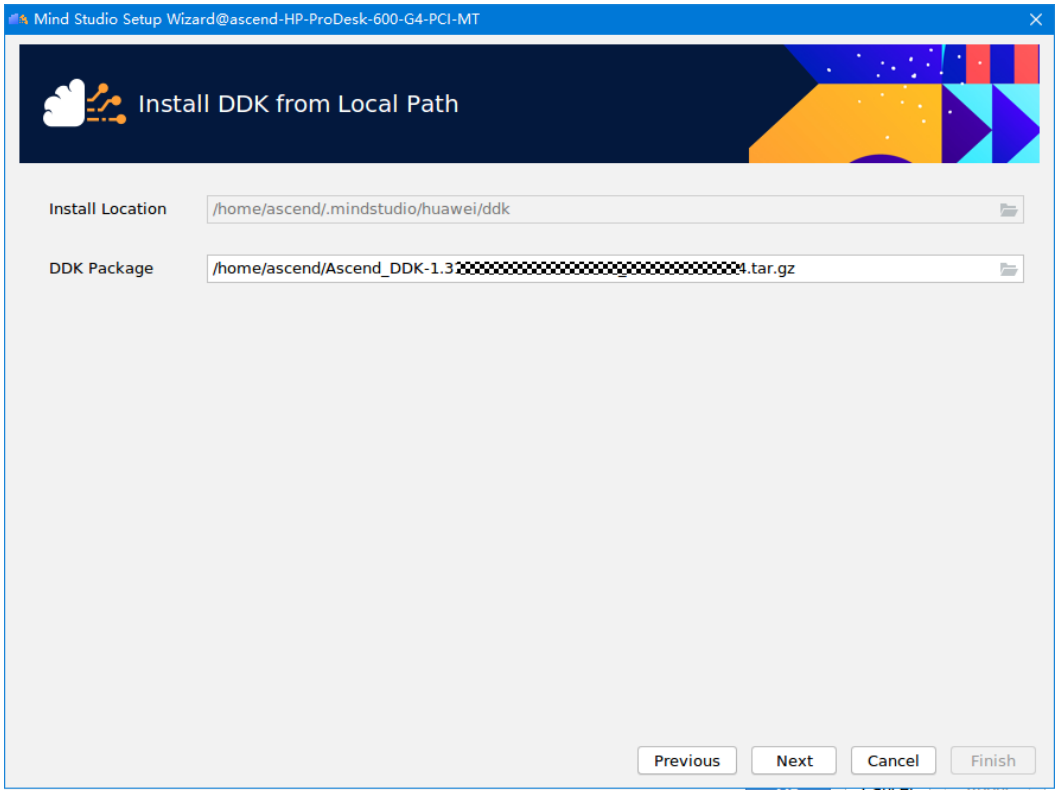
   需要操作系统内存>=4G才允许安装，否则停止安装；检查通过后，会在检查项下面给出总的内存空间。

说明

图2-5中的可用磁盘空间以及总的内存空间只是样例，请以实际服务器显示为准。

步骤5 单击“Next”进入DDK软件包选择页面，如图2-6所示。

图 2-6 DDK 安装包设置页面



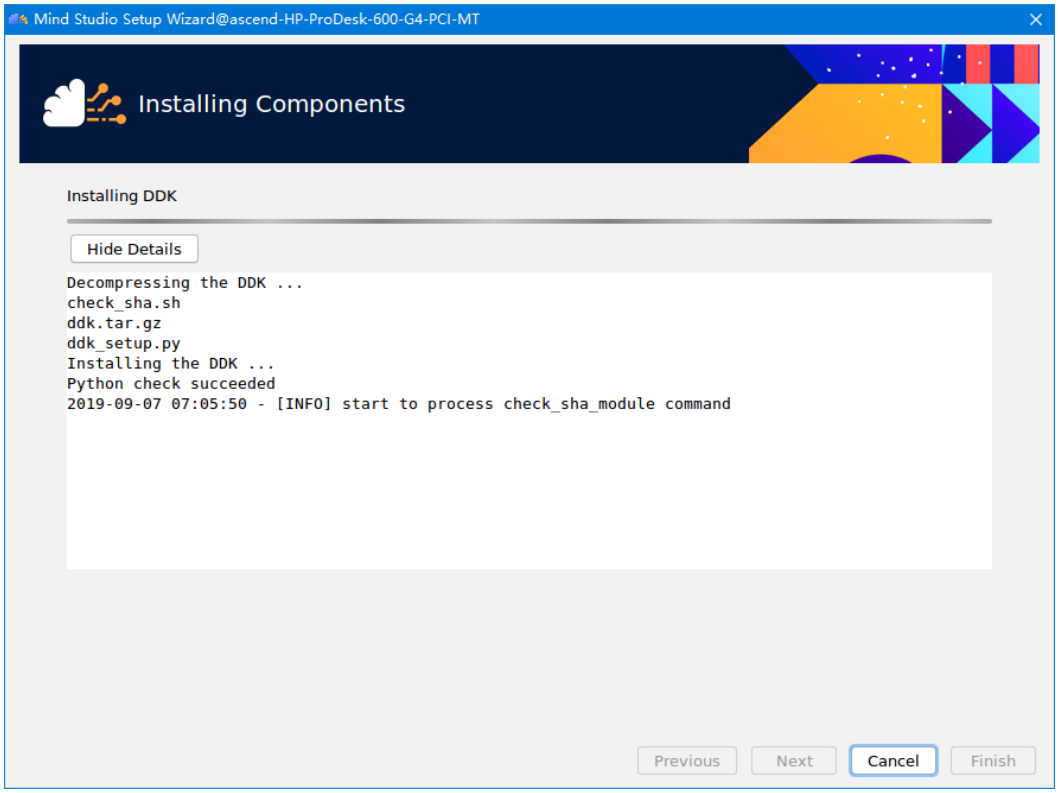
- Install Location：DDK在Mind Studio安装服务器上的安装路径，不支持修改。默认为：$HOME/.mindstudio/huawei/ddk
- DDK Package：DDK软件包。需要用户单击右侧的，然后选择**上传软件包**所在路径下的DDK软件包。

须知

Mind Studio安装完毕后，在使用期间禁止通过后台方式删除$HOME/.mindstudio/huawei/ddk安装目录，否则可能造成Mind Studio功能不可用。

步骤6 单击“Next ”解压并安装DDK软件包，如**图2-7**所示。

图 2-7 安装 DDK



单击“Show Details”可以查看DDK安装详细信息，单击“Hide Details”隐藏相关信息。

步骤7 若Details区域中出现如下信息表明DDK安装成功，此时“Finish”按钮变成可单击状态，其余按钮置灰。

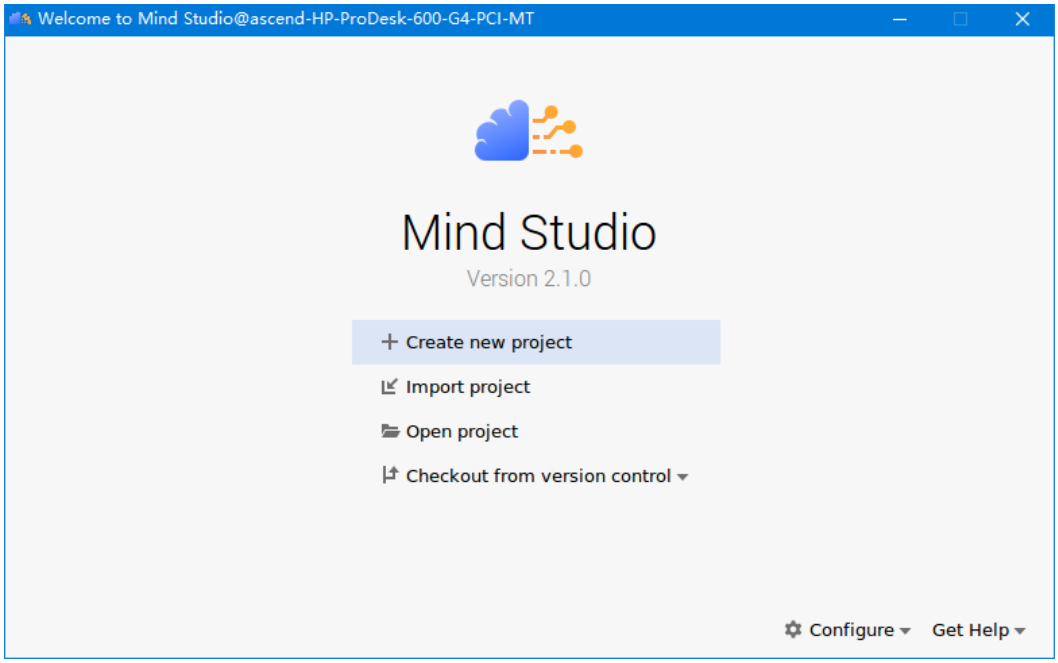
```
Successfully installed the DDK!
DDK installed
DDK {software version} is installed
Installation succeeded. Please click "Finish"
```

步骤8 单击“Finish”，进入如图2-8所示工程创建页面。如果没有报错信息且能正常进入如下界面，则表示Mind Studio安装成功。

安装成功后， 在$HOME路径会出现如下目录：

- .MindStudioMS-2.1/：
  - config/：用户配置数据目录，包括工程界面的个性化设置（例如背景色等信息）以及当前版本安装的DDK版本信息。
  - system/：系统目录，其中Mind Studio安装以及启动后的运行日志信息存储在“system/log/idea.log”文件中。
- .mindstudio/：DDK安装目录。
- AscendProjects：用户创建的工程。
- .mozilla：用户安装的浏览器所在目录。

图 2-8 工具安装完毕界面



界面参数解释如下，您可以通过单击“Create new project”创建新工程或选择“Import project”导入已有工程。

- Create new project：创建新工程。
- Import project：导入已有工程：将Web环境下的工程导入到本地IDE中。
- Open project：打开已有工程。
- Checkout from version control：从git等版本控制工具导出工程。

----结束

2.6 查询 Mind Studio 版本

安装完成后您可以通过如下方式查询所安装Mind Studio版本是否正确：

- Mind Studio安装完成后，在图2-9所示欢迎界面中，单击右下角的“Get Help > About”查看所安装的版本信息，如图2-9所示：

图 2-9 查询版本号

- 进入Mind Studio工程页面，单击菜单栏的“Help > About”弹出版本信息，查看版本信息是否正确。

说明

图2-9中的版本号只是样例，请以实际安装的版本为准。

2.7 启动 Mind Studio

如果用户关闭了Mind Studio，想要重新启动，则可以参见如下方式操作：

使用Mind Studio的安装用户，进入软件包解压后的“MindStudio-ubuntu/bin”目录，执行如下命令启动Mind Studio：

```
./MindStudio.sh
```

启动Mind Studio后，查询进程命令为：

```
ps -ef|grep idea
```

2.8 停止 Mind Studio

如果用户想要关闭Mind Studio，在Mind Studio运行界面直接单击窗口右上角的退出即可。

2.9 更新 Mind Studio

建议使用最新版本的Mind Studio，以便获得最新功能。如果尚未安装Mind Studio，请参见**2.5 安装Mind Studio**安装。

Mind Studio为绿色版软件，如果用户想更新版本，请参见**卸载Mind Studio**章节卸载已安装的Mind Studio， 然后参见**2.4 软件包完整性校验**和**2.5 安装Mind Studio**重新安装新版本Mind Studio。

2.10 卸载 Mind Studio 与 DDK

如果用户不再使用Mind Studio与DDK功能，则可以参见该章节将其卸载。如果用户只卸载DDK，则请慎重，可能会造成Mind Studio其他功能不可用。如果用户想更换DDK版本，则可以参见**8.3 DDK版本管理**章节安装新版本DDK。

卸载 Mind Studio

使用Mind Studio的安装用户，在linux系统的$HOME目录下删除Mind Studio安装包解压后的目录：MindStudio-ubuntu。

删除命令为：

```
rm -rf MindStudio-ubuntu
```

- 删除Mind Studio不会删除如下目录中的数据：
  - .MindStudioMS-2.1/：用户配置数据。
  - .mindstudio/：DDK安装目录。
  - AscendProjects：用户创建的工程。
  - .mozilla：用户安装的浏览器所在目录。
- 如果用户想要完全恢复Mind Studio初始配置，则请使用Mind Studio安装用户分别执行如下命令，删除用户配置数据：
  ```
  rm -rf .MindStudioMS-2.1/
  sudo rm -rf .mindstudio/
  ```

卸载 DDK

以Mind Studio安装用户执行如下卸载步骤，DDK安装完成后，根据**图2-6**所示，DDK安装目录为：“$HOME/.mindstudio/huawei/ddk”。

步骤1 登录Mind Studio安装服务器，进入DDK安装目录如下路径：

```
cd $HOME/.mindstudio/huawei/ddk/{software version}/ddk/scripts
```

步骤2 执行如下命令卸载DDK：

```
python ddk_setup.py uninstall
```

步骤3 界面会出现如下提示信息：

```
Please confirm that you want to clear this ddk path:/home/username/huawei/ddk? Please type Y or N
```

输入“Y”执行卸载，输入“N”取消卸载。

若提示如下信息，则说明DDK卸载成功：

```
Successfully uninstalled the DDK!
```

----结束

2.11 FAQ

2.11.1 安装完 Mind Studio 后，无法远程连接 host 为 CentOS 操作系统的服务器

问题描述

安装完Mind Studio后，无法远程连接到host侧为CentOS操作系统的服务器。

可能原因

CentOS系统防火墙是默认是开启的，CentOS系统重启后也会自动开启防火墙，导致Log Port和Daemon Port端口被防火墙阻隔。

解决方法

- Daemon Port（22118）端口解决方法：

  以HwHiAiUser用户登录Host侧服务器，执行如下命令检查Daemon Port端口是否被阻隔：

  ```
  IDE-daemon-client --host hostIP:22118 --detect
  ```

  若命令执行后卡住未返回SUCCESS，则请切换到root用户执行如下命令配置防火墙策略（允许22118端口以ssh方式连接Host侧服务器）：

  ```
  iptables -I INPUT -p tcp --dport 22118 -j ACCEPT
  ```

  配置完成后，执行**iptables -L**命令查看结果。

  ```
  # iptables -L
  Chain INPUT (policy ACCEPT)
  target     prot opt source               destination
  ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:22118
  ```

- Log Port（18080）端口解决方法：

  以HwHiAiUser用户登录Host侧服务器，然后切换到root用户执行如下命令允许访问18080端口：

  ```
  iptables -I INPUT -p tcp --dport 18080 -j ACCEPT
  ```

2.11.2 Mind Studio 或 DDK 安装过程中提示 pip2 或 pip 不可用

问题描述

Mind Studio或DDK安装过程中提示pip2或pip不可用，并退出安装，提示信息如图2-10、图2-11所示。

图 2-10 pip2 不可用提示信息

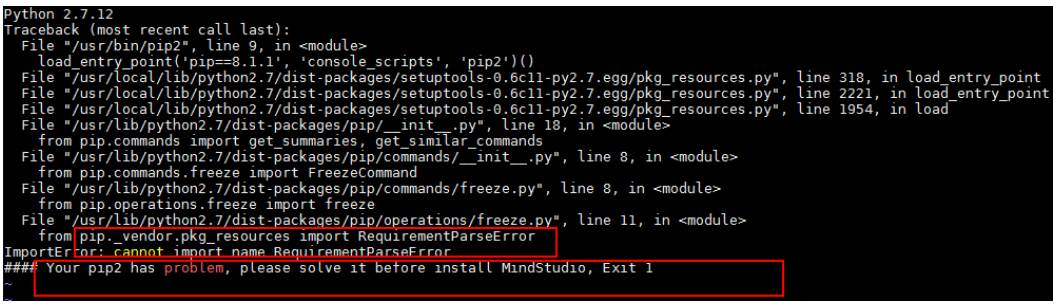


图 2-11 pip 不可用提示信息

```
Traceback (most recent call last):
  File "/usr/bin/pip", line 9, in <module>
    from pip import main
ImportError: cannot import name main
```

可能原因

该问题可能是由于pip重新安装过程中，pip2没有正确更新。

解决方法 1

步骤1 **su root**切换到root用户，执行**pip list**，如果返回信息没有错误提示，说明pip可用；执行**pip2 list**提示错误，说明pip2不可用。

步骤2 root用户下删除pip2，删除命令为：**rm /usr/bin/pip2**。

步骤3 将pip2软连接到pip，命令为：**ln -s pip pip2**。

步骤4 再次执行**pip2 list**，若返回信息没有错误提示，则说明问题解决。

若pip、pip2仍不可用，则参见**解决方法2**。

----结束

解决方法 2

如果依赖安装过程中pip安装异常，请依次执行如下命令。

```
sudo apt-get remove python-pip python3-pip
wget https://bootstrap.pypa.io/get-pip.py
python get-pip.py --user
python3 get-pip.py  --user
```

2.11.3 软件包完整性校验返回 WARNING 或 FAIL

软件包完整性校验如果返回WARNING或 FAIL，则表示验证未通过，请参见**表2-4**处理建议解决。

表 2-4 场景举例

验证结果场景	输出信息举例	验证结果	处理建议
签名验证通过，没有异常	gpg: Signature made Thu Jan 9 15:29:06 2014 CST using RSA key ID 27A74824 gpg: Good signature from "OpenPGP signature key for Huawei software (created on 30th Dec,2013) <support@huawei.com>"	PASS	NA

验证结果场景	输出信息举例	验证结果	处理建议
签名验证失败	gpg: Signature made Thu Jan 9 15:29:06 2014 CST using RSA key ID 27A74824 gpg: BAD signature from "OpenPGP signature key for Huawei software (created on 30th Dec,2013) <support@huawei.com>"	FAIL	重新下载目标文件。
找不到公钥	gpg: Signature made Thu Jan 9 15:20:01 2014 CST using RSA key ID 27A74824 gpg: Can't check signature: public key not found	FAIL	重新下载公钥，请参见“配置OpenPGP公钥-配置公钥”章节。
签名验证通过，但是公钥没有被设置为完全信任	gpg: Signature made Thu Jan 9 15:29:06 2014 CST using RSA key ID 27A74824 gpg: Good signature from "OpenPGP signature key for Huawei software (created on 30th Dec,2013) <support@huawei.com>" gpg: WARNING: This key is not certified with a trusted signature! gpg: There is no indication that the signature belongs to the owner. Primary key fingerprint: B100 0AC3 8C41 525A 19BD C087 99AD 81DF 27A7 4824	WARNING	确认KeyID为27A74824后，将华为公钥设置为可信，请参见“配置OpenPGP公钥-配置公钥-验证公钥”章节。
找不到对应的源文件	gpg: no signed data gpg: can't hash datafile: No data	FAIL	重新下载目标文件。
签名已到期	gpg: Signature made 04/24/13 10:50:29 CST using RSA key ID 133B64E5 gpg: Expired signature from " OpenPGP signature test key <support@huawei.com>" gpg: Signature expired 04/25/13 10:50:29 CST	FAIL	下载更新过签名的目标文件。
签名验证通过，但是公钥已被撤销	gpg: Signature made 06/13/13 11:14:49 CST using RSA key ID 133B64E5 gpg: Good signature from " OpenPGP signature test key <support@huawei.com>" gpg: WARNING: This key has been revoked by its owner! gpg: This could mean that the signature is forged. gpg: reason for revocation: Key is no longer used gpg: revocation comment:	WARNING	下载最新公钥和更新了签名的目标文件。

验证结果场景	输出信息举例	验证结果	处理建议
源文件找不到对应的签名文件	无	WARNING	下载目标文件对应的签名文件。

3 编译环境配置

3.1 配置流程

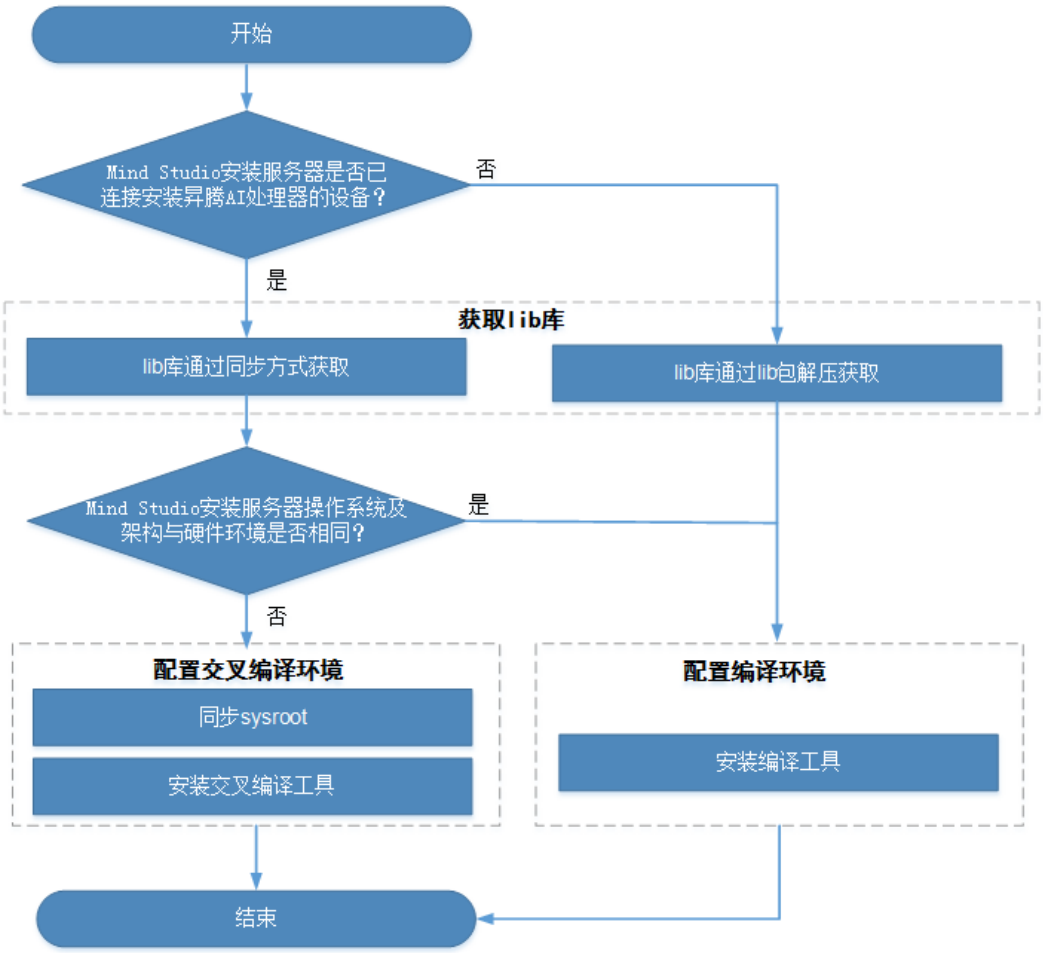
3.2 获取lib库

3.3 配置编译环境

3.1 配置流程

安装完Mind Studio并进入相关工程页面后，如果用户使用算子工程或应用工程开发了相关程序，需要进行工程的编译和运行，用于生成相关二进制文件，进行工程编译前，需要配置编译环境，配置流程如**图3-1**所示。

图 3-1 编译环境配置流程



关键概念解释如下：

- 交叉编译：是指在当前操作系统（Ubuntu16.04+x86的Mind Studio安装服务器）编译出在其他操作系统（Ubuntu16.04+Arm64、CentOS7.4+x86、CentOS7.6+Arm64）下运行的程序。
- sysroot：是安装昇腾AI处理器硬件环境（Ubuntu16.04+Arm64、CentOS7.4+x86、CentOS7.6+Arm64）的头文件和库目录，当用户在Mind Studio安装服务器进行交叉编译时，需要依赖硬件环境的sysroot。
- 硬件环境：安装昇腾AI处理器的环境，分为**Host/Device**，后面统称为硬件环境。

配置编译环境包括如下几方面，如果不涉及编译运行相关操作，则该章节请忽略。

获取 lib 库

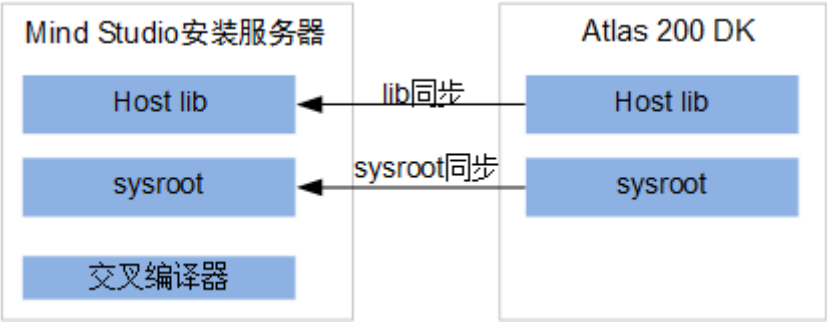
编译程序时所依赖的硬件环境的lib库。lib库获取方式有两种，用户根据**表3-1**使用场景获取其中一种方式获取即可。

表 3-1 lib 库获取方式

获取方式	使用场景	说明
3.2.1 lib库通过同步方式获取	• Mind Studio安装服务器已连接安装昇腾AI处理器的设备。 • 硬件环境所安装的软件包版本与DDK版本保持一致。	使用Mind Studio设备管理下面的同步按钮，将硬件环境的lib库同步到Mind Studio安装服务器。Atlas 200 DK形态lib获取只能通过该方式获取，其同步原理请参见图3-2。Atlas 300形态同步原理请参见图3-3~图3-5。 **说明** • 同步lib时，硬件环境只能被一台Mind Studio安装服务器所连接，如果多台Mind Studio安装服务器同时连接硬件环境进行同步，可能会同步失败。 • 请确保Mind Studio安装服务器剩余足够磁盘空间，至少为3G，同步lib过程中，如果有同步sysroot场景，由于sysroot文件过大，可能会造成磁盘空间不足，造成同步失败。
3.2.2 lib库通过lib包解压获取	• Mind Studio安装服务器操作系统及架构与lib包中Host侧操作系统架构相同。 • 用户对lib库路径配置有个性化要求。 • lib包版本与DDK版本保持一致。	lib包放在Mind Studio安装服务器 DDK安装路径下并解压。

- Atlas 200 DK 同步原理

图 3-2 Atlas 200 DK 同步原理

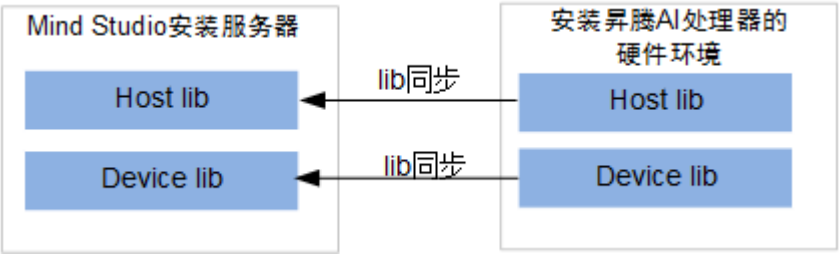


- Atlas 300 同步原理
  - Mind Studio安装在Host服务器

图 3-3 Atlas 300 同步原理_场景 1

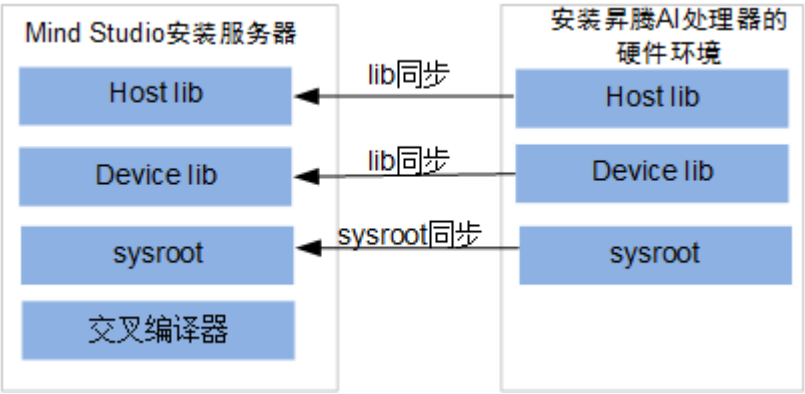
– Mind Studio安装服务器（Ubuntu_x86）与安装昇腾AI处理器的硬件环境（Ubuntu_x86）相同，且Mind Studio与Host在不同服务器。

图 3-4 Atlas 300 同步原理_场景 2



– Mind Studio安装服务器（Ubuntu_x86）与安装昇腾AI处理器硬件环境（CentOS_x86或CentOS_Arm）不同，且Mind Studio与Host在不同服务器。

图 3-5 Atlas 300 同步原理_场景 3



配置编译环境

不同产品形态下lib库获取方式不同，配置编译环境方法不同，如下表所示。

- Atlas 200 DK形态下的交叉编译环境配置

表 3-2 交叉编译环境配置

lib库获取方式	编译场景	Mind Studio安装服务器操作系统及架构	硬件环境操作系统及架构	编译环境配置
lib库通过同步方式获取，且Mind Studio安装服务器操作系统以及架构与Atlas 200 DK硬件环境不同	配置交叉编译环境	Ubuntu16.04+x86	Ubuntu16.04+Arm64	**1.同步sysroot**。 **2.安装交叉编译工具**。

- Atlas 300形态下的编译环境配置

表 3-3 不同 lib 库获取方式下的编译环境配置

lib库获取方式	编译场景	Mind Studio安装服务器操作系统及架构	硬件环境操作系统及架构	编译环境配置
lib库通过同步方式获取，且Mind Studio安装服务器操作系统以及架构与Atlas 300硬件环境不同	配置交叉编译环境	Ubuntu16.04+x86	CentOS7.4+x86、 CentOS7.6+Arm64	**1.同步sysroot**。 **2.安装交叉编译工具**。
• lib库通过同步方式获取，且Mind Studio安装服务器操作系统以及架构与Atlas 300硬件环境相同 • lib库通过lib包解压获取	配置编译环境		Ubuntu16.04+x86	安装5.4.0版本g++编译工具，请使用Mind Studio安装用户，在Mind Studio安装服务器执行如下命令： `sudo apt-get install g++`

3.2 获取 lib 库

3.2.1 lib 库通过同步方式获取

安装完Mind Studio后，如果用户进行编译运行相关操作，则需要参见该章节，将硬件环境的lib库同步到Mind Studio安装服务器。

前提条件

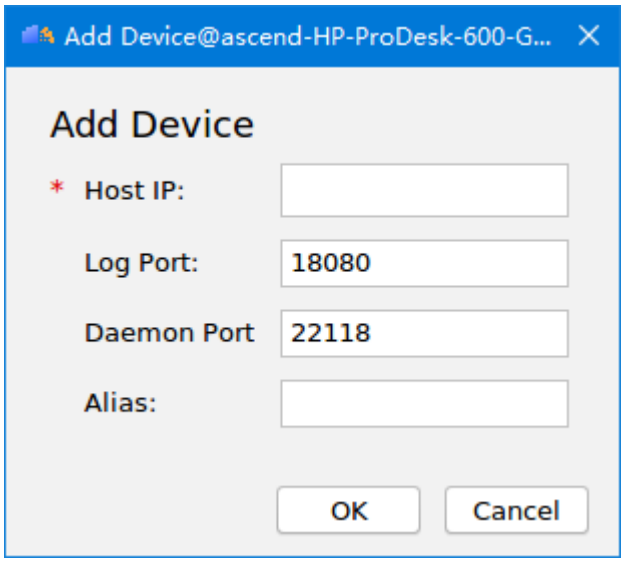
- 已经完成**2 安装**。

- 请确保DDK版本号与硬件环境所安装的软件包版本号一致。

同步步骤

步骤1 在Mind Studio工程界面依次选择“Tools > Device Management”，弹出设备管理界面，单击右侧工具栏的 + 增加设备，如图3-6所示。

图 3-6 增加设备



参数说明如表3-4所示。

表 3-4 增加设备界面参数

参数	说明
Host IP	Host侧的IP地址。
Log Port	Log端口号，取值范围为[18000~18080]，默认为18080。
Daemon Port	IDE_Daemon_client与IDE_Daemon_host通信使用的端口号，取值范围为[20000~25000]，默认为22118。
Alias	设备的别名，如果有多个设备时，方便用户管理设备信息。

说明

添加完表3-4设备参数，单击OK后，还会弹出增加设备进度条，刷新设备信息，该进度条根据用户网络状况，出现时间长短不定，待获取到设备信息后进入图3-7。

步骤2 填写相关参数后单击“OK”，所增加的设备会更新到“Available Device”区域，如图3-7所示。

图 3-7 设备增加完成状态

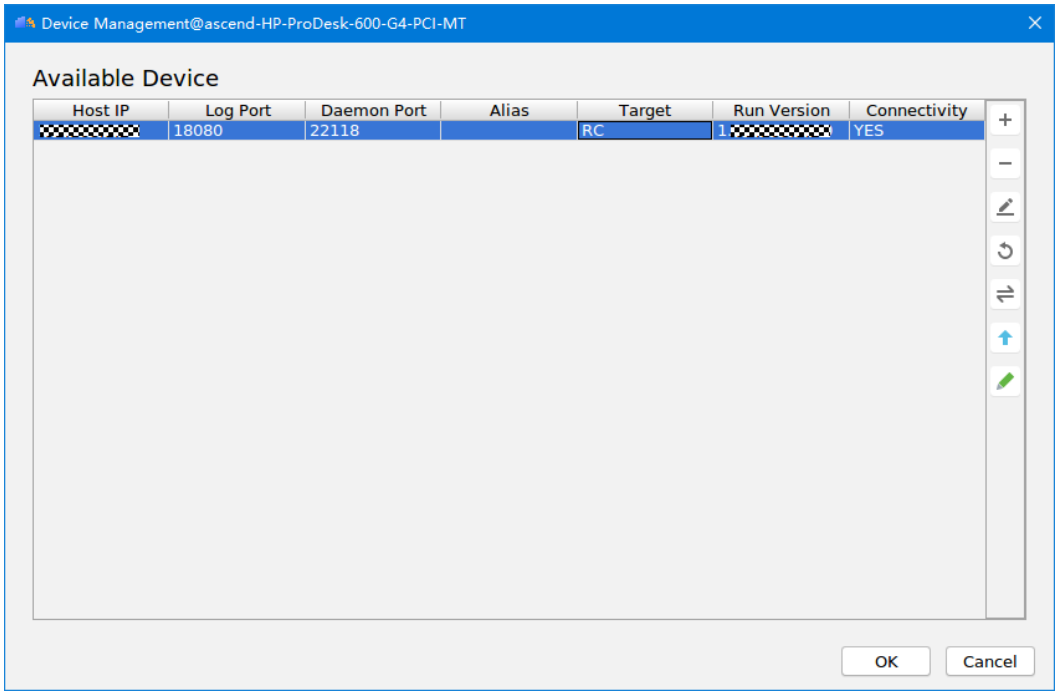
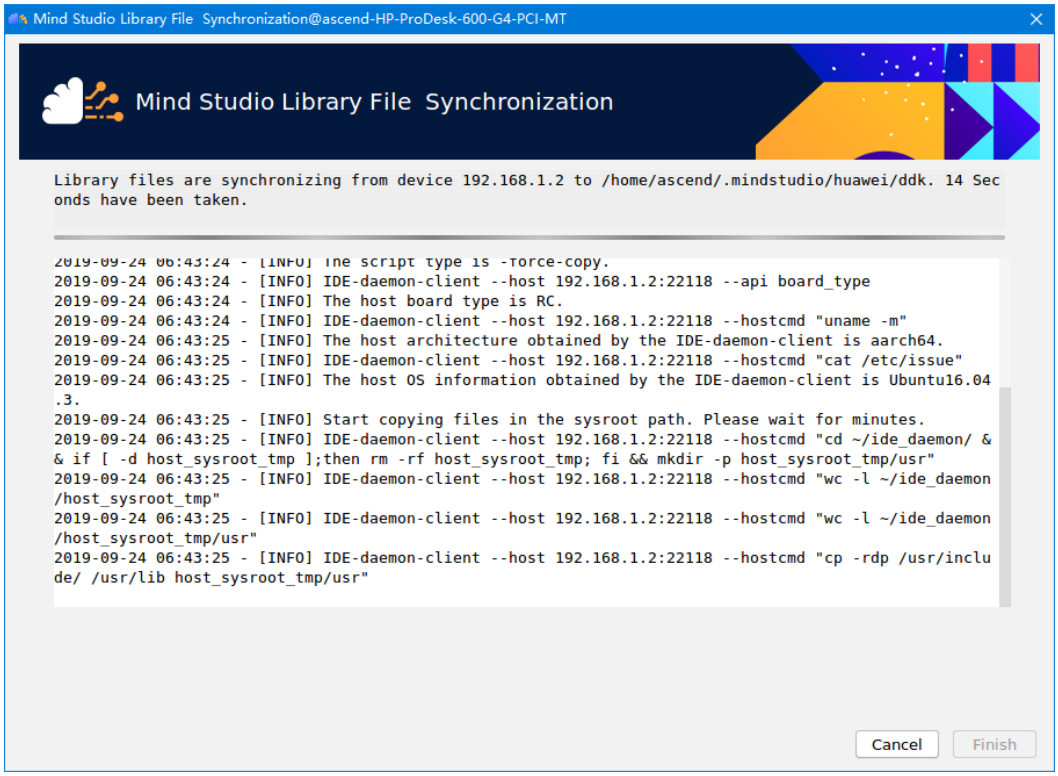


表 3-5 设备增加完成状态界面参数

参数	说明
Target	设备类型： ● EP：Atlas 300形态 ● RC：Atlas 200 DK形态
Run Version	设备软件包版本号。
Connectivity	设备连接状态： ● YES：连接成功。 ● NO：连接失败。 – 如果增加完设备，Connectivity始终显示NO，则请参见**异常处理**依次进行排查。 – 如果Host为CentOS操作系统，Mind Studio无法连接该环境时，请参见**2.11.1 安装完Mind Studio后，无法远程连接host为CentOS操作系统的服务器**解决。

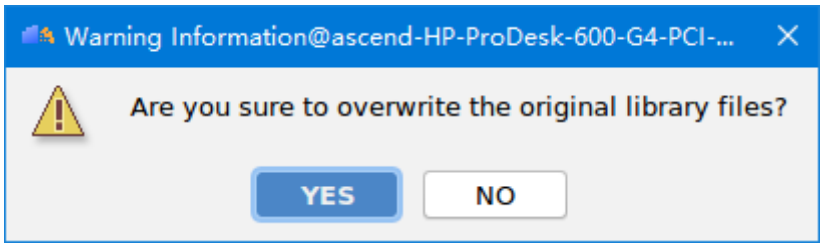
步骤3 选中增加完成的设备，单击右侧的 ⇌ 同步lib。弹出**图3-8**所示界面。

图 3-8 同步界面



- 若Mind Studio安装服务器“$HOME/.mindstudio/huawei/ddk”路径下已经有之前同步过的lib文件，会弹出如图3-9所示界面。

  图 3-9 覆盖 lib 库文件信息

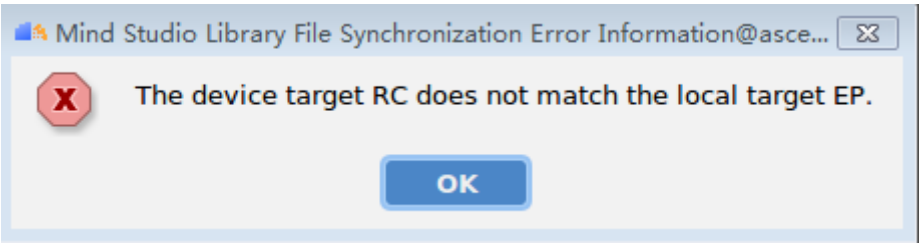
  

  单击“YES”，启动同步，覆盖原有Lib内容；单击“NO”，取消同步，返回设备管理界面。

- 若工程的类型与需要同步的硬件设备类型不一致，会出现“The device target xx does note match the local target xx”的错误提示。

  如图3-10所示，工程的Target为Atlas 300，设备为Atlas 200 DK，则需要选中工程，右键选择**Edit Operators Property**，修改工程属性的T**arget**为**Atlas 200 DK**。

图 3-10 同步失败错误提示



- 同步lib时，脚本会自动检测Host IP所连接的设备信息：
  - 如果为Atlas 300形态，则会将Host以及Device侧lib库同步到Mind Studio所在服务器。
  - 如果为Atlas 200 DK形态，则会将Host侧（Host与Device合并）lib库同步到Mind Studio所在服务器。

步骤4 同步完成后且窗口中提示如下信息，则说明同步lib成功，此时Finish按钮变成可单击状态，其余按钮置灰。图3-8上方给出同步后lib库在Mind Studio所在服务器路径，并给出同步所耗费时间。

```
Library file synchronization succeeded. Please click "Finish".
```

步骤5 单击“Finish”完成同步，返回设备管理界面。

同步的lib库会按照host侧和device侧操作系统架构、操作系统以及操作系统版本在Mind Studio安装服务器*{x-compile path}*路径创建对应目录，分别存储host侧lib库、device侧lib库。lib库存储路径为：

- Atlas 300形态：则会将Host以及Device侧lib库同步到Mind Studio安装服务器
  - Host侧lib库存储路径：{x-compile path}/{version}/形态/host-{host arch}_{host OS}/lib
  - Device侧lib库存储路径：{x-compile path}/{version}/形态/device -{device arch}_{device OS}/lib
- Atlas 200 DK形态：则会将Host侧lib库同步到Mind Studio安装服务器

  Host侧lib库存储路径：{x-compile path}/{version}/形态/host-{host arch}_{host OS}/lib

参数说明如表3-6所示。

表 3-6 lib 库存储路径对应参数说明

参数	说明
{x-compile path}	在Mind Studio安装服务器存放路径，默认为：$HOME/.mindstudio/huawei/ddk，不支持修改。
{version}	lib包版本。
形态	包括如下两种形态： RC：Atlas 200 DK形态 EP：Atlas 300形态
{host arch}	Host侧操作系统架构，例如x86_64。
{device arch}	Device侧操作系统架构，例如aarch64。

参数	说明
{host OS}	Host侧操作系统，例如Ubuntu16.04。
{device OS}	Device侧操作系统，例如miniOS，Ubuntu16.04.3。

图3-11为形态为EP时同步成功的样例。图3-12为形态为RC时同步成功的样例。

图 3-11 lib 同步成功样例_EP

```
ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~/.mindstudio/huawei/ddk/1.31.[redacted]9/EP$
total 16
drwxrwxr-x 4 ascend ascend   4096 Aug 29 03:45 ./
drwxrwxr-x 3 ascend ascend   4096 Aug 29 03:44 ../
drwxrwxr-x 3 ascend ascend   4096 Aug 29 03:45 device-aarch64_miniOS/
drwxrwxr-x 3 ascend ascend   4096 Aug 29 03:44 host-x86_64_Ubuntu16.04/
```

图 3-12 lib 同步成功样例_RC

```
ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~/.mindstudio/huawei/ddk/1.31.T[redacted]0/RC$ ll
total 12
drwxr-x--- 3 ascend ascend 4096 Sep 24 06:44 ./
drwxr-x--- 4 ascend ascend 4096 Sep 24 06:44 ../
drwxr-x--- 3 ascend ascend 4096 Sep 24 06:44 host-aarch64_Ubuntu16.04.3/
```

----结束

3.2.2 lib 库通过 lib 包解压获取

使用Mind Studio安装用户，将2.2 下载软件包中获取的lib包上传到Mind Studio安装服务器DDK安装路径下并解压，获得lib库。

DDK安装路径为：$HOME/.mindstudio/huawei/ddk/{software version}/，详情请参见2.5 安装Mind Studio中的步骤5。

lib包说明请参见13.1 软件包说明 ，lib包的完整性校验请参见2.4 软件包完整性校验中的DDK软件包校验部分。

说明

第一次解压完成后再次解压获取lib前，或第一次解压完成后再使用3.2.1 lib库通过同步方式获取方式获取lib前，请使用Mind Studio安装用户在Mind Studio安装服务器lib解压后的路径，执行如下命令，确保后续解压或同步能覆盖上次解压的内容：

```
chmod u+w -R EP
或
chmod u+w -R RC
```

请根据具体形态选择相应命令。

3.3 配置编译环境

lib库获取方式不同，编译环境配置方法不同，下面分别介绍。

通过设备同步获取 lib 后配置编译环境

根据Mind Studio安装服务器操作系统及架构与硬件环境是否相同，编译环境分为如下两种场景：

- 若Mind Studio安装服务器操作系统及架构与硬件环境相同，则配置编译环境。方法如下：

  请使用Mind Studio安装用户，在Mind Studio安装服务器安装5.4.0版本g++编译工具，命令为：

  ```
  sudo apt-get install g++
  ```

- 若Mind Studio安装服务器操作系统及架构与硬件环境不同，则配置交叉编译环境。方法如下：

  a. 同步sysroot

  在执行**3.2.1 lib库通过同步方式获取**时，会同步把硬件环境的sysroot同步到Mind Studio所在服务器，同步完成后，会按照Host侧操作系统架构、操作系统以及操作系统版本在Mind Studio安装服务器创建对应目录，存储sysroot。

  sysroot存储路径为：{x-compile path}/sysroot/{arch}_{os}_{os_version}

  表 3-7 sysroot 存储路径对应参数说明

参数	说明
{x-compile path}	sysroot在Mind Studio安装服务器存放路径，默认为：$HOME/.mindstudio/huawei/ddk，不支持修改。
{arch}	Host侧操作系统架构，例如aarch64。
{os}	Host侧操作系统，例如Ubuntu。
{os_version}	Host侧操作系统版本，例如16.04.3。

  图3-13为形态为RC时同步成功的样例。

  图 3-13 sysroot 同步成功样例

  ```
  ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~/.mindstudio/huawei/ddk/sysroot$ ll
  total 12
  drwxr-x--- 3 ascend ascend 4096 Sep 24 06:43 ./
  drwxr-x--- 4 ascend ascend 4096 Sep 24 06:43 ../
  drwxr-x--- 4 ascend ascend 4096 Sep 24 06:44 aarch64_Ubuntu16.04.3/
  ```

  b. 安装交叉编译工具

  交叉编译环境工具根据硬件环境操作系统以及架构不同，所需交叉编译工具不同，详情如下。请使用Mind Studio安装用户，在Mind Studio安装服务器执行安装命令：

  - Atlas 200 DK形态

表 3-8 交叉编译工具使用命令

Mind Studio安装服务器环境	设备侧操作系统及架构	交叉编译工具	对应产品形态
Ubuntu16.04+x86	Ubuntu16.04+Arm64	安装命令： `sudo apt-get install g++-aarch64-linux-gnu` 编译工具及版本：aarch64-linux-gnu-g++ 5.4.0	Atlas 200 DK形态

- Atlas 300形态

表 3-9 交叉编译工具使用命令

Mind Studio安装服务器环境	设备侧操作系统及架构	交叉编译工具	对应产品形态
Ubuntu16.04+x86	CentOS7.4+x86	安装命令： `sudo apt-get install -y g++-4.8` 编译工具及版本：g++-4.8 4.8.5	Atlas 300形态
	CentOS7.6+Arm64	安装命令： `sudo apt-get install -y g++-4.8-aarch64-linux-gnu` 编译工具及版本：aarch64-linux-gnu-g++-4.8 4.8.5	Atlas 300形态

通过解压 lib 包后配置编译环境

若lib库通过该方式获取，则配置编译环境只需安装5.4.0版本g++编译工具即可，请使用Mind Studio安装用户，在Mind Studio安装服务器执行如下命令：

```
sudo apt-get install g++
```

4 快速入门

4.1 创建首个AI应用
4.2 快速体验首个算子工程

4.1 创建首个 AI 应用

该章节带您在5分钟内快速创建一个简单的基于分类网络的AI应用样例，并快速熟悉我们的应用开发流程。

该样例通过输入三张图片，经过resnet18分类网络进行推理计算，得出图片的分类结果。

工程创建

步骤1 进入工程创建页面。

- 首次登录**Mind Studio**：在**Welcome to Mind Studio**窗口中，单击**Create New Project**。
- 非首次登录**Mind Studio**：在顶部菜单栏中选择**File > New > Project...**。

步骤2 在**New Project**窗口中，选择**Ascend App**，配置工程相关参数如表4-1所示。

表 4-1 工程参数说明

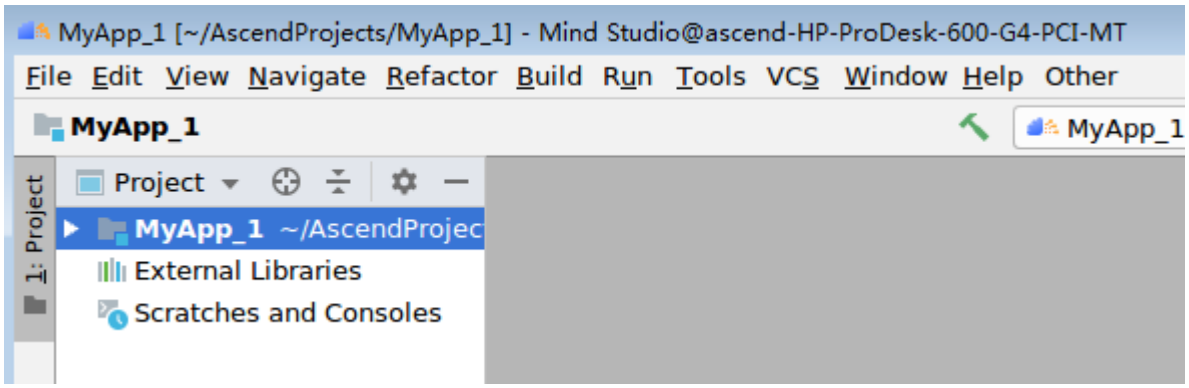
参数	说明
Name	工程名称，自行配置。 要求配置为以字母开头的字符串，最大长度支持64位。仅支持字母、数字、中划线、下划线。
Description	工程描述信息，自行配置。
DDK Version	当前DDK的版本号。
Target	运行环境，请根据实际硬件环境进行选择。
Project location	工程的存储路径，用户可自定义。

步骤3 单击**Next**，**Project Type**选择Classification Network。

步骤4 单击**Finish**，完成工程创建。

界面显示当前工程页面，如图4-1所示。

图 4-1 工程页面



步骤5 展开工程名查看工程目录结构，如表4-2所示。

说明

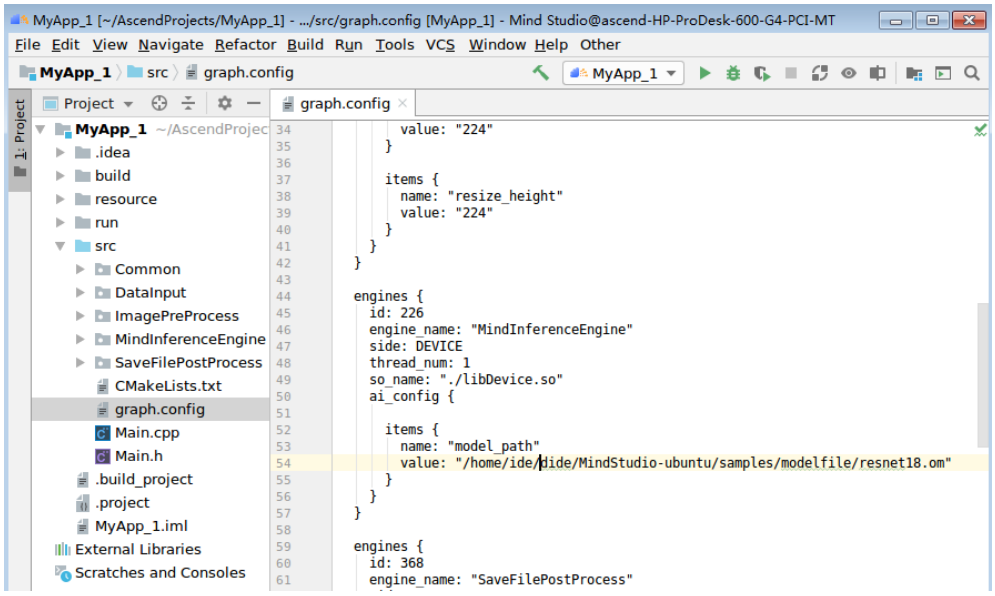
作为首次体验，无需对这些文件进行编辑。

表 4-2 Classification Network 目录结构

一级目录	二级目录	说明
resource	data	数据集文件。
src	Common	公共结构体。
	DataInput	数据读取Engine，包括实现文件和头文件。
	ImagePreProcess	数据预处理Engine，包括实现文件和头文件。
	MindInferenceEngine	推理Engine，包括实现文件和头文件。
	SaveFilePostProcess	后处理Engine，包括实现文件和头文件。
	graph.config	Graph配置文件，用于定义Engine节点间的串接关系和Engine节点属性（运行该节点所需的参数）。
	Main.cpp	主函数实现文件。
	Main.h	主函数头文件。
	CMakeLists.txt	编译脚本。
.project	-	工程信息文件，包含工程类型、工程描述、运行目标设备类型以及DDK版本。
其他目录为工具框架依赖，和工程无关，用户无需关注。		

该工程需要的能够适配昇腾AI处理器的离线模型在如图4-2所示的路径下，您无需自行准备。

图 4-2 离线模型文件路径



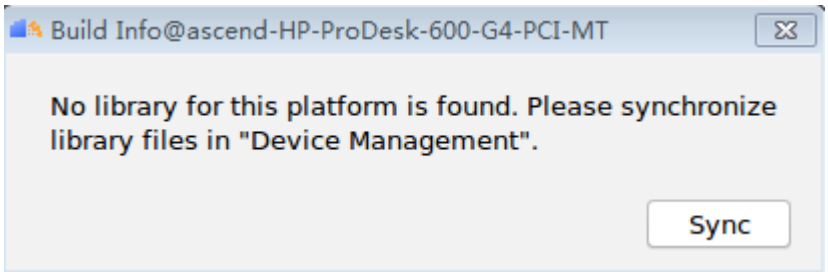
----结束

工程编译

步骤1 在**Mind Studio**工程界面，依次选择**Build > Edit Build Configuration...**，进入编译配置页面。

如果出现如下提示，说明编译依赖的lib库没有从运行环境同步到开发环境上，请单击**Sync**进入设备管理界面，然后参考**3 编译环境配置**进行相关操作。

图 4-3 未同步 lib 库



正常情况下进入编译配置页面，如图4-4所示，请参考表4-3进行编译配置。

图 4-4 编译配置页面

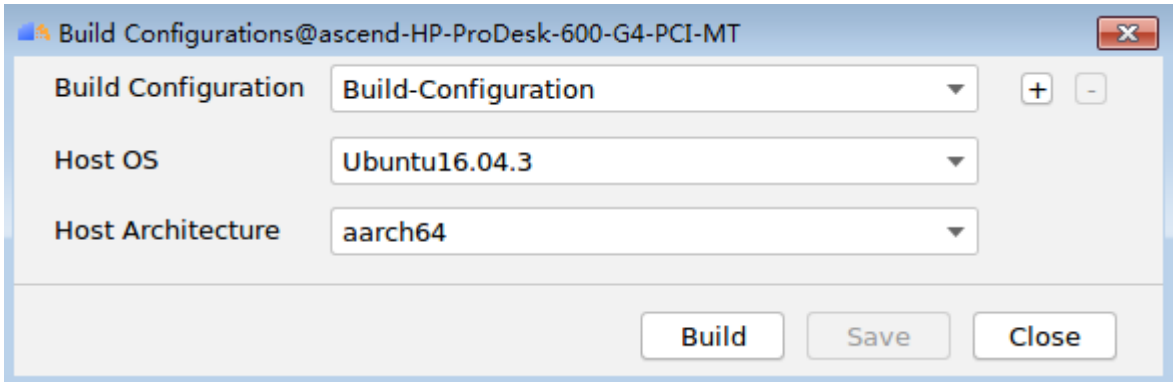


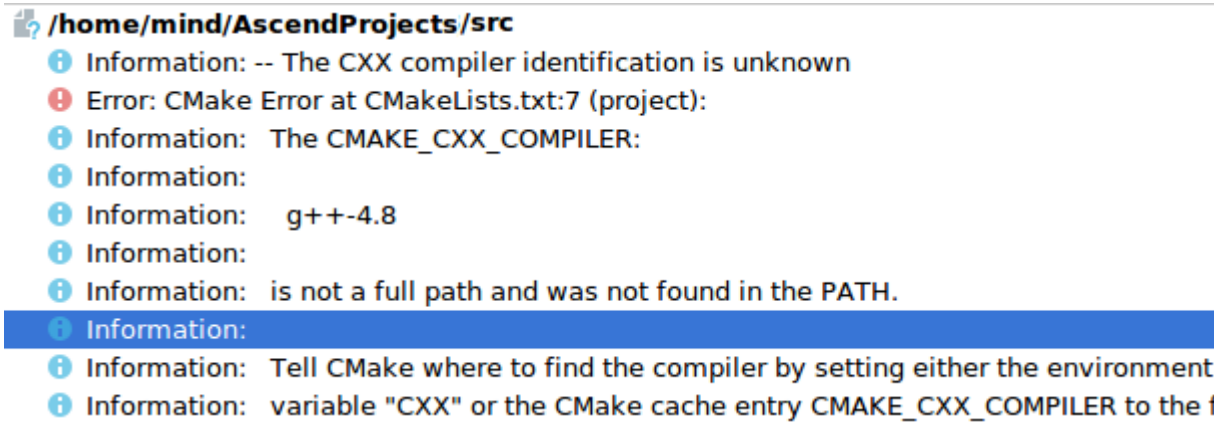
表 4-3 编译配置参数说明

参数	说明
Host OS	选择硬件环境的Host侧操作系统。
Host Architecture	选择硬件环境的Host侧操作系统架构。

步骤2 单击**Build**进行工程编译。

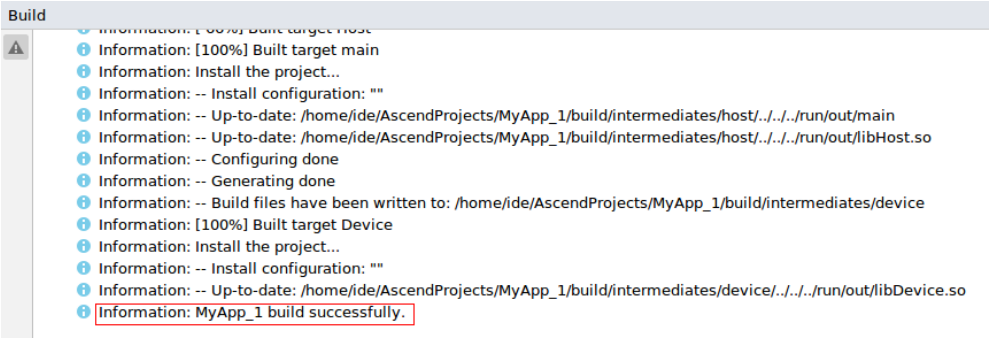
如果出现类似如下问题，请按照**3 编译环境配置**部署g++-4.8编译器。

图 4-5 编译环境问题导致编译失败



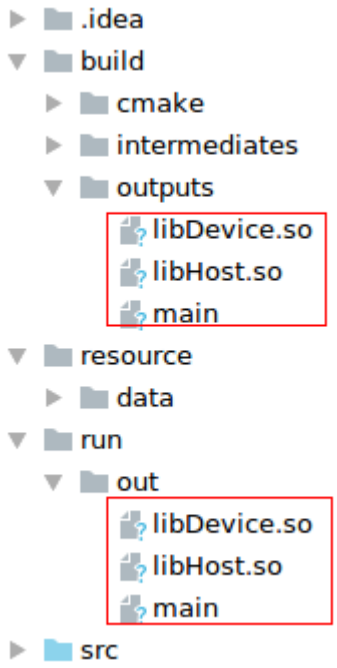
步骤3 确认工程编译结束。

图 4-6 编译成功提示



步骤4 编译完成后在工程目录下的build/outputs文件夹下，生成libDevice.so、libHost.so和可执行文件main。同时自动生成run目录，并将上述文件自动拷贝到run/out文件夹下，为工程运行做准备。

图 4-7 编译生成文件



----结束

工程运行

步骤1 在**Mind Studio**工程界面，依次选择**Run > Edit Configuration...**，进入运行配置页面。

步骤2 选择“Ascend APP > *工程名称*”配置应用工程运行参数，参数说明如**表4-4**所示。

表 4-4 应用工程运行配置参数说明

配置参数	参数说明
Name	运行配置名称，可修改。

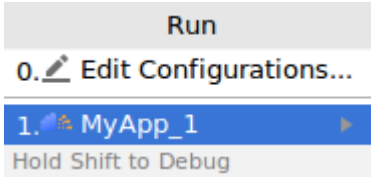
配置参数	参数说明
Target Host Ip	下拉选择运行硬件设备的Host侧的IP地址，若未添加设备，请单击⊞进行添加。 详细的添加设备的方法请参见**8.2.2 增加设备**。
Profilling	可保持默认关闭。如需分析应用工程性能数据，请参考**9 Profiling工具**进行配置。
Before launch	此栏中可配置运行前的预处理操作。

步骤3 配置完运行工程后，单击“Apply”保存运行配置，单击“OK”，关闭运行配置窗口。

步骤4 在**Mind Studio**工程界面，依次选择**Run > Run**。

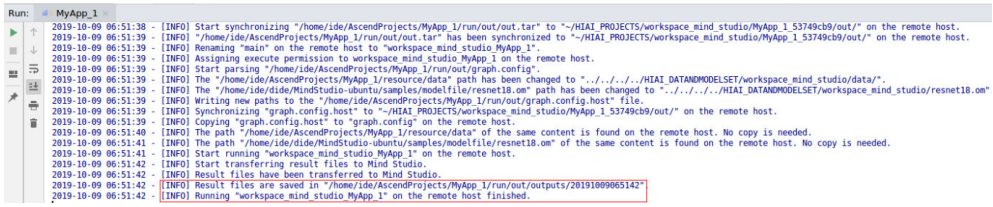
在弹出框中选择已创建好的运行配置信息，进行工程运行，如**图4-8**所示。

图 4-8 选择运行配置信息



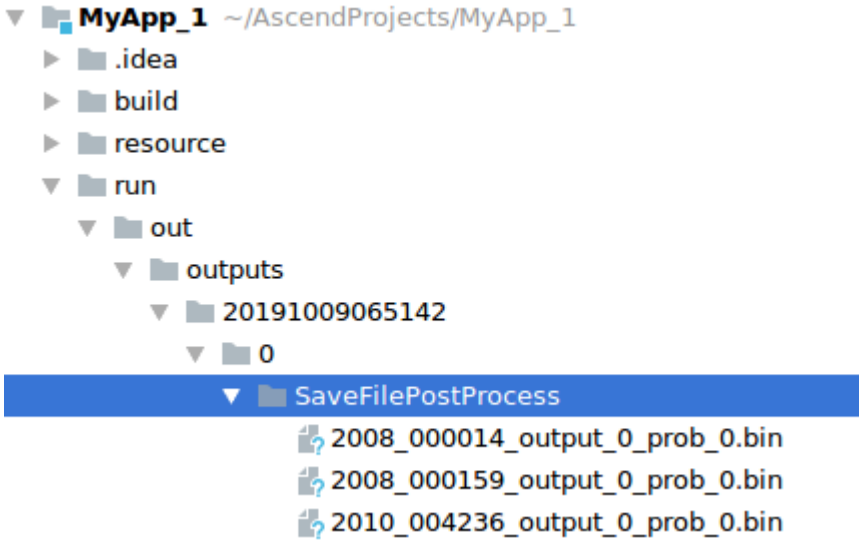
步骤5 确认运行结束，并可以查看到运行结果存放路径，如**图4-9**所示。

图 4-9 运行成功提示



步骤6 查看运行结果，如**图4-10**所示。

图 4-10 查看运行结果



----结束

4.2 快速体验首个算子工程

该章节以Caffe网络模型Lenet-5内置Reduction算子为例，带您快速体验算子及算子插件的开发流程，并给出样例输入数据与期望输出数据，运行单算子验证Reduction算子的正确性，最后通过加载Reduction算子插件成功进行Mylenet网络模型转换。

背景信息

本示例中Reduction算子实现了如下运算：

将每一个输入数据乘以一个系数coeff（本示例中coeff=2），然后在轴axis（本示例中axis=1）上进行累加的操作。

其中输入数据的shape为（2,3,4），一共2*3*4=24个数，每个数的值都为-1。在轴1上进行累加，所以输出数据的shape为（2），最终的输出是两个数（共24个数，则分别是12个-1乘系数2后然后累加），所以最终的输出结果为-24,-24。

创建算子工程

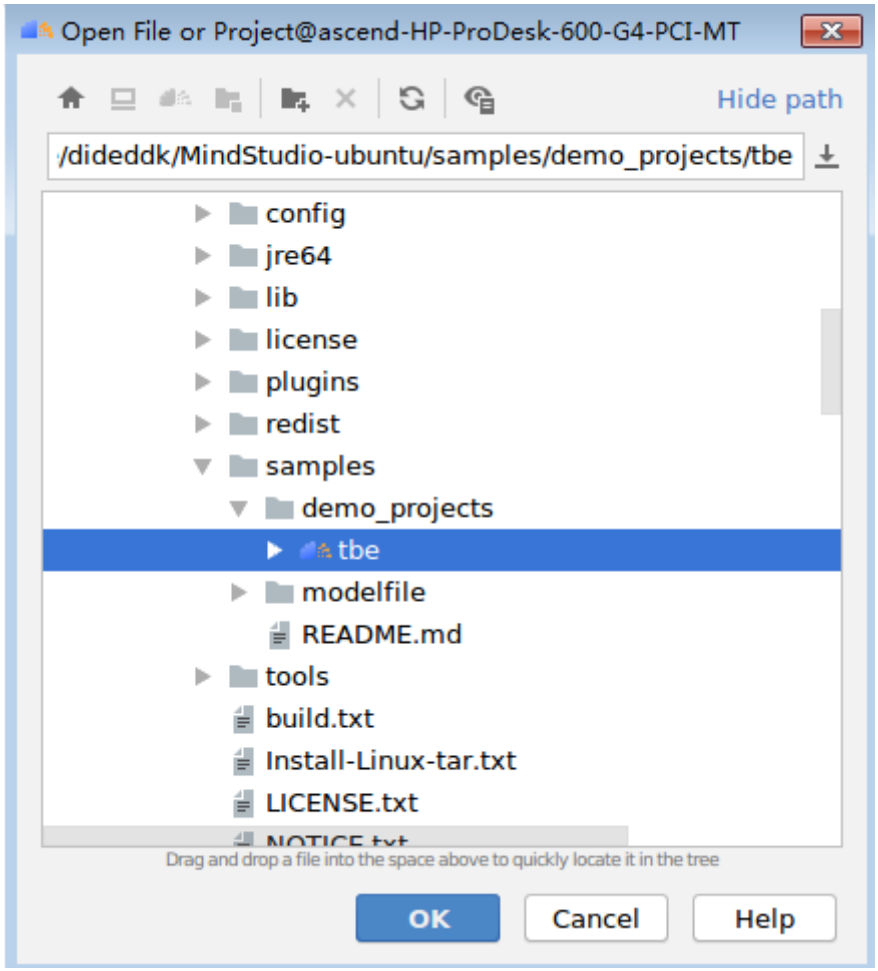
通过导入样例算子工程的方式创建算子工程。

步骤1 导入样例算子工程。

1. 进入工程导入页面。
   - 首次登录Mind Studio：在**Welcome to Mind Studio**窗口中，单击**Open project**。
   - 非首次登录Mind Studio：在顶部菜单栏中选择**File > Open...**。
2. 在**Open File or Project**窗口中，选择样例工程所在文件夹。

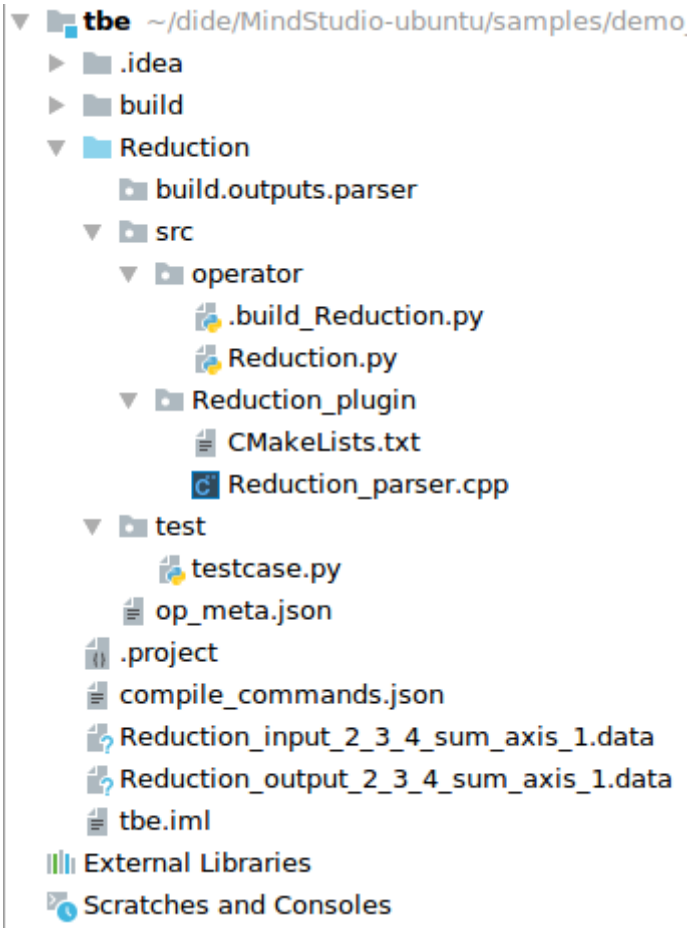
   如图4-11所示，TBE样例工程所在路径为**Mind Studio**安装路径下的“samples/demo_projects/tbe”。

图 4-11 选择 TBE 样例工程所在文件夹



3. 单击**OK**，导入TBE样例工程。
   系统显示当前工程界面，如图4-12所示。

图 4-12 TBE 样例工程界面



若打开工程时出现如图4-13所示提示，是因为样例工程配置文件中的DDK版本号与当前已安装的DDK版本号不匹配，需要单击**OK**修改为当前实际使用的DDK版本号并配置Target，如图4-14所示。

图 4-13 打开工程错误提示

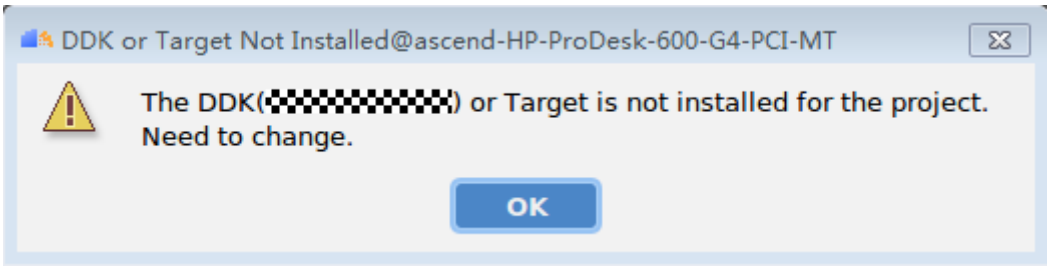
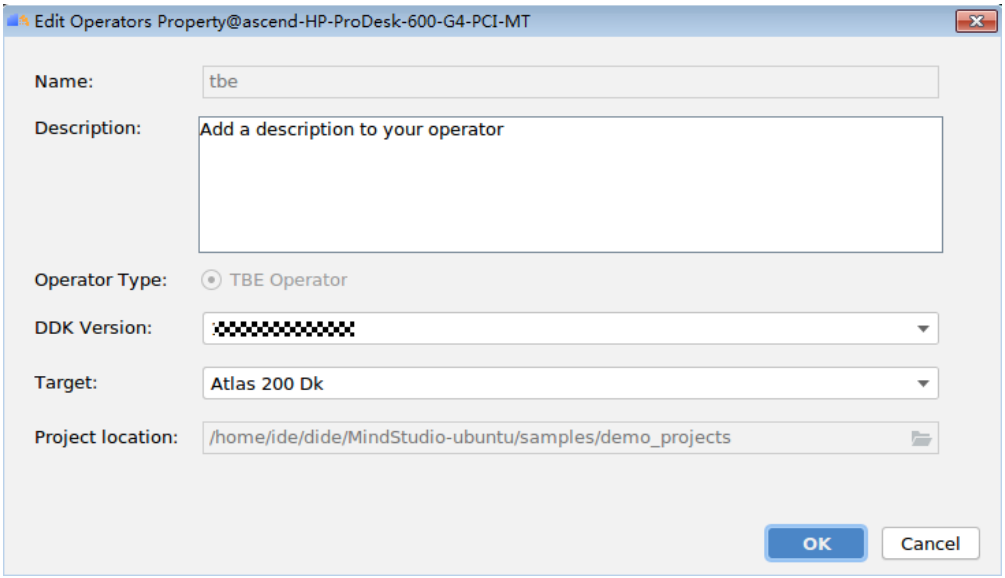


图 4-14 工程属性修改



步骤2 了解工程文件。

展开工程名，您可以看到该工程所有文件，如表4-5所示。

说明

作为首次体验，无需对这些文件进行编辑。

表 4-5 工程文件

一级目录	二级目录	文件内容	说明
.idea	-	-	工程信息。.idea文件夹下的其他文件为idea框架默认，用户无需关注。
Reduction	src	operator	Reduction.py为算子实现文件，.build_Reduction.py为算子编译所需配置文件。
		Reduction_plugin	算子插件文件所在目录。其中Reduction_parser.cpp为算子插件实现代码，CMakeLists.txt为算子插件编译规则文件。
	test	testcase.py	单算子运行所需测试数据配置文件。
Reduction_input_2_3_4_sum_axis_1.data	-	-	Reduction算子的二进制格式输入数据文件。

一级目录	二级目录	文件内容	说明
Reduction_output_2_3_4_sum_axis_1.data	-	-	Reduction算子二进制格式输出数据验证文件，用于验证算子运行后的输出结果是否正确。
.project	-	-	工程信息，包含工程类型、工程描述、运行目标设备类型以及DDK版本。
其他目录和文件用户无需关注。			

----结束

修改样例代码

修改插件样例代码src/Reduction_plugin/Reduction_parser.cpp中的**FileFath**路径。

图 4-15 修改插件样例代码中 FilePath 路径

```
// Parse input shape, validate if it is equal to 4
if (input_desc.GetShape().GetDimNum() != 4)
{
    printf("The shape size is %d, which is not 4!", (int32_t)input_desc.GetShape().GetDimNum());
    return FAILED;
}

FilePath   = "Reduction/src/operator/Reduction"  // Add absolute project path to the front.
FuncName   = "Reduction";
KernelName = "Reduction";

/* TODO: Pass the parameters to the api function */
te::BuildTeCustomOp(te_bin_info.ddk_version, op.GetName(), FilePath, FuncName,
        "(i,i,i,i), s, s, i, f, s",
        input_desc.GetShape().GetDim(0), input_desc.GetShape().GetDim(1),
        input_desc.GetShape().GetDim(2), input_desc.GetShape().GetDim(3),
        "float16", operation.c_str(), axis, coeff,
        KernelName.c_str());
```

FilePath需要为算子实现文件所在的绝对路径+算子py文件名称。

例如：FilePath = "/home/ascend/MindStudio-ubuntu/samples/demo_projects/tbe/Reduction/src/operator/Reduction"

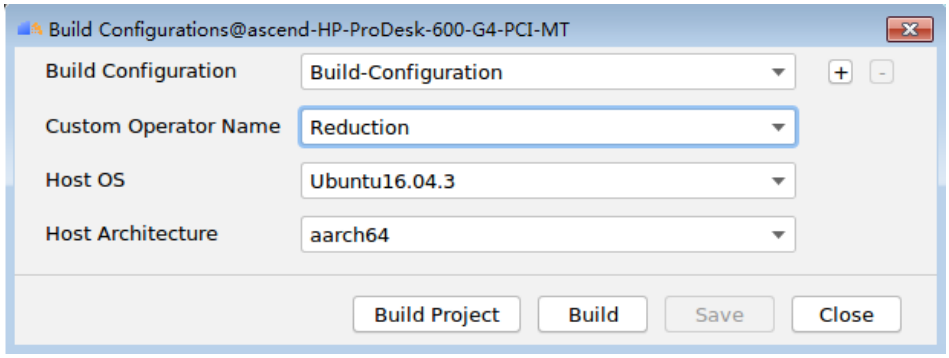
其中：

- /home/ascend/MindStudio-ubuntu/samples/demo_projects/tbe/Reduction/src/operator/为算子实现文件的绝对路径。
- Reduction为算子实现文件的名称。
- /home/ascend为Mind Studio安装目录示例。

工程编译构建

步骤1 选中创建的TBE工程，在顶部菜单栏中依次单击“Build > Edit Build Configuration”如下图所示。

图 4-16 编译自定义算子

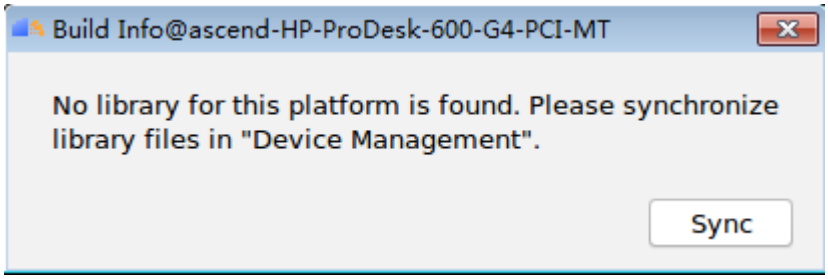


其中：

- Custom Operator Name：需要编译的自定义算子。
- Host OS：硬件侧Host的操作系统版本。
- Host Architecture：硬件侧Host操作系统的类型。

如果出现如下提示，说明编译依赖的lib库没有从运行环境同步到开发环境上，请单击**Sync**进入设备管理界面，然后参考**3 编译环境配置**进行相关操作。

图 4-17 未同步 lib 库



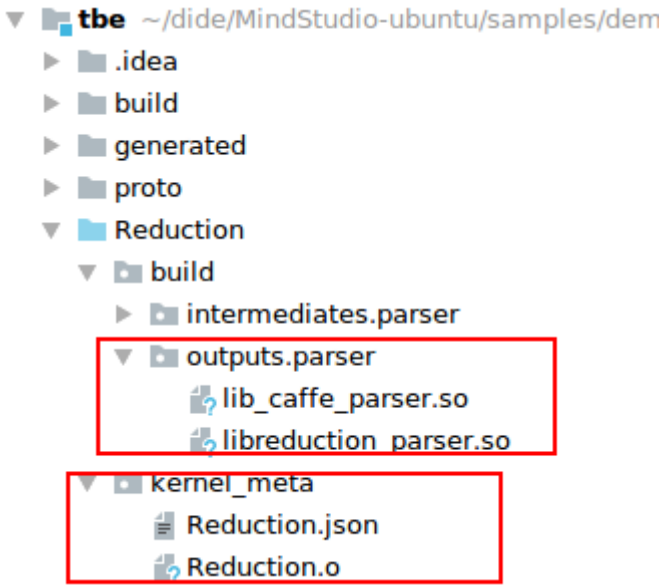
步骤2 单击**Build**进行编译。

说明

- **Build**为编译当前选中的单个算子。
- **Build Project**为编译当前工程中的所有算子。
- 编译过程中会同步编译算子插件。

- 自定义算子编译完成后，会在算子工程目录下创建kernel_meta文件夹，并在kernel_meta文件夹下生成算子二进制文件Reduction.o及算子描述文件Reduction.json。
- 算子插件编译完成后，会在算子工程目录的“build > outputs.parser”目录下生成caffe.proto的二进制文件**lib_caffe_parser.so**与算子插件二进制文件**libreduction_parser.so**。

图 4-18 算子工程编译结果文件



----结束

单算子运行验证

步骤1 选中算子工程，单击顶部菜单栏的“Run > Edit Configurations > ...”，进入运行配置界面。

步骤2 选择“Ascend Operator > Reduction”配置算子工程运行参数，参数说明如表4-6所示。

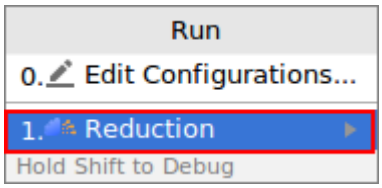
表 4-6 算子工程运行配置参数说明

配置参数	参数说明
Target Host Ip	下拉选择运行硬件设备的Host侧的IP地址，若未添加设备，请单击⊞进行添加。 详细的添加设备的方法请参见8.2.2 增加设备。
Operator Type	算子类型，若此工程中有多个算子，此处可下拉选择需要运行的算子。
Before launch	此栏中可配置运行前的预处理操作。

步骤3 配置完运行工程后，单击“Apply”保存算子运行配置，单击“OK”，关闭运行配置窗口。

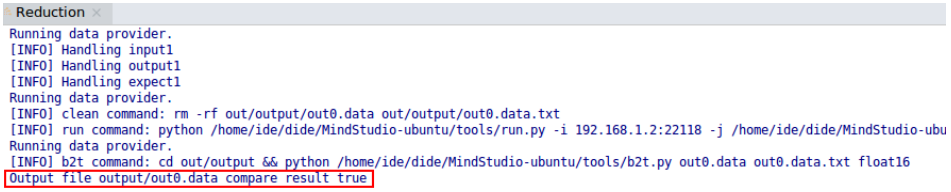
步骤4 单击菜单栏中的“Run > Run”，在如图4-19所示界面中选择Reduction，代表运行Reduction算子。

图 4-19 运行选择界面



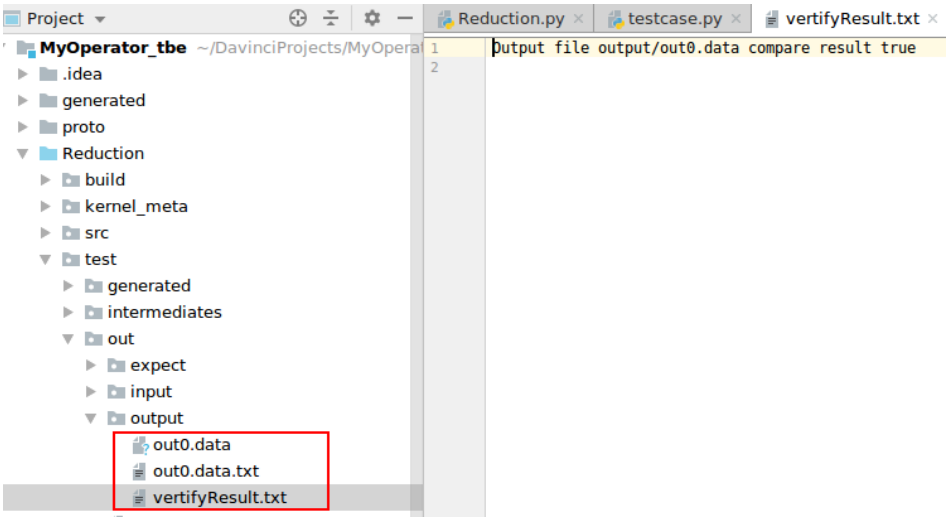
- 运行成功后，输出如图4-20所示。

图 4-20 运行输出



红框中**compare result true**代表算子实际结果与预期输出结果比对成功，另外在工程的“Reduction > test > out > output”目录下会生成如下文件。

图 4-21 运行结果文件



其中：

- out0.data为输出数据二进制文件。
- out0.data.txt为输出数据的可视化的文本文件。
- vertifyResult.txt为比对结果文件。

- 若运行失败，用户可在Log页签查看详细的System Log进行排查。
  详细的日志查看方法可参考**10 日志工具**。

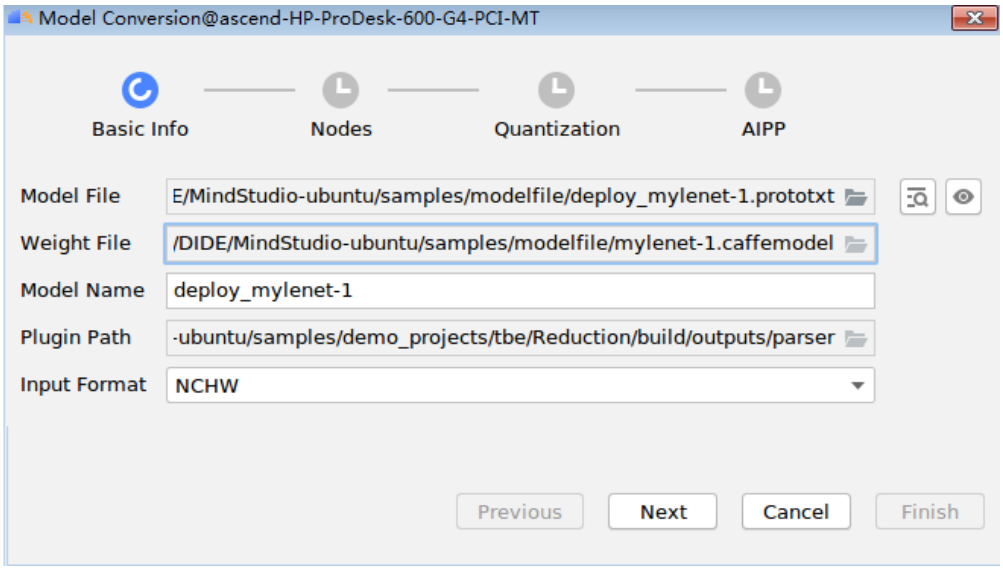
----结束

加载插件转换模型

步骤1 选中算子开发工程名，在菜单栏选择“Tools > Model Convert”。

步骤2 在弹出的“Model Conversion”窗口中，选择模型文件、权重文件、算子插件所在目录及输入数据类型，自定义模型名称，如图4-22所示。

图 4-22 模型转换基本配置



- Model File单击后面左侧的...，选择需要转化的模型对应的*.prototxt文件，并上传。

  本示例使用的“deploy_mylenet-1.prototxt”文件在Mind Studio安装路径下的“samples/modelfile/”目录下。

- Weight File单击后面左侧的...，选择需要转化的模型对应的*.caffemodel文件，并上传。

  本示例使用的“mylenet-1.caffemodel”文件在Mind Studio安装路径下的“samples/modelfile/”目录下。

- Model Name：生成的模型文件名称，用户自定义，保持唯一即可。
- Plugin Path：算子插件二进制文件**libreduction_parser.so**以及caffe.proto二进制文件**lib_caffe_parser.so**文件所在路径。
- Input Format：此处保持默认值。

说明

此处只给出当前示例的配置值说明，详细的模型转换的参数说明请参见7.2 模型转换。

步骤3 单击“Next”，进入**Nodes**配置界面。

Nodes界面配置保持默认值即可。

步骤4 单击“Next”，进入Quantization配置界面。

对于Lenet-5网络，不需要进行量化配置，所以保持默认关闭。

步骤5 单击“Next”，进入AIPP预处理界面。

对于Lenet-5网络，不需要进行AIPP预处理，所以将Image Pre-processing按钮关闭。

步骤6 单击“Next”，进入Encryption配置界面。

此处不使用加密配置，所以保持默认关闭。

步骤7 单击“Finish”，开始进行模型转换。

模型转换成功后，输出信息如图4-23所示。

图 4-23 模型转换成功输出信息示例

```
2019-10-09 09:03:30  Offline model saving is complete. (5/5)
2019-10-09 09:03:30  [INFO] FMK:2019-10-09-09:03:30.781.825 Generate:framework/domi/omg/omg.cpp:867:"OMG generate end."
2019-10-09 09:03:30  [INFO] RUNTIME:2019-10-09-09:03:30.781.880 18953 runtime/feature/src/stream.cc:71 ~Stream:free stream id = 896
2019-10-09 09:03:30  OMG generate offline model success.
2019-10-09 09:03:30  [INFO] RUNTIME:2019-10-09-09:03:30.786.223 18953 runtime/feature/src/driver.cc:57 ~Driver:deconstruct driver
2019-10-09 09:03:30  Model convert success
2019-10-09 09:03:30  Model output path:/home/ide/modelzoo/deploy_mylenet-13
2019-10-09 09:03:30  Model convert log file path:/home/ide/modelzoo/deploy_mylenet-13/ModelConvert.log
2019-10-09 09:03:30  convert device model command:
export SLOG_PRINT_TO_STDOUT=1 && export PATH=${PATH}:/home/ide/.mindstudio/huawei/ddk/1.31.T12.B120/ddk/uihost/toolchains/ccec-linux/bin && export LD_LIBRARY_PATH
```

转换成功的模型文件在Mind Studio安装用户家目录的“modelzoo”目录下。

例如:

- 转换好的模型文件：“$HOME/modelzoo/deploy_mylenet-1/device/deploy_mylenet-1.om”。
- 模型转换日志：“$HOME/modelzoo/deploy_mylenet-1/ModelConvert.log”。

----结束

5 工程迁移

本章节讲解如何将原有基于Eclipse CHE架构的Mind Studio工程导入到基于IntelliJ IDEA架构的Mind Studio中，并给出导入后，工程编译以及运行的方法。

前提条件

- 基于Eclipse CHE架构的Mind Studio前提条件
  a. 在基于Eclipse CHE架构的Mind Studio环境创建.mind工程，并成功完成工程的编译以及运行。
  b. 选中所创建的工程，在基于Eclipse CHE架构的Mind Studio环境中依次单击“File > Download Project”，将工程导出到本地，导出的工程会以zip包的形式导出的本地。
- 基于IntelliJ IDEA架构的Mind Studio前提条件
  a. 已经完成基于IntelliJ IDEA架构的Mind Studio安装，请参见**2 安装**。
  b. 参见**8.2 设备管理**将工程运行的设备添加到设备管理界面，并完成Lib库同步。
  c. 以基于IntelliJ IDEA架构的Mind Studio安装用户，将基于Eclipse CHE架构的Mind Studio工程使用的数据集和模型文件上传到基于IntelliJ IDEA架构的Mind Studio后台服务器任意路径。
     - 若基于Eclipse CHE架构的Mind Studio工程使用的数据集和模型都为内置数据集和内置模型，则导出的zip包工程中不包括数据集和模型信息，需要用户从基于Eclipse CHE架构的Mind Studio工程“$HOME/tools/che/datasets/built-in-datasets/”目录将所使用的内置数据集导出，从“$HOME/tools/che/model-zoo/built-in-model”目录将所使用的内置模型导出，然后以基于IntelliJ IDEA架构的Mind Studio安装用户上传到基于IntelliJ IDEA架构的Mind Studio后台服务器任意路径。
     - 若基于Eclipse CHE架构的Mind Studio工程使用的数据集或模型为自定义数据集或自定义模型，则导出的zip包工程中包括该自定义数据集或自定义模型信息，用户需要以基于IntelliJ IDEA架构的Mind Studio安装用户将“MyModel/*modulename*/device”路径下的.om自定义模型文件或“MyDataset/dataname”路径下的自定义数据集文件上传到基于IntelliJ IDEA架构的Mind Studio后台服务器任意路径。

迁移范围

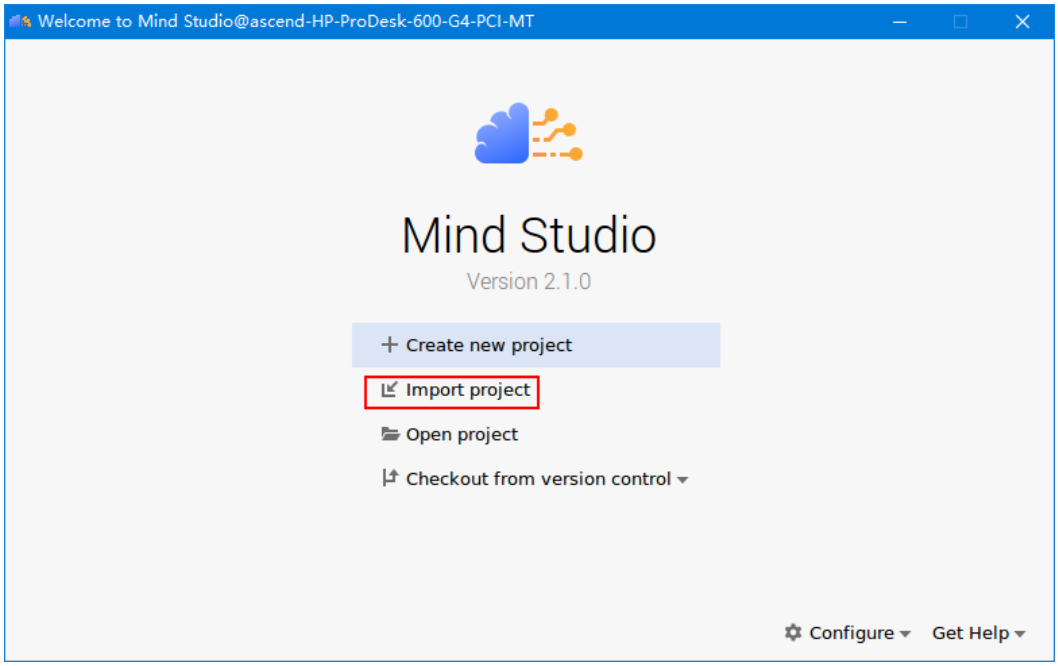
- 只支持.mind工程的迁移，不支持publish工程。
- 不支持Target为Local，Caffe仿真环境的迁移。
- 不支持自定义算子工程的迁移。

操作步骤

本章节以mobilenet.zip工程为例，为您介绍如何操作。

步骤1 打开基于IntelliJ IDEA架构的Mind Studio，在欢迎页面选择“Import project”，如图5-1所示。您也可以通过工程界面的“File >New >Import project”导入已有工程。

图 5-1 工程导入界面

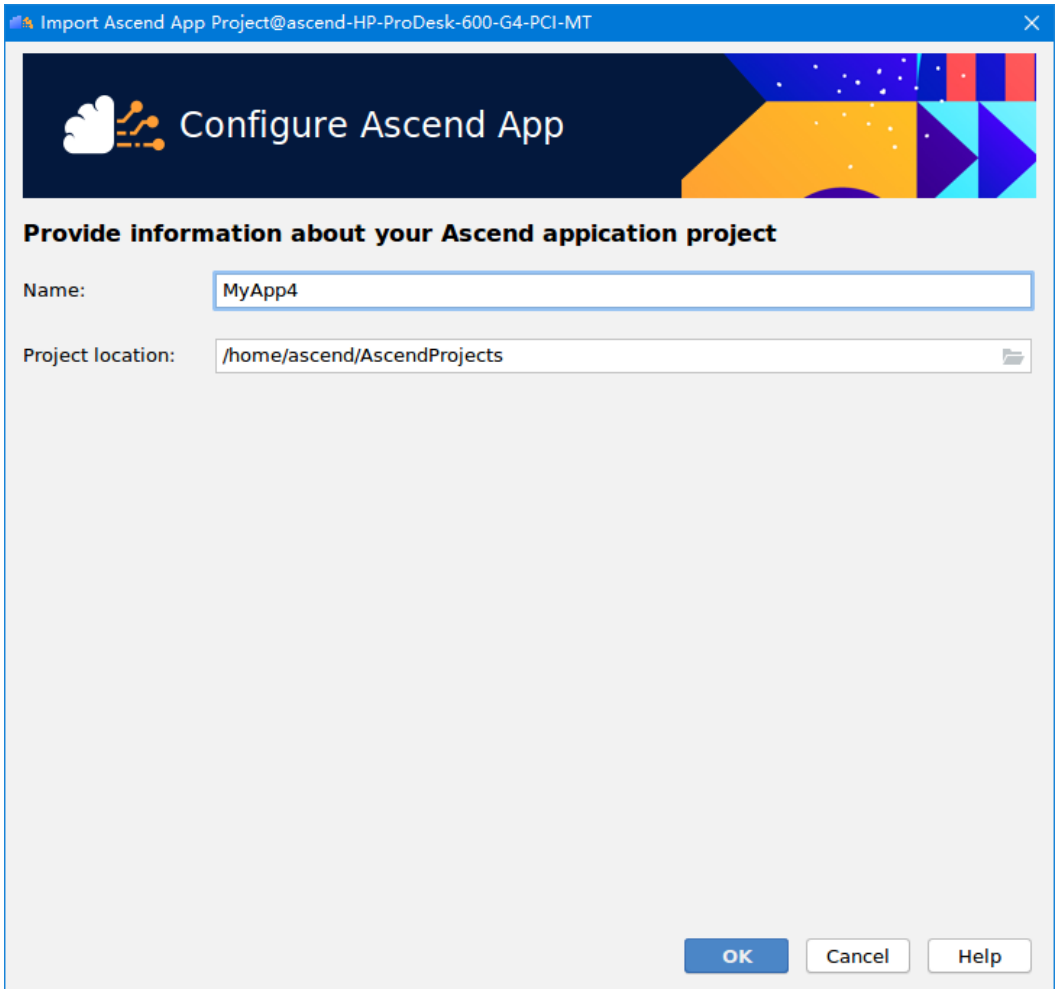


说明

如果用户在执行工程迁移前，已经打开其他工程，例如算子开发工程或应用工程，则需要参见关闭工程先将其他工程关闭，然后才能进入图5-1界面。

步骤2 在弹出界面中，选择要导入的基于Eclipse CHE架构的Mind Studio工程，单击“OK”。基于IntelliJ IDEA架构的Mind Studio中会弹出新的工程界面，用于记录基于Eclipse CHE架构的Mind Studio在基于IntelliJ IDEA架构的Mind Studio中的工程名，如图5-2所示。

图 5-2 基于 Eclipse CHE 架构的 Mind Studio 在基于 IntelliJ IDEA 架构的 Mind Studio 中的工程名



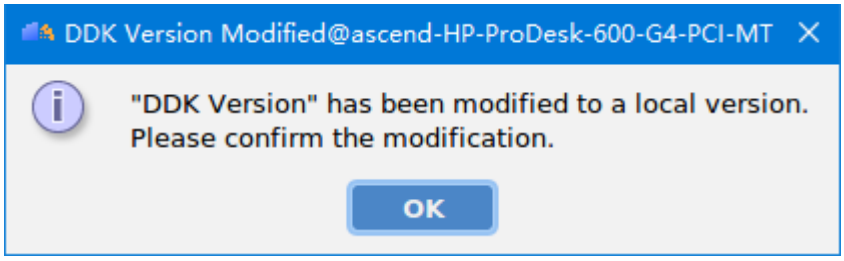
界面参数解释如表5-1所示。

表 5-1 界面参数

参数	说明
Name	基于Eclipse CHE架构的Mind Studio在基于IntelliJ IDEA架构的Mind Studio中的工程名，用户根据自己需要进行修改。
Project location	基于Eclipse CHE架构的Mind Studio导入基于IntelliJ IDEA架构的Mind Studio后，工程所在路径，用户根据自己需要进行修改。

步骤3 单击“OK”，弹出DDK版本变化提示信息，如图5-3所示。

图 5-3 DDK 版本变化提示信息

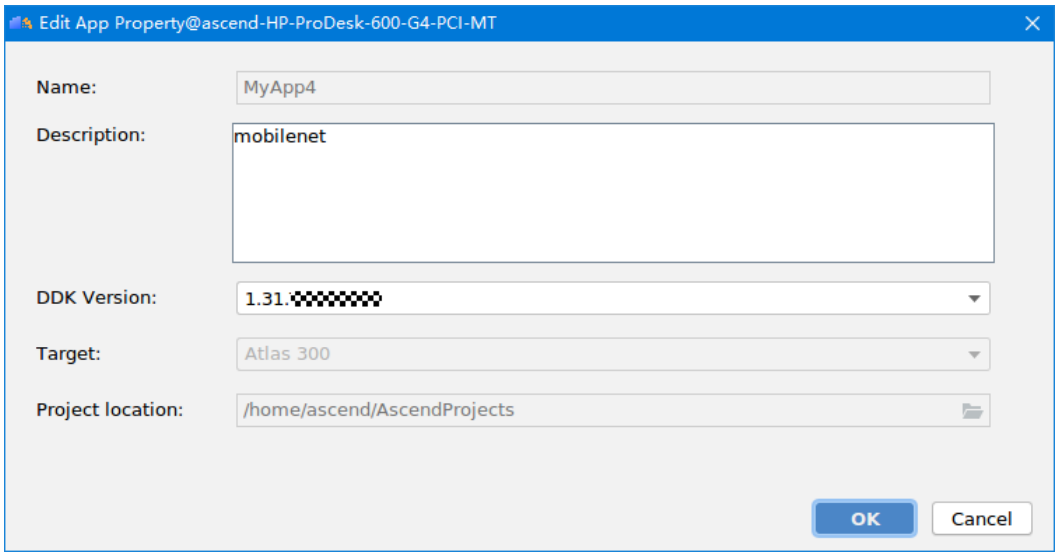


说明

图5-3所示页面只在首次导入工程时出现，用户导入工程后，再次打开工程不会出现该页面。

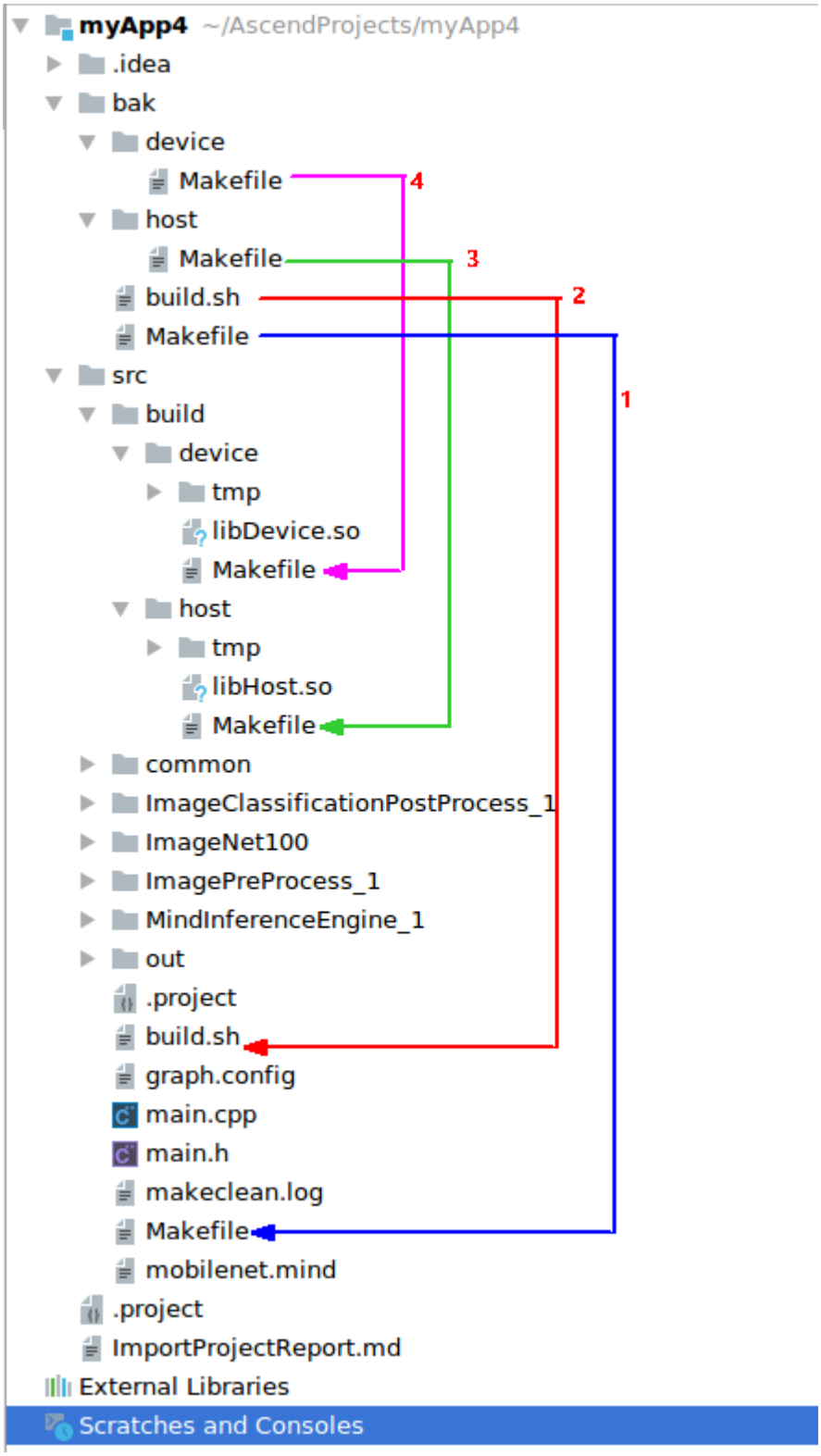
步骤4 单击“OK”，弹出DDK版本选择界面，如图5-4所示。

图 5-4 选择 DDK 版本



步骤5 选择完之后，单击“OK”，进入工程打开界面，如图5-5所示。

图 5-5 迁移后的工程界面



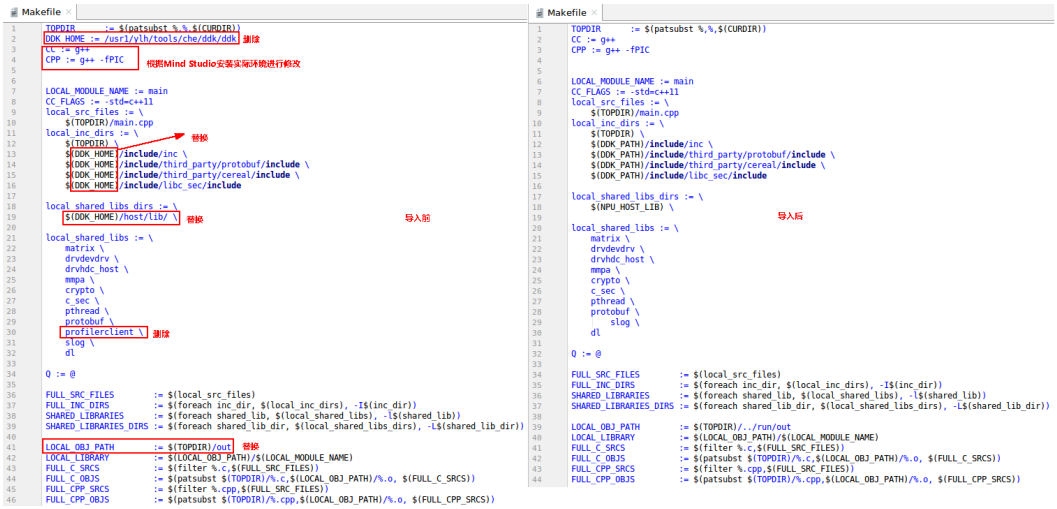
迁移过程中的文件修改如下：

如下四个文件，导入工程时脚本会自动处理，除了根据Mind Studio安装服务器修改CC或CPP 编译器版本号外，其他无需用户手动处理；如果编译不通过，需要用户自行手

动修改Makefile文件进行适配。build/device/Makefile和build/host/Makefile根据用户的工程决定，实际导入的工程可能没有这两个文件。

1. 修改Makefile（对应图5-5中的序号1）：

图 5-6 修改 makefile



须知

图5-6中“根据Mind Studio安装实际环境进行修改”，如果Mind Studio安装服务器为CentOS7.4+x86或者CentOS7.6+Arm64，则需要修改为如下内容：

- Mind Studio安装服务器为CentOS7.4+x86

```
TOPDIR      := $(patsubst %,%,$(CURDIR))
CC := g++-4.8
CPP := g++-4.8
```

- Mind Studio安装服务器为CentOS7.6+Arm64

```
TOPDIR      := $(patsubst %,%,$(CURDIR))
CC := aarch64-linux-gnu-g++-4.8
CPP := aarch64-linux-gnu-g++-4.8 -fPIC
```

2. 修改build.sh（对应图5-5中的序号2）：

图 5-7 修改 build.sh

build.sh

```
#!/bin/sh
export LD_LIBRARY_PATH=/usr1/ylh/tools/che/ddk/ddk/uihost/lib/
TOP_DIR=${PWD}
BUILD_DIR=${TOP_DIR}/build
LOCAL_OBJ_PATH=${TOP_DIR}/out
cd ${TOP_DIR} && make
if [ $? != 0 ]
then
  rm ${LOCAL_OBJ_PATH}/*.o ${LOCAL_OBJ_PATH}/main
  exit
fi
cp graph.config ./out/
cd ${BUILD_DIR}
if [ -d "host" ];then
  if [ -f "host/Makefile" ];then
    cd host && make install
    if [ $? != 0 ]
    then
      rm ${LOCAL_OBJ_PATH}/*.o ${LOCAL_OBJ_PATH}/main
      rm ${LOCAL_OBJ_PATH}/*.so
      exit
    fi
  fi
fi
cd ${BUILD_DIR}
if [ -d "device" ];then
  if [ -f "device/Makefile" ];then
    cd device && make install
    if [ $? != 0 ]
    then
      rm ${LOCAL_OBJ_PATH}/*.o ${LOCAL_OBJ_PATH}/main
      rm ${LOCAL_OBJ_PATH}/*.so
      exit
    fi
  fi
fi
echo "============ build success ! ============"
```

删除

替换

替换

导入前

build.sh

```
#!/bin/sh
TOP_DIR=${PWD}
BUILD_DIR=${TOP_DIR}/build
LOCAL_OBJ_PATH=${TOP_DIR}/../run/out
cd ${TOP_DIR} && make
if [ $? != 0 ]
then
  rm ${LOCAL_OBJ_PATH}/*.o ${LOCAL_OBJ_PATH}/main
  exit
fi
cp graph.config ../run/out
cd ${BUILD_DIR}
if [ -d "host" ];then
  if [ -f "host/Makefile" ];then
    cd host && make install
    if [ $? != 0 ]
    then
      rm ${LOCAL_OBJ_PATH}/*.o ${LOCAL_OBJ_PATH}/main
      rm ${LOCAL_OBJ_PATH}/*.so
      exit
    fi
  fi
fi
cd ${BUILD_DIR}
if [ -d "device" ];then
  if [ -f "device/Makefile" ];then
    cd device && make install
    if [ $? != 0 ]
    then
      rm ${LOCAL_OBJ_PATH}/*.o ${LOCAL_OBJ_PATH}/main
      rm ${LOCAL_OBJ_PATH}/*.so
      exit
    fi
  fi
fi
echo "============ build success ! ============"
```

导入后

3. 若有build/host/Makefile（对应图5-5中的序号3）：

图 5-8 修改 build/host/Makefile

Makefile

```
SHELL := /bin/bash
all : libHost.so
#HOST COMPILER
DDK_HOME = /usr1/ylh/tools/che/ddk/ddk
CC := g++
#Attention: no space at the end of the line, or it will be added to dir name which will will cause error
LOCAL_DIR  :=  ../../ImageNet100 ../../ImageClassificationPostProcess_1 ../../common

SRC_DIR = $(LOCAL_DIR)
BUILD_DIR = tmp
OUT_DIR = ../../out
OBJ_DIR = $(BUILD_DIR)/obj
DEPS_DIR  = $(BUILD_DIR)/deps

#Add other .h file paths here
INC_DIR = \
    -I$(DDK_HOME)/include/inc \
    -I$(DDK_HOME)/include/third_party/protobuf/include \
    -I$(DDK_HOME)/include/third_party/cereal/include \
    -I$(DDK_HOME)/include/libc_sec/include \
    -I/usr/include/python2.7 \
    -I../../ImageNet100 \
    -I../../ImageClassificationPostProcess_1 \
    -I../../common/include

#Add compile arguments here
CC_FLAGS := $(INC_DIR) -std=c++11 -fPIC
LNK_FLAGS := \
    -L$(DDK_HOME)/host/lib/ \
    -lmatrix \
    -lhiai_common \
    -shared
```

删除

根据Mind Studio部署实际环境进行修改

替换

替换

导入前

替换

Makefile

```
SHELL := /bin/bash
all : libHost.so
#HOST COMPILER
CC := g++
#Attention: no space at the end of the line, or it will be added to dir name which will will cause error
LOCAL_DIR  :=  ../../ImageNet100 ../../ImageClassificationPostProcess_1 ../../common

SRC_DIR = $(LOCAL_DIR)
BUILD_DIR = tmp
OUT_DIR = ../../../run/out
OBJ_DIR = $(BUILD_DIR)/obj
DEPS_DIR  = $(BUILD_DIR)/deps

#Add other .h file paths here
INC_DIR = \
    -I$(DDK_PATH)/include/inc \
    -I$(DDK_PATH)/include/third_party/protobuf/include \
    -I$(DDK_PATH)/include/third_party/cereal/include \
    -I$(DDK_PATH)/include/libc_sec/include \
    -I/usr/include/python2.7 \
    -I../../ImageNet100 \
    -I../../ImageClassificationPostProcess_1 \
    -I../../common/include

#Add compile arguments here
CC_FLAGS := $(INC_DIR) -std=c++11 -fPIC
LNK_FLAGS := \
    -L$(NPU_HOST_LIB) \
    -lmatrix \
    -lhiai_common \
    -shared
```

导入后

4. 若有build/device/Makefile（对应图5-5中的序号4）：

图 5-9 修改 build/device/Makefile

Makefile

```
SHELL := /bin/bash
all : libDevice.so
#DEVICE COMPILER
DDK_HOME = /usr1/ylh/tools/che/ddk/ddk
CC := $(DDK_HOME)/toolchains/aarch64-linux-gcc6.3/bin/aarch64-linux-gnu-g++
#Attention: no space at the end of the line, or it will be added to dir name which will will cause error
LOCAL_DIR  :=  ../../ImagePreProcess_1 ../../MindInferenceEngine_1 ../../common

SRC_DIR = $(LOCAL_DIR)
BUILD_DIR = tmp
OUT_DIR = ../../out
OBJ_DIR = $(BUILD_DIR)/obj
DEPS_DIR  = $(BUILD_DIR)/deps

#Add other .h file paths here
INC_DIR = \
    -I$(DDK_HOME)/include/inc \
    -I$(DDK_HOME)/include/third_party/protobuf/include \
    -I$(DDK_HOME)/include/third_party/cereal/include \
    -I$(DDK_HOME)/include/libc_sec/include \
    -I/usr/include/python2.7 \
    -I../../ImagePreProcess_1 \
    -I../../MindInferenceEngine_1 \
    -I../../common/include

#Add compile arguments here
CC_FLAGS := $(INC_DIR) -g -std=c++11 -fPIC -DCPU_ONLY
LNK_FLAGS := \
    -L$(DDK_HOME)/device/lib/ \
    -lDvpp_api \
    -lDvpp_jpeg_decoder \
    -lDvpp_jpeg_encoder \
    -lDvpp_vpc \
    -lidedaemon \
    -shared
```

删除

替换

导入前

替换

Makefile

```
SHELL := /bin/bash
all : libDevice.so
#DEVICE COMPILER
CC := $(DDK_PATH)/toolchains/aarch64-linux-gcc6.3/bin/aarch64-linux-gnu-g++
#Attention: no space at the end of the line, or it will be added to dir name which will will cause error
LOCAL_DIR  :=  ../../ImagePreProcess_1 ../../MindInferenceEngine_1 ../../common

SRC_DIR = $(LOCAL_DIR)
BUILD_DIR = tmp
OUT_DIR = ../../../run/out
OBJ_DIR = $(BUILD_DIR)/obj
DEPS_DIR  = $(BUILD_DIR)/deps

#Add other .h file paths here
INC_DIR = \
    -I$(DDK_PATH)/include/inc \
    -I$(DDK_PATH)/include/third_party/protobuf/include \
    -I$(DDK_PATH)/include/third_party/cereal/include \
    -I$(DDK_PATH)/include/libc_sec/include \
    -I/usr/include/python2.7 \
    -I../../ImagePreProcess_1 \
    -I../../MindInferenceEngine_1 \
    -I../../common/include

#Add compile arguments here
CC_FLAGS := $(INC_DIR) -g -std=c++11 -fPIC -DCPU_ONLY
LNK_FLAGS := \
    -L$(NPU_DEV_LIB) \
    -lDvpp_api \
    -lDvpp_jpeg_decoder \
    -lDvpp_jpeg_encoder \
    -lDvpp_vpc \
    -lidedaemon \
    -shared
```

导入后

须知

图5-8中“根据Mind Studio安装实际环境进行修改”，如果Mind Studio安装服务器为CentOS7.4+x86或者CentOS7.6+Arm64，则需要修改如下内容：

- Mind Studio安装服务器为CentOS7.4+x86

```
TOPDIR      := $(patsubst %,%,$(CURDIR))
CC := g++-4.8
```

- Mind Studio安装服务器为CentOS7.6+Arm64

```
TOPDIR      := $(patsubst %,%,$(CURDIR))
CC := aarch64-linux-gnu-g++-4.8
```

步骤6 修改ImagePreProcess.cpp文件。

1. 删除或注释掉**#include "custom/toolchain/ide_daemon_api.h**头文件。
2. 删除或注释掉StorePreprocessImage函数相关代码，例如下图中红框内的代码：

图 5-10 StorePreprocessImage 函数相关代码

```
558     if (1 == dvppConfig_->dump_value) {
559         HIAI_ENGINE_LOG(HIAI_IDE_INFO, "[ImagePreProcess_1]preprocess need dump, width:%u, height:%u", image->img.width, image->img.height);
560         DvppPreprocessInfo info = {resizeWidth, resizeHeight, outputConfigure->widthStride, outputConfigure->heightStride, imageFrameID_, orderInFrame_};
561         return StorePreprocessImage(outBuffer.get(), image->img.size, info);
562     }
563
564     return HIAI_OK;
565 }
566
567 int ImagePreProcess_1::StorePreprocessImage(const u_int8_t *outBuffer, const uint32_t &size, const DvppPreprocessInfo &info)
568 {
569     if(NULL == outBuffer) {
570         HIAI_ENGINE_LOG(HIAI_IDE_ERROR, "[ImagePreProcess_1]StorePreprocessImage, outBuffer is null!");
571         return HIAI_ERROR;
572     }
573     time_t now = time(0);
574     //important:after dvpp, the output data is that width is aligned with 128, and height is aligned with 16.
575     std::string fileName = dvppConfig_->userHome + "/projects/" + dvppConfig_->project_name + "/out/PreProcess/ImagePreProcess_1/"
576         + std::to_string(imageFrameID_) + "_" + std::to_string(orderInFrame_ + 1) + "_" + std::to_string(now) + "_preprocessYUV";
577     IDE_SESSION session = ideOpenFile(NULL, (char *)fileName.c_str());
578     HIAI_ENGINE_LOG("save fileName:%s!", fileName.c_str());
579     if (NULL == session) {
580         HIAI_ENGINE_LOG(HIAI_IDE_ERROR, "[ImagePreProcess_1]StorePreprocessImage, ideOpenFile is null!");
581         return HIAI_ERROR;
582     }
583
584     uint32_t sendSize = 0;
585     char *tmp = (char *)outBuffer;
586     while (sendSize < size) {
587         uint32_t everySize = (sendSize + SEND_BUFFER_SIZE > size)  ? (size - sendSize) : SEND_BUFFER_SIZE;
588         if (IDE_DAEMON_NONE_ERROR != ideWriteFile(session, tmp, everySize)) {
589             HIAI_ENGINE_LOG(HIAI_IDE_ERROR, "[ImagePreProcess_1]StorePreprocessImage, ideWriteFile error!");
590             ideCloseFile(session);
591             return HIAI_ERROR;
592         }
593         tmp += everySize;
594         sendSize += everySize;
595     }
596
597     HIAI_ENGINE_LOG("preprocess info, resiz_width:%u, resize_height:%u, preprocess_width:%u, preprocess_height:%u, frameID:%u, order:%u",
598         info.resize_width, info.resize_height, info.preprocess_width, info.preprocess_height, info.frameID, info.orderInFrame);
599     //add info of yuv file to file end which are used to transfer to jpg
600     if (IDE_DAEMON_NONE_ERROR != ideWriteFile(session, &info, sizeof(info))) {
601         HIAI_ENGINE_LOG(HIAI_IDE_ERROR, "[ImagePreProcess_1]StorePreprocessImage, ideWriteFile error!");
602         ideCloseFile(session);
603         return HIAI_ERROR;
604     }
605
606     usleep(SEND_DATA_SLEEP_MS); //assure ide-daemon-host catch all data.
607     if (IDE_DAEMON_NONE_ERROR != ideCloseFile(session)) {
608         HIAI_ENGINE_LOG(HIAI_IDE_ERROR, "[ImagePreProcess_1]StorePreprocessImage, ideCloseFile error!");
609         return HIAI_ERROR;
610     }
611     HIAI_ENGINE_LOG(HIAI_IDE_INFO, "[ImagePreProcess_1]StorePreprocessImage, storage success!, sendSize:%u, size:%u", sendSize, size);
612     return HIAI_OK;
613 }
```

步骤7 修改graph.config文件。

```
graphs {
  priority: 0
    engines {
      id: 580
      engine_name: "ImageNet100"    /*数据集节点*/
      side: HOST
      thread_num: 1
      so_name: "./libHost.so"
      ai_config {

    items {
      name: "path"
      value: "/home/username/webproject/tools/data/ImageNet100/"   /*请替换为数据集在基于IntelliJ
```

```
IDEA架构的Mind Studio服务器上的实际地址*/
      }
    }
  }
      engines {
        id: 887
        engine_name: "MindInferenceEngine_1"    /*推理引擎节点*/
        side: DEVICE
        thread_num: 1
        so_name: "./libDevice.so"
        ai_config {
          items {
        name: "model_path"
        value: "/home/username/webprojects/tools/modelzoo/mobilenet.om"   /*请替换为模型文件在基于
IntelliJ IDEA架构的Mind Studio服务器上的实际地址*/
      }
    }
  }
      engines {
        id: 553
        engine_name: "ImageClassificationPostProcess_1"      /*后处理节点*/
        side: HOST
        thread_num: 1
        so_name: "./libHost.so"
        ai_config {

      items {
        name: "path"
        value: "/home/username/webproject/tools/data/ImageNet100/"    /*请替换为数据集在基于IntelliJ
IDEA架构的Mind Studio服务器上的实际地址*/
      }

      items {
        name: "mode"
        value: "normal"
      }
    }
  }
      engines {
        id: 212
        engine_name: "ImagePreProcess_1"     /*预处理节点*/
        side: DEVICE
        thread_num: 1
        so_name: "./libDevice.so"
        ai_config {

      items {
        name: "userHome"
        value: "/home/username/"     /*用户$HOME目录*/
      }
    }
  }
```

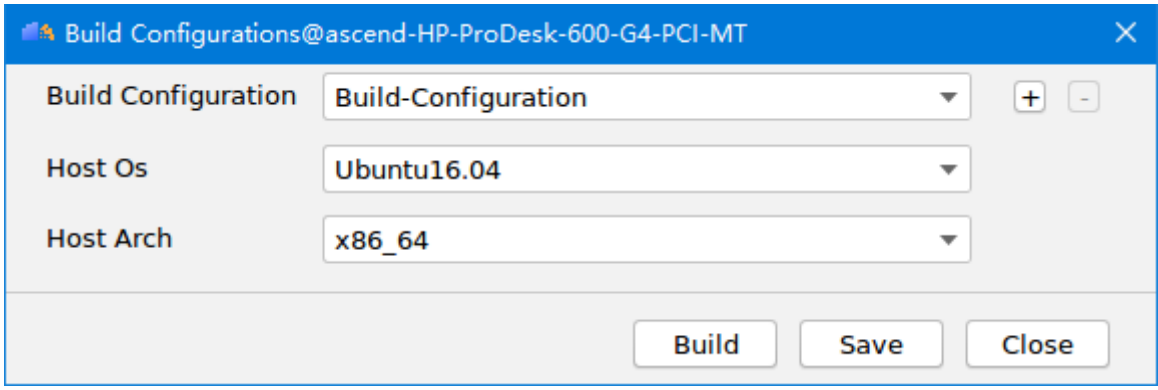
说明

上述graph.config文件只是样例，请以实际代码为准，其中“/home/*username*/webproject/tools/”为测试路径，请以用户实际存放的数据集以及模型文件路径为准。

步骤8 编译工程

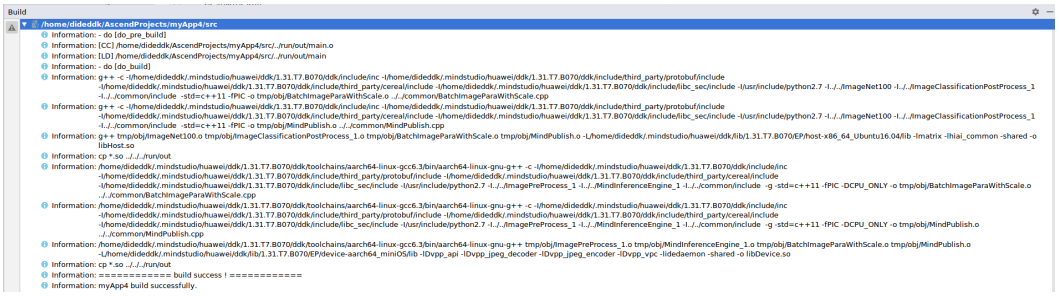
1. 选中导入后的工程，单击上方菜单栏的“Build > Edit Build Configuration”，弹出图5-11所示界面。

图 5-11 工程编译界面



2. 选择相应的Host操作系统和架构后，单击“Build”。在工程下方会弹出编译日志信息，如图5-12所示。

图 5-12 编译日志

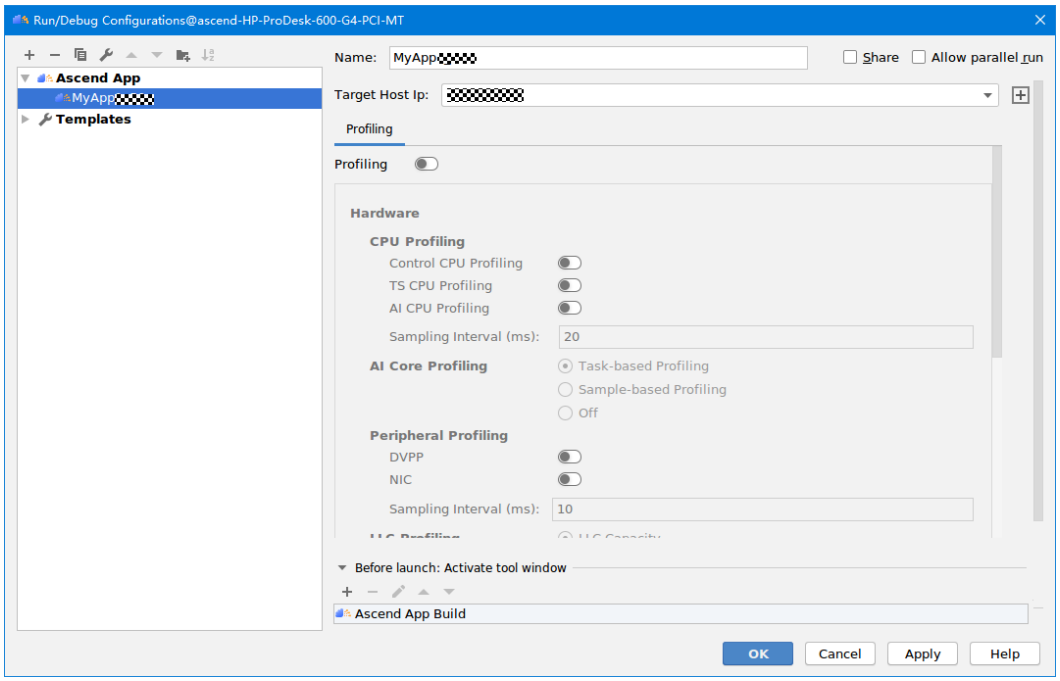


3. 若日志信息中提示"*project* build successfully"，则说明编译成功。

步骤9 运行工程

1. 选中运行的工程，单击上方菜单栏的“Run > Edit configuration”配置运行信息，在弹出界面中选择运行的工程，弹出图5-13所示界面。

图 5-13 运行配置界面

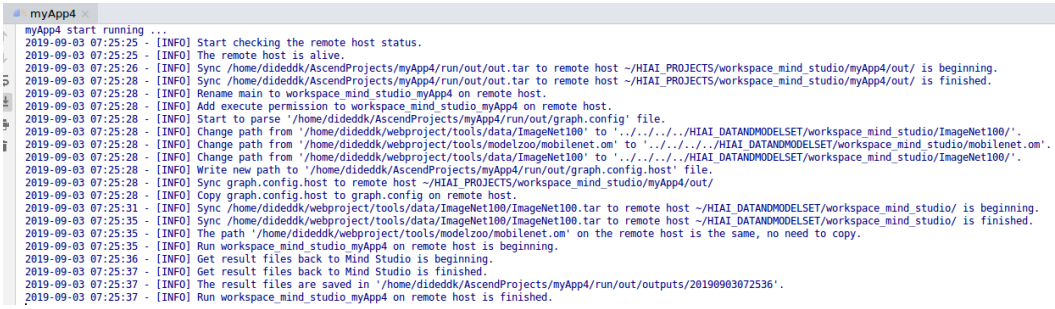


说明

- 若设备管理界面有多个设备，则需要用户在图5-13界面中的“Target Host IP”中选择需要运行的设备。
- 运行导入的工程时，运行的target要和在基于Eclipse CHE架构的Mind Studio运行的target保持一致。

2. 根据用户需要选择相关配置，配置完成后单击“OK”。
3. 选中工程单击菜单栏“Run > Run”或“Run > Run *projectname*”，运行工程。在工程界面下方会弹出运行日志，如图5-14所示。

图 5-14 运行日志



```
myApp4
myApp4 start running ...
2019-09-03 07:25:25 - [INFO] Start checking the remote host status.
2019-09-03 07:25:25 - [INFO] The remote host is alive.
2019-09-03 07:25:26 - [INFO] Sync /home/dideddk/AscendProjects/myApp4/run/out/out.tar to remote host ~/HIAI_PROJECTS/workspace_mind_studio/myApp4/out/ is beginning.
2019-09-03 07:25:28 - [INFO] Sync /home/dideddk/AscendProjects/myApp4/run/out/out.tar to remote host ~/HIAI_PROJECTS/workspace_mind_studio/myApp4/out/ is finished.
2019-09-03 07:25:28 - [INFO] Rename main to workspace_mind_studio_myApp4 on remote host.
2019-09-03 07:25:28 - [INFO] Add execute permission to workspace_mind_studio_myApp4 on remote host.
2019-09-03 07:25:28 - [INFO] Start to parse '/home/dideddk/AscendProjects/myApp4/run/out/graph.config' file.
2019-09-03 07:25:28 - [INFO] Change path from '/home/dideddk/webproject/tools/data/ImageNet100' to '../../../../HIAI_DATANDMODELSET/workspace_mind_studio/ImageNet100/'.
2019-09-03 07:25:28 - [INFO] Change path from '/home/dideddk/webproject/tools/modelzoo/mobilenet.om' to '../../../../HIAI_DATANDMODELSET/workspace_mind_studio/mobilenet.om'.
2019-09-03 07:25:28 - [INFO] Change path from '/home/dideddk/webproject/tools/data/ImageNet100' to '../../../../HIAI_DATANDMODELSET/workspace_mind_studio/ImageNet100/'.
2019-09-03 07:25:28 - [INFO] Write new path to '/home/dideddk/AscendProjects/myApp4/run/out/graph.config.host' file.
2019-09-03 07:25:28 - [INFO] Sync graph.config.host to remote host ~/HIAI_PROJECTS/workspace_mind_studio/myApp4/out/
2019-09-03 07:25:28 - [INFO] Copy graph.config.host to graph.config on remote host.
2019-09-03 07:25:31 - [INFO] Sync /home/dideddk/webproject/tools/data/ImageNet100/ImageNet100.tar to remote host ~/HIAI_DATANDMODELSET/workspace_mind_studio/ is beginning.
2019-09-03 07:25:35 - [INFO] Sync /home/dideddk/webproject/tools/data/ImageNet100/ImageNet100.tar to remote host ~/HIAI_DATANDMODELSET/workspace_mind_studio/ is finished.
2019-09-03 07:25:35 - [INFO] The path '/home/dideddk/webproject/tools/modelzoo/mobilenet.om' on the remote host is the same, no need to copy.
2019-09-03 07:25:35 - [INFO] Run workspace_mind_studio_myApp4 on remote host is beginning.
2019-09-03 07:25:36 - [INFO] Get result files back to Mind Studio is beginning.
2019-09-03 07:25:37 - [INFO] Get result files back to Mind Studio is finished.
2019-09-03 07:25:37 - [INFO] The result files are saved in '/home/dideddk/AscendProjects/myApp4/run/out/outputs/20190903072536'.
2019-09-03 07:25:37 - [INFO] Run workspace_mind_studio_myApp4 on remote host is finished.
```

4. 若运行界面提示如下信息，且无错误，则说明运行成功。

   [INFO] Run workspace_mind_studio_*myApp4* on remote host is finished.

   工程运行结果存储在后台服务器“$HOME/AscendProjects/*myApp4*/run/out/outputs/*20190903072536*”路径下。

步骤10 进入“$HOME/AscendProjects/*myApp4*/run/out/outputs/*20190903072536*/0/ImageClassificationPostProcess_1”路径，查看结果文件，结果文件数量要与基于Eclipse CHE架构的Mind Studio原始工程*projectname*/out/*20190828185251*/0/

ImageClassificationPostProcess_1路径下结果文件的名称、数量保持一致，相同文件内容需要保持一致。

----结束

6 教程

6.1 应用开发

工程创建

步骤1 进入工程创建页面。

- 首次登录**Mind Studio**：在**Welcome to Mind Studio**窗口中，单击**Create New Project**。
- 非首次登录**Mind Studio**：在顶部菜单栏中选择**File > New > Project...**。

步骤2 在**New Project**窗口中，选择**Ascend App**，配置工程相关参数如表6-1所示。

表 6-1 工程参数说明

参数	说明
Name	工程名称，自行配置。 要求配置为以字母开头的字符串，最大长度支持64位。仅支持字母、数字、中划线、下划线。
Description	工程描述信息，自行配置。
DDK Version	当前DDK的版本号。
Target	运行环境，请根据实际硬件环境进行选择。
Project location	工程的存储路径，用户可自定义。

步骤3 单击**Next**，通过**Project Type**参数选择工程模板。

我们为您提供四种应用开发模板，请用户根据实际情况进行选择。

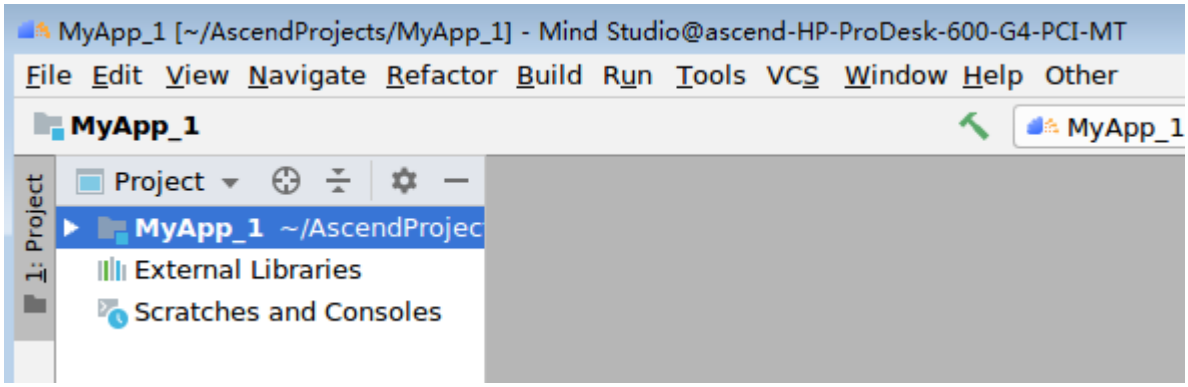
- Classification Network：是基于分类网络的模板，包括四个Engine。
- Object Detection Network(FastRCNN)：是基于检测网络的模板，且模型推理引擎是双输入，包括五个Engine。

- Object Detection Network(SSD)：是基于检测网络的模板，包括四个Engine。
- Custom：是一个相对简单的模板，如果您是全新创建自定义工程，建议选择该模板。

步骤4 单击**Finish**，完成工程创建。

界面显示当前工程页面，如图6-1所示。

图 6-1 工程页面



步骤5 展开工程名查看工程目录结构，一般情况下如表6-2所示。

表 6-2 目录结构

一级目录	二级目录	说明
resource	data	数据集文件。
lib	device	用户第三方库文件存放目录。必须存放在该目录下，工程运行时才可以被拷贝到运行环境上去。
	host	
src	-	Engine的实现文件和头文件。
	graph.config	Graph配置文件，用于定义Engine节点间的串接关系和Engine节点属性（运行该节点所需的参数）。
	Main.cpp	主函数实现文件。
	Main.h	主函数头文件。
	CMakeLists.txt	编译脚本。
.project	-	工程信息文件，包含工程类型、工程描述、运行目标设备类型以及DDK版本。
其他目录为工具框架依赖，和工程无关，用户无需关注。		

----结束

工程实现

请参考配套版本的《Ascend 310 应用开发指导（Mind Studio方式）》。

工程编译

请确保已经完成**3 编译环境配置**。

步骤1 编辑工程目录下的“src/CMakeLists.txt”文件。

表 6-3 CMakeLists 文件参数说明

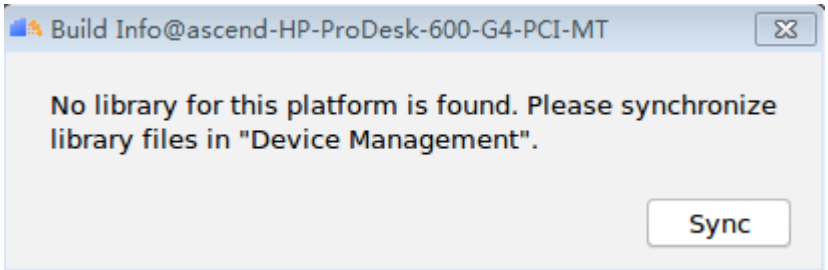
参数	说明
DDK_PATH	DDK安装目录环境变量，用户无需配置。具体路径为：$HOME/.mindstudio/huawei/ddk/{ddk_version}/ddk/
NPU_HOST_LIB	Host侧lib库的环境变量，用户无需配置。具体路径请见**3.2.1 lib库通过同步方式获取**。
NPU_DEV_LIB	Device侧lib库的环境变量，用户无需配置。具体路径请见**3.2.1 lib库通过同步方式获取**。
include_directories	添加头文件目录（支持添加用户第三方库头文件），示例： ```# Header path include_directories(. $ENV{DDK_PATH}/include/inc/ $ENV{DDK_PATH}/include/third_party/protobuf/include $ENV{DDK_PATH}/include/third_party/cereal/include $ENV{DDK_PATH}/include/libc_sec/include Common DataInput ImagePreProcess SaveFilePostProcess MindInferenceEngine)``` **说明** 用户的第三方库文件路径必须放在工程目录下的lib/host和lib/device，工程运行时才可以被拷贝到运行环境上去。
link_directories	添加需要链接的库文件目录。如果工程依赖用户第三方库，则需要添加第三方库文件目录，请见加粗部分： ```# add host lib path link_directories($ENV{NPU_HOST_LIB} **../lib/host**) # add device lib path link_directories($ENV{NPU_DEV_LIB} **../lib/device**)``` **说明** 用户的第三方库文件路径必须放在工程目录下的lib/host和lib/device，工程运行时才可以被拷贝到运行环境上去。
target_link_libraries	设置要链接的库文件的名称，作用为将目标文件与库文件进行链接。该命令的第一个参数是指通过add_executable()和add_library()命令生成已经创建的目标文件；后面的参数表示库文件没有后缀的名字。 如果工程依赖用户第三方库，则需要添加第三方库文件名，请见加粗部分： ```target_link_libraries(main matrixdaemon mmpa pthread dl rt) target_link_libraries(Host matrixdaemon hiai_common **abc**) target_link_libraries(Device Dvpp_api Dvpp_jpeg_decoder Dvpp_jpeg_encoder Dvpp_vpc idedaemon hiai_common **cdf**)``` **说明** 用户的第三方库文件路径必须放在工程目录下的lib/host和lib/device，工程运行时才可以被拷贝到运行环境上去。

参数	说明
add_executable	添加一个可执行文件构建目标。该命令的第一个参数是exe程序的名称；后面的参数，是添加到这个exe中的源码文件，示例： add_executable(main Main.cpp)
add_library	使用指定的源文件向工程中添加一个库。该命令的第一个参数是库名，该库文件将会根据调用的命令里列出的源文件来创建。示例： add_library(Host SHARED Common/Common.cpp DataInput/DataInput.cpp SaveFilePostProcess/SaveFilePostProcess.cpp) add_library(Device SHARED Common/Common.cpp ImagePreProcess/ImagePreProcess.cpp MindInferenceEngine/MindInferenceEngine.cpp)
add_compile_options	编译选项。 Ubuntu环境的so是基于gcc 5.4版本编译的，gcc 5.4版本中编译宏_GLIBCXX_USE_CXX11_ABI只能设置为1。如果用户在ubutun环境上编译自己的应用，将_GLIBCXX_USE_CXX11_ABI设为0，会导致编译宏冲突，从而出现链接失败。此时建议用户用_GLIBCXX_USE_CXX11_ABI=0编译的so，要用_GLIBCXX_USE_CXX11_ABI=1重新编译。在centos环境上编译应用时，建议编译宏_GLIBCXX_USE_CXX11_ABI也设为1，保持统一。

步骤2 在**Mind Studio**工程界面，依次选择**Build > Edit Build Configuration...**，进入编译配置页面。

如果出现如下提示，说明编译依赖的lib库没有从运行环境同步到开发环境上，请单击**Sync**进入设备管理界面，然后参考**3 编译环境配置**进行相关操作。

图 6-2 未同步 lib 库



正常情况下进入编译配置页面，如**图6-3**所示，请参考**表6-4**进行编译配置。

图 6-3 编译配置页面

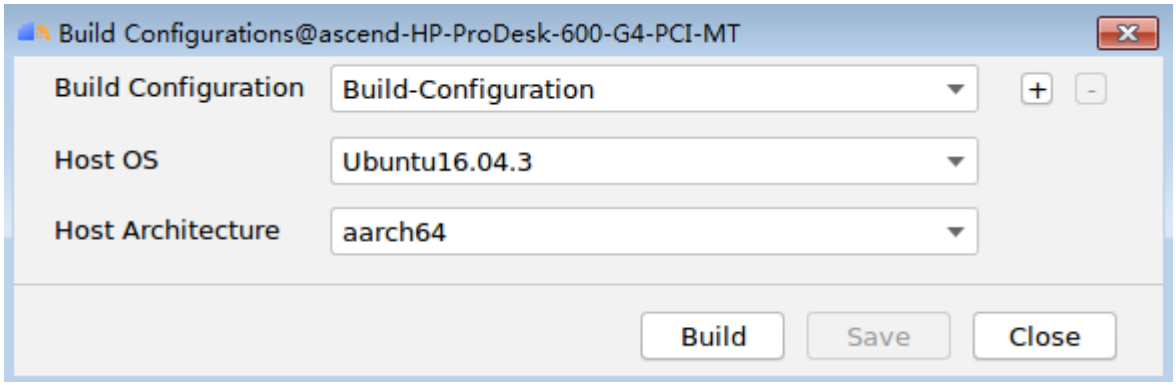


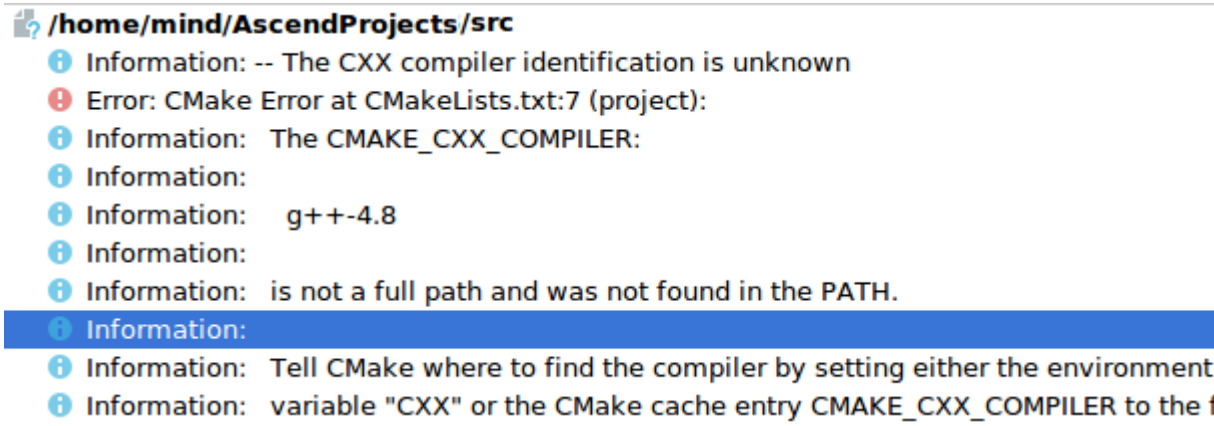
表 6-4 编译配置参数说明

参数	说明
Host OS	选择硬件环境的Host侧操作系统。
Host Architecture	选择硬件环境的Host侧操作系统架构。

步骤3 单击**Build**进行工程编译。

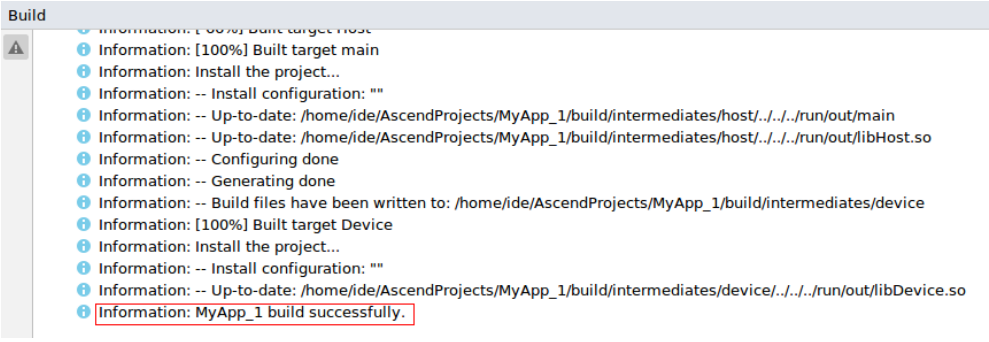
如果出现类似如下问题，请按照**3 编译环境配置**部署g++-4.8编译器。

图 6-4 编译环境问题导致编译失败



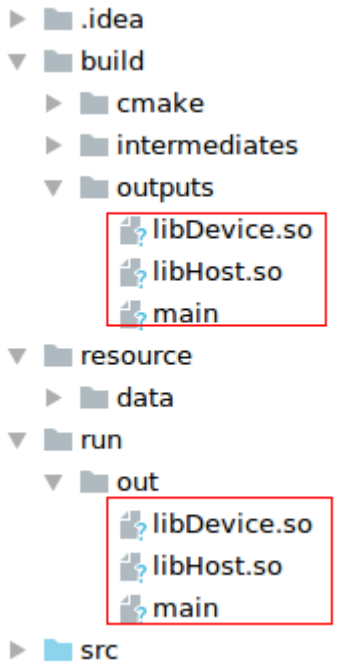
步骤4 确认工程编译结束。

图 6-5 编译成功提示



步骤5 编译完成后在工程目录下的build/outputs文件夹下，生成libDevice.so、libHost.so和可执行文件main。同时自动生成run目录，并将上述文件自动拷贝到run/out文件夹下，为工程运行做准备。

图 6-6 编译生成文件



----结束

工程运行

步骤1 在**Mind Studio**工程界面，依次选择**Run > Edit Configuration...**，进入运行配置页面。

步骤2 选择“Ascend APP > *工程名称*”配置应用工程运行参数，参数说明如**表6-5**所示。

表 6-5 应用工程运行配置参数说明

配置参数	参数说明
Name	运行配置名称，可修改。

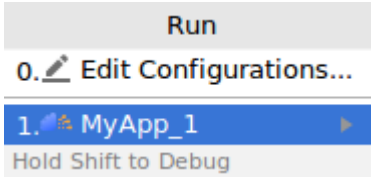
配置参数	参数说明
Target Host Ip	下拉选择运行硬件设备的Host侧的IP地址，若未添加设备，请单击⊞进行添加。 详细的添加设备的方法请参见**8.2.2 增加设备**。
Profilling	可保持默认关闭。如需分析应用工程性能数据，请参考**9 Profiling工具**进行配置。
Before launch	此栏中可配置运行前的预处理操作。

步骤3 配置完运行工程后，单击“Apply”保存运行配置，单击“OK”，关闭运行配置窗口。

步骤4 在**Mind Studio**工程界面，依次选择**Run > Run**。

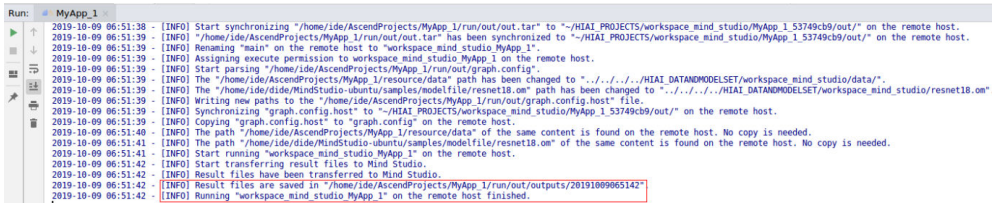
在弹出框中选择已创建好的运行配置信息，进行工程运行，如**图6-7**所示。

图 6-7 选择运行配置信息



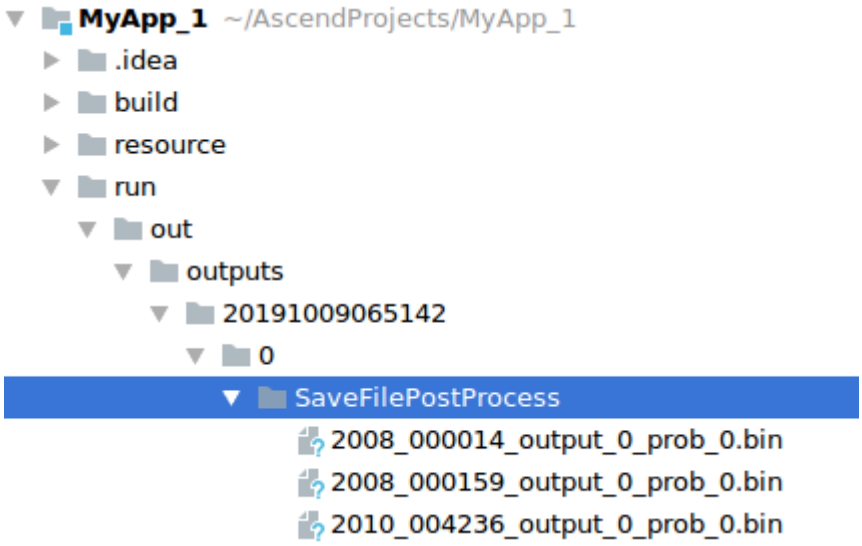
步骤5 确认运行结束，并可以查看到运行结果存放路径，如**图6-8**所示。

图 6-8 运行成功提示



步骤6 查看运行结果，如**图6-9**所示。

图 6-9 查看运行结果



----结束

6.2 自定义算子开发

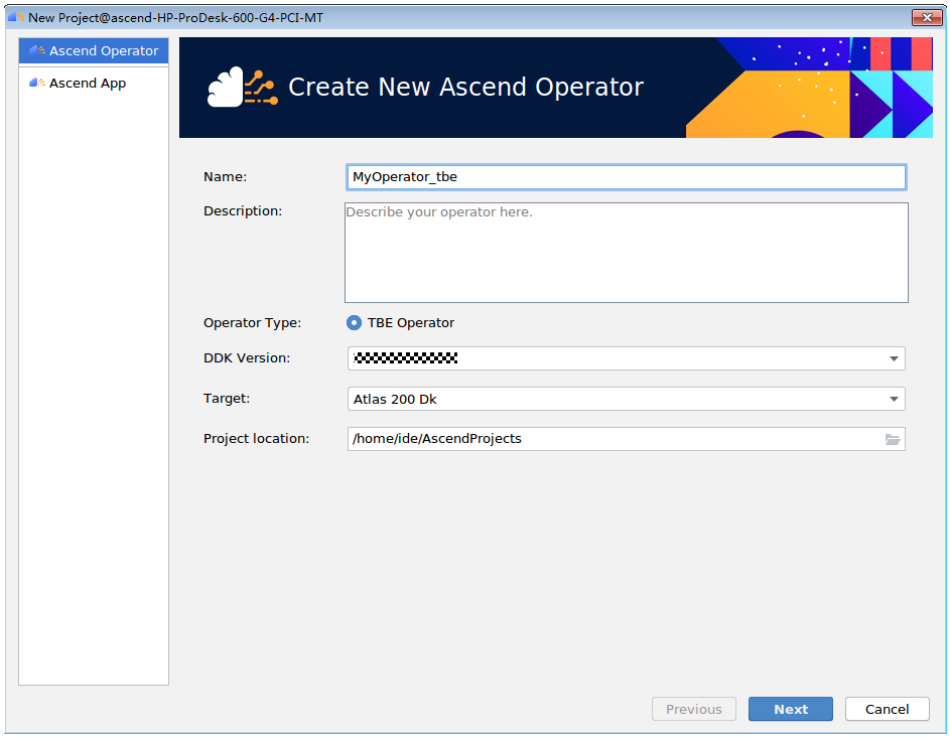
本章节介绍端到端进行算子开发的操作流程，自定义算子开发前，需要参考**2 安装**以及**3 编译环境配置**完成Mind Studio的部署以及交叉编译环境的环境。

工程创建

步骤1 进入算子工程创建界面。

- 首次登录Mind Studio。
  a. 启动Mind Studio。
  b. 在Mind Studio欢迎界面中单击“Create new project”，进入创建工程界面。
     如**图6-10**所示。

图 6-10 Mind Studio 创建工程界面



- 非首次登录Mind Studio
  a. 启动Mind Studio。
  b. 在顶部菜单栏中选择“File > New > Project...”，进入创建工程界面，如图6-10所示。

步骤2 创建TBE算子工程。

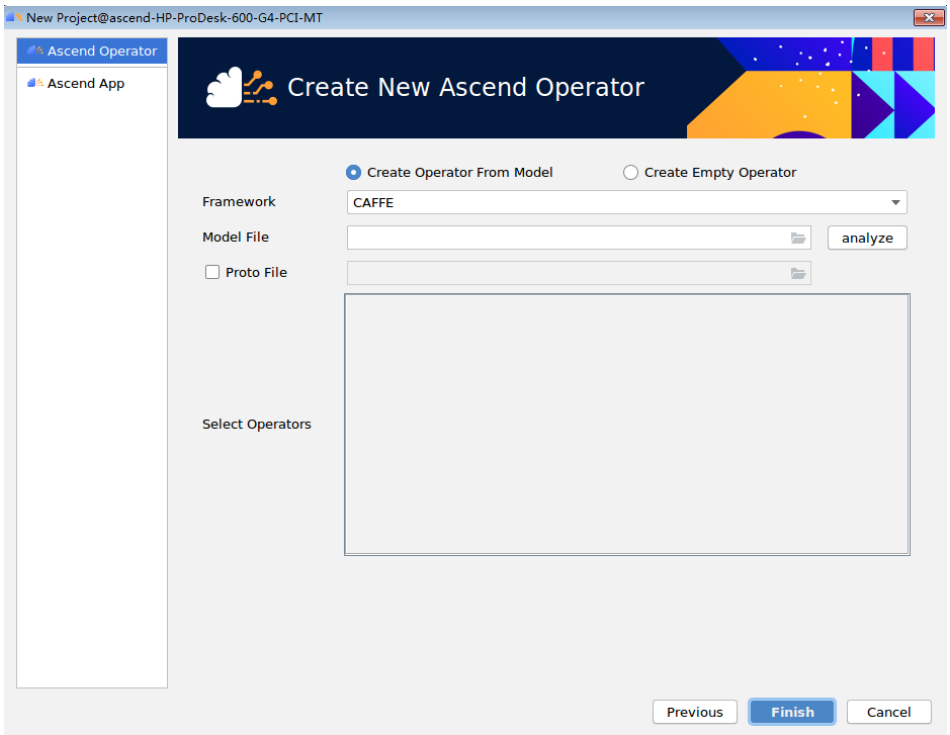
1. 在左侧导航栏选择“Ascend Operator”，在右侧配置工程信息，参数配置如表6-6所示。

表 6-6 工程参数配置说明

参数	参数说明
Name	工程名称，自行配置。 要求配置为以字母开头的字符串，最大长度支持64位。仅支持字母、数字、中划线、下划线。
Description	工程描述信息，自行配置。
Operator Type	TBE Operator，代表创建TBE自定义算子工程。
DDK Version	当前DDK的版本号。
Target	运行环境，请根据实际硬件环境进行选择。
Project location	工程的存储路径，用户可自定义。

2. 单击“Next”，弹出如图6-11所示界面。

图 6-11 TBE 算子工程创建



其中：

- Create Operator From Model：表示通过解析模型创建算子开发工程，若选择此选项 =>步骤2.3。
- Create Empty Operator：表示直接输入算子名称创建算子开发工程，若选择此选项 =>步骤2.4。

3. 选择“Create Operator From Model”，通过解析模型创建算子工程，参数说明如表6-7所示。

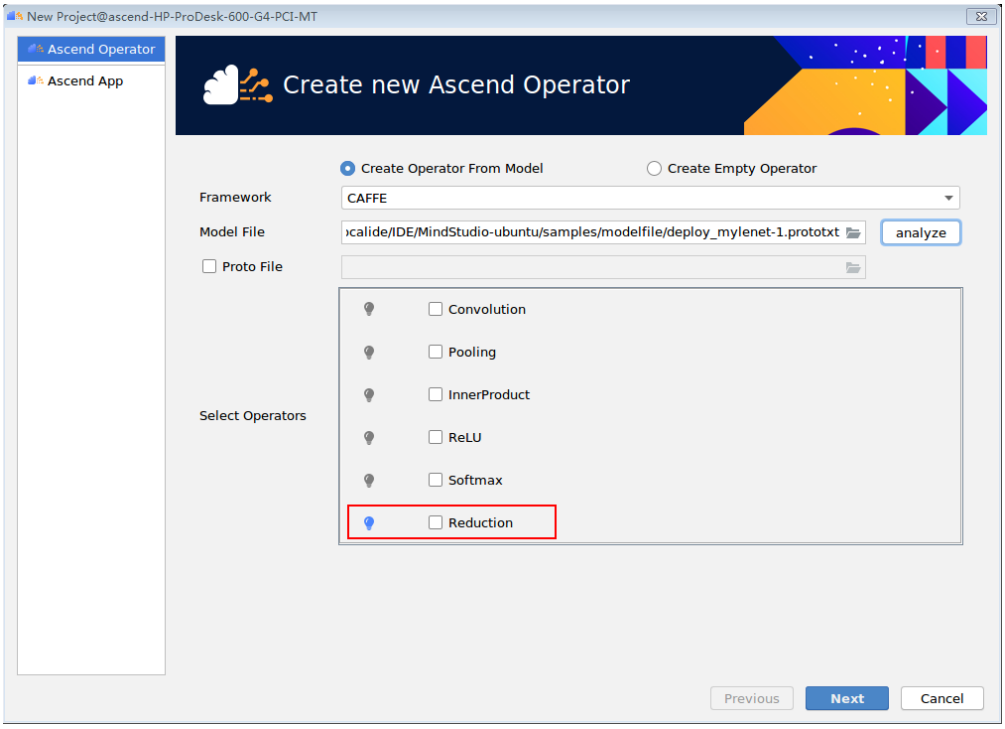
表 6-7 解析模型创建算子工程参数配置说明

参数	参数说明
Framework	模型文件框架类型，当前只支持CAFFE框架。
Model File	单击输入框右侧的，选择服务器中存在的模型文件，然后单击“analyze”进行解析。

参数	参数说明
Proto File	对于CAFFE类型的模型，如果上传的prototxt中存在系统内置caffe.proto中未定义的算子，会导致第一次解析算子失败，此时需要勾选Proto File前的复选框，然后单击上传用户自定义caffe.proto文件。系统会把所需字段合并到内置的caffe.proto中，自定义caffe.proto的方法请参考**13.2 caffe.proto文件算子定义**。 **说明** 系统内置的caffe.proto的存储路径为DDK安装路径下的“/ddk/include/inc/custom/proto/caffe/caffe.proto”。 DDK的默认安装路径为Mind Studio安装用户家目录的.mindstudio/huawei/ddk/*ddk_version*，其中*ddk_version*请替换为实际DDK版本号）
Select Operators	此区域会列出模型中的每一层算子，此处可以选择一个或多个算子进行算子的自定义。

Mind Studio中内置了Mylenet网络的模型文件，存储路径为Mind Studio安装路径下的“samples/modelfile/deploy_mylenet-1.prototxt”，选择deploy_mylenet-1.prototxt文件，并单击“analyze”，选择自定义Reduction算子为例进行工程创建，如**图6-12**所示。

图 6-12 基于 CAFFE 模型的 TBE 工程创建示例



昇腾AI处理器中已经支持的算子前图标为 ，昇腾AI处理器中不支持的算子前图标为 。

须知

若模型中有多个不支持的算子，请同时勾选所有不支持算子，使用同一工程进行自定义算子的开发。

下一步，请执行**步骤2.5**。

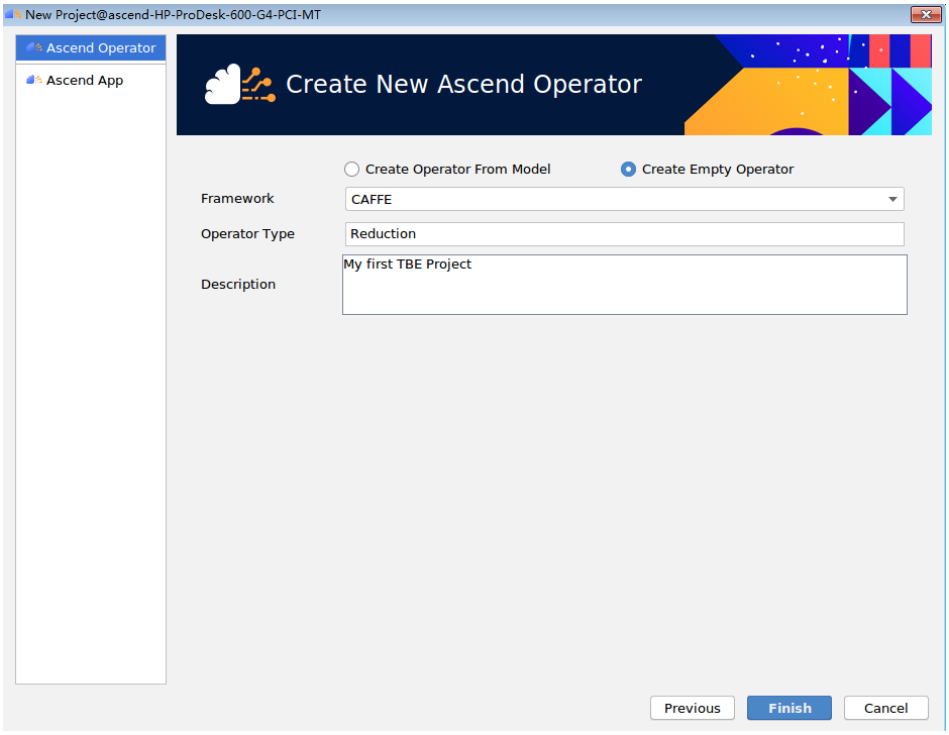
4. 选择“Create Empty Operator”，直接输入算子名称创建算子开发工程，参数说明如**表6-8**所示。

表 6-8 直接输入算子名称创建工程参数配置

参数	参数说明
Framework	模型文件框架类型，当前只支持CAFFE框架。
Operator Type	自定义的算子类型。
Description	算子描述，自行配置。 需要为字符串，最大256字节。

配置示例如**图6-13**所示。

图 6-13 直接输入算子名称创建工程配置示例

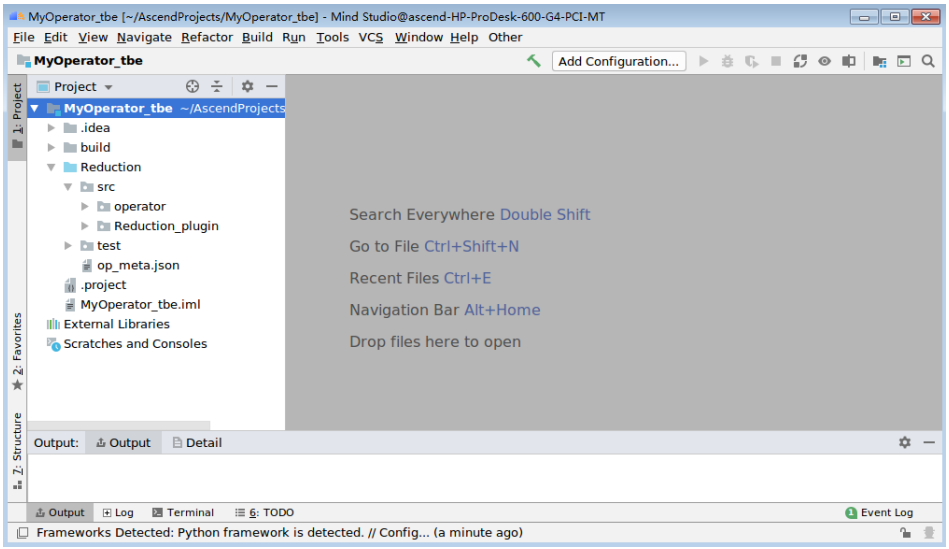


下一步，请执行**步骤2.5**。

5. 单击“Finish”，完成算子工程的创建。

   会进入如**图6-14**所示的算子工程开发界面。

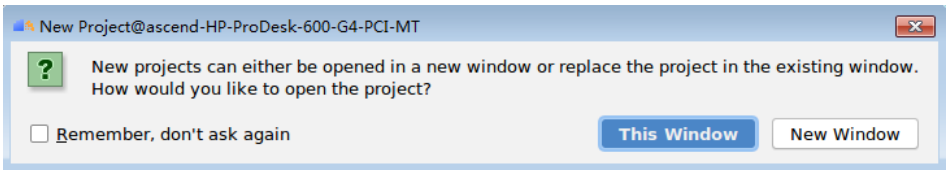
图 6-14 自定义算子工程开发窗口



若工作窗口已打开其他工程，会出现如图6-15所示提示。

- 选择“This Window”，则直接在当前工作窗口打开新创建的工程。
- 选择“New Window”，则新建一个工作窗口打开新创建的工程。

图 6-15 工程创建提示



说明

新建工程的默认存储路径为Mind Studio安装用户家目录的“AscendProjects”文件夹下。

----结束

算子代码开发

步骤1 双击打开“*算子名称*/src/operator/*算子名称.py*”文件。

步骤2 实现自定义算子。

在下图所示区域用Python语言调用TBE DSL接口进行算子的逻辑实现。

详细的算子代码实现请参考《 TBE自定义算子开发指导（ Mind Studio ）》。

图 6-16 TBE 算子逻辑代码实现

```
import te.lang.cce
from te import tvm
from topi import generic


def Reduction(shape, dtype, operation, axis, coeff, kernel_name = "Reduction", need_bu
    """

    Parameters
    ----------

    kernel_name : kernel name, default value is "Reduction"

    need_buid : if need to build CCEC kernel, default value is False

    need_print : if need to print the ir, default value is False

    Returns
    -------
    None

    """

    """
    TODO:
    Please refer to the TE DSL Manual, And code here with TE DSL.
    """

#if __name__ == "__main__":
```

说明

Mind Studio新建工程不支持自动实现对TBE Python依赖库的解析，开发者可参考**13.3 设置Mind Studio支持自动解析TBE Python依赖库**进行配置，实现Mind Studio对Python依赖库的自动解析。

步骤3 实现单算子执行代码，用于单算子的运行验证。

在下图所示区域，实现单算子的执行代码。

图 6-17 单算子调用代码实现

```
import te.lang.cce
from te import tvm
from topi import generic


def Reduction(shape, dtype, operation, axis, coeff, kernel_name = "Reduction", need_bu
    """

    Parameters
    ----------

    kernel_name : kernel name, default value is "Reduction"

    need_buid : if need to build CCEC kernel, default value is False

    need_print : if need to print the ir, default value is False

    Returns
    -------
    None

    """

    """
    TODO:
    Please refer to the TE DSL Manual, And code here with TE DSL.
    """

#if __name__ == "__main__":
```

例如：

以shape为(2,3,4)，datatype为float16，axis为1，op为SUM，coeff为2，算子名称为Reduction的参数来调用reduction算子。

示例代码如下：

```
if __name__ == "__main__":
    Reduction((2, 3, 4), "float16", "SUM", 1, coeff=2, kernel_name="Reduction")
```

----结束

算子插件代码开发

在自定义算子工程的“*算子名称* > src > *算子名称_plugin* > *算子名称_parse.cpp*”文件中。

Mind Studio已自动生成了算子插件框架，详细的插件代码实现请参考《 TBE自定义算子开发指导 （ Mind Studio ）》。

步骤1 实现算子解析函数Status xxxParseParams(const Message* op_origin, ge::Operator& op_dest)，将原始模型中的算子定义转换为适配昇腾处理器的离线模型中的算子定义。

- 若算子工程是通过“Create Operator From Model”方式创建，则此函数Mind Studio会自动实现int、float以及enum类型参数的解析，repeated类型的参数仍需用户自定义实现解析。

- 若算子工程通过“Create Empty Operator”方式创建，则此函数需要用户自定义实现。

步骤2 实现推理算子输出张量描述：Status xxxInferShapeAndType(const ge::Operator& op, vector<ge::TensorDesc>& v_output_desc)。

Mind Studio生成的框架函数已经实现了获取算子输入对象的tensorDesc以及其shape信息，然后直接将tensorDesc信息赋值给输出对象。

若输出张量描述需要根据算子逻辑进行推理生成，需要用户自定义实现。

步骤3 实现算子编译函数，构建算子的二进制文件及描述文件：Status xxxBuildTeBin(const ge::Operator& op, TEBinInfo& te_bin_info)

- “FilePath”请替换为当前算子文件所在绝对路径+算子py文件名称，例如“/home/ascend/MindStudio-ubuntu/samples/demo_projects/tbe/Reduction/src/operator/Reduction”。
- “FuncName”为Reduction.py文件中算子实现的函数名称，例如Reduction。
- “KernelName”为Reduction.py文件中设置的算子的KernelName，例如Reduction。
- te::BuildTeCustomOp( )函数中的算子参数调用顺序需要与算子实现文件Reduction.py中算子实现方法def Reduction( )中定义的算子参数顺序保持一致。

  例如，Reduction.py中Reduction算子实现方法定义如下所示：

```
def Reduction(shape, dtype, operation, axis, coeff, kernel_name = "Reduction", need_build = True,
need_print = False):
```

  插件代码Reduction_parse.cpp中BuildTeCustomop（ ）的调用如下：

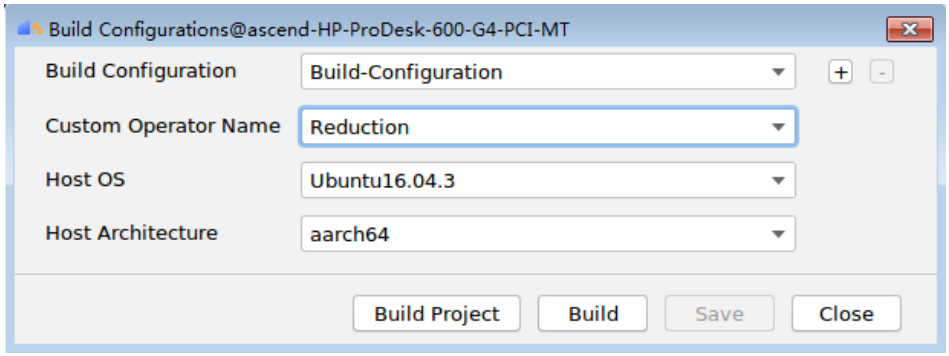
```
te::BuildTeCustomOp(te_bin_info.ddk_version, op.GetName(), FilePath, FuncName,
                "(i,i,i,i), s, s, i, f, s",input_desc.GetShape().GetDim(0),
input_desc.GetShape().GetDim(1),              input_desc.GetShape().GetDim(2),
input_desc.GetShape().GetDim(3), "float16", operation.c_str(), axis, coeff,   KernelName.c_str());
```

----结束

工程编译构建

步骤1 选中创建的TBE工程，在顶部菜单栏中依次单击“Build > Edit Build Configuration”如下图所示。

图 6-18 编译自定义算子



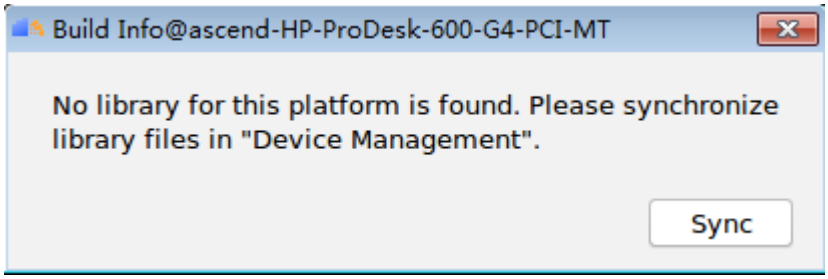
其中：

- Custom Operator Name：需要编译的自定义算子。

- Host OS：硬件侧Host的操作系统版本。
- Host Architecture：硬件侧Host操作系统的类型。

如果出现如下提示，说明编译依赖的lib库没有从运行环境同步到开发环境上，请单击**Sync**进入设备管理界面，然后参考**3 编译环境配置**进行相关操作。

图 6-19 未同步 lib 库



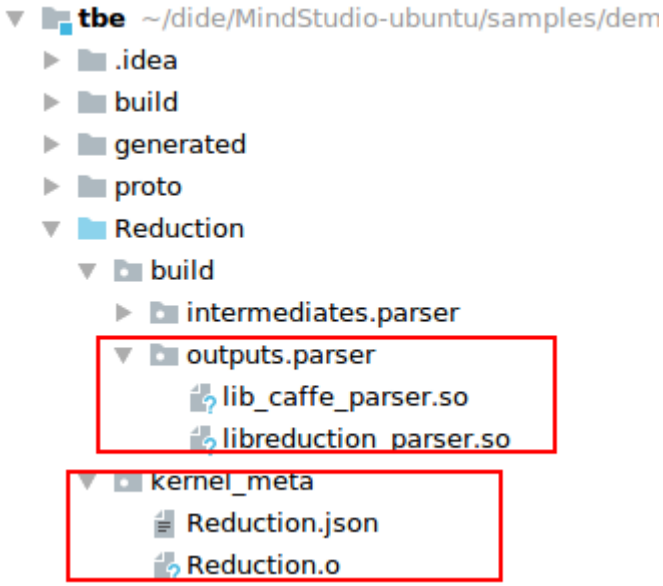
步骤2 单击**Build**进行编译。

说明

- **Build**为编译当前选中的单个算子。
- **Build Project**为编译当前工程中的所有算子。
- 编译过程中会同步编译算子插件。

- 自定义算子编译完成后，会在算子工程目录下创建kernel_meta文件夹，并在kernel_meta文件夹下生成算子二进制文件Reduction.o及算子描述文件Reduction.json。
- 算子插件编译完成后，会在算子工程目录的“build > outputs.parser”目录下生成caffe.proto的二进制文件**lib_caffe_parser.so**与算子插件二进制文件**libreduction_parser.so**。

图 6-20 算子工程编译结果文件



----结束

单算子运行验证

对编译生成的算子二进制文件（kernel_meta目录下的*.o文件）进行验证，验证单算子输出结果是否正确。

步骤1 构造单算子运行测试数据。

算子数据包含输入数据，期望输出数据与实际输出数据，详细描述如表6-9所示。

表 6-9 算子输入输出数据描述

类别	描述
Input	算子的输入数据文件。
Expect	用于校验实际输出结果和用户预期结果是否一致。 ● Precision Deviation：精度偏差，表示单数据相对误差允许范围，数值范围为[0，1），精度偏差越小表示精度越高。 ● Statistical Discrepancy：统计偏差，整个数据集中精度偏差不满足门限的数据量占比，数值范围（0，1），统计偏差越小表示精度越高。
Output	描述算子的输出数据信息及指定输出文件路径。 ● Path：输出数据文件，生成于output文件夹下。 ● Size：输出数据大小，单位字节。 ● Data Type：输出数据类型，配置1(float16)或者2(float32)。

用户需要在算子工程目录的**test/testcase.py**文件中配置算子数据，定义运行代码，详细实现流程如下所示。

- 在def example()中配置算子数据。
  a. 实例化DataProvider对象。
  ```
  dp1 = dp.DataProvider()
  ```
  b. 构造输入数据。

     输入数据通过已存在的输入数据文件构造。
  ```
  example_input_path = os.path.join('path', 'to', 'input_data')  //设置输入数据文件的路径，路径是由输入参数的各个字段拼接组成，需要为相对于testcase.py文件的相对路径
  dp1.add_input_by_data_path(example_input_path)        //将输入数据文件的路径设置到dp1对象中
  ```
     对于os.path.join('path', 'to', 'input_data')，其中“input_data”赋值为输入数据文件的名称，“path”与“to”可以赋值为输入数据文件所在的路径，三个字段之间用“/”进行拼接。

     例如os.path.join('../../', 'redution_input_1.data')与os.path.join('..', '..','redution_input_1.data')，都表示输入数据文件为“../../redution_input_1.data”。
  c. 构造输出数据。
  ```
  size = 2        //设置输出的大小，单位字节
  data_type = 1        //设置输出的类型，1：float16, 2：float32
  dp1.add_output('out0.data', size, data_type)//将输出数据文件设置到dp1对象中，文件名称为out0.data
  ```
  d. 构造期望输出数据。

     期望输出数据通过添加已存在的期望输出数据文件构造。

```
example_expect_path = os.path.join('path', 'to', 'expect_data')  //设置期望输出数据文件的路径，路径是由输入参数的各个字段拼接组成，需要为相对于testcase.py文件的相对路径
dp1.add_expect_by_data_path(example_expect_path)        //将输出数据文件的路径设置到dp1对象中
```

对于os.path.join('path', 'to', 'expect_data')，其中“expect_data”赋值为期望输出数据文件的名称，“path”与“to”可以赋值为期望输出数据文件所在的路径，三个字段之间用“/”进行拼接。

例如os.path.join('../../', 'redution_expect_1.data')与os.path.join('..', '..','redution_expect_1.data')，都表示期望输出数据文件为“../../redution_expect_1.data”。

e. （可选）设置精度。

```
dp1.set_precision_deviation(0.2)
dp1.set_statistical_discrepancy(0.2)
```

set_precision_deviation：精度偏差，表示数据相对误差的允许范围，取值为[0,1)，精度偏差越小表示精度越高。

set_statistical_discrepancy：统计偏差，整个数据集中精度偏差不满足门限的数据量占比，数值范围（0，1），统计偏差越小表示精度越高。

f. 返回构造好的DataProvider。

```
return dp1
```

- 在def testcase( )中自定义实现代码，并返回已经实例化的DataProvider对象。

```
......
return example()
```

步骤2 配置运行参数。

1. 选中工程名称，单击顶部菜单栏的“Run > Edit Configurations”，进入运行配置界面。
2. 选择“**Ascend Operator > *算子名称***”配置算子工程运行参数，参数说明如**表6-10**所示。

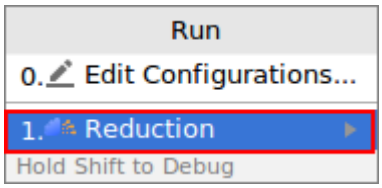
表 6-10 算子工程运行配置参数说明

配置参数	参数说明
Target Host Ip	下拉选择运行硬件设备的Host侧的IP地址，若未添加设备，请单击⊞进行添加。 详细的添加设备的方法请参见**8.2 设备管理**。
Operator Type	算子类型，若此工程中有多个算子，此处可下拉选择需要运行的算子。
Before launch	此栏中可配置运行前的预处理操作。

3. 配置完运行工程后，单击“Apply”保存算子运行配置。
4. 单击“OK”，关闭运行配置窗口。

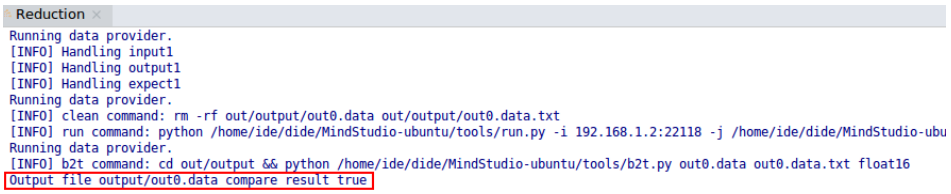
步骤3 单击菜单栏中的“Run > Run”，在如**图6-21**所示界面中选择需要运行的算子。

图 6-21 运行选择界面



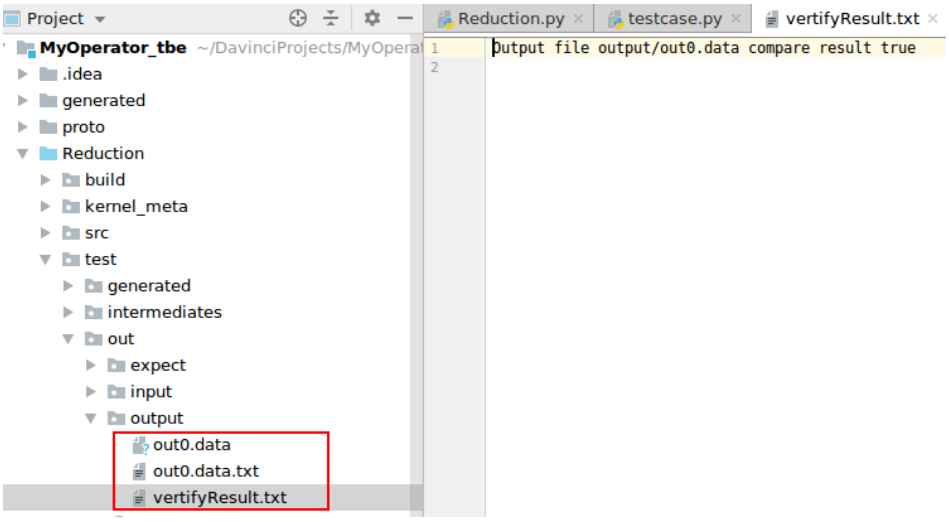
- 运行成功后，输出如图6-22所示。

图 6-22 运行输出



红框中compare result true代表运行成功，另外在工程的“***算子名称*** > **test** > **out** > **output**”目录下会生成如下文件。

图 6-23 运行结果文件



其中：

- out0.data为输出数据二进制文件。
- out0.data.txt为输出数据的可视化的文本文件。
- vertifyResult.txt为比对结果文件。
- 若运行失败，用户可在Log页签查看详细的System Log进行排查。

- 若运行失败，用户可在Log页签查看详细的System Log进行排查。
  详细的日志查看方法可参考**10 日志工具**。

----结束

加载插件转换模型

算子代码及算子插件代码开发完成后，开发者可加载**工程编译构建**章节生成算子插件文件进行模型转换 ，详细模型转换操作可参考**7.2 模型转换**。

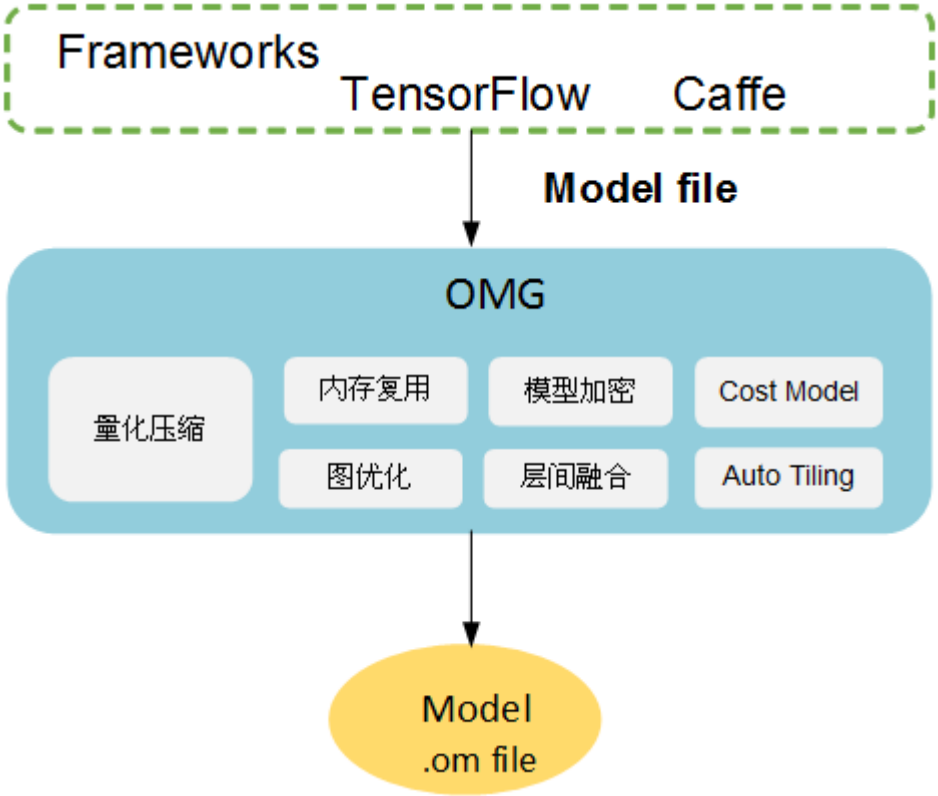
7 模型转换

7.1 概述
7.2 模型转换
7.3 模型可视化

7.1 概述

用户使用Caffe/Tensorflow等框架训练好的模型，可通过OMG工具将其转换为昇腾AI处理器支持的离线模型，模型转换过程中可以实现算子调度的优化、权重数据重排、量化压缩、内存使用优化等，可以脱离设备完成模型的预处理，详细架构如**图7-1**所示，OMG详细概念介绍请参见**OMG/OME**。

图 7-1 OMG 工具架构



约束说明

在进行模型转换前，请务必查看如下约束要求：

- 只支持原始框架类型为caffe和tensorflow的模型转换，当原始框架类型为caffe时，输入数据类型为FLOAT；当原始框架类型为tensorflow时，输入数据类型为INT32、BOOL、UINT8、FLOAT。
- 当原始框架类型为caffe时，模型文件（.prototxt）和权重文件（.caffemodel）的op name、op type必须保持名称一致（包括大小写）。
- 当原始框架类型为Caffe时，除了top与bottom相同的layer以外（例如BatchNorm，Scale，ReLU等），其他layer的top名称需要与其name名称保持一致。
- 当原始框架类型为tensorflow时，只支持FrozenGraphDef格式。
- 不支持动态shape的输入，例如：NHWC输入为[?,?,?,3]多个维度可任意指定数值。模型转换时需指定固定数值。
- 输入数据最大支持四维，转维算子（reshape、expanddim等）不能输出五维。
- 模型中的所有层算子除const算子外，输入和输出需要满足dim!=0。
- 模型转换不支持含有训练算子的模型。
- 模型中的算子只支持2D卷积，暂不支持3D卷积。
- 只支持《算子规格说明》中的算子，并需满足算子限制条件。

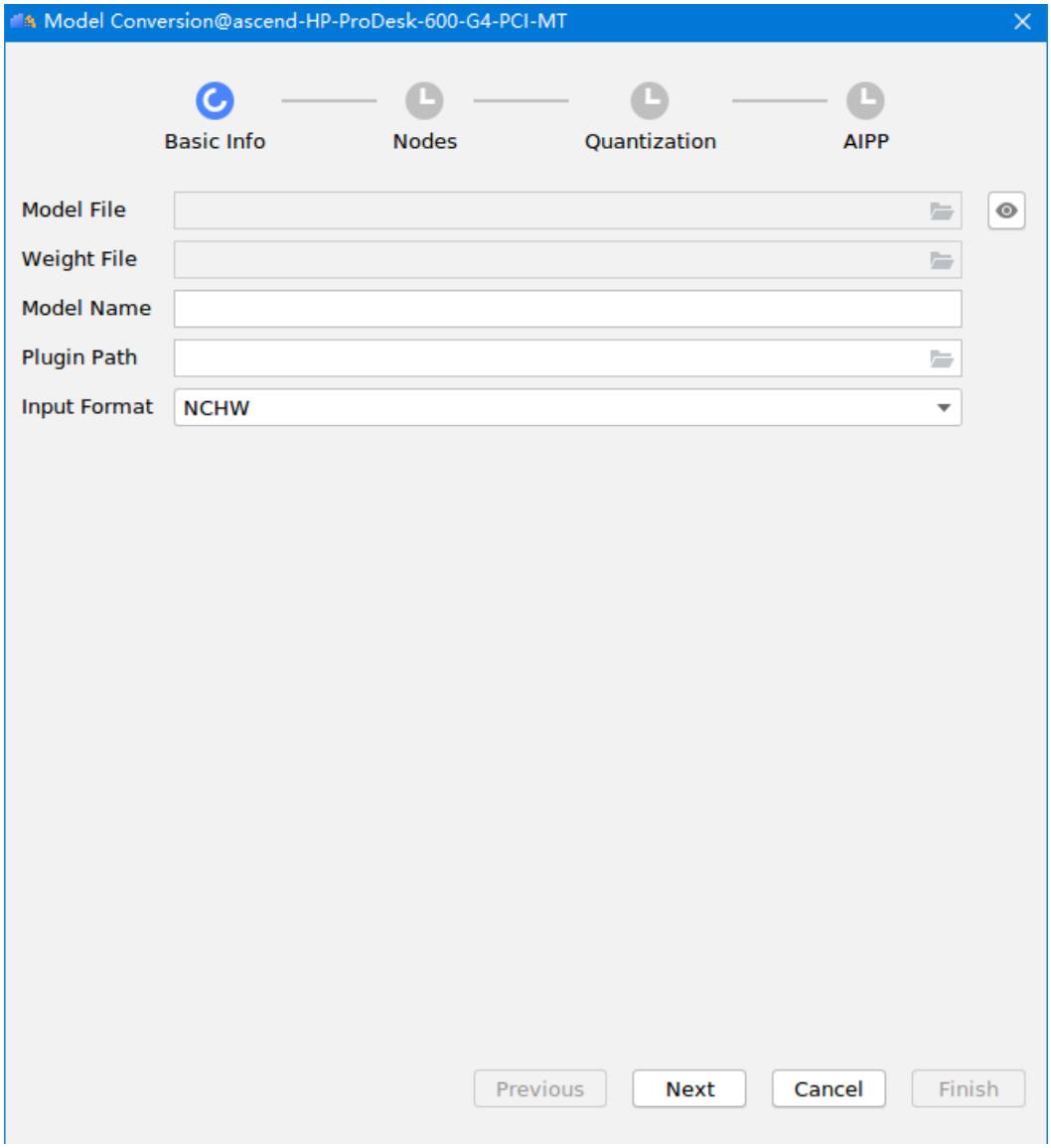
模型转换入口

可以通过如下两种方式进入模型转换界面。

- 在菜单栏选择“Tools > Model Convert”。
- 在菜单栏选择“View > Toolbar”，菜单栏下方会出现一行工具栏，选择左侧的 。

模型转换页面如图7-2所示。

图 7-2 模型转换界面



7.2 模型转换

本章节以Caffe框架的resnet18网络模型为例进行讲解。

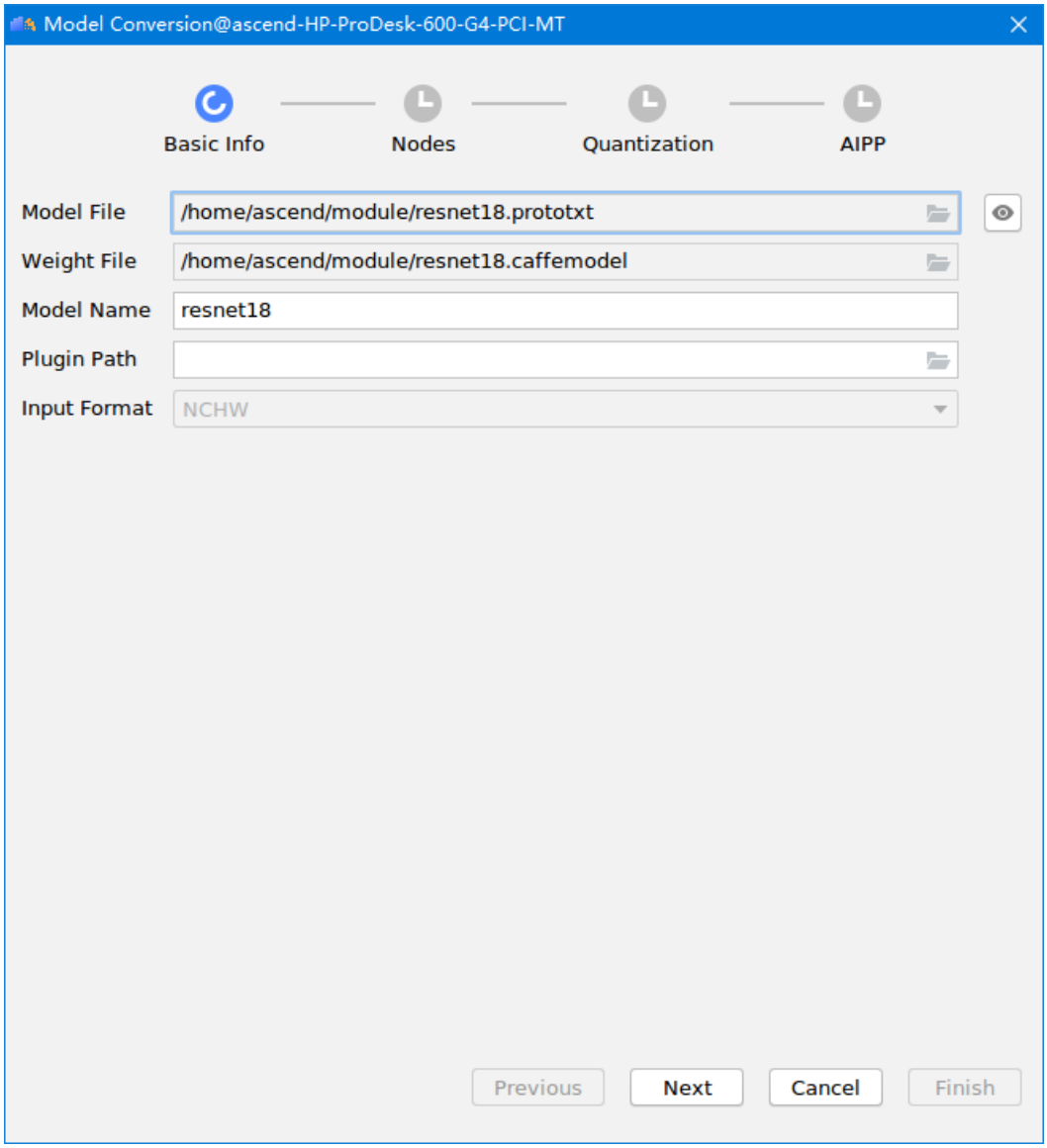
前提条件

进行模型转换前，使用Mind Studio安装用户，将所转换模型的模型文件以及权重文件上传到Mind Studio安装服务器。

操作步骤

步骤1 打开模型转换页面，在“Basic”页签中上传模型文件和权重文件，如图7-3所示。

图 7-3 上传模型文件



参数解释如表7-1所示。

表 7-1 Basic 界面参数配置

参数	说明
Model File	模型文件。 单击右侧的文件夹图标，在后台服务器路径选择需要转化的模型对应的*.prototxt文件，并上传。 如果之前上传过模型，并模型转换成功，则会在右侧出现查询按钮，单击该按钮，可以从历史记录中获取相应的文件再次上传。

参数	说明
Weight File	权重文件。 ● 当原始框架是Caffe时，该参数存在： – 如果模型文件和权重文件存在于后台服务器同一目录下，且名称和模型文件名称相同，则选择模型文件后，权重文件会自动填充。 – 如果模型文件和权重文件存在于后台服务器不同目录下，或者在同一目录下，但名称和模型文件名称不相同，则需要手动上传权重文件，单击右侧的文件夹图标，在后台服务器路径选择模型文件对应的*.caffemodel权重文件，上传即可。 ● 当原始框架是Tensorflow时，该参数不存在。
Model Name	模型文件名称，上传模型文件后，该参数会自动填充，用户可以根据需要自行修改名称，要求如下： 只支持a-z、A-Z、0-9、下划线以及短划线的组合，最多支持40个字符，名称需要保持唯一。
Plugin Path	自定义算子插件二进制文件所在目录，若当前工程中有多个自定义算子插件，请选择同时包含所有插件目录的顶级目录。 如果有针对所导入模型的算子进行自定义开发，此处需要导入自定义算子对应对的算子插件文件。 **说明** 模型转换时，在Plugin Path参数处指定了自定义算子插件，则配置量化时，请确保**步骤3**中Quantization Configuration开关关闭。
Input Format	输入图片格式。 ● 当原始框架是Caffe时，Input Format默认值是NCHW，不支持修改。 ● 当原始框架是Tensorflow时，Input Format默认是NHWC，用户可以根据需要修改为NCHW格式。

用户可以单击右侧的 按钮，查看该模型的原始网络结构图：

- 选中某层算子，该层算子会出现绿色选中框，右上角区域会展示该算子的详细信息，包括算子名称，算子类型以及算子的参数信息等，单击相关参数后面的▶可以查看参数的详细信息，展示的信息同*.prototxt文件中该层算子的信息。
- 选中某层算子，右击选择“mark output”，该层变成红色，转换完成后，选中层的输出会直接作为模型的输出，如**图7-4**所示。右击选择“cancel mark”取消选中。本节以“mark output”名称为“res2b”，类型为“Eltwise”算子为例进行说明。

  图7-4右下角会展示该网络模型的整体结构，包括左侧展示框中的算子在整体网络结构中的位置，即图中的蓝色框选择位置。左侧展示区域下拉，右侧整体区域中的蓝区选择框随之往下移动。左侧展示框放大和缩小，右侧蓝色框随之缩小和放大。

说明

- 如果不标记某层算子，则模型的输出默认为最后一层的算子信息，某些情况下，用户想要查看某层算子参数是否合适，则需要将该层算子的参数输出，既可以在选择模型时通过将该层算子标记为“mark output”，模型转换后，在相应.om模型文件最后一层即可以看到标记层算子的参数信息。详细信息请参见**步骤2**。
- 如果模型转换过程中该算子被融合掉，则该算子不能“mark output”标记为输出。

- 在**图7-4**所示界面左侧空白区域单击“Ctrl+F”快捷键，在界面右下方会弹出搜索区域，如**图7-5**所示。

  搜索区域中给出了该模型所用到的所有算子，您可以在搜索区域对话框中输入算子名称，下方搜索区域会列出相关的算子。选择其中一个算子，左侧网络拓扑结构中相应算子会显示绿色选中框，右上方会展示该算子的详细信息，包括算子名称、算子输入、输出等信息。单击搜索区域中的“Cancel”退出搜索页面。

图 7-4 模型网络结构

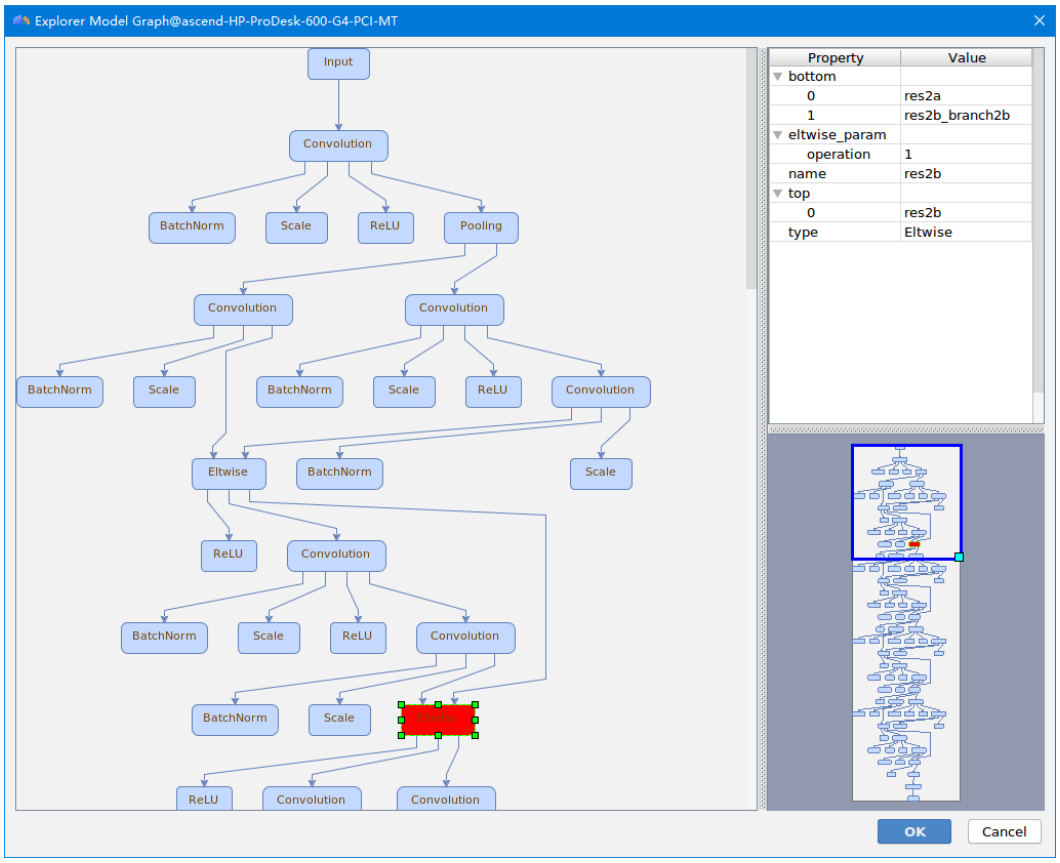
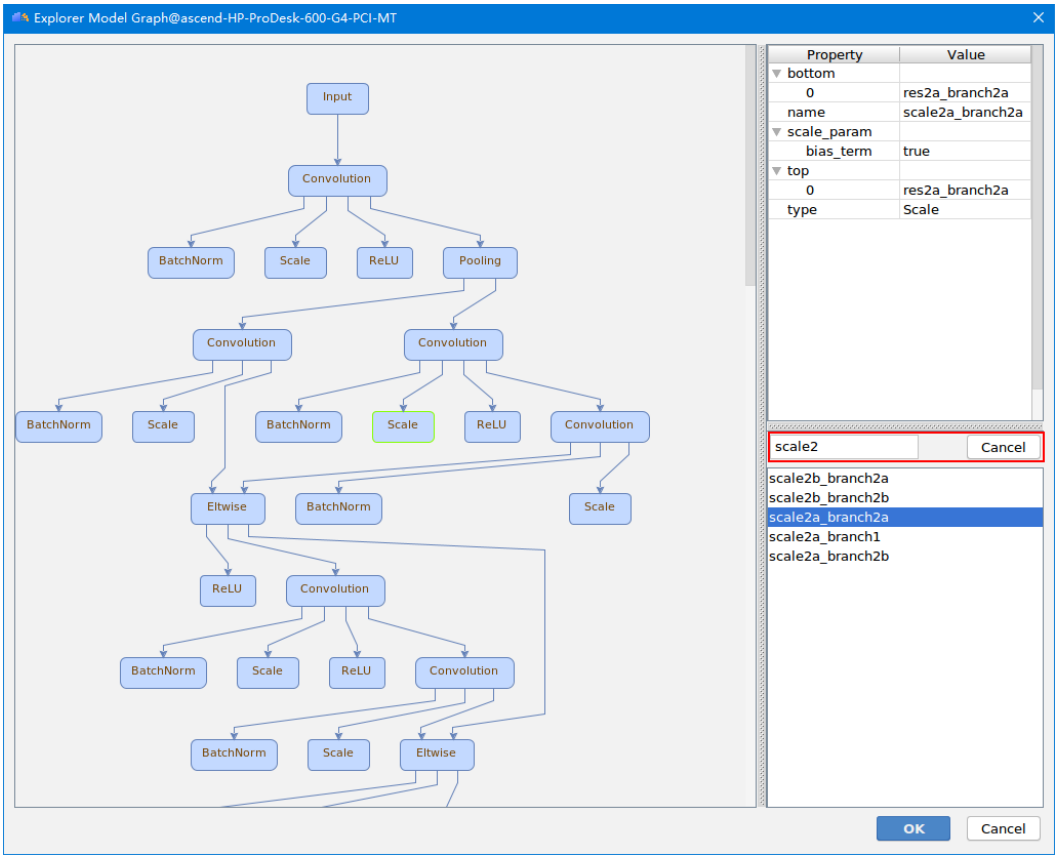


图 7-5 模型可视化界面的搜索功能



单击“OK”退出编辑。

步骤2 单击“Next”，进入“Nodes”节点配置页签，如图7-6、图7-7所示。

图 7-6 配置 Nodes 信息（Static）

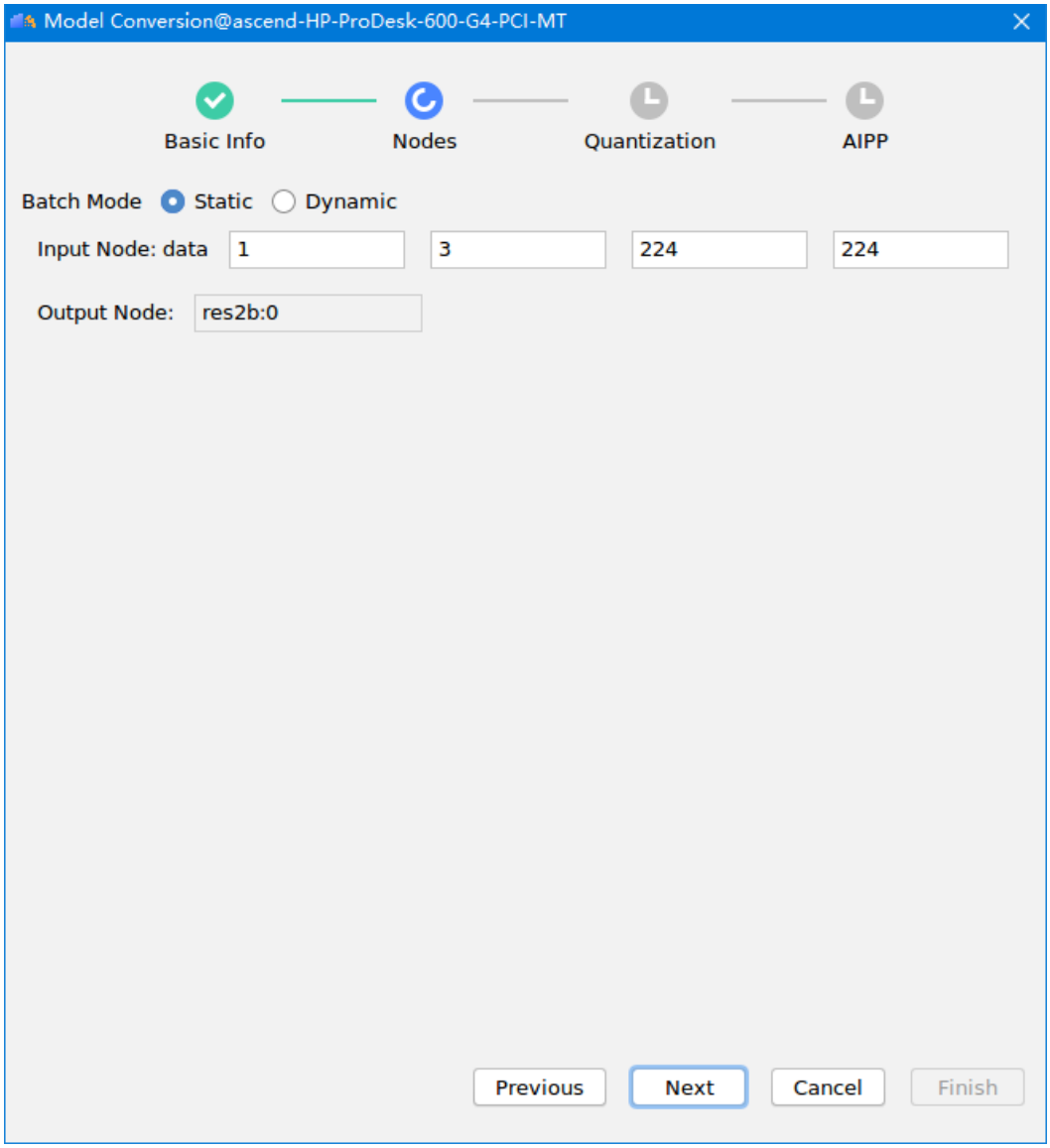
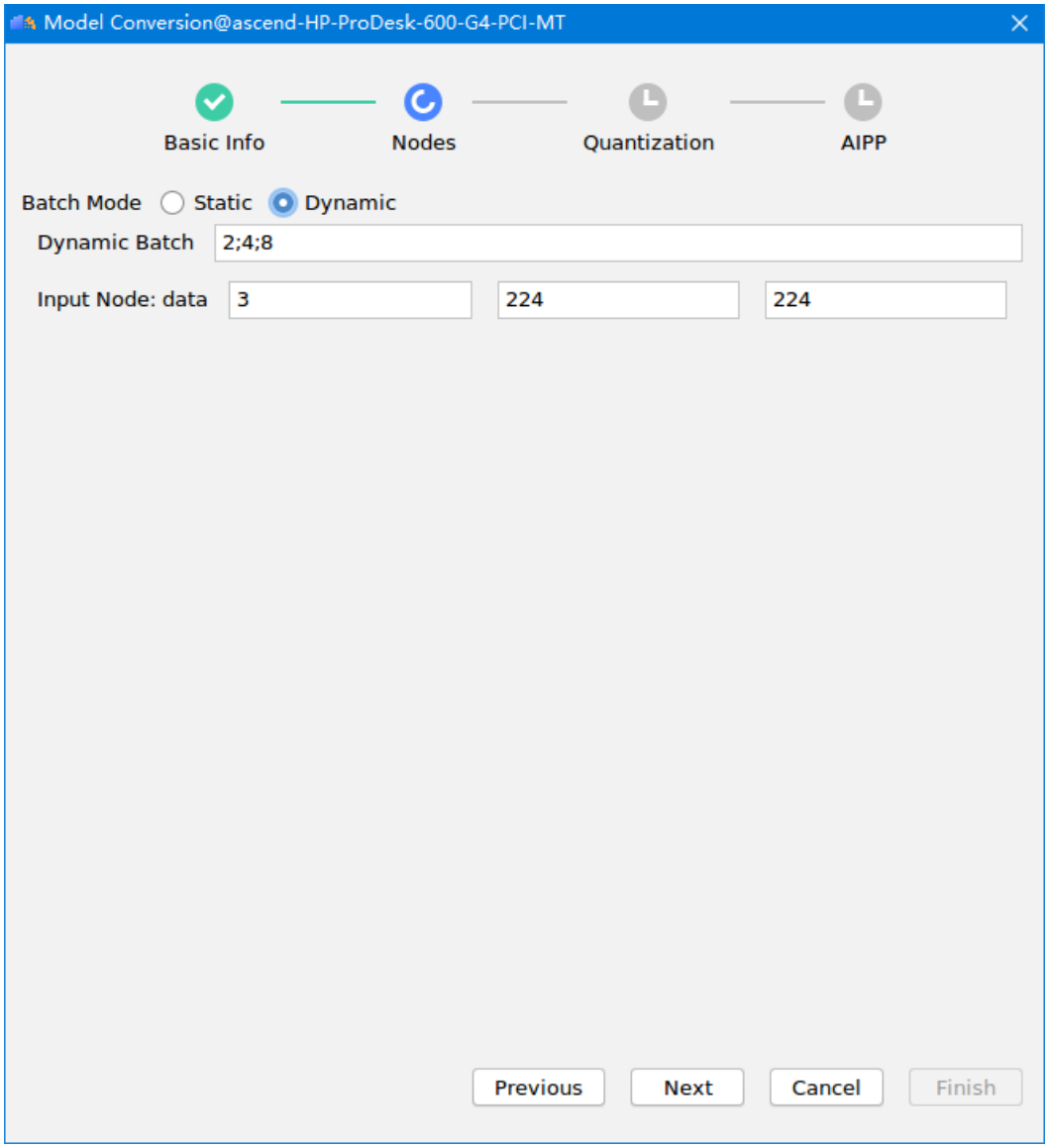


图 7-7 配置 Nodes 信息（Dynamic）



参数说明如表7-2所示。

表 7-2 节点配置参数

参数	说明
Batch Mode	处理图片的模式，默认为Static。本示例采用Static。 ● Static：静态模式，若设置为该值，则模型一次处理的图片数量为Input Node:data第一个参数的取值。适用于业务每次推理图片数量固定的场景。 ● Dynamic：动态模式，若设置为该值，则由用户根据Dynamic Batch参数中具体选定的Batch数，决定模型一次处理图片的数量。适用于业务每次推理图片数量不固定的场景，可以提高处理性能，减少模型空间和内存占用。

参数	说明
Dynamic Batch	当“Batch Mode”取值为“Dynamic”出现。 动态batch数，最多支持8档配置，每一档通过英文分号分隔，每个档位数值限制为：[1~2048]。例如图7-7中所示的[2;4;8]，建议配置为3档。每档最大数值建议不超过32。 由于内存限制，支持用户可配置的档位数值以及档位数不同： ● 支持的最大档位数值受Input Node大小影响，不同的Input Node大小（N*C*H*W）支持的最大档位数值不同：Input Node值越大，支持用户可配置的最大档位数值应该越小。 ● 支持的最大档数受模型大小（.prototxt文件大小）影响，不同的模型大小支持的最大档不同：模型越大，支持用户可配置的最大档数应该越小。 **说明** ● 具体申请多少档位由用户根据实际业务情况决定，例如用户执行推理业务时需要每次处理2张，4张，8张图片，则可以申请档位[2;4;8]，申请了档位后，模型推理时会根据档位中的最大数值申请内存。若用户申请了档位后，执行推理业务时需要处理的图片数量与档位中的数值不匹配，则取与实际图片数量相近的最大档位进行处理，例如用户实际需要处理5张图片，则执行推理业务时获取的Dynamic Batch取值为8。 ● 执行推理业务时，模型一次处理图片的数量要小于等于Dynamic Batch档位中的最大值。 ● 如果用户设置的档位超过3档或档位数值过大，可能会导致模型转换失败，此时建议用户减少档位或调低档位数值。 ● 如果使用了该参数，则图7-4中不支持标记某层算子的输出。

参数	说明
Input Node:data	输入数据。 ● Batch Mode取值为Static： – 对于Caffe模型： 右侧四个输入框分别代表输入数据的N（模型一次处理的图片个数），C（Channel，例如彩色RGB图像的Channel数为3），H（Height），W（Width）。 – 对于Tensorflow模型： 右侧四个输入框分别代表输入数据的N（模型一次处理的图片个数），H（Height），W（Width），C（Channel，例如彩色RGB图像的Channel数为3）。 ● Batch Mode取值为Dynamic： – 对于Caffe模型： 右侧三个输入框分别代表输入数据的C（Channel，例如彩色RGB图像的Channel数为3），H（Height），W（Width）。 – 对于Tensorflow模型： 右侧三个输入框分别代表输入数据的H（Height），W（Width），C（Channel，例如彩色RGB图像的Channel数为3）。 Input Node会自动解析对应模型文件中“input_param”的参数并填充相应的N、C、H、W；如果无法解析，需要用户根据实际.prototxt文件中的“input: "data"”对应的参数填写。如果模型有多个输入，模型转换界面还会出现“Input Node:im_info”参数，该参数的取值需要根据.prototxt文件中“input: "im_info"”对应的参数填写。
Output Node	输出节点信息。 ● 如果图7-4中没有标记任何一层算子，则该参数不会出现。 ● 如果标记了某层算子，则输出节点为标记层的算子名。由于图7-4中标记了名称为“res2b”，类型为“Eltwise”算子，则该参数才会展示。 **说明** 如果模型转换过程中该算子被融合掉，则该算子不能作为输出节点。

步骤3 单击“Next”，进入“Quantization”界面。

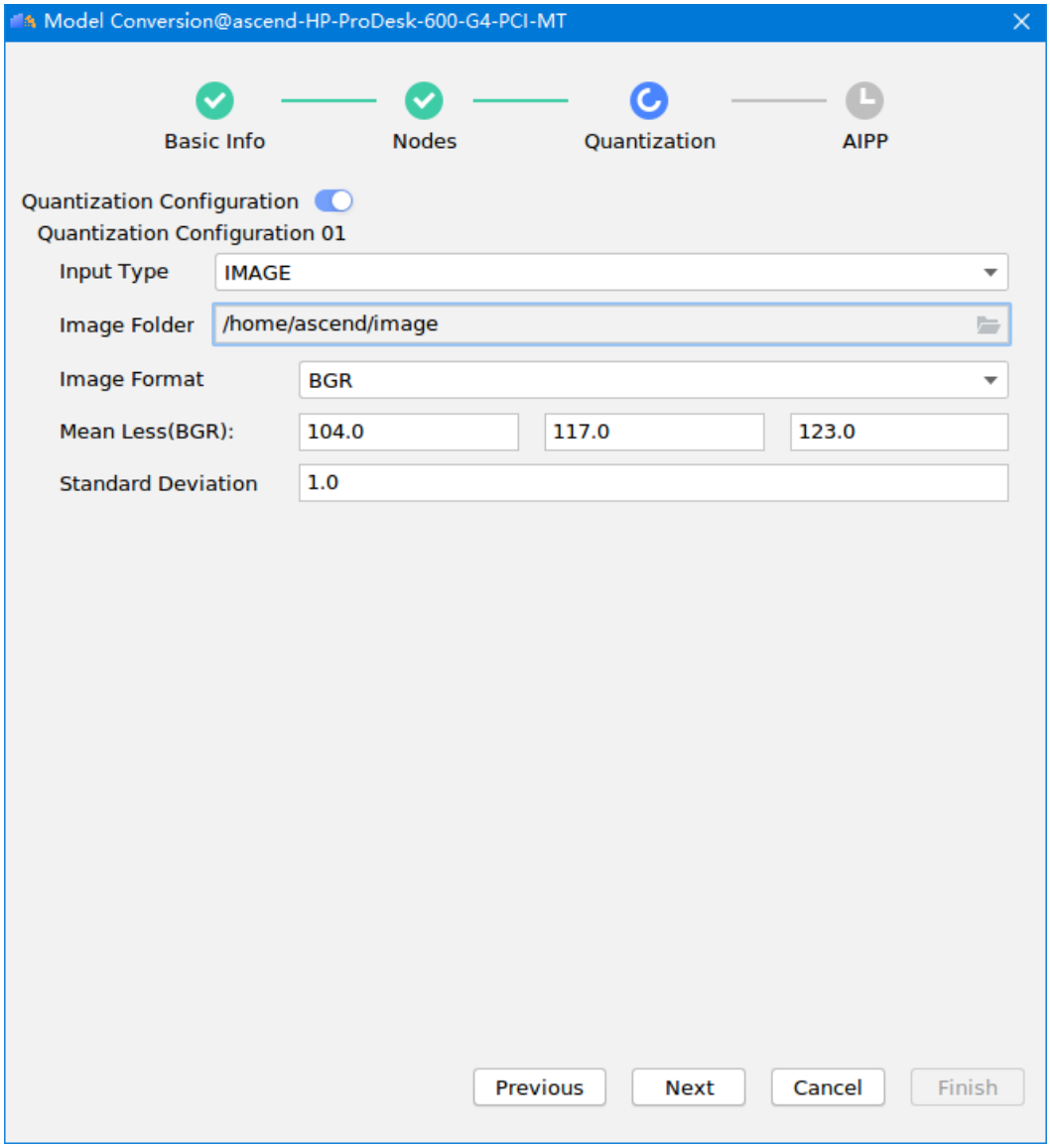
在对模型大小和性能有更高要求的时候可以选择执行量化操作。模型转换过程中量化会将高精度数据向低比特数据进行量化，让最终生成的模型更加轻量化，从而达到节约网络存储空间、降低传输时延以及提高运算执行效率的目的。

说明

在量化过程中，由于模型存储大小受参数影响很大，因此模型转换重点支持卷积算子（Convolution）、全连接算子（FullConnection）以及深度可分离卷积（ConvolutionDepthwise）等带有参数算子的量化。

单击“Quantization Configuration”右侧的开关，打开量化配置功能，如图7-8所示。如果不涉及量化，则不用开启量化开关， 直接进入步骤4。

图 7-8 量化配置



参数说明如表7-3所示。

表 7-3 量化配置参数

参数	说明
Input Type	量化数据（校准集：用于训练量化参数、保证精度）类型。当前支持图片（IMAGE）和二进制文件（BINARY）两种格式。 ● IMAGE：图片格式。 支持的图片格式包括：".bmp", ".dib", ".jpeg", ".jpg", ".jpe", ".jp2", ".png", ".webp",".pbm", ".pgm", ".ppm", ".sr", ".ras", ".tiff", ".tif",".BMP", ".DIB", ".JPEG", ".JPG", ".JPE", ".JP2", ".PNG", ".WEBP",".PBM", ".PGM", ".PPM", ".SR", ".RAS", ".TIFF", ".TIF"。 ● BINARY：二进制文件（Binary格式）格式，请参见表7-4，对于非四维（num、channels、height、width）的数据，需要补齐到四维，补齐的维度值为1。
Image Folder	量化数据所在目录，即校准集图片所在路径。当Input Type选择IMAGE时出现。 选择图片前请先将图片上传到服务器Mind Studio安装用户所在目录下，建议使用与应用场景相关的图片。 **说明** 进行量化时，建议选择多张图片进行量化，如果数量太少，可能会影响精度；最多建议不超过50张，数量太多，有可能因为量化时间太长导致进程超时（限定3小时）。
Image File	量化数据所在文件，即校准集二进制文件所在路径。当Input Type选择BINARY时出现。 选择文件前请先将文件上传到服务器Mind Studio安装用户所在目录下，建议使用与应用场景相关的二进制文件。
Image Format	模型训练时图片三通道排序格式。当Input Type选择IMAGE时需要配置。 支持BRG、RGB两种格式，默认为BRG。
Mean Less(BGR)	图片预处理单个通道的均值，默认开启。当Input Type选择IMAGE时需要配置。 三个通道的值默认为104.0、117.0、123.0。
Standard Deviation	图像预处理的标准方差。当Input Type选择IMAGE时需要配置。默认为1.0。

表 7-4 二进制文件格式

文件头/数据	地址偏移	type	value	description
文件头（共20字节）	0000	32bit int	510	magic number 当magic number = 510，用来校验文件的合法性。

文件头/数据	地址偏移	type	value	description
	0004	32bit int	*50*	input num 请根据实际输入数据的N进行修改，例如图7-6中的1。
	0008	32bit int	*3*	input channels 请根据实际输入数据的C进行修改，例如图7-6中的3。
	0012	32bit int	*28*	input height 请根据实际输入数据的H进行修改，例如图7-6中的224。
	0016	32bit int	*28*	input width 请根据实际输入数据的W进行修改，例如图7-6中的224。
数据	…	float	126	数据数量等于N*C*H*W。

步骤4 单击“Next”，进入“AIPP”预处理界面。

AIPP是昇腾AI处理器提供的硬件图像预处理模块，包括色域转换，图像归一化（减均值/乘系数）和抠图（指定抠图起始点，抠出神经网络需要大小的图片）等功能。DVPP模块输出的图片多为对齐后的YUV420SP类型，不支持输出RGB图片。因此，业务流需要使用AIPP模块转换对齐后YUV420SP类型图片的格式，并抠出模型需要的输入图片。

单击“Image Pre-processing”右侧的开关，打开AIPP预处理功能，如图7-9、图7-10所示。如果不涉及预处理，则不用开启预处理开关， 直接进入步骤5。

图 7-9 AIPP 预处理界面（static）

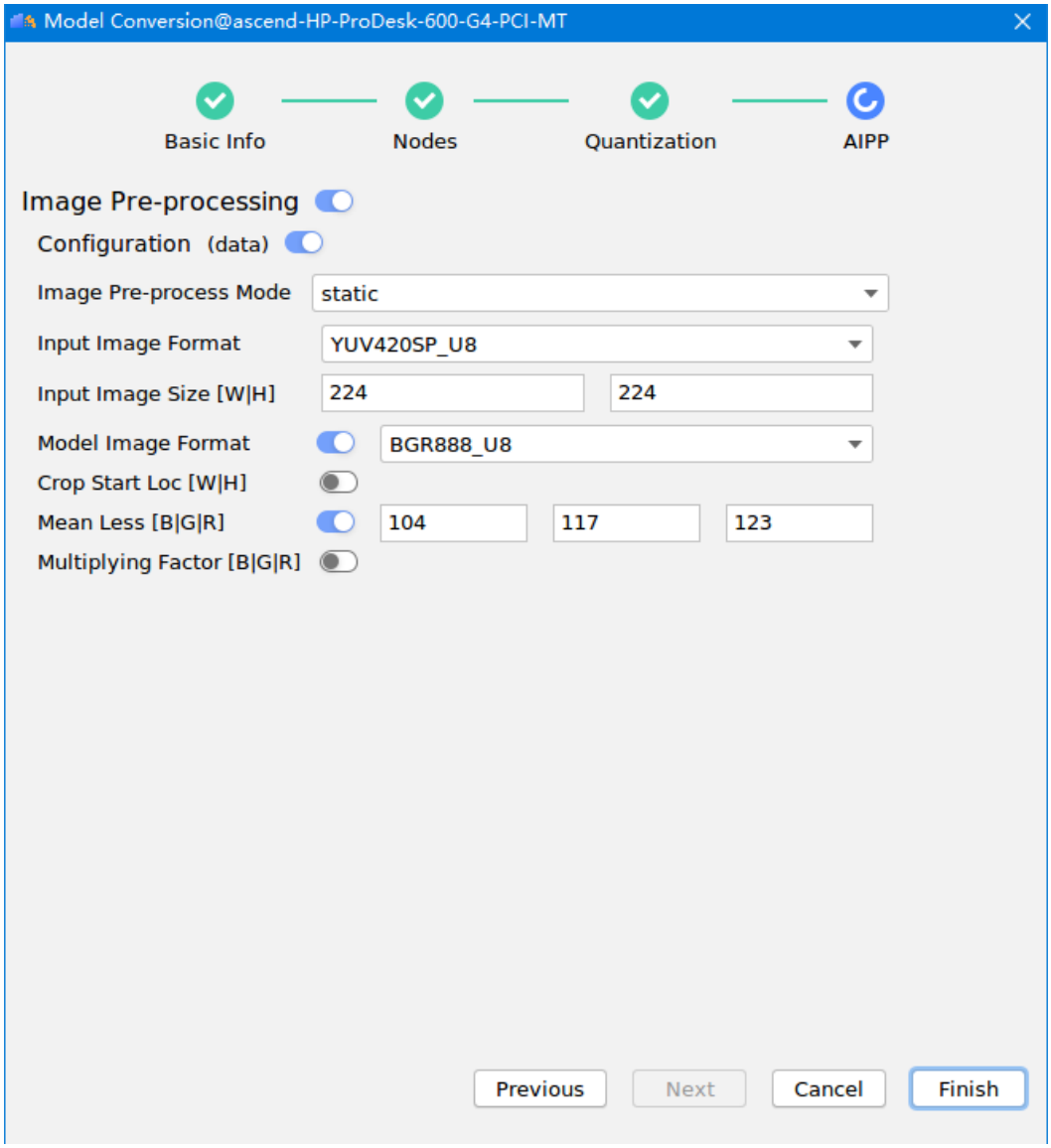
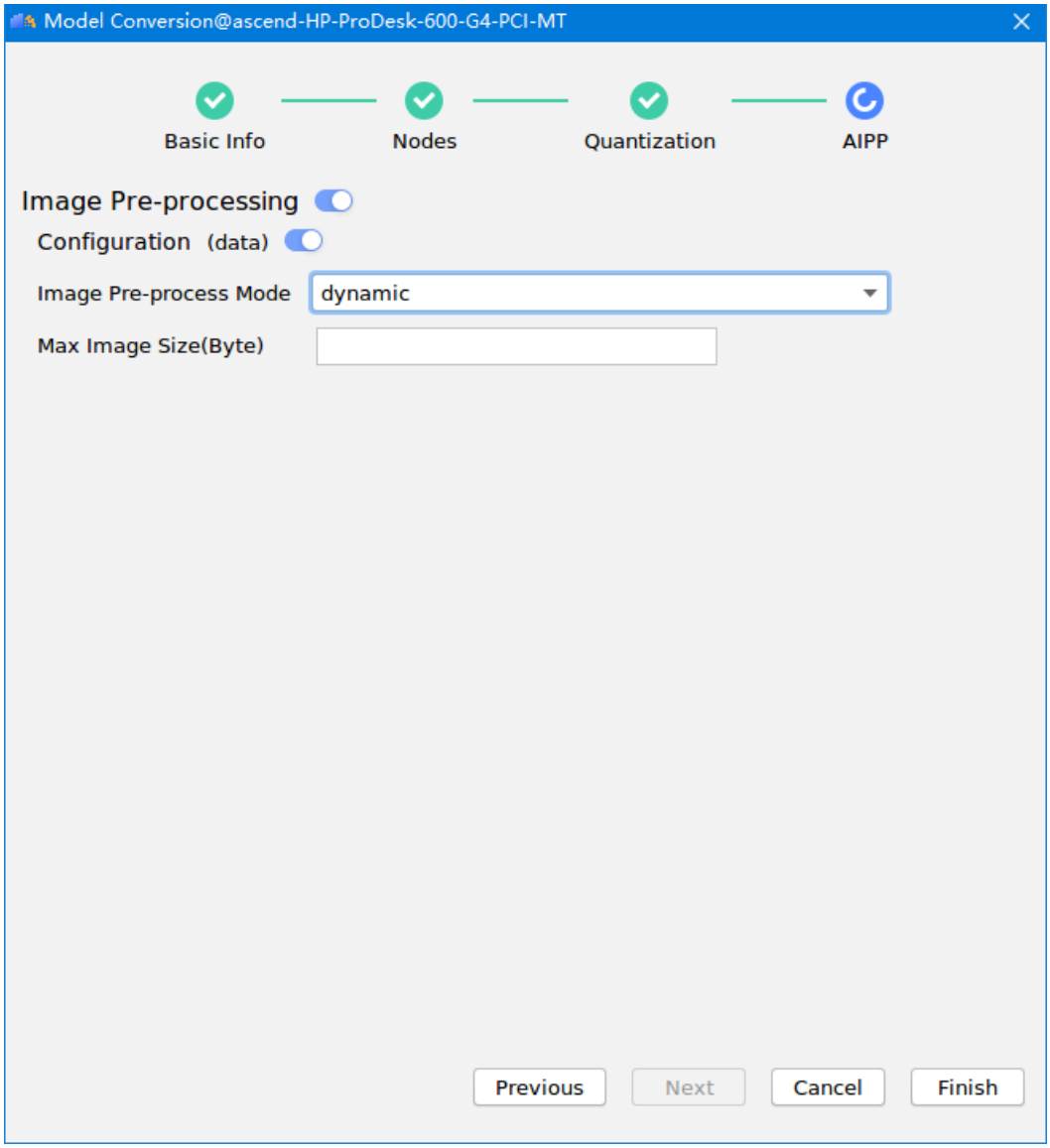


图 7-10 AIPP 预处理界面（dynamic）



参数说明如表7-5所示。

表 7-5 AIPP 预处理配置参数

参数	说明
Configuration (data)	节点配置开关，如果有多个节点，可以控制每个节点的开关情况。

参数	说明
Image Pre-process Mode	AIPP图片预处理模式，默认为static。本示例采用static。 ● static：静态AIPP。在该模式下，模型生成后，AIPP参数值被保存在生成的.om模型中，模型推理过程采用固定的AIPP预处理（无法修改）。 ● dynamic：动态AIPP。在该模式下，模型生成后，每次模型推理前，根据需求在推理Engine的代码中设置动态AIPP参数值，然后在模型推理时可使用不同的AIPP参数。动态AIPP在根据业务要求改变预处理参数的场合下使用（如不同摄像头采用不同的归一化参数，输入图片格式需要兼容YUV420和RGB等）。 **须知** – 当“Image Pre-process Mode”取值static时，系统显示参数包括Input Image Format、Input Image Size [W|H]、Model Image Format、Crop Start Loc [W|H]、Mean Less [B][G][R]和Multiplying Factor [B][G][R]；当“Image Pre-process Mode”取值dynamic时，系统只显示参数Max Image Size(Byte)。 – 当“Image Pre-process Mode”取值dynamic时，如果在模型转换后需要修改图片的输入/输出格式，请修改应用工程推理引擎节点**MindInferenceEngine.cpp**的如下代码： hiai::AippInputFormat inputImageFormat = hiai::*YUV420SP_U8*; hiai::AippModelFormat modelImageFormat = hiai::MODEL_*BGR888_U8*; 如果需要修改图片的输入/输出之外的其他信息，请参见《 Matrix API参考 》中的“模型管家接口>动态AIPP配置接口”。 – 在使用dynamic动态AIPP参数时， 请确保所创建的工程为Ascend APP工程。
Input Image Format	输入图片格式。 选项：YUV420SP_U8、XRGB8888_U8、RGB888_U8、YUV400_U8。 **说明** 模型转换完毕后，在对应的*.om模型文件中，上述四个参数分别以1、2、3、4参数呈现。
Input Image Size [W|H]	输入图片大小，由模型文件Input层的宽和高分别16和2对齐得到。 **说明** 此处需要用户手动进行对齐。
Model Image Format	模型处理图片格式。 选项：YUV444SP_U8，YVU444SP_U8，RGB888_U8，BGR888_U8，GRAY。 该参数同时也是色域转换开关，默认开启。 当输入图片格式与模型处理文件格式不一致时需要开启。
Crop Start Loc [W|H]	抠图开始位置，默认关闭，开启开关后可以设置起始位置。

参数	说明
Mean Less [B][G][R]	减均值，默认开启。 三个通道的值默认为104、117、123。
Multiplying Factor [B][G][R]	乘系数（均方差或(max-min)的倒数），默认关闭。
Max Image Size(Byte)	当“Image Pre-process Mode”取值“dynamic”时出现。 处理图片时一次申请的内存空间大小，取值范围为(0, 4294967295]。 如果用户需要修改该参数，则取值等于N * H * W * 4。（N,H,W来自“Input Node”参数。W和H默认值由模型文件Input层的宽和高分别16和2对齐得到。4表示系数，系统校验时该系数根据输入图片格式有不同的取值，分别是YUV400_U8：1、YUV420SP_U8：1.5、RGB888_U8：3、XRGB8888_U8：4，为了保证空间充足，该参数使用最大值4）。

须知

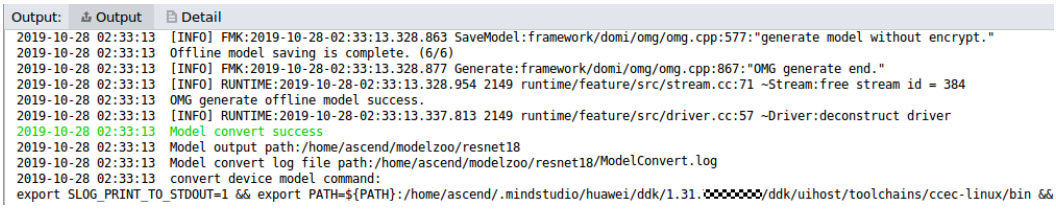
模型转换是否开启AIPP功能，执行推理业务时，对输入图片数据的要求：

- 模型转换时开启AIPP：Input Image Format选择XRGB8888_U8或RGB888_U8，使用该种配置转换后的模型，在进行推理业务时，输入图片数据要求为NHWC排布。
- 模型转换时没有开启AIPP，模型转换完毕，在进行推理业务时，输入图片数据要求为NCHW排布，因此需要用户自行把NHWC排布的原始图片数据转换为NCHW排布。

步骤5 单击“Finish”，开始进行模型转换。

在Mind Studio界面下方，Output窗口会显示模型转换的结果、模型输出路径以及模型转换日志，如图7-11所示。

图 7-11 模型转换结果



步骤6 模型转换完毕，在服务器后台路径“$HOME/modelzoo/resnet18/device”下会生成.om的模型文件，如果开启量化和AIPP预处理功能，在.om同级目录下，还会生成量化配置文件（cal.cfg）和AIPP预处理的配置信息文件（insert_op.cfg），如图7-12所示。

模型转换的日志文件（ModelConvert.log）所在路径为：“$HOME/modelzoo/resnet18”

图 7-12 后台服务器生成的文件

```
ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~/modelzoo/resnet18/device$ ll
total 11700
drwxr-x--- 2 ascend ascend     4096 Oct 28 02:33 ./
drwxr-x--- 3 ascend ascend     4096 Oct 28 02:33 ../
-rw-r----- 1 ascend ascend      236 Oct 28 02:33 cal.cfg
-rw-r----- 1 ascend ascend      188 Oct 28 02:33 insert_op.cfg
-rw------- 1 ascend ascend 11964325 Oct 28 02:33 resnet18.om
```

----结束

说明

若图7-7配置的“Batch Mode”取值为“Dynamic”或图7-10中配置的“Image Pre-process Mode”取值为“dynamic”，模型转换完毕后， 如果要在应用工程中使用该.om模型文件，需要进行应用工程适配，详细适配方法请参见《Ascend 310 应用开发指导（Mind Studio方式）》中的*离线模型推理>AIPP功能或Batch功能*。

异常处理

如果模型转换失败，可以参见该章节进行处理。

1. 如果转换失败，弹出Model Conversion Failed模型转换失败报告页面：
   a. 如果失败类型是需要重新指定算子，弹出图7-13所示界面。

图 7-13 模型转换失败_重新指定算子

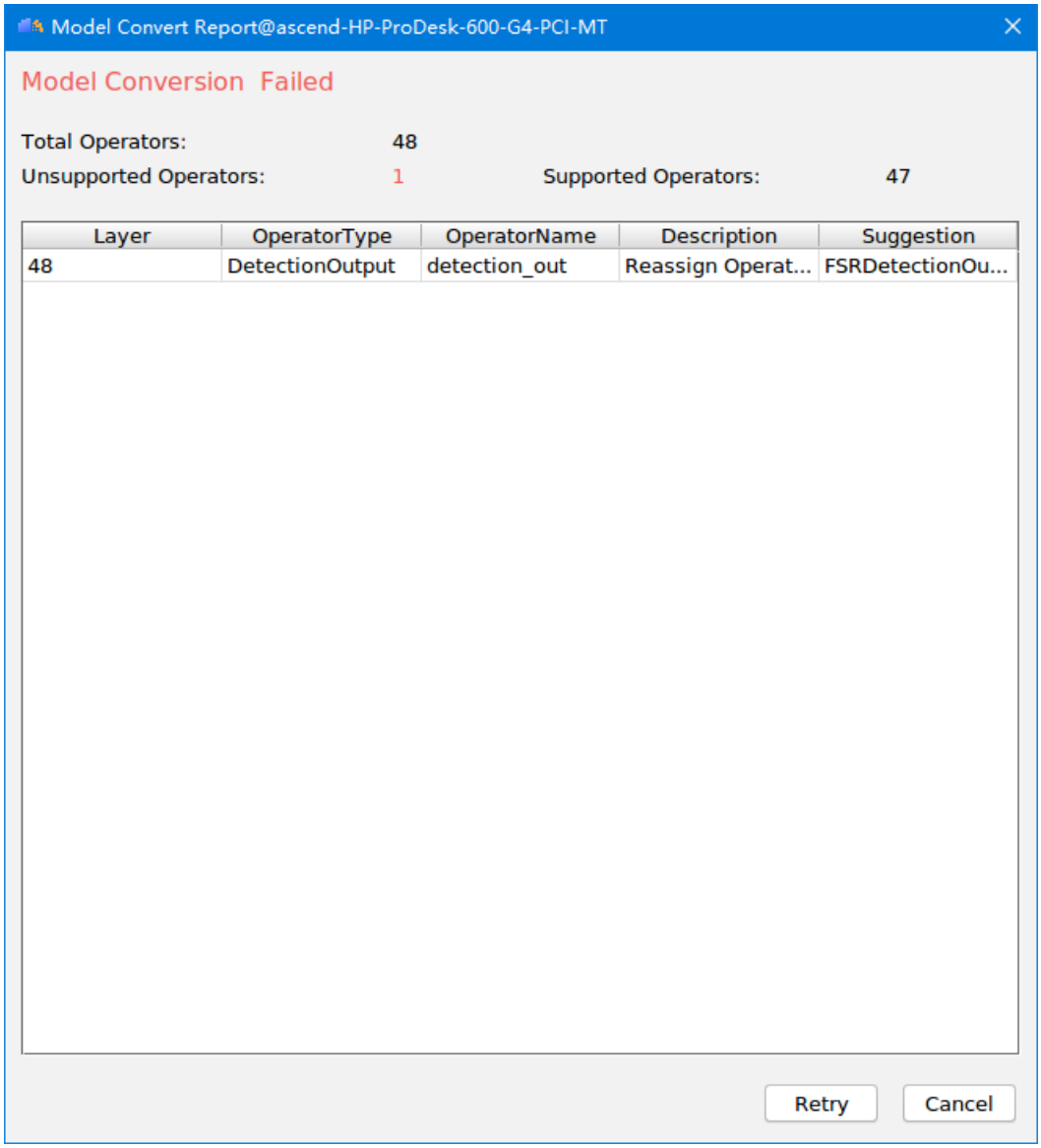
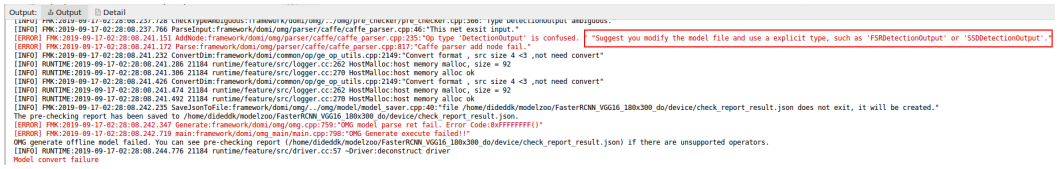


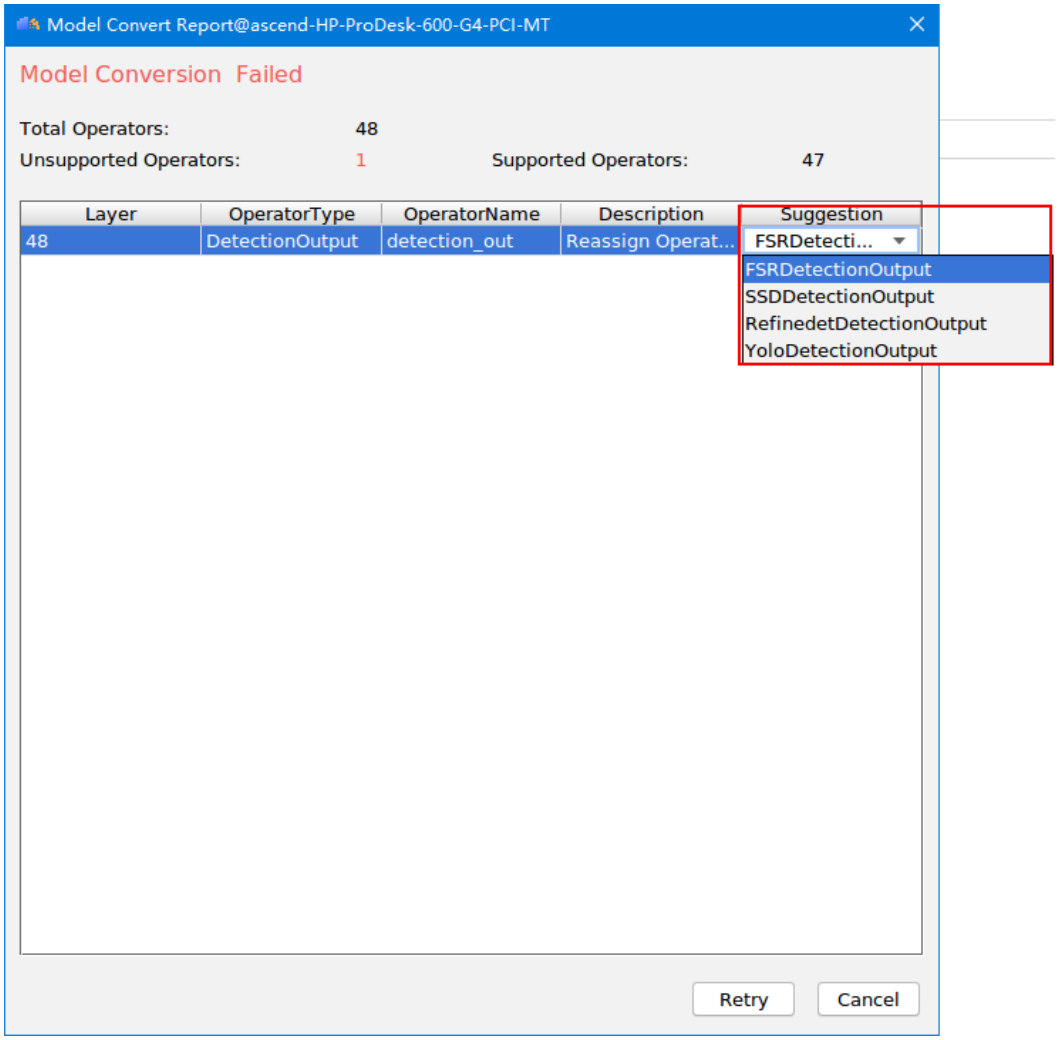
图7-13中会给出总的算子数，不支持的算子数，不支持算子的类型、不支持算子的名称、不支持的原因，以及处理建议。同时Output窗口会给出详细日志信息，如图7-14所示。

图 7-14 日志信息



从下拉框选择一个算子后，单击“Retry”进行重试，如图7-15所示。

图 7-15 重新指定算子



b. 如果失败类型包含不支持的算子，则弹出图7-16所示界面。

图 7-16 模型转换失败_包含不支持算子

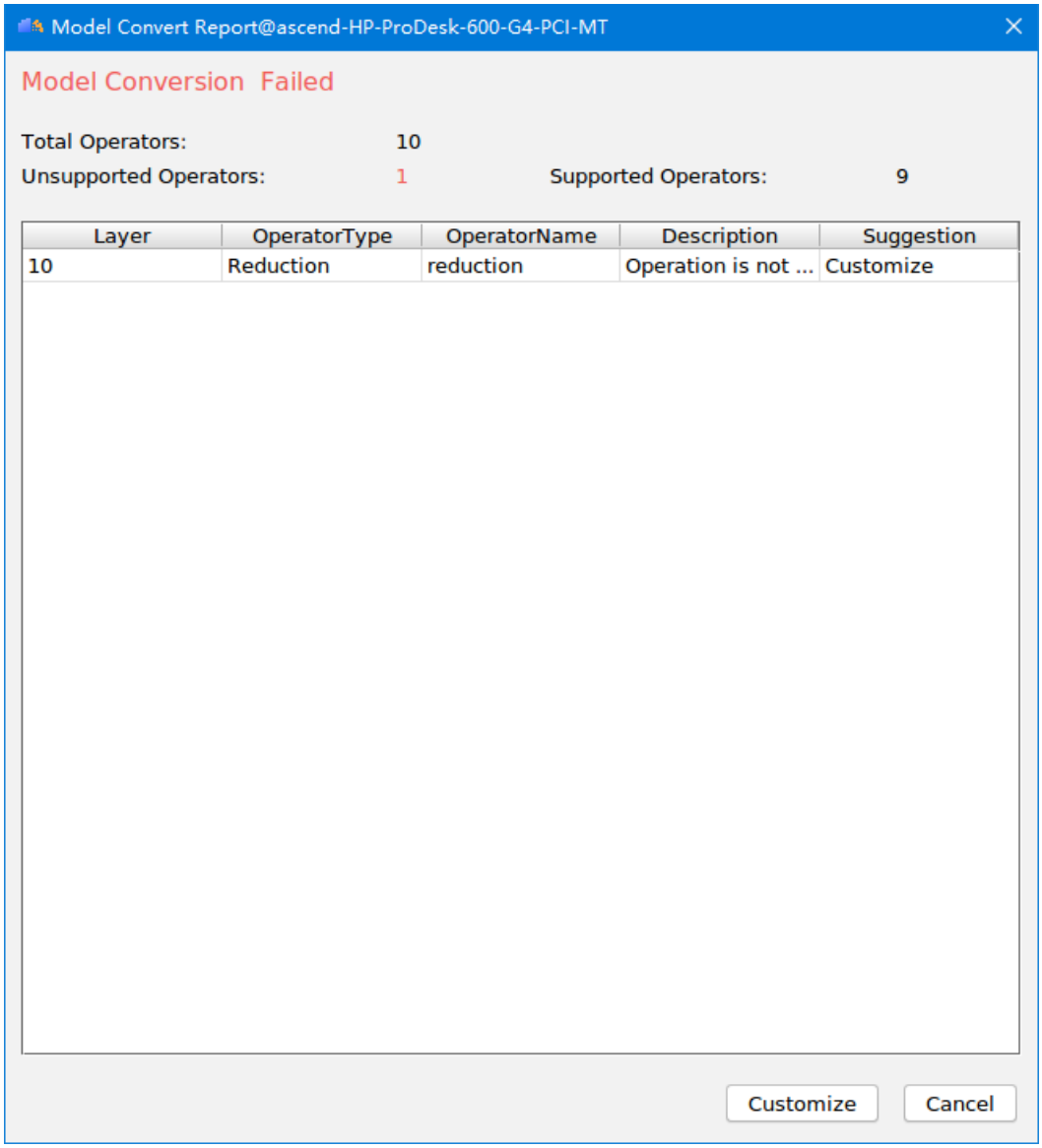
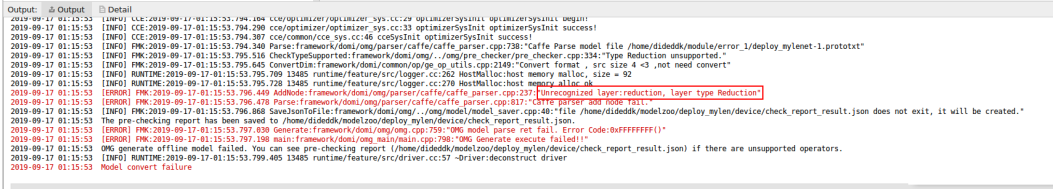


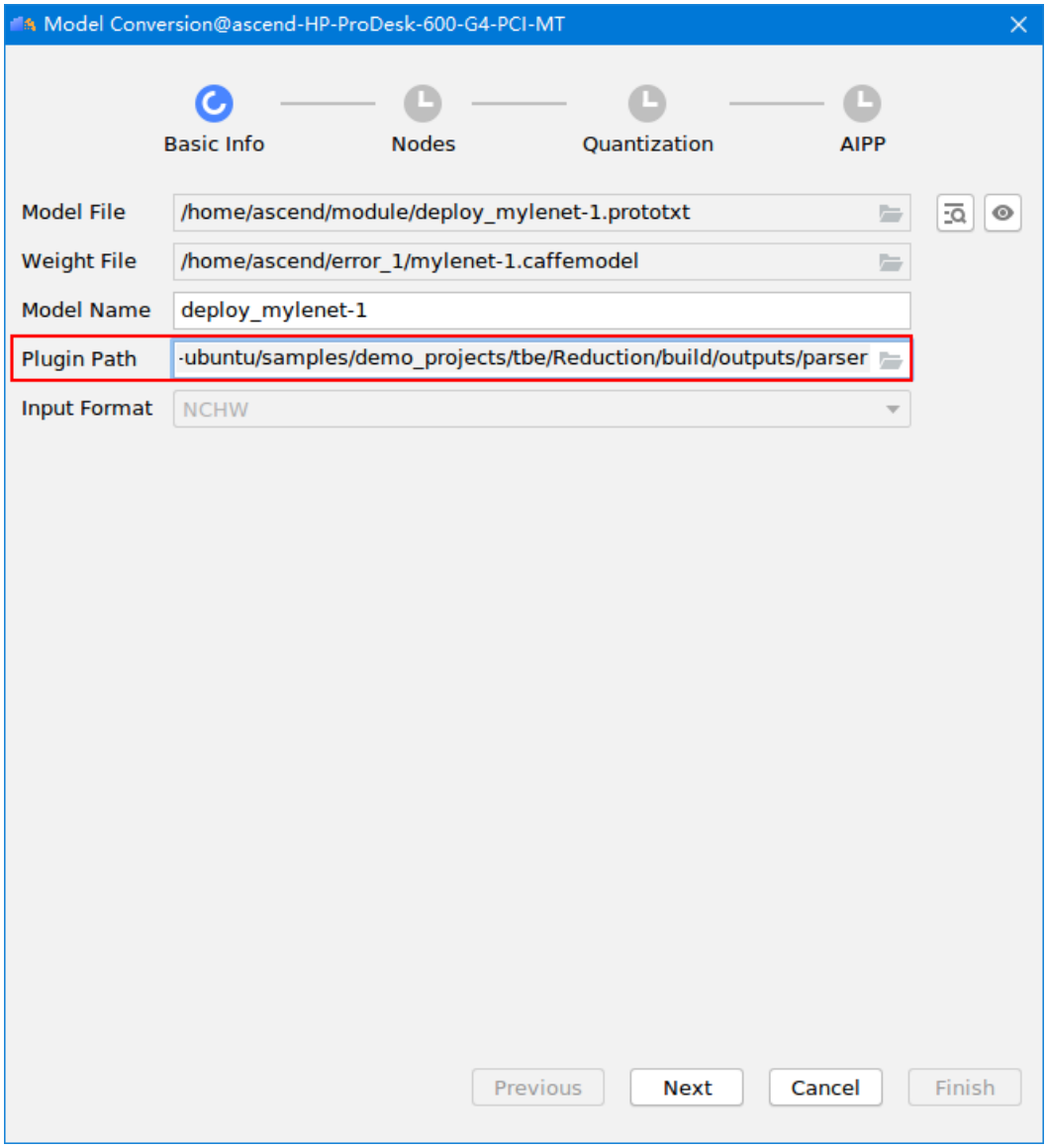
图7-16中会给出模型转换总的算子数、不支持的算子数、不支持算子的类型、不支持算子的名称、不支持的原因，以及处理建议，同时Output窗口会给出详细日志信息，如图7-17所示。

图 7-17 日志信息



单击“Customize”对该类型的算子创建工程，进行自定义算子插件开发，详细请参见《TBE自定义算子开发指导（Mind Studio方式）》。自定义算子插件开发完成后，重新进行离线模型导入，导入时需要选择开发好的自定义算子插件，如图7-18所示。

图 7-18 选择自定义开发算子插件



然后参见**步骤2**~**步骤6**重新进行模型转换。

2. 如果模型转换时设置了“Batch Mode”为“Dynamic”，模型转换失败，提示类似如**图7-19**所示错误信息，其中*1024*（具体报错信息以用户实际设置的Batch值为准）为用户设置的Dynamic Batch中的值。

图 7-19 模型转换失败日志

```
2019-09-24 08:31:54  [INFO] FMK:2019-09-24-08:31:54.174.967 GetShapeClassifyNode:framework/domi/omg/../omg/model/model_builder.cpp:4531:"Create shape classify node for dynamic multi batch, node name:shape
2019-09-24 08:31:54  [ERROR] FMK:2019-09-24-08:31:54.180.089 AddSwitchAndSliceNodes:framework/domi/omg/../omg/model/model_builder.cpp:3952:"batch_node is null, node name:conv1_b1024__b1024__b1024_."
2019-09-24 08:31:54  [ERROR] FMK:2019-09-24-08:31:54.180.103 GenBatchGraph:framework/domi/omg/../omg/model/model_builder.cpp:3782:"fail to add switch and slice nodes."
2019-09-24 08:31:54  [ERROR] FMK:2019-09-24-08:31:54.180.112 Build:framework/domi/omg/../omg/model/model_builder.cpp:3716:"GenBatchGraph failed!"
2019-09-24 08:31:54  [ERROR] FMK:2019-09-24-08:31:54.180.116 Generate:framework/domi/omg/omg.cpp:842:"OMG builder Build() return fail."
2019-09-24 08:31:54  [ERROR] FMK:2019-09-24-08:31:54.236.736 main:framework/domi/omg_main/main.cpp:798:"OMG Generate execute failed!!"
2019-09-24 08:31:54  OMG generate offline model failed. You can see pre-checking report (/home/ascend/modelzoo/resnet18_1/device/check_report_result.json) if there are unsupported operators.
2019-09-24 08:31:54  [INFO] RUNTIME:2019-09-24-08:31:54.238.702 4320 runtime/feature/src/driver.cc:57 ~Driver:deconstruct driver
2019-09-24 08:31:54  Model convert failure
2019-09-24 08:31:54  No error report was generated
```

或模型转换成功，但执行推理业务时提示**图7-20**所示异常信息。

图 7-20 推理阶段失败日志

```
3,matrixdaemon):2019-09-24-09:44:18.994.666 [AICPUFW] [aicpufw_sup_thread_proc_193] sup_cmd_handler return error type 0, ret = 4294967295.
(1500,graph_318127):2019-09-24-09:44:21.728.786 1506 runtime/feature/src/npu_driver.cc:632 DevMemAllocOffline:drvMemAllocManaged , error = 6
(1500,graph_318127):2019-09-24-09:44:21.728.886 1506 runtime/feature/src/logger.cc:240 DevMalloc:dev memory alloc error, errno = 14
3,graph_318127):2019-09-24-09:44:21.728.934 MallocVarMem:framework/domi/ome/../omm/session.cpp:281:"Call rt api failed, ret: 0xE"
3,graph_318127):2019-09-24-09:44:21.728.971 InitModelMemNew:framework/domi/ome/model_manager/davinci_model.cpp:4268:"malloc var memory"
3,graph_318127):2019-09-24-09:44:21.772.285 LoadModel:framework/domi/ome/ome.cpp:546:"LoadModel: Load failed. ret = 50331651"
3,graph_318127):2019-09-24-09:44:21.728.993 Load:framework/domi/ome/model_manager/model_manager.cpp:1360:"DavinciInit failed."
3,graph_318127):2019-09-24-09:44:21.785.327 LoadModelFromFile:hiaiengine/node/hcs/npu_controller/ai_npu_executor.cpp:24:"Load model from file path failed!"
3,graph_318127):2019-09-24-09:44:21.772.345 LoadModelFromFile:framework/domi/ome/ome.cpp:664:"LoadModel: Load failed. ret = 50331651"
3,graph_318127):2019-09-24-09:44:21.785.394 LoadModels:hiaiengine/node/ai_model_manager_impl.cpp:292:"Load model \"VGG16_1_4_5_8\" failed!"
3,graph_318127):2019-09-24-09:44:21.785.373 LoadModelFromFile:hiaiengine/node/ai_model_manager_impl.cpp:360:"Load model from \"/home/HwHiAiUser/matrix/3181274022/VGG16_1_4_5_8.om\" failed!"
3,graph_318127):2019-09-24-09:44:21.785.420 Init:hiaiengine/node/ai_model_manager_impl.cpp:171:"Load models failed!"
3,graph_318127):2019-09-24-09:44:21.785.475 [MindInferenceEngine] Failed to call Init for AIModelManager.,[/home/gaouser/AscendProjects/app/20190924/MyApp_1/src/MindInferenceEngine/MindInferenceEngine.cpp:80:Init], Msg:
INE(1500,graph_318127):2019-09-24-09:44:21.785.527 [JobID=2019092409441799 6][GraphID=3181274022],engineID=226,[hiaiengine/connector/device/../common/src/graph_impl.cpp:3550:InitEngines], Msg: engine init failed
3,graph_318127):2019-09-24-09:44:21.994.753 [AICPUFW] [aicpufw_sup_thread_proc_193] sup_cmd_handler return error type 0, ret = 4294967295.
INE(1500,graph_318127):2019-09-24-09:44:22.099.887 DeviceCreateGraph result ret,[hiaiengine/connector/device/../common/src/graph_impl.cpp:1649:DeviceCreateGraph], Msg: engine init failed
3,graph_318127):2019-09-24-09:44:24.995.056 [AICPUFW] [aicpufw_sup_thread_proc_193] sup_cmd_handler return error type 0, ret = 4294967295.
INE(1500,graph_318127):2019-09-24-09:44:22.099.934 DeviceAddHostRecvAddr cannot find graph by msg.graph_id ,[hiaiengine/connector/device/../common/src/graph_impl.cpp:5802:GetMemoryPoolByGraphID], Msg: graph not exist
INE(1500,graph_318127):2019-09-24-09:44:25.199.065 Chlid process recv msg error due to the hdc session was disconnected,[hiaiengine/connector/device/src/hdc/hdc_server.cpp:334:RecvData], Msg: recv msg error
INE(1500,graph_318127):2019-09-24-09:44:25.199.187 Chlid process recv msg error due to the hdc session was disconnected,[hiaiengine/connector/device/src/daemon_process.cpp:130:RecvData], Msg: recv msg error
```

出现图7-19、图7-20问题是由于用户设置的Dynamic Batch值过大导致，建议用户调低Batch值（每Batch最大数值建议不超过32），重新进行模型转换。

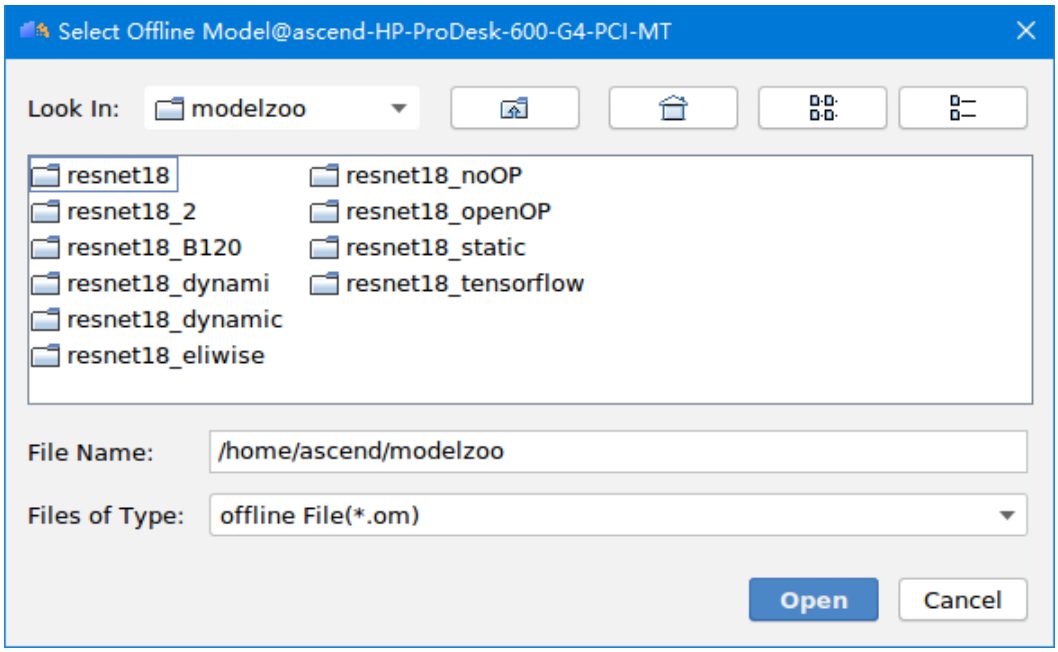
7.3 模型可视化

对于已经转换成功的.om模型文件，可以在Mind Studio界面呈现其网络拓扑结构，并可以查看模型所使用的算子。

查看模型网络拓扑结构

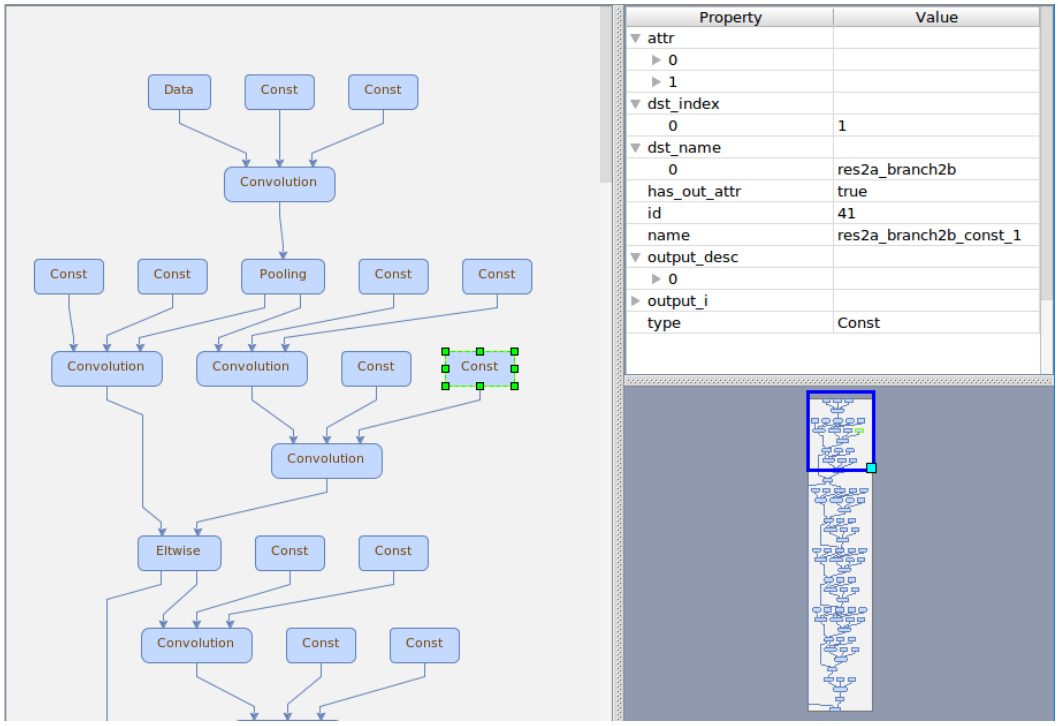
步骤1 依次单击菜单栏“Tools > View Model”，在图7-21所示界面中选择要可视化的模型，例如此处的resnet18模型文件。

图 7-21 选择要打开的模型



步骤2 依次选择“resnet18 > device”，选择7.2 模型转换中已经转换成功的resnet18.om模型文件，单击“Open”，弹出图7-22所示界面。

图 7-22 模型可视化界面



- 图7-22展示了模型文件中所有的算子，单击某层算子，该层算子会出现绿色选中框，右上角区域会展示该算子的详细信息，包括算子名称、算子输入、输出等信息，单击相关参数后面的▶可以查看参数的详细信息。右下角会展示该网络模型的整体结构，包括左侧展示框中的算子在整体网络结构中的位置，即图中的蓝色框选择位置。左侧展示区域下拉，右侧整体区域中的蓝区选择框随之往下移动。左侧展示框放大和缩小，右侧蓝色框随之缩小和放大。
- AIPP配置信息查看：
  - 图7-22网络拓扑结构中的第一个Convolution算子，记录了模型转换过程中所配置的AIPP信息，您可以单击该算子，查看相关信息（key值为aipp，对应的value即为AIPP配置信息）。
  - 在图7-22网络拓扑结构界面左侧空白地方单击，在右上方会显示“Aipp Properties”参数，单击该参数后面的▶也可以查看AIPP配置信息，如图7-23所示为静态AIPP的配置参数展示样例。

图 7-23 AIPP 配置信息

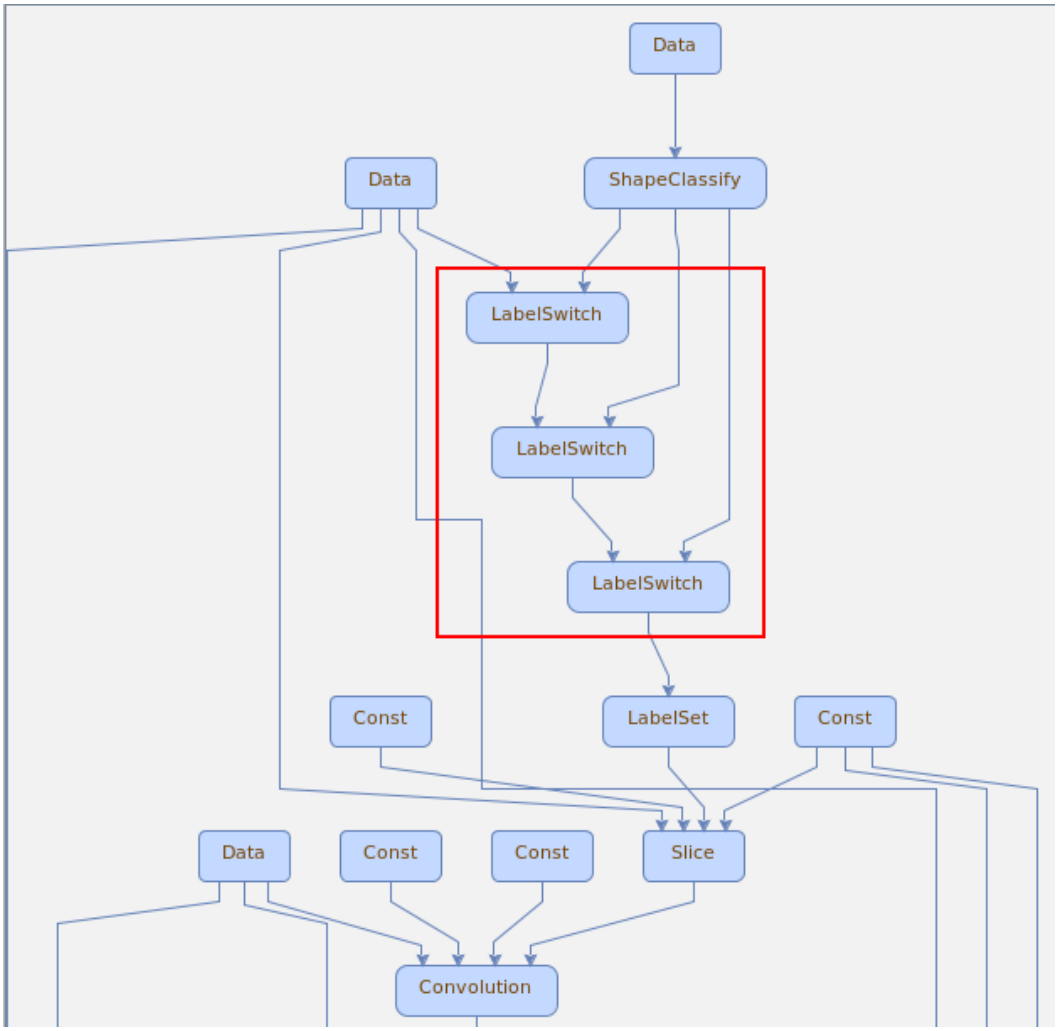
Property	Value
Aipp Properties	
aipp_mode	1
ax_swap_switch	false
bottom_padding_size	0
cpadding_value	0.0
crop	true
crop_size_h	224
crop_size_w	224
csc_switch	false
input_format	1
left_padding_size	0
load_start_pos_h	0
load_start_pos_w	0
mean_chn_0	0
mean_chn_1	0
mean_chn_2	0
min_chn_0	104.0
min_chn_1	117.0
min_chn_2	123.0
padding	false
rbuv_swap_switch	false
resize	false
resize_output_h	0
resize_output_w	0
right_padding_size	0
single_line_mode	false
src_image_size_h	224
src_image_size_w	224
top_padding_size	0
var_reci_chn_0	1.0
var_reci_chn_1	1.0
var_reci_chn_2	1.0
Model Name	ResNet-18
Model Version	2

关于AIPP更多参数详细解释，请参见《OMG模型转换工具使用指导》中的“AIPP配置 > 配置文件模板”章节。

- 动态Batch网络模型查看：

  如果模型转换时设置“Batch Mode”为“Dynamic”，如7.2 模型转换中的图7-7所示，则在模型可视化界面可以看到有对应档位数的labelSwitch算子，图7-7中设置了三个档位，则对应有三个labelSwitch算子，如图7-24所示。

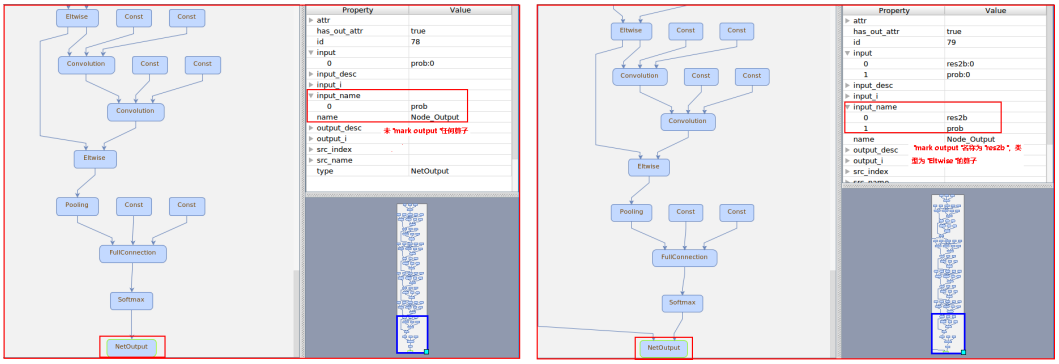
图 7-24 Batch Mode 为 dynamic 时模型可视化界面的 labelSwitch 算子



- 模型输出节点信息查看：

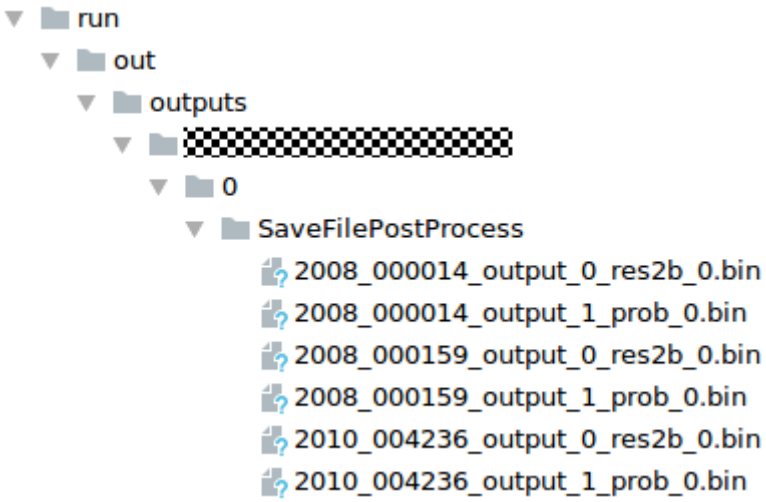
  由于模型转换时mark output了名称为“res2b”，类型为“Eltwise”算子，则在.om模型文件最后一层算子会有两个输入以及两个输出，图7-25给出了mark output算子前后.om模型文件输出节点的变化情况：

图 7-25 mark output 算子前后区别



在执行推理业务时，mark output层的算子也会单独输出推理结果，如图7-26所示。

图 7-26 mark output 算子时的推理结果

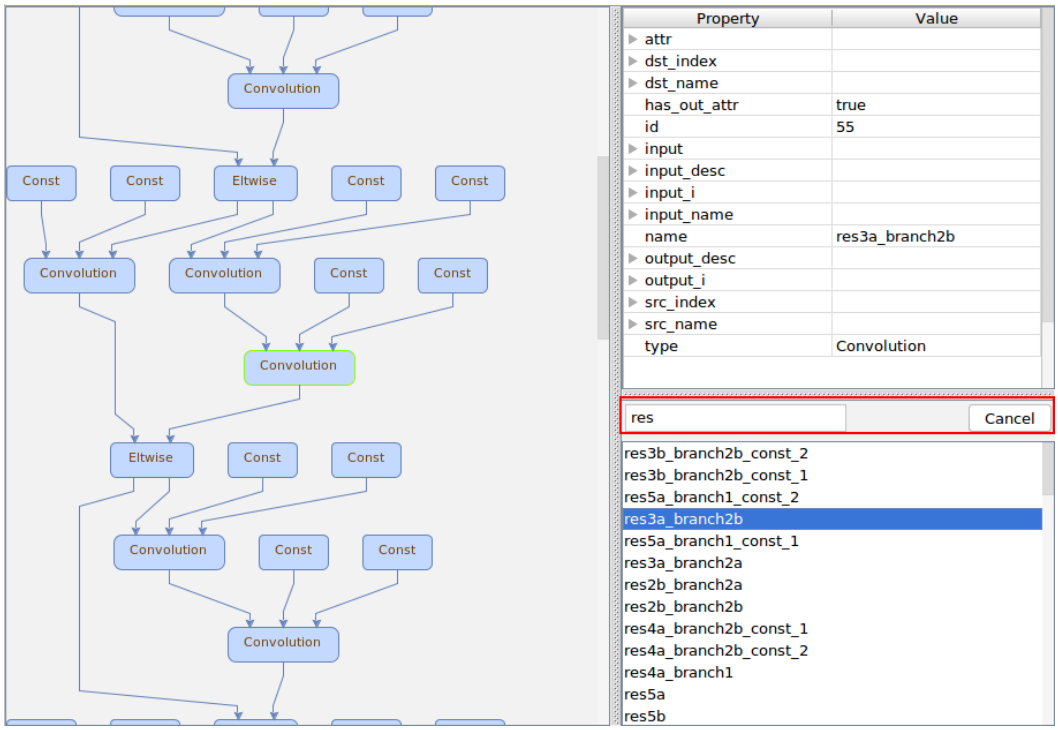


----结束

搜索模型所使用的算子

在图7-22所示界面左侧任意地方单击“Ctrl+F”快捷键，在界面右下方会弹出搜索区域，如图7-27所示。

图 7-27 模型可视化界面的搜索功能



搜索区域中给出了该模型所用到的所有算子，您可以在搜索区域对话框中输入算子名称，下方搜索区域会列出相关的算子。选择其中一个算子，左侧网络拓扑结构中相应算子会显示绿色选中框，右上方会展示该算子的详细信息，包括算子名称、算子输入、输出等信息。单击“Cancel”，取消搜索功能。

8 操作指南

8.1 工程管理
8.2 设备管理
8.3 DDK版本管理

8.1 工程管理

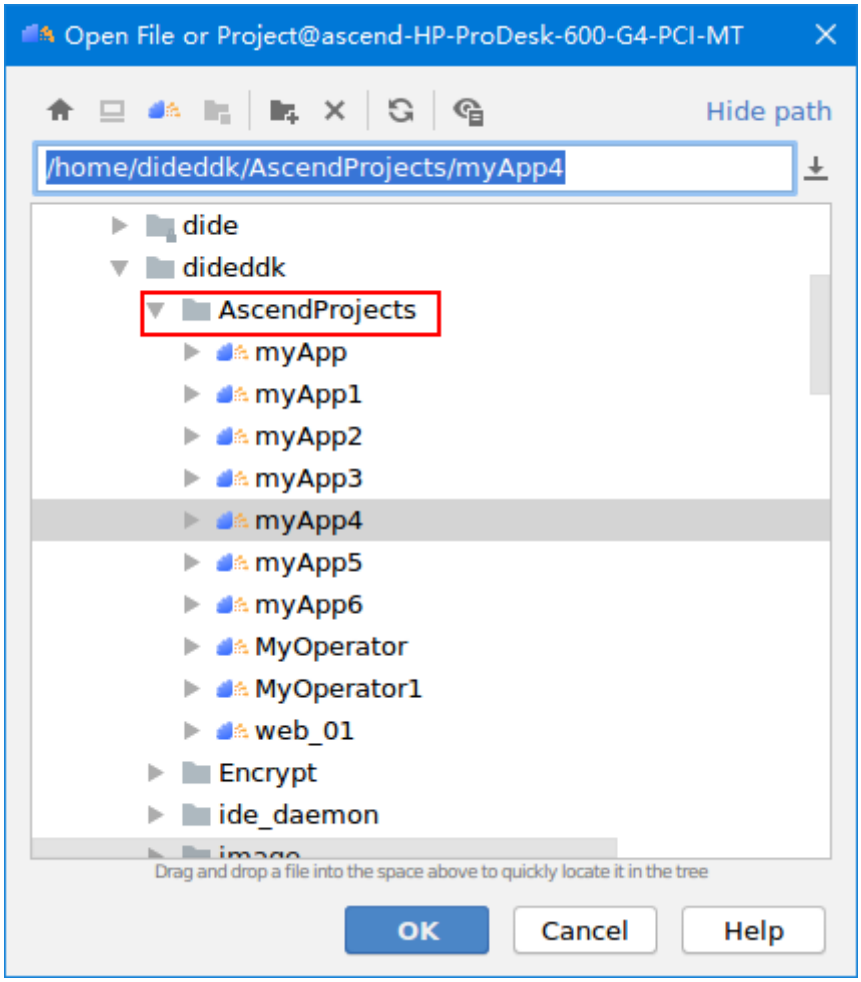
8.1.1 打开/关闭/移除/还原工程

打开工程

如果用户之前创建过工程，则可以通过如下两种方式打开之前创建的工程。

1. 在工程打开界面，依次选择“File > Open”，选择工程所在路径后，弹出**图8-1**所示界面。

图 8-1 打开已有工程



选中相应工程，单击“OK”。

2. 在工程欢迎界面，单击“Open project”，在弹出界面中选择工程所在路径后，进入图8-1所示界面，选择相应工程，单击OK。

说明

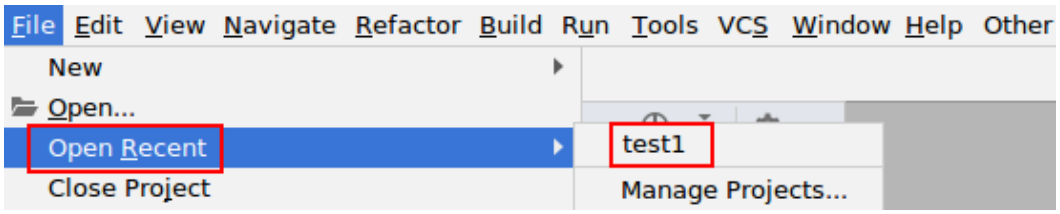
打开工程时，如果用户按住Ctrl键同时选择多个工程，例如图8-1中的myApp工程~myApp6工程，则默认打开选中的第一个工程，即默认打开myApp工程；如果用户按住Ctrl键同时选择多个文件，则选中的文件可以同时打开。

打开最近工程

在已经有Project关闭的情况下，可以通过如下两种方式，打开最近使用的工程，方法如下：

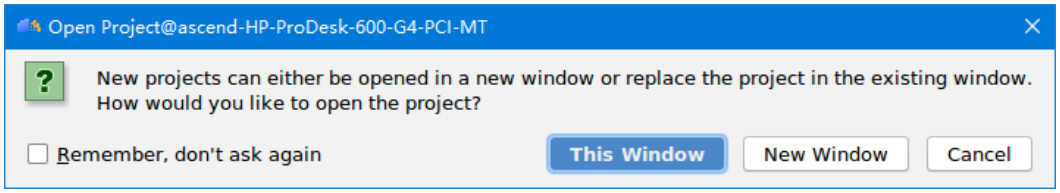
1. 依次单击菜单栏的“File > Open Recent”，在右侧弹出页面中选择想要打开的工程，如图8-2所示中的**test1**工程。

图 8-2 Open Project



单击相应工程后弹出如图8-3所示页面。

图 8-3 工程打开提示



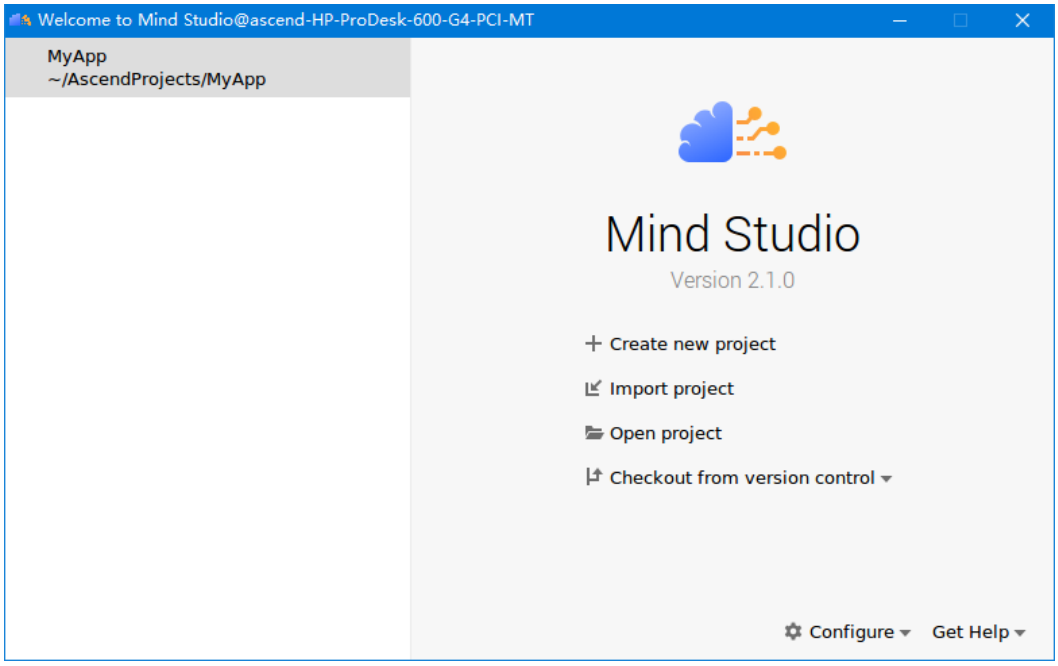
- 选择“This Window”，则直接在当前工作窗口打开选中工程。
- 选择“New Window”，则新建一个工作窗口打开选中的工程。

2. 依次单击菜单栏的“File > Open Recent > Manage Projects”，在弹出页面中选择想要打开的工程，选中后会在新窗口中打开相应工程。

关闭工程

单击File菜单中的“Close Project”，会把当前已经打开的工程关闭，弹出如图8-4所示界面。

图 8-4 Close Project



您可以根据需要选择重新打开已关闭工程、切换到其他工程或者单击右侧区域“Create new project”新建工程。

移除工程

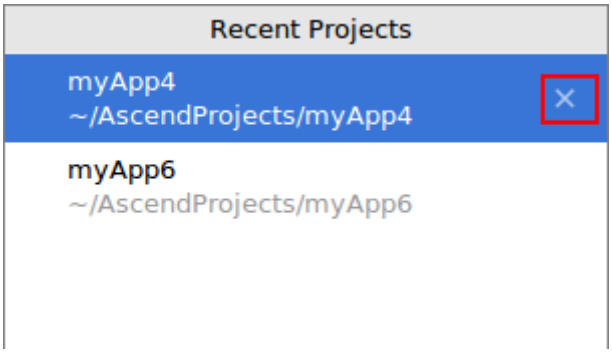
工程关闭后，如果您不想继续使用相关工程，可以选择将其移除。移除工程有两种方法：

说明

移除工程并不是从服务器上删除该工程，只是不在Mind Studio界面展示，工程移除后，用户可以参见还原工程将工程还原。

1. 在图8-4所示界面中，选中想移除的工程，单击右侧的或者右击选择“Remove Selected from Welcome Screen”。
2. 单击File菜单中的“Open Recent > Manage Projects”，在弹出页面中选中想移除的工程，单击右侧的移除。

图 8-5 移除工程

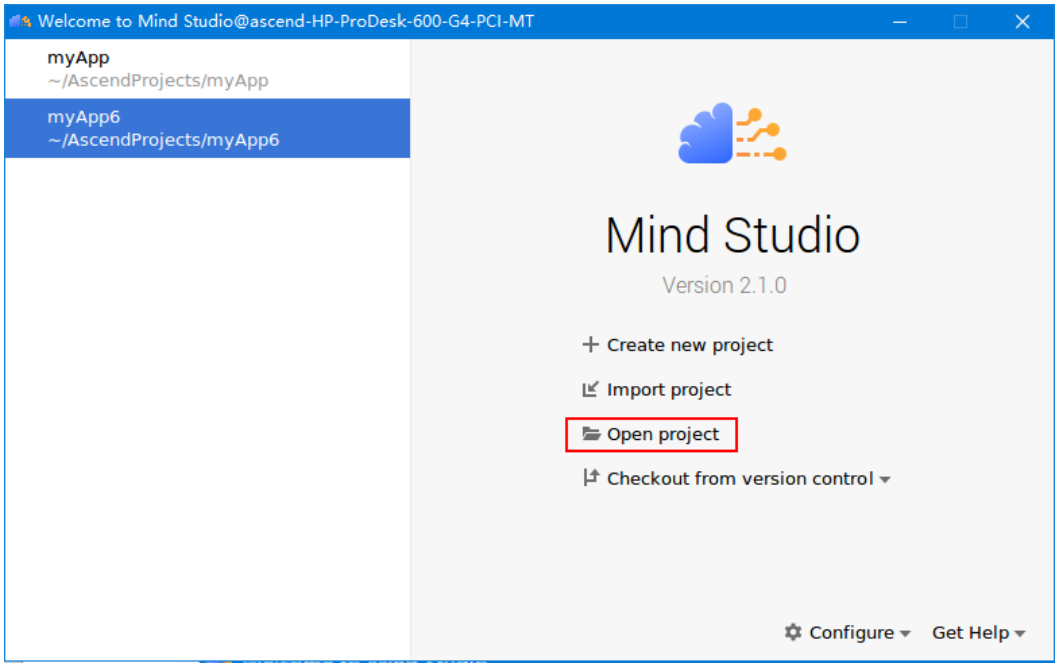


还原工程

工程移除后，如果想还原工程，则方法有如下两种方式：

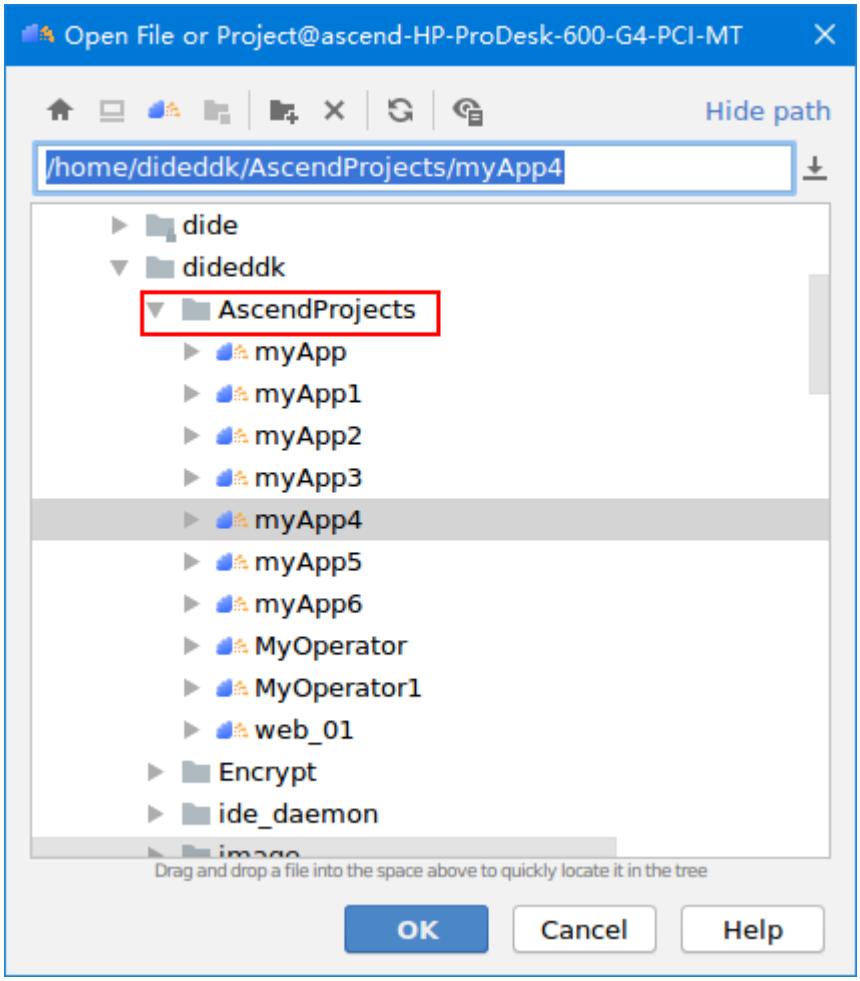
1. 从欢迎页面还原工程
   a. 在欢迎界面，单击右侧的“Open project”，如图8-6所示。

图 8-6 欢迎界面



b. 在弹出页面中，选中工程路径下想要还原的工程，如图8-7所示，单击“OK”。

图 8-7 选择想要打开的工程



c. 选中工程会在新窗口中打开。

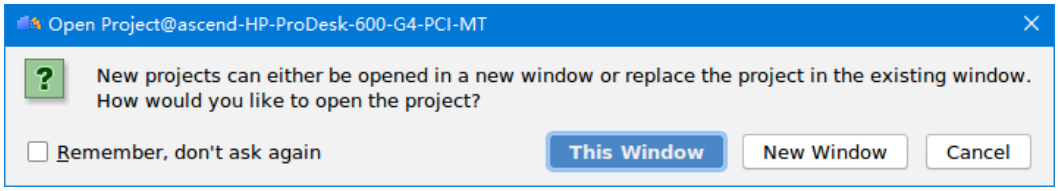
2. 从工程打开页面还原工程

如果已经有其他工程打开，则可以从其他工程打开页面，将原有移除的工程打开，方法如下。

a. 依次单击菜单栏“File > Open”，选择工程路径后，进入图8-7所示界面。

b. 选中想要打开的工程，单击OK，弹出图8-8所示界面。

图 8-8 工程打开提示



- 选择“This Window”，则直接在当前工作窗口打开选中工程。
- 选择“New Window”，则新建一个工作窗口打开选中的工程。

c. 根据需要选择相关按钮，将工程打开。

8.1.2 新建/删除文件

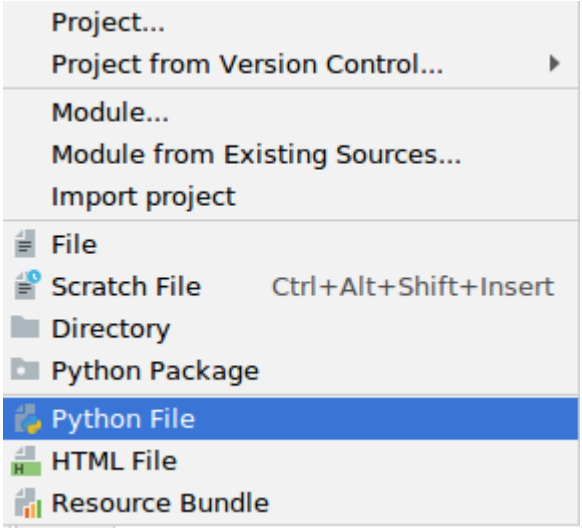
新建文件

在Projects视图中，选中相应工程，单击鼠标右键选择“New”（或者在File菜单中选择New），在弹出的子菜单中选择需要创建的文件类型或直接单击File，可以新建文件，新建文件有如下两种方式：

- 选择文件类型后直接新建文件

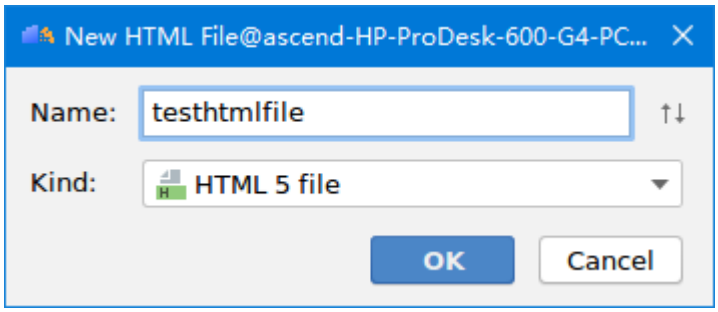
  a. 选中相应工程，单击“File>New”，弹出图8-9所示界面。

  图 8-9 选择文件类型

  b. 用户可以选择新建Python或HTML两种文件类型，选择其中一种文件类型后，弹出图8-10所示界面。

  图 8-10 输入文件名

  用户输入文件名后，单击“OK”即可。

  在“Kind”区域，用户还可以选择不同的文件类型：

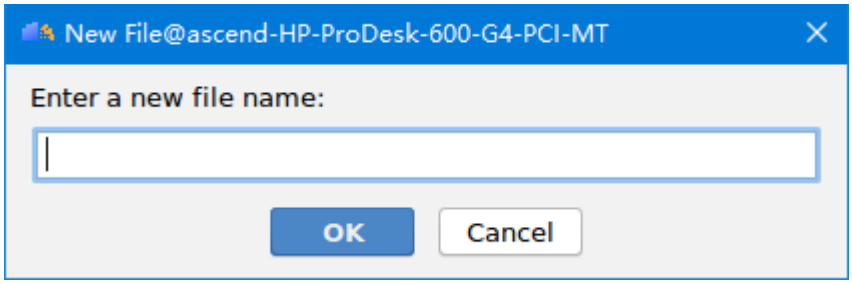
  - Python格式的文件类型：Python file、Python unit test、Python stub
  - HTML格式的文件类型：HTML 5 file、HTML 4 file、XHTML file

  若上述文件类型无法满足要求，则可以通过如下方式新建其他类型的文件。

- 先新建文件再选择文件类型

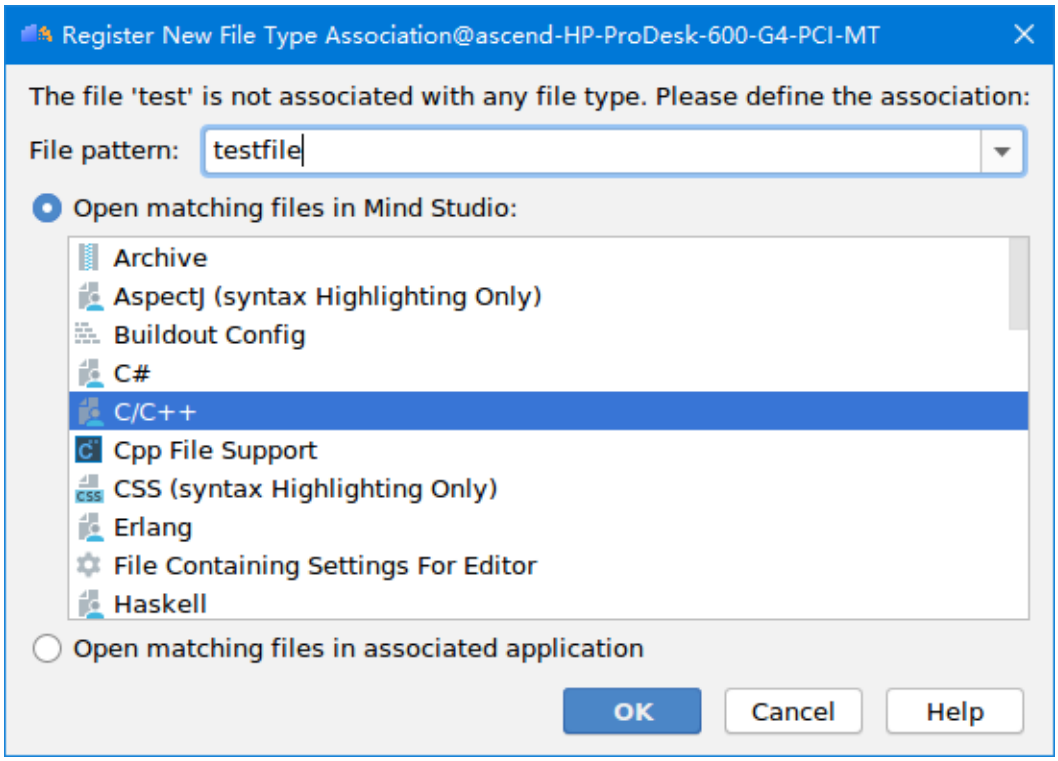
  a. 选中相应工程，依次单击“File > New > File”，弹出如图8-11所示页面。

  图 8-11 输入新建文件名



  b. 输入文件名后，单击“OK”，弹出如图8-12所示选择文件类型界面，用户可以根据需要选择新建文件的类型，选择完成后，单击“OK”即可。

  图 8-12 选择新建文件类型



删除文件

选中文件，鼠标右键，在弹出的页面中单击“Delete”即可。

8.1.3 修改工程属性

创建完算子工程或应用工程后，支持修改工程属性，包括DDK版本以及产品形态，方便用户通过一个工程运行不同的产品形态。

前提条件

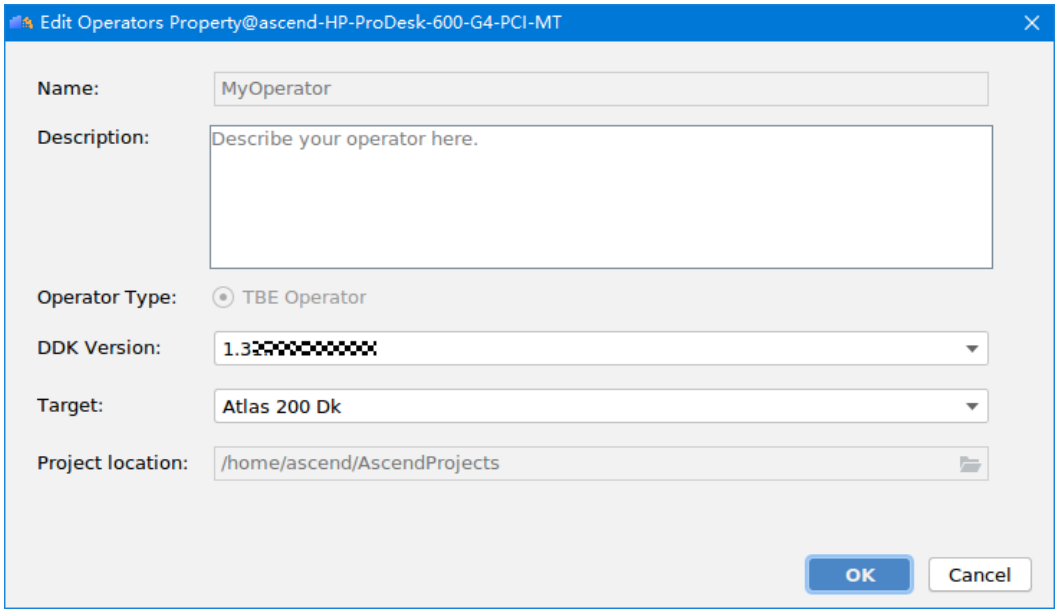
已经创建算子工程或应用工程，算子工程创建详细操作请参见6.2 自定义算子开发，应用工程创建详细操作请参见6.1 应用开发。

操作步骤

修改算子工程或应用工程属性方法类似，下面以修改算子工程属性为例进行说明。

步骤1 选中已经创建的算子工程，右击选择“Edit Operators Property”，弹出图8-13所示界面。

图 8-13 工程属性修改



界面参数解释如表8-1所示。

表 8-1 工程属性修改界面参数

参数	说明
Name	工程名称，创建算子工程时决定，不支持修改。
Description	工程描述信息，支持修改。
Operator Type	算子类型，创建算子工程时决定，不支持修改。 TBE Operator，代表创建TBE自定义算子工程。
DDK Version	当前DDK的版本号，不支持修改。
Target	运行环境，可以根据实际情况更换，目前支持如下几种形态： ● Atlas 200 DK形态 ● Atlas 300产品形态 **说明** 如果工程为通过5 工程迁移导入的工程，则Target属性不支持修改。

参数	说明
Project location	工程的存储路径，创建算子工程时决定，不支持修改。

步骤2 根据实际情况修改“DDK Version”以及“Target”，修改完后，单击“OK”。进行其他工程开发。

----结束

8.2 设备管理

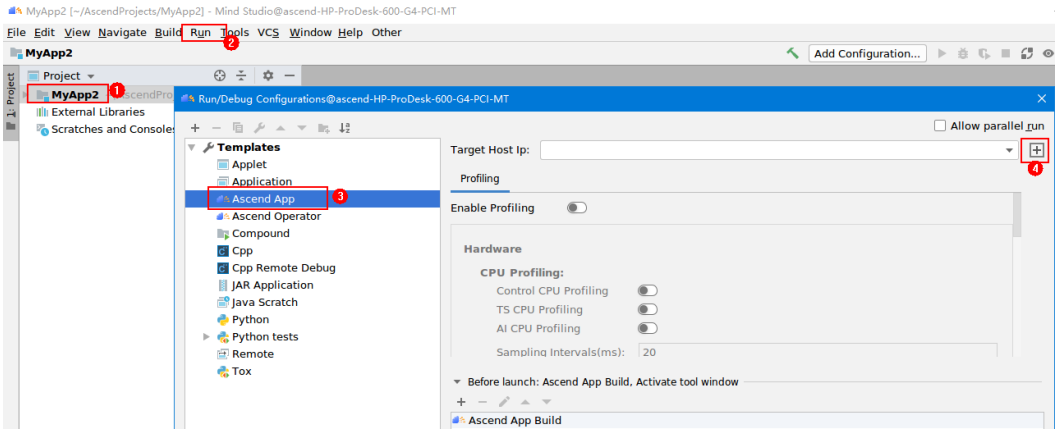
8.2.1 概述

本章节讲解如何通过Mind Studio界面的Device Management菜单管理设备信息。

设备管理功能提供了增加、删除、修改等能力，方便使用者对Device设备进行管理。您可以通过以下方式进行设备管理：

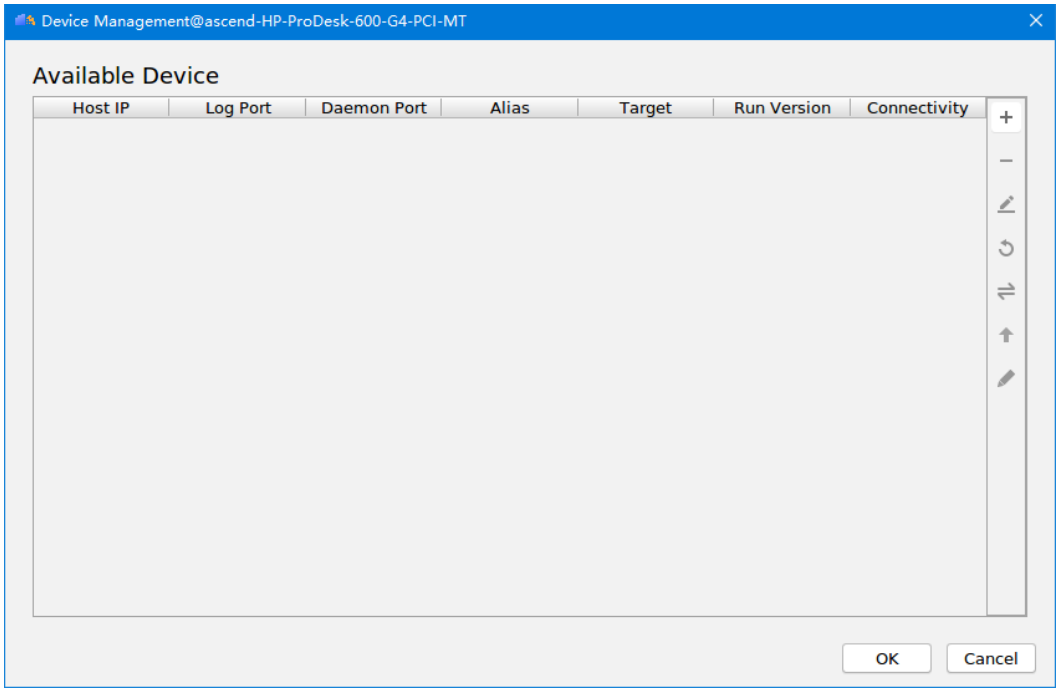
1. 在Mind Studio主界面，单击菜单栏“Tools > Device Management”，弹出设备管理窗口。
2. 在Mind Studio界面的“Log”窗口，单击图标，打开Device Management窗口。
3. 在已经打开的算子工程或App工程中，选中相应工程，单击菜单栏的“Run”，在弹出界面中选择相关配置，然后单击右侧的，如图8-14所示。

图 8-14 工程界面打开设备管理窗口



三种方式打开的设备管理窗口如图8-15所示。

图 8-15 Device Management 窗口



界面参数以及图标解释如表8-2所示。

表 8-2 设备管理界面参数以及图标说明

参数以及图标	说明
Host IP	设备侧IP地址。
Log Port	Log端口号，取值范围为[18000~18080]。请确保该端口不被占用。查询端口是否被占用的命令是：**netstat -an | grep *端口号*** 默认为18080。
Daemon Port	IDE_Daemon_client与IDE_Daemon_host通信使用的端口号，取值范围为[20000~25000]，请确保该端口不被占用。查询端口是否被占用的命令是：**netstat -an | grep *端口号*** 默认为22118。 ● 如果设备类型为Atlas 300形态，该参数需要和Host侧服务器“~/ide_daemon”目录下，ide_daemon.cfg配置文件中的HOST_PORT端口保持一致。~表示启动IDE-daemon-host进程的用户的家目录。 ● 如果设备类型为Atlas 200 DK形态，该参数需要和开发者板“~/ide_daemon”目录下，ide_daemon.cfg配置文件中的HOST_PORT端口保持一致。~表示启动IDE-daemon-host进程的用户的家目录。
Alias	设备的别名，在有多个设备时，方便用户管理设备信息。

参数以及图标	说明
Target	设备类型： ● EP：Atlas 300形态 ● RC：Atlas 200 DK形态
Run Version	设备软件包版本号。
Connectivity	Mind Studio与设备连接状态： YES：连接成功。 NO：连接失败。
+	增加设备，已经完成设备增加后，可以继续单击该按钮，增加多个设备。
−	删除设备，选中需要删除的设备单击该按钮进行删除。
	编辑设备，选中需要编辑的设备，单击该按钮，修改Host IP、Log Port、Daemon Port、Alias信息。
	检查设备连接状态，修改了设备信息或升级单板后，可以通过单击该按钮刷新设备连接状态，软件版本号以及设备类型。
	同步按钮，如果用户使用Mind Studio进行编译运行相关操作，则需要通过该按钮将设备侧的lib库以及sysroot同步到Mind Studio所在服务器。
	Atlas 200 DK开发者板升级按钮。
	修改Atlas 200 DK开发者板的IP地址，该按钮只有Target为RC时，选中设备信息后，该按钮才变成可编辑状态。

8.2.2 增加设备

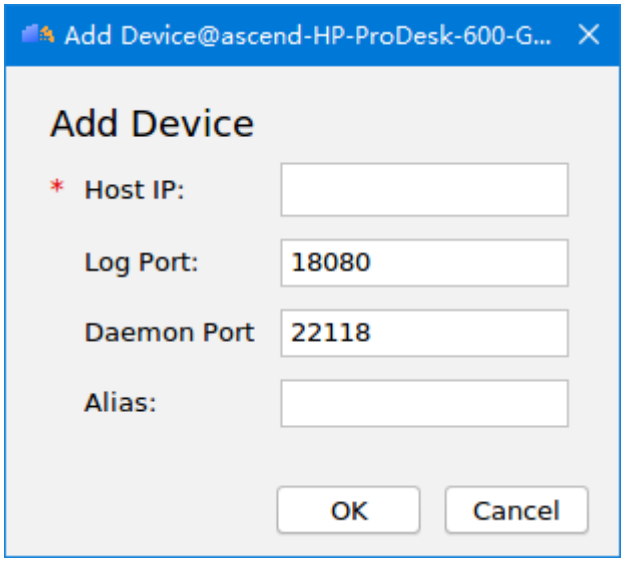
前提条件

请确保Mind Studio安装服务器与安装昇腾AI处理器设备网络连通。

增加设备

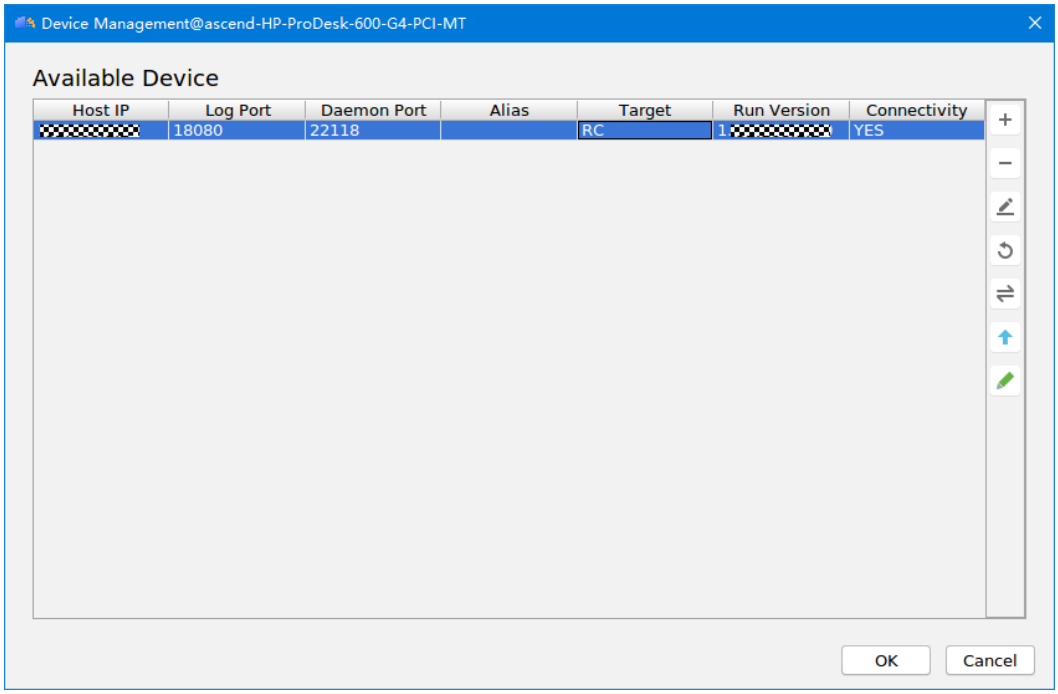
步骤1 单击设备管理界面右侧的 + 增加设备，弹出图8-16所示界面，参数解释如表8-2所示。

图 8-16 增加设备



步骤2 填写完相关信息后，单击“OK”，后台脚本会根据Host IP连接设备，并获取设备的Target以及版本信息。之后设备信息会增加到设备管理界面“Available Device”区域，如图8-17所示。

图 8-17 设备增加完成状态



说明

- 添加完图8-16设备参数，单击OK后，还会弹出增加设备进度条，刷新设备信息，该进度条根据用户网络状况，出现时间长短不定，待获取到设备信息后进入图8-17。
- 增加的设备IP地址必须保持唯一。
- 如果Host为CentOS操作系统，Mind Studio无法连接该环境时，请参见**2.11.1 安装完Mind Studio后，无法远程连接host为CentOS操作系统的服务器**解决。
- 暂不支持Host侧为虚拟机以及虚拟机中创建docker容器场景下的增加设备功能。

步骤3 单击“OK”，退出设备管理界面，下次再次进入设备管理界面，增加的设备仍会保留在图8-17所示界面中。

说明

如果不单击OK，增加完设备后直接单击窗口右上角的关闭界面，则设备信息不会被保留在图8-17所示界面中。

----结束

异常处理

设备增加完成后，若“Connectivity”参数显示为“NO”，根据**8.2.5 更新设备信息**更新设备后仍旧显示“NO”，则可以根据如下方法进行排查：

以Mind Studio安装用户登录Mind Studio所在后台服务器。

1. 检查Mind Studio安装用户根目录的可用磁盘空间是否小于3G。

   先使用**id *Mind Studio安装用户***命令检查Mind Studio安装用户的ID，然后使用**df -h**命令检查对应ID根目录的磁盘空间是否充足，如果可用磁盘空间小于3GB，则设备无法连接；如果磁盘空间大于3G，设备仍旧无法连接，则参见2继续排查。

2. 在Mind Studio安装服务器输入**ping *host IP***检查网络是否连通。

   如果能连通，以Mind Studio安装用户在DDK安装目录“$HOME/.mindstudio/huawei/ddk/{software version}/ddk/scripts”下输入如下命令查看连通设备所用的时间：

   ```
   python make_uihost_env.py hostIP.22118 ../ -ask-host-connect
   ```

   如果返回时间>5s，则即使网络能连通，“Connectivity”参数仍旧显示为“NO”，设备无法连接；只有网络连通且连通设备所用时间<5s时，“Connectivity”参数才有可能显示为“YES”，如果设备仍旧无法连接，则参见3继续排查。

3. IDE-daemon 通路检查。

   a. 检查输入的“Host IP”地址是否正确，如果正确则进入下一步。

   b. 检查“Daemon Port”接口是否正确，如果正确则进入下一步。

      在host侧服务器输入**netstat -an**命令，查看host侧 IDE-daemon-host 监听的端口22118 是否存在，如果存在，则查看检测的端口和host侧IP地址是否对应。

   c. 检查端口是否被防火墙阻隔：

      以HwHiAiUser用户登录Host侧服务器，执行如下命令检查Daemon Port端口是否被阻隔

      ```
      IDE-daemon-client --host hostIP.22118 --detect
      ```

      若命令执行后卡住未返回SUCCESS，则请切换到root用户执行如下命令配置防火墙策略（允许22118端口以ssh方式连接Host侧服务器）：

```
iptables -I INPUT -p tcp --dport 22118 -j ACCEPT
```

配置完成后，执行**iptables -L**命令查看结果。

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:22118
```

d. 使用命令**ps -ef | grep IDE-daemon-host**，检查IDE-daemon-host进程是否启动。

在Host侧服务器上重启IDE-daemon-host进程。

- 若为Atlas 300形态：

  以启动IDE-daemon-host进程的用户登录Host侧服务器，切换到“/usr/local/HiAI/driver/tools”目录下，依次执行如下命令重启IDE-daemon-host进程。

  ```
  kill IDE-daemon-host 进程ID
  ./IDE-daemon-host &
  ```

- 若为Atlas 200 DK形态：

  以HwHiAiUser用户ssh登录到Atlas 200 DK开发者板，切换到/var目录，执行如下命令重启IDE-daemon-host进程。

  ```
  kill IDE-daemon-host 进程ID
  ./IDE-daemon-host &
  ```

e. 检查Mind Studio安装服务器与host侧服务器openssl证书是否一致。

在Mind Studio安装服务器DDK安装目录“$HOME/.mindstudio/huawei/ddk/*{software_version}*/ddk/conf/ide_daemon”下和host侧服务器“/usr/local/HiAI/driver/tools”路径下分别执行输入如下命令：

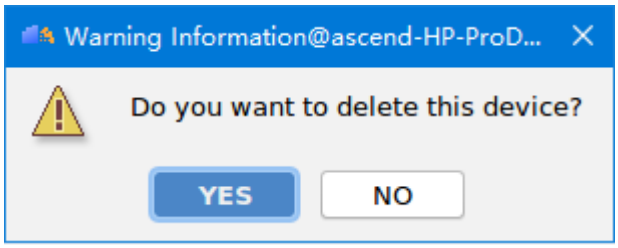
```
openssl x509 -inform pem -noout -text -in ide_daemon_cacert.pem
```

查看两者“Serial Number”对应的序列号是否一致。

8.2.3 删除设备

选中**8.2.2 增加设备**章节中**图8-17**想要删除的设备，单击右侧菜单栏的 会弹出如**图8-18**所示界面，单击“YES”确认删除，单击“NO”取消删除。

图 8-18 删除设备



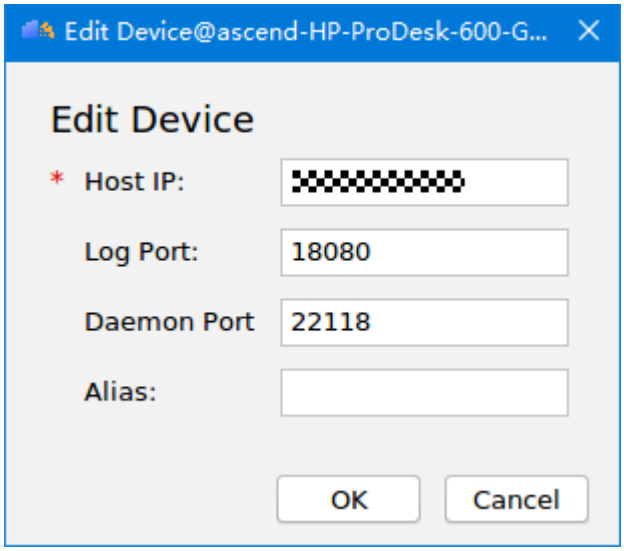
说明

删除设备不会删除已经同步完成的lib文件。

8.2.4 编辑设备

步骤1 选中8.2.2 增加设备章节中图8-17需要编辑的设备，单击右侧菜单栏的 弹出如图8-19所示界面，界面参数解释如8.2.1 概述章节中的表8-2所示。

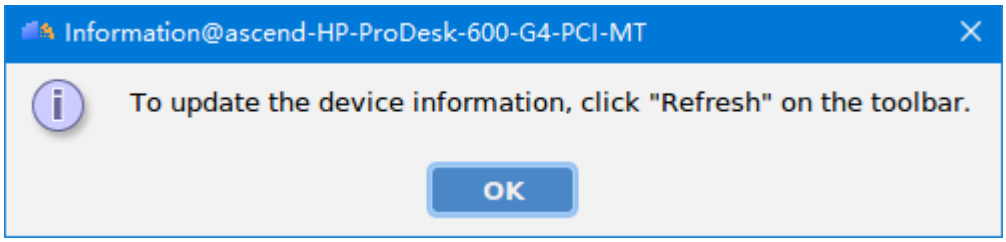
图 8-19 编辑设备



您可以修改“Host IP”、“Log Port”、“Daemon Port”、“Alias”参数。

步骤2 修改完成后，单击“OK”，会弹出图8-20所示界面。

图 8-20 编辑设备后的处理



步骤3 单击“OK”。如果想要更新设备信息，则单击设备管理界面的 ，更新设备信息请参见8.2.5 更新设备信息。

说明

修改“Host IP”后，如果用户涉及编译运行相关操作，则需要重新同步Lib库信息，详细操作请参见8.2.6 同步lib。

----结束

8.2.5 更新设备信息

编辑完设备信息或升级单板后，例如修改“Host IP”导致设备类型变化，则需要单击右侧菜单栏的 更新设备信息。该功能只会更新设备管理界面的如下三个参数：

- Target：设备类型。
- Run Version：设备软件包版本号。
- Connectivity：设备连接状态。

参数详细说明请参见8.2.1 概述章节中的表8-2。操作方法如下：

步骤1 选中需要更新信息的设备，单击右侧菜单栏的 ，弹出图8-21所示界面。

图 8-21 更新设备信息

步骤2 更新设备信息完成，检查上述三个参数是否更新。

- 如果参数更新，说明设备信息有变更。
- 如果参数没有更新，说明设备信息没有变更。

----结束

8.2.6 同步 lib

前提条件

- 同步lib之前请确保已经完成8.2.2 增加设备。
- 请确保DDK版本号与设备侧安装的软件包版本保持一致。
- 请确保已经打开工程的target（可以通过工程的.project查看）和所同步设备的target保持一致，如果不一致，可以参见8.1.3 修改工程属性尝试修改工程属性。
- 请确保Mind Studio安装服务器剩余足够磁盘空间（至少为3G），同步lib过程中，如果有同步sysroot场景，由于sysroot文件过大，可能会造成磁盘空间不足，导致同步失败。

说明

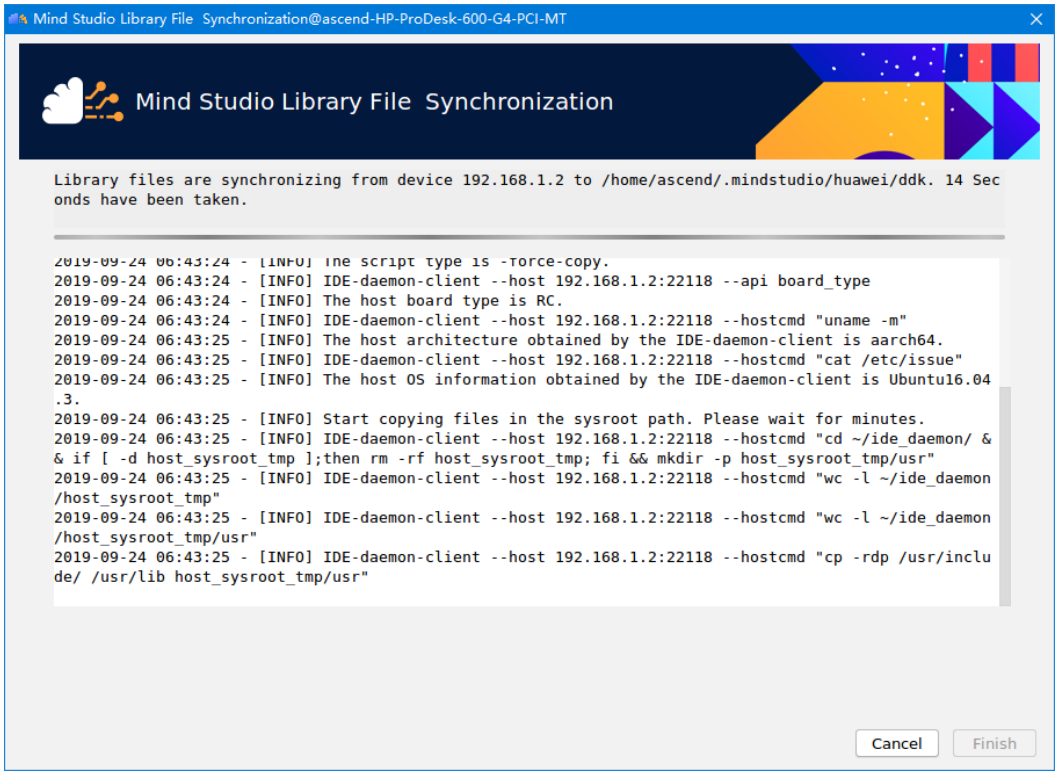
同步lib时，设备侧服务器只能被一台Mind Studio安装服务器所连接，如果多台Mind Studio安装服务器同时连接设备侧进行同步，可能会同步失败。

操作步骤

步骤1 选中增加的设备，单击 启动同步，弹出图8-22所示同步界面。

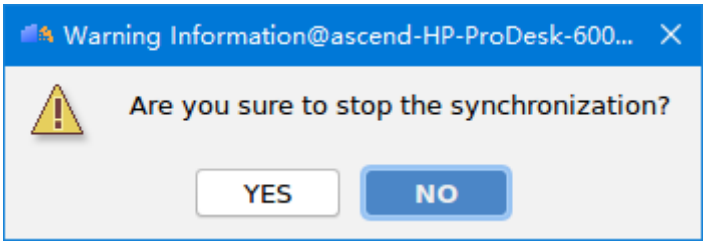
在弹出图8-22界面之前，还会弹出“Pre Synchronization Check”界面更新设备信息，该界面根据用户网络状况，出现时间长短不定，待更新完毕再进入图8-22。

图 8-22 同步界面



- 若同步过程中单击“Cancel”取消同步，弹出图8-23所示界面。

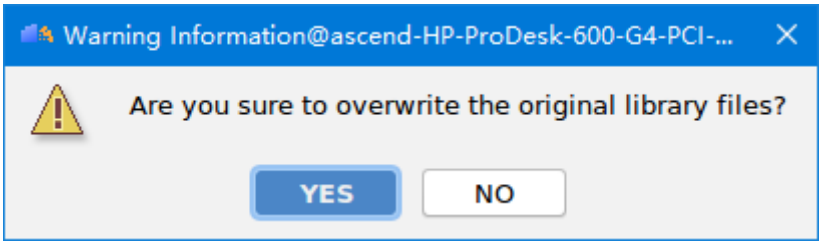
图 8-23 取消同步信息



单击“YES”取消同步，返回设备管理界面，单击“NO”继续同步进入步骤2。

- 若Mind Studio安装服务器“$HOME/.mindstudio/huawei/ddk”路径下已经有之前同步过的lib文件，会弹出如图8-24所示界面。

图 8-24 覆盖 lib 库文件信息



单击“YES”，启动同步，覆盖原有Lib内容；单击“NO”，取消同步，返回设备管理界面。

说明

- 单击 ⇌ 同步按钮后，脚本会自动检查所连接设备的连通性，设备连通后，再检查设备的软件版本号与当前Mind Studio所使用的DDK版本是否一致，上述都检查通过后，才会启动同步。
- 若本地已经创建工程，在当前工程中单击同步lib时，请确保工程的Target与所同步设备Target保持一致。如果不一致，请参见8.1.3 修改工程属性修改工程属性或更换设备类型。

步骤2 同步完成且窗口中提示如下信息，则说明同步lib成功，此时Finish按钮变成可单击状态，其余按钮置灰。图8-22上方给出同步后lib在Mind Studio安装服务器路径，并给出同步所耗费时间。

```
Library file synchronization succeeded. Please click "Finish".
```

说明

同步失败Finish按钮也会变成可单击状态，其余按钮置灰，在同步界面会显示日志信息。

步骤3 单击“Finish”完成同步，返回设备管理界面。

同步的lib会按照host侧和device侧操作系统架构、操作系统以及操作系统版本在Mind Studio安装服务器*{x-compile path}*路径创建对应目录，分别存储host侧lib库、device侧lib库。lib库存储路径为：

- Atlas 300形态：则会将host以及device侧lib库同步到Mind Studio安装服务器
  - host侧lib库存储路径：{x-compile path}/{version}/形态/host-{host arch}_{host OS}/lib
  - device侧lib库存储路径：{x-compile path}/{version}/形态/device -{device arch}_{device OS}/lib
- Atlas 200 DK形态：则会将host侧lib库同步到Mind Studio安装服务器

  host侧lib库存储路径：{x-compile path}/{version}/形态/host-{host arch}_{host OS}/lib

参数说明如表8-3所示。

表 8-3 lib 库存储路径对应参数说明

参数	说明
{x-compile path}	在Mind Studio安装服务器存放路径，默认为：$HOME/.mindstudio/huawei/ddk，不支持修改。
{version}	lib包版本。
形态	包括如下两种形态： RC：Atlas 200 DK形态 EP：Atlas 300形态
{host arch}	host侧操作系统架构，例如x86_64。
{device arch}	device侧操作系统架构，例如aarch64。
{host OS}	host侧操作系统，例如Ubuntu16.04。

参数	说明
{device OS}	device侧操作系统，例如miniOS、Ubuntu16.04.3。

图8-25为形态为EP时同步成功的样例，图8-26为形态为RC时同步成功的样例。

图 8-25 lib 同步成功样例_EP

```
ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~/.mindstudio/huawei/ddk/1.31.[illegible]9/EP$
total 16
drwxrwxr-x 4 ascend ascend   4096 Aug 29 03:45 ./
drwxrwxr-x 3 ascend ascend   4096 Aug 29 03:44 ../
drwxrwxr-x 3 ascend ascend   4096 Aug 29 03:45 device-aarch64_miniOS/
drwxrwxr-x 3 ascend ascend   4096 Aug 29 03:44 host-x86_64_Ubuntu16.04/
```

图 8-26 lib 同步成功样例_RC

```
ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~/.mindstudio/huawei/ddk/1.31.T[illegible]0/RC$ ll
total 12
drwxr-x--- 3 ascend ascend 4096 Sep 24 06:44 ./
drwxr-x--- 4 ascend ascend 4096 Sep 24 06:44 ../
drwxr-x--- 3 ascend ascend 4096 Sep 24 06:44 host-aarch64_Ubuntu16.04.3/
```

若Mind Studio安装服务器操作系统架构与设备侧操作系统架构不同，同步lib时，脚本会自动将设备侧的sysroot同步到Mind Studio安装服务器，同步完成后，会按照host侧操作系统架构、操作系统以及操作系统版本在Mind Studio安装服务器创建对应目录，存储sysroot。

sysroot存储路径为：{x-compile path}/sysroot/{arch}_{os}_{os_version}

表 8-4 sysroot 存储路径对应参数说明

参数	说明
{x-compile path}	sysroot在Mind Studio安装服务器存放路径，默认为：$HOME/.mindstudio/huawei/ddk，不支持修改。
{arch}	host侧操作系统架构，例如aarch64。
{os}	host侧操作系统，例如Ubuntu。
{os_version}	host侧操作系统版本，例如16.04.3。

图8-27为形态为RC时同步成功的样例。

图 8-27 sysroot 同步成功样例

```
ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~/.mindstudio/huawei/ddk/sysroot$ ll
total 12
drwxr-x--- 3 ascend ascend 4096 Sep 24 06:43 ./
drwxr-x--- 4 ascend ascend 4096 Sep 24 06:43 ../
drwxr-x--- 4 ascend ascend 4096 Sep 24 06:44 aarch64_Ubuntu16.04.3/
```

----结束

8.2.7 升级 Atlas 200 DK

用户可以通过设备管理界面中的升级功能，在线升级Atlas 200 DK开发者板。

说明

设备管理界面中的 升级按钮只支持升级Atlas 200 DK开发者板。

前提条件

- 参见8.2.2 增加设备已经将Atlas 200 DK增加到设备管理界面中，并且“Target”参数显示为“RC”，“Connectivity”参数显示为“YES”。
- 升级前请使用Mind Studio安装用户将Atlas 200 DK开发者板升级包上传到Mind Studio安装服务器：
  - mini_developerkit-xxx.rar：安装包
  - mini_developerkit-xxx.rar.asc：安装包完成性校验文件

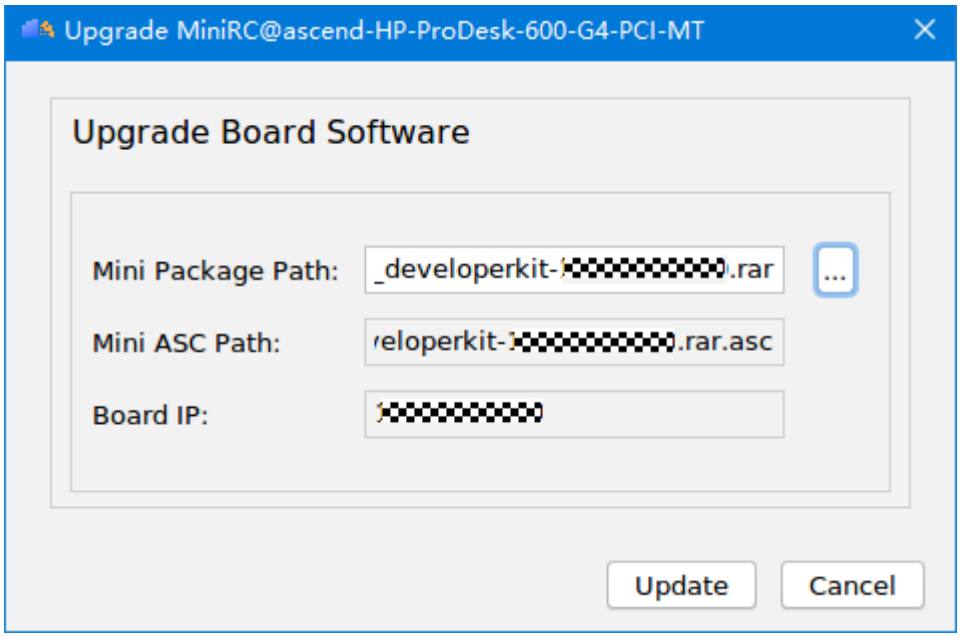
  说明

  xxx代表具体版本号，安装包与安装包校验文件需放在同一目录。

升级操作

步骤1 在设备管理界面选中要升级的开发者板，单击右侧的 升级按钮，弹出如图8-28所示升级界面。

图 8-28 升级开发者板



界面参数说明如下：

- Mini Package Path：开发者板安装包所在路径。

- Mini Asc Path：安装包校验文件所在路径，选择安装包后，.asc校验文件会自动填充。
- Board IP：Atlas 200 DK开发者板IP地址。

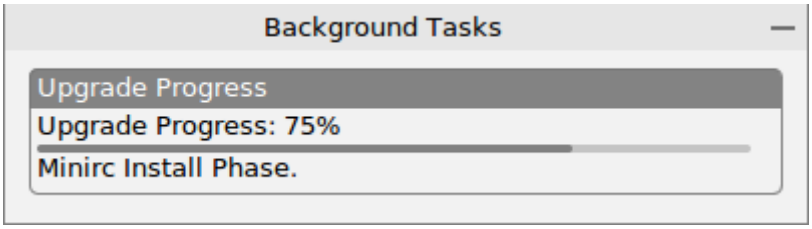
说明

开发者板升级过程中，升级脚本会自动调用.asc文件校验安装包的完整性。

步骤2 选择相关安装包和校验文件后，单击“Update”进行升级。

- 单击“OK”关闭设备管理窗口，在Mind Studio界面左下方的Output窗口中查看升级过程的错误信息，以及是否结束升级信息，在Detail窗口打印升级过程中的日志信息。
- 升级过程中Mind Studio界面最底层中间会给出升级进度，右下方会给出升级进度条，单击进度条也可以查看升级进度，如图8-29所示。

图 8-29 升级进度



步骤3 升级成功后，Output窗口打印如下信息：

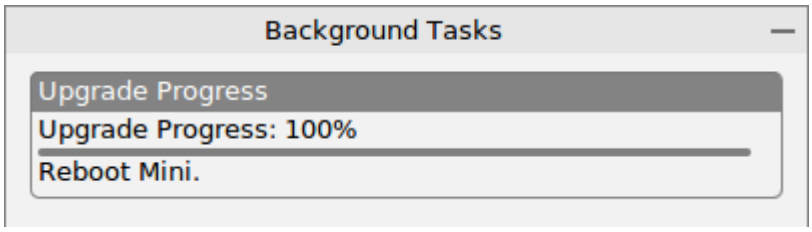
```
Upgrade MiniRC successfully. Please go to Tools -> Device Management, click "Refresh" button to update the device information.
```

Detail窗口打印如下信息：

```
All hardware successes to upgrade
date Upgrade finish.
```

单击进度条，显示升级成功，如图8-30所示。

图 8-30 进度条升级成功显示信息



说明

如果升级失败，请在“$HOME/upgradeLogForMindStudio/upgradeMini.log”文件中查看详细日志信息。

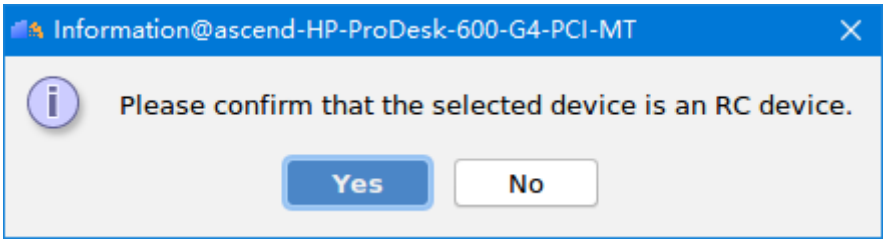
步骤4 打开设备管理界面，选中刚升级的开发者板，单击设备管理界面右侧的 ，刷新升级后的设备信息。

----结束

异常处理

1. 如果用户在设备管理界面增加的开发者板Target参数显示为空，Connectivity参数显示为YES，选中设备单击 升级时会弹出图8-31所示界面。

   图 8-31 确认升级的设备是否为 RC

   由用户确认设备为RC后，单击“Yes”，然后进入图8-28所示界面，单击“No”返回设备管理界面。

2. 如果用户输入的安装包和安装包校验文件不完整或者无效，会在升级界面弹出如图8-32所示窗口，并在Detail窗口出现日志信息，如图8-33所示。

   图 8-32 异常信息

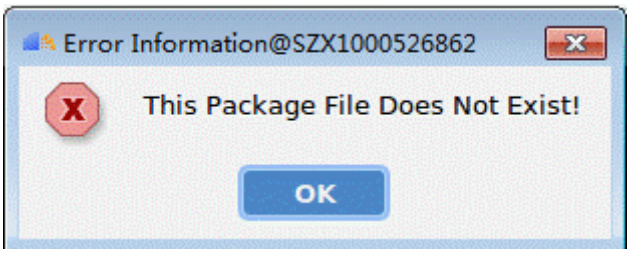
   

   图 8-33 detail 窗口的日志信息 1

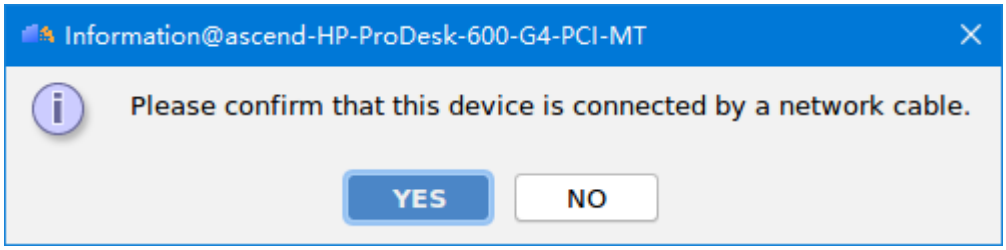
8.2.8 修改 Atlas 200 DK IP 地址

说明

- 设备管理界面中的 按钮只有Target参数为RC时，选中设备该按钮才变成可编辑状态。
- 当前Atlas 200 DK IP地址修改不支持USB通讯方式，只支持NIC通讯方式。

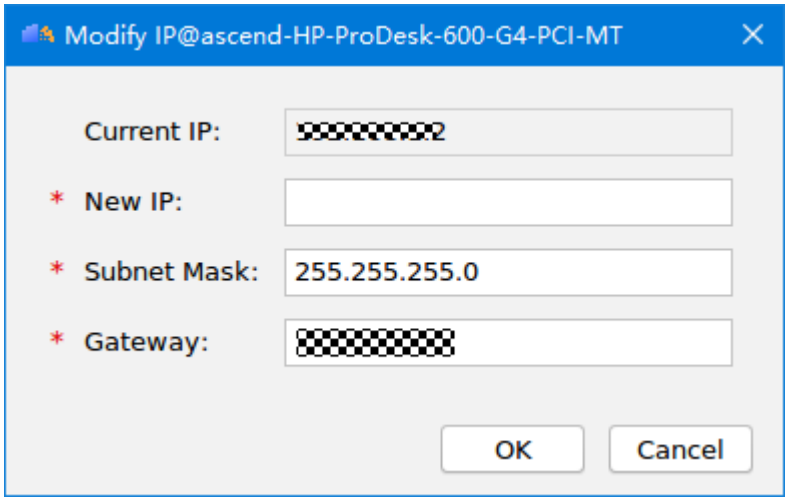
步骤1 在设备管理界面选中要修改IP地址的的开发者板，单击右侧的 按钮，弹出图8-34所示界面。

图 8-34 确认开发者板是否连接



步骤2 确认设备连接后，单击“YES”，弹出图8-35所示界面。

图 8-35 修改 IP 地址界面



界面参数解释如表8-5所示。

表 8-5 修改 IP 地址界面参数

参数	说明
Current IP	当前Atlas 200 DK IP地址。不支持修改。
New IP	新的Atlas 200 DK IP地址。 **说明** 修改后的IP地址必须为有效且未被分配过的（没有设备使用）。
Subnet Mask	子网掩码。
Gateway	网关。

说明

New IP、Subnet Mask、Gateway必须保持一致。

步骤3 单击“OK”完成IP地址修改。弹出“Modify IP Success! Please click "Refresh" button after rebooting.”提示信息。关闭窗口后，单击 刷新设备信息，设备管理界面更新为新的开发者板IP地址。

----结束

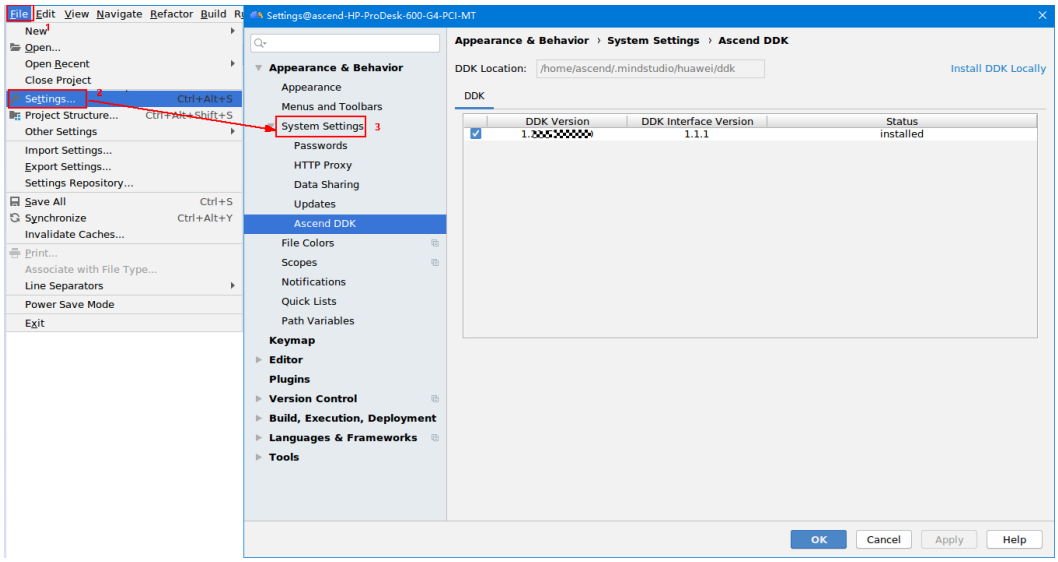
8.3 DDK 版本管理

DDK Manager为用户提供了查看已经安装DDK版本信息的功能，同时还为用户提供了在不重装Mind Studio的前提下，替换DDK版本的功能。

8.3.1 前提条件

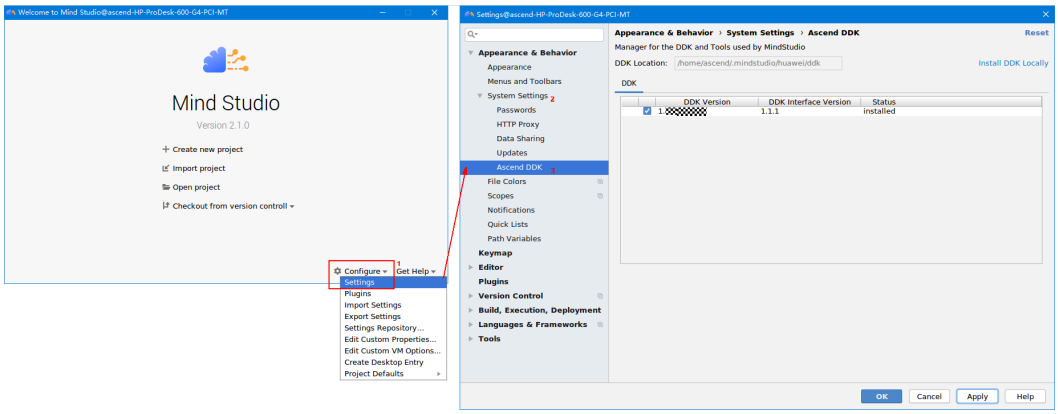
- 根据2 安装已经完成Mind Studio以及DDK的安装。
- 将所需的新版本DDK安装包上传到Mind Studio所在后台服务器。
- DDK Manager入口：
  a. 在Mind Studio工程界面依次选择“File > Settings > System Settings > Ascend DDK”，弹出图8-36所示界面。

图 8-36 DDK 版本管理界面入口 1



  b. 在Mind Studio欢迎页面，依次选择“Configure > Settings > System Settings > Ascend DDK”，弹出图8-37所示界面。

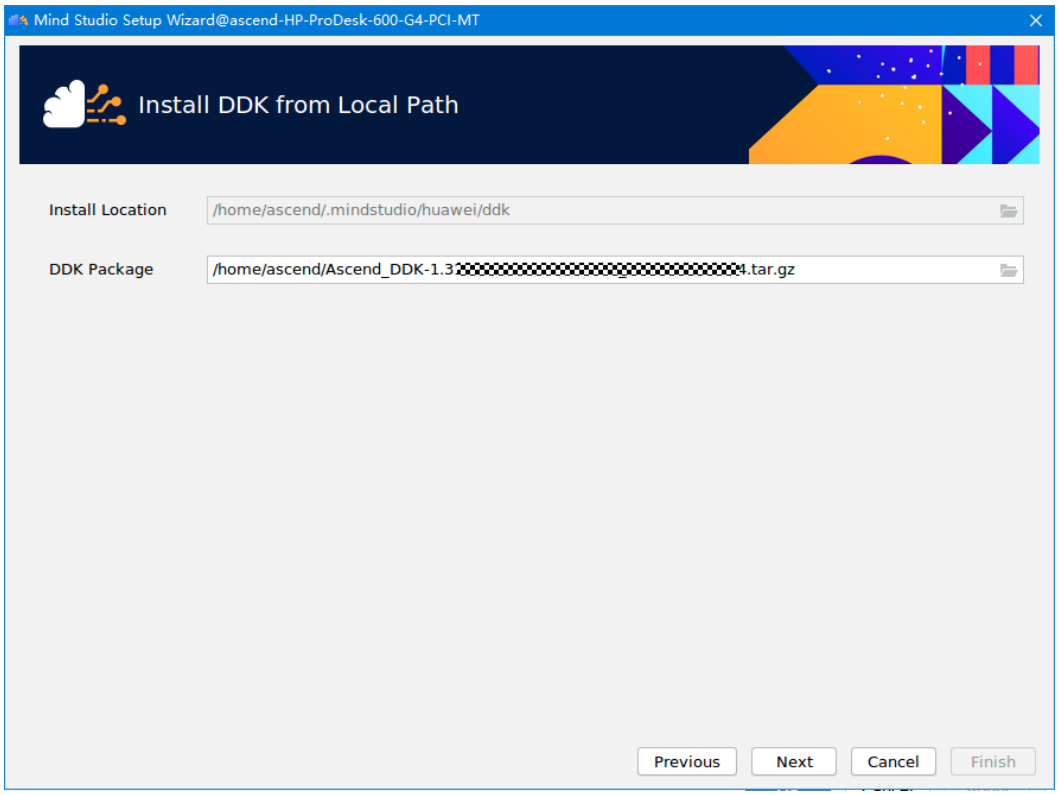
图 8-37 DDK 版本管理界面入口 2



8.3.2 安装新版本 DDK 版本

步骤1 单击DDK版本管理界面右侧的“Install DDK Locally”，弹出重新安装DDK界面，如图8-38所示。

图 8-38 重新安装 DDK 界面



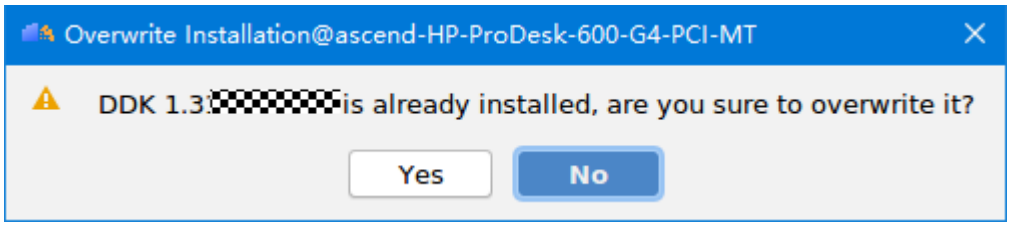
界面参数解释请参见2.5 安装Mind Studio章节中步骤5，在“DDK Package”中选择新的DDK安装包。

说明

Install Location：DDK安装包在部署服务器上的路径，不支持修改。

步骤2 单击“Next”进入如图8-39所示确认是否继续安装界面。

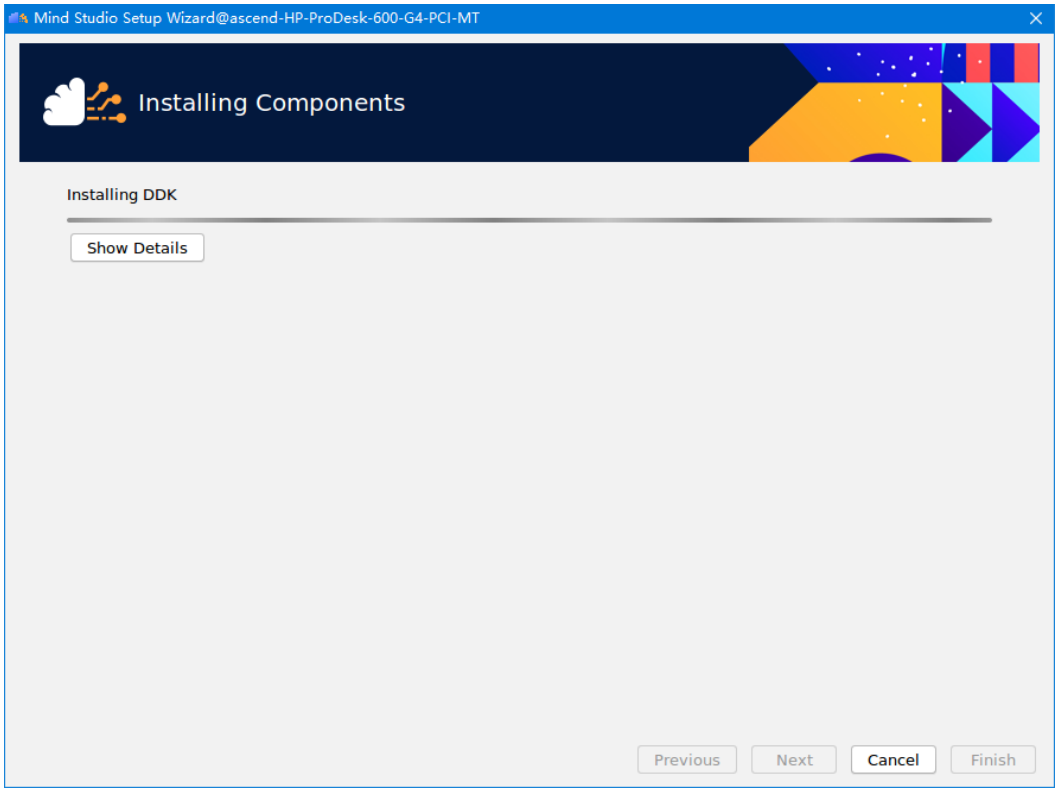
图 8-39 确认继续安装界面



单击“Yes”确认继续安装，进入下一步；单击“No”不安装，返回DDK版本管理界面。

步骤3 DDK安装界面如图8-40所示。

图 8-40 DDK 安装界面

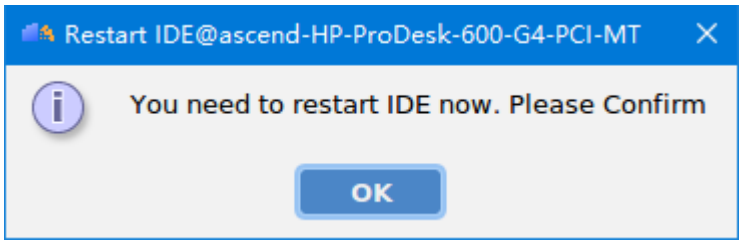


单击“Show Details”可以显示详细安装日志信息，如果显示如下信息，则说明DDK安装成功。此时“Finish”按钮变成可单击状态，其余按钮置灰。

```
Successfully installed the DDK!
DDK installed
DDK {software version} is installed
Installation succeeded. Please click "Finish"
```

步骤4 单击“Finish”弹出如图8-41所示重启IDE提示信息。

图 8-41 重启 IDE 提示



步骤5 单击OK，退出Mind Studio。然后参见**2.7 启动Mind Studio**重新启动Mind Studio。

重新启动Mind Studio后，用户重新进入DDK版本管理界面，可以看到DDK版本已经是最新版本。

说明

- 支持相同版本的重复安装。
- 如果用户安装新版本DDK时已经打开某个工程，则安装完毕重新进入Mind Studio时，默认重新加载该工程时还会弹出**图8-42**所示界面，用户单击OK，在弹出**图8-43**界面中重新加载DDK版本即可。安装完毕后重新打开全新工程也会弹出**图8-42**所示界面，用户重新加载DDK即可。

图 8-42 DDK 版本替换后打开工程提示信息

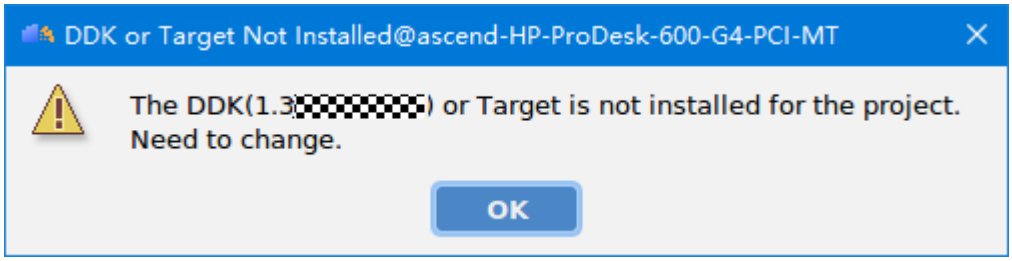
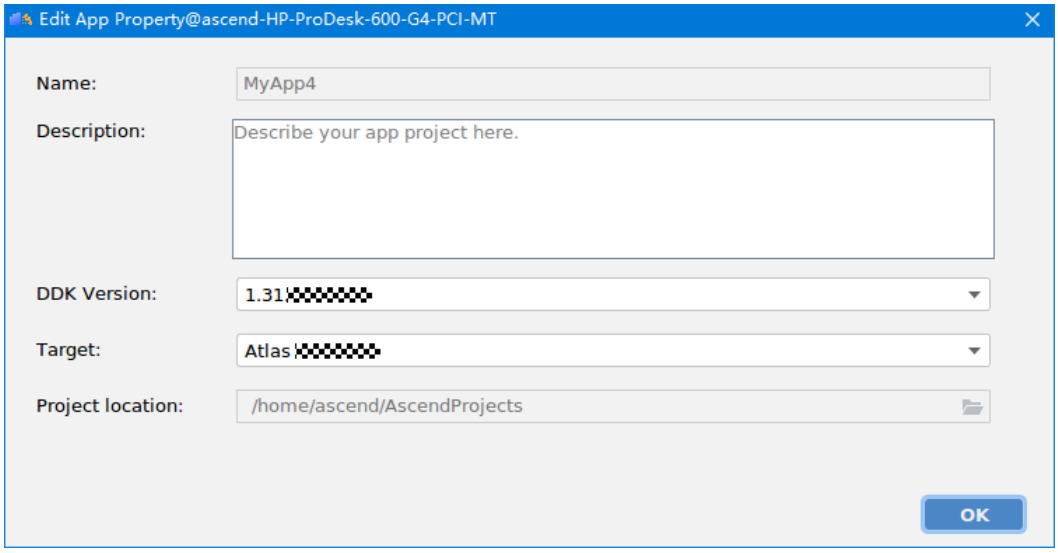


图 8-43 重新加载 DDK



----结束

9 Profiling 工具

9.1 简介

9.2 APP工程Profiling

9.3 参考

9.1 简介

工具介绍

Mind Studio针对昇腾AI处理器提供高效、易用、可灵活扩展的系统化性能分析工具Profiling，便于快速识别产品的关键性能瓶颈并提出针对性能优化的建议，最终实现产品的极致性能。

Profiling提供针对Ascend APP工程的硬件和软件性能数据采集、分析、汇总展示。总体流程如下：

1. 配置需要采集的数据，包括硬件和软件的性能数据。

   硬件的性能数据包括：Control CPU、Task Schedule CPU、AI CPU、AI Core、外围设备（DVPP、NIC）、LLC、DDR的性能数据等。

   软件的性能数据包括：HIAI Engine（本文其他地方也称为Matrix）、OME、RTS等模块的性能数据。

2. 采集性能数据。

   配置Mind Studio与Host侧的连接后，可开始采集性能数据。

3. 汇总展示性能分析的结果。

   在Mind Studio界面上以Timeline、Hardware、AI CPU Function&Control CPU Function&TS CPU Funtion、Summary四个维度展示性能分析的结果。

工具使用约束

以下情况不能使用Profiling功能：

- Profiling不支持基于Matrix框架编码出的多Graph、多进程并行运行的用例程序。
- Profiling不支持同时从Mind Studio安装侧发起2个基于相同结果目录的Profiling。

- Profiling功能与Dump功能不能同时使用，即打开Profiling开关前，请关闭数据Dump。

9.2 APP 工程 Profiling

9.2.1 工程准备

存在已编译过的、“Target”为Atlas 200 DK或Atlas 300的Ascend APP工程，可以通过工程目录下的“.project”文件的“target”项查看工程类型，对应值为“RC”或“EP”的工程。

9.2.2 启动 Profiling

通过以下操作方法启动Profiling：

步骤1 选择已编译完成的工程。

步骤2 单击菜单栏“Run > Run”，选择工程运行，在弹出的“Edit configurations”窗口中找到“Profiling”区域并启用“Profiling”参数，配置Profiling参数项。

说明

因为同时开启Dump数据采集和Profiling功能，会影响Profiling性能数据准确性，所以，当启用“Profiling”参数时，会自动判断工程是否开启Dump数据采集功能，如果已开启Dump数据采集功能，则无法启用“Profiling”参数；如果需要继续使用Profiling功能，需要关闭Dump数据采集功能后启用“Profiling”参数。

图 9-1 运行界面

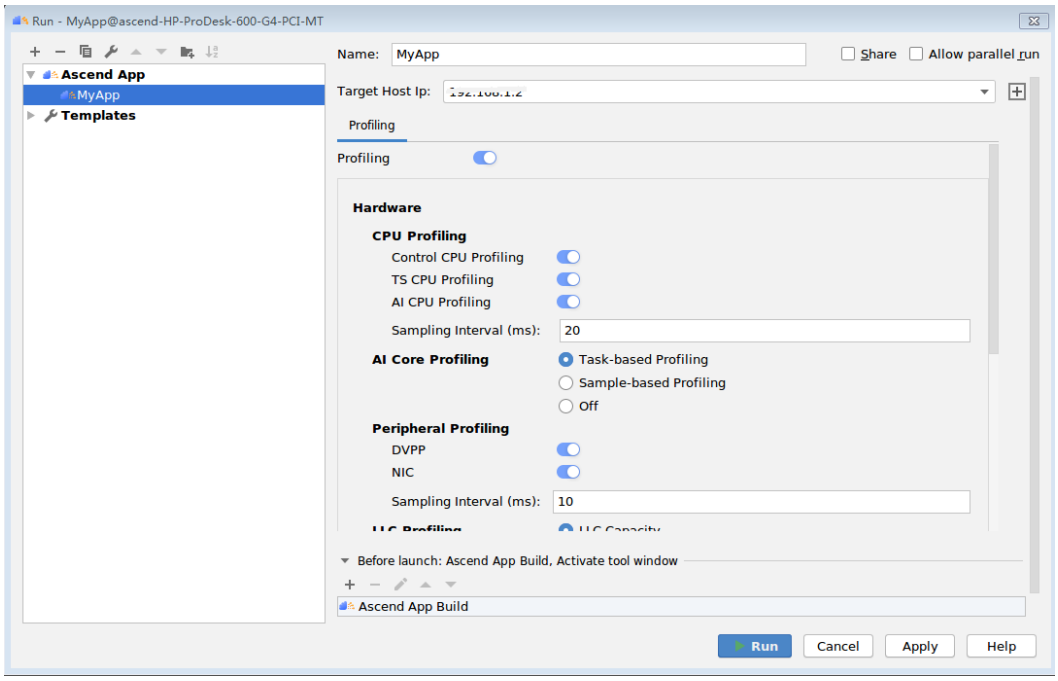


表 9-1 Profiling 参数说明

分类	参数		说明
Hardware	**CPU Profiling**	**Control CPU Profiling**	表示NPU的主控CPU，支持ON/OFF开关，设置为ON后，采集Control CPU上的PMU事件，当前支持的默认事件包括0x11和0x8，其中0x11表示CPU执行的Cycles数，0x8表示已执行完的Instructions数。
		TS CPU Profiling	表示AI task调度CPU，支持ON/OFF开关，设置为ON后，采集TS CPU上的PMU事件，当前支持的默认事件包括0x11和0x8，其中0x11表示CPU执行的Cycles数，0x8表示已执行完的Instructions数。
		AI CPU Profiling	表示处理AI算子（非AI core）的CPU，支持ON/OFF开关，设置为ON后，采集AI CPU上的PMU事件，当前支持的默认事件包括0x11和0x8，其中0x11表示CPU执行的Cycles数，0x8表示已执行完的Instructions数。
		Sampling Interval(ms)	配置Control CPU、TS CPU和AI CPU的PMU事件采集周期，默认值为20ms。
	AI Core Profiling	**Task-Based Profiling**	负责scalar、cube、fp等AI计算的处理单元采集，当前支持Task-Based和Sample-Based采集，支持的默认事件包括0xa、0x8、0x9、0xe、0x3a、0x3b、0x4a和0x49，0xa表示执行的cube类型指令的cycle数，0x8表示执行的vector类型指令执行的cycle数，0x9表示执行的scalar类型指令执行的cycle数，0xe表示指的是总指令cycle数，0x3a表示scalar访问UB读请求类型指令执行的cycle数，0x3b表示scalar访问UB写请求类型指令执行的cycle数，0x4a表示cube int类型指令执行的cycle数，0x49表示cube FP类型执行执行的cycle数。 Task-Based是以task为粒度进行性能数据采集（应用于性能调优），Sample-Based是以固定的时间周期进行性能数据采集（应用于系统长稳性能调优）。 **说明** ● 启用“Task-Based Profiling”模式，必须同时开启“RTS Profiling”。该场景下AI Core数据采集与RTS模块相关。 ● 启用“Sample-Based Profiling”模式，需配置采集周期“Interval”，默认取值10ms。
		Sample-Based Profiling	
	Peripheral Profiling		配置外设采集，当前支持的外设包括DVPP和NIC，默认采集周期（**Sampling Interval**）是10ms。
	LLC Profiling	**LLC capacity**	支持ON/OFF开关，表示采集LLC （Last Level Cache）的使用容量信息。与**LLC bandwidth**开关互斥，只能同时采集一种LLC数据，“Sampling Interval”表示LLC采集周期，若有输入需确认输入是否在[100, 1000] ms之间，默认值为100ms。

分类	参数		说明
		LLC bandwidth	支持ON/OFF开关，表示采集LLC（Last Level Cache）的读写带宽信息。与**LLC capacity**开关互斥，只能同时采集一种LLC数据，“Sampling Interval”表示LLC采集周期，若有输入需确认输入是否在[100, 1000] ms之间，默认值为100ms。
	DDR Profiling		支持ON/OFF开关，表示采集DDR的读写带宽。配置为on为开启采集，配置为off为不采集，默认为off。 “Master ID”表示具体采集哪个Core的读写带宽，当前可选0,1,2,3,4,5,6,7。0,1,2,3对应Control CPU core ID；4,5,6,7对应AI CPU core ID。 “Sampling Interval”表示DDR采集周期，若有输入需确认输入是否在[100, 1000] ms之间，默认值为100ms。
Software	**HIAI Engine Profiling**		支持ON/OFF开关，设置为ON后，采集Engine性能数据。
	OME Profiling		支持ON/OFF开关，设置为ON后，采集OME性能数据。 **说明** 开启“OME Profiling”，必须同时开启“RTS Profiling”。OME数据仅记录融合算子对应的task信息，获取算子时间时，需要通过解析对应的RTS Task才可以计算。
	RTS Profiling		支持ON/OFF开关，设置为ON后，采集RTS性能数据。

步骤3 单击“Run”，执行工程的Profiling。

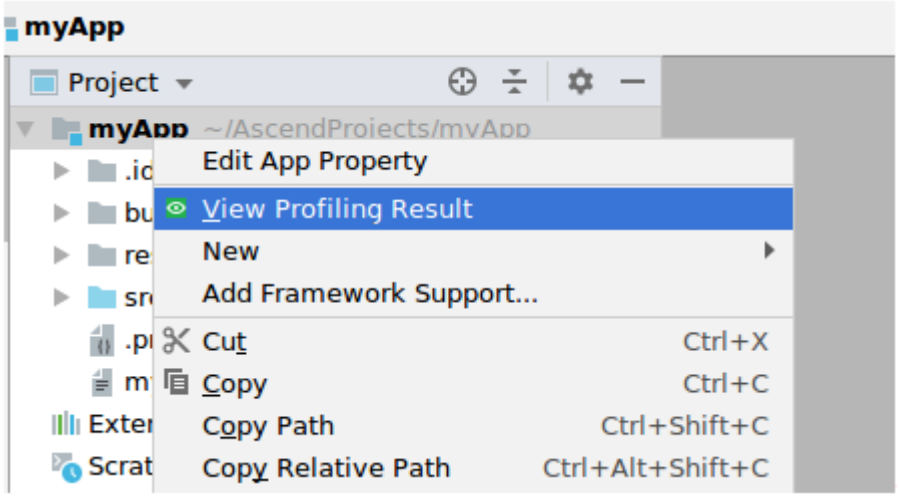
工程执行完成后，Mind Studio窗口下方展示Profiling结果视图。

----结束

9.2.3 查看 Profiling 数据

Profiling执行成功后，通过右键单击工程目录，选择“View Profiling Result”菜单，查看Profiling数据结果。展示的性能数据主要包括：Timeline、Hardware、AI CPU Function、Control CPU Function、TS CPU Function、Summary。

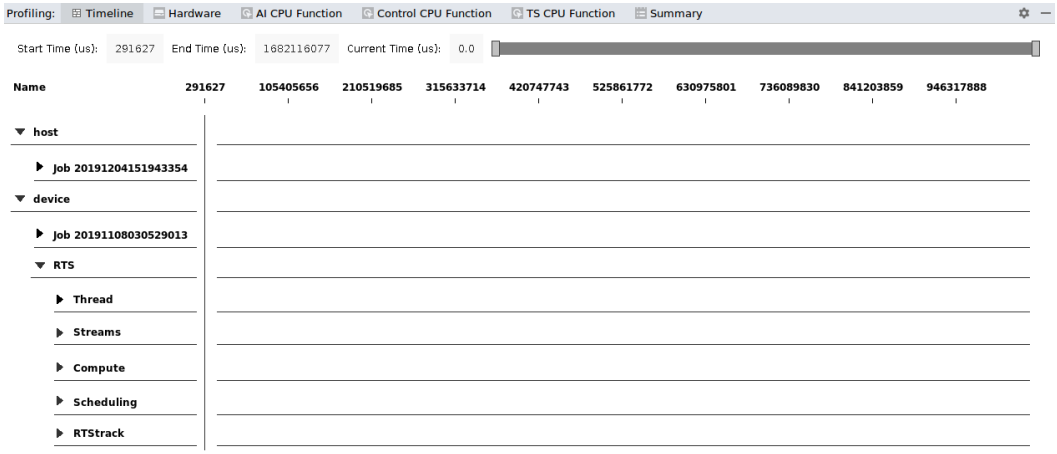
图 9-2 查看 Profiling 数据



Timeline 视图

支持分析Matrix、OME和RTS性能时序信息，如图9-3所示。

图 9-3 Timeline 视图

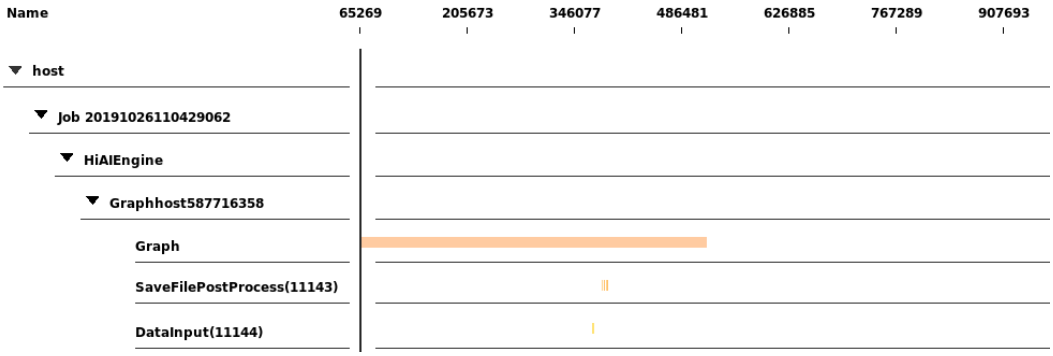


说明

- “Start Time”和“End Time”表示数据展示的时间范围。图中数据展示的时间范围可通过调整右上方的滑块进行变更，也可以通过“Ctrl+鼠标滑轮滚动”进行变更。
- 鼠标放到对应的时间块上，可以查看具体时间段信息；拖动页面右上方时间进度条，界面会重新请求数据，部分模块在该时间段因为没有相关数据而不展示出来。
- Atlas 200开发者板场景，没有Host侧Matrix视图。

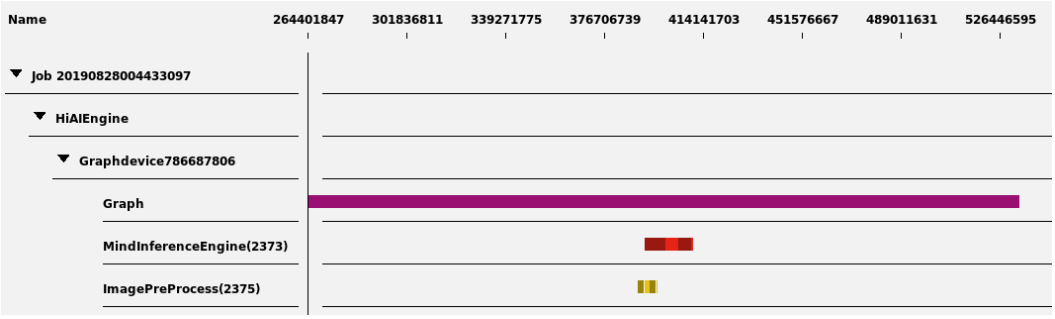
Host侧Matrix：每个Engine运行的起止时间，如图9-4所示。

图 9-4 Host 侧 Matrix 分析



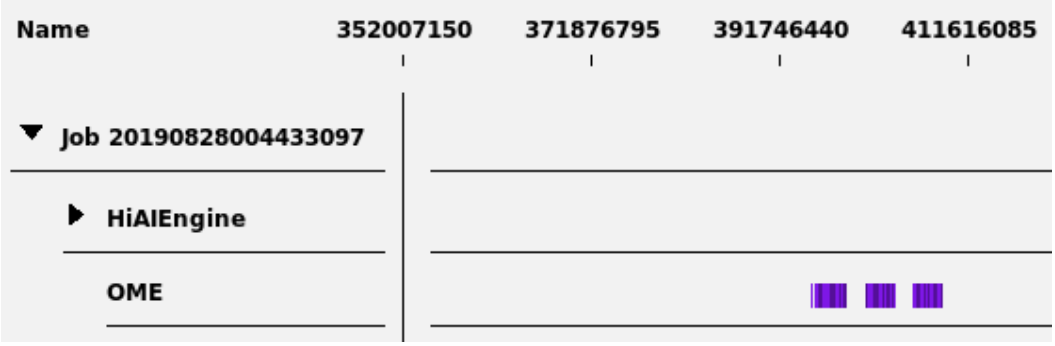
Device侧Matrix：每个Engine运行的起止时间，如图9-5所示。通过Timeline页签可以直观获取推理任务的时间耗时。

图 9-5 Device 侧 Matrix 分析



OME：每个op运行的起止时间和相关信息，如图9-6所示。统计每一个算子的耗时是观察推理任务耗时瓶颈的关键指标。通过Timeline页签，可以直接确认哪一个算子是整个推理任务的耗时瓶颈，并根据定位结果进行进一步优化。

图 9-6 OME 界面



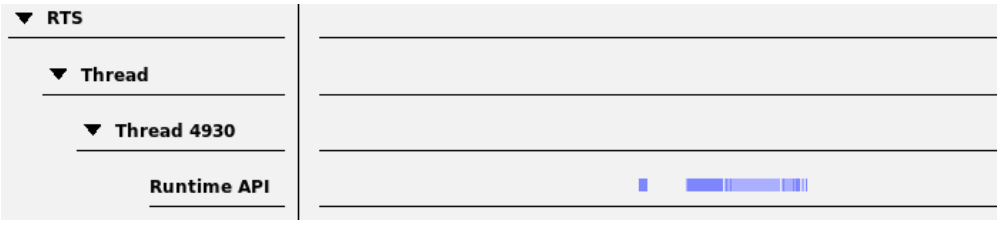
RTS包括如下部分：

说明

鼠标放到对应的时间块上除了展示时间段信息外，compute视图还展示task运行状态pending、waiting、running；stream、scheduling视图展示task的running运行状态。

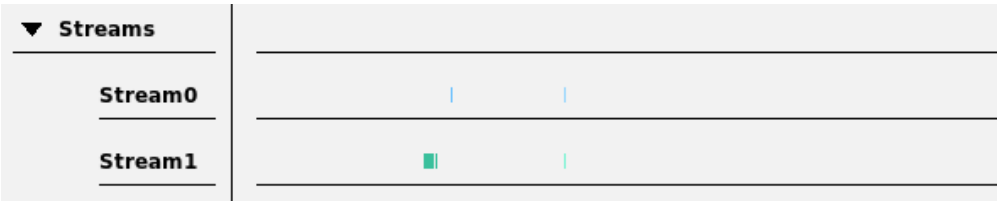
- Thread：每个线程调用runtime API的时序信息，如图9-7所示。

  图 9-7 Thread 分析

- Streams：Stream内的task运行时序信息，包括流间同步信息，如图9-8所示。

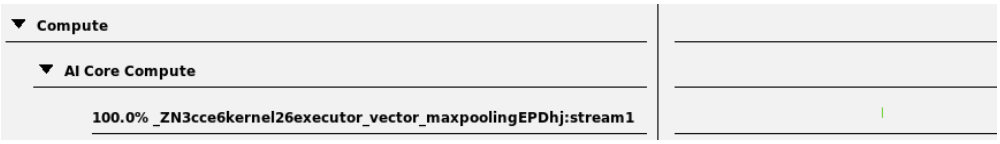
  图 9-8 Streams 分析

- compute：AI Core上kernel函数的运行时序信息，单击每个kernel函数运行的矩形，可以显示kernel函数运行的起止时间和task-based模式采集的AI Core PMU值，如图9-9所示。

  此项指标名称由AI Core kernel函数耗时占比、函数名称以及所属stream ID组成，通过此项指标，用户可以直观获取哪一个kernel函数占用了较多的推理时间，并可以根据此项指标进行kernel函数优化。

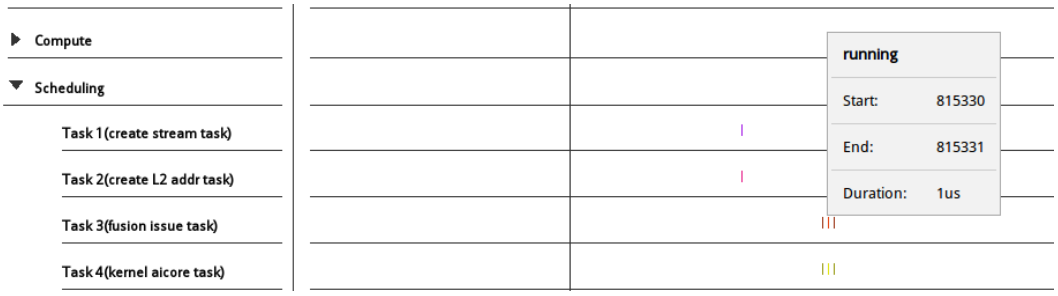
  图 9-9 compute 分析

- Scheduling：Task Scheduler调度的task的运行时序信息，如图9-10所示。用户可以根据在OME数据中观察到的算子耗时，获取算子对应的task ID信息，并在Scheduling指标中查看相应task ID的具体运行信息。

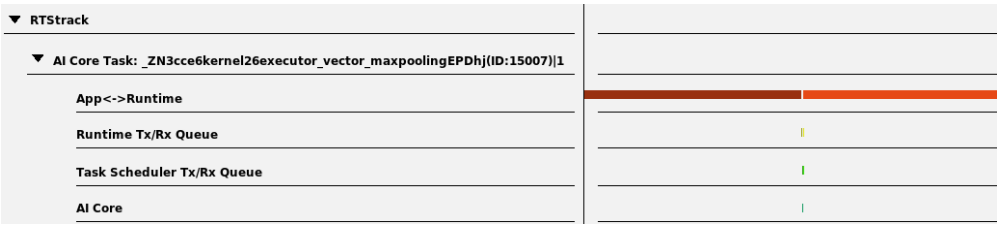
  图中running状态展示该Task的某一个时间块的运行起止时间。

  图 9-10 Scheduling 分析

- RTStrack：显示每个AI Core Task的上下行数据，如图9-11所示。

图 9-11 RTStrack 分析界面



Hardware 视图

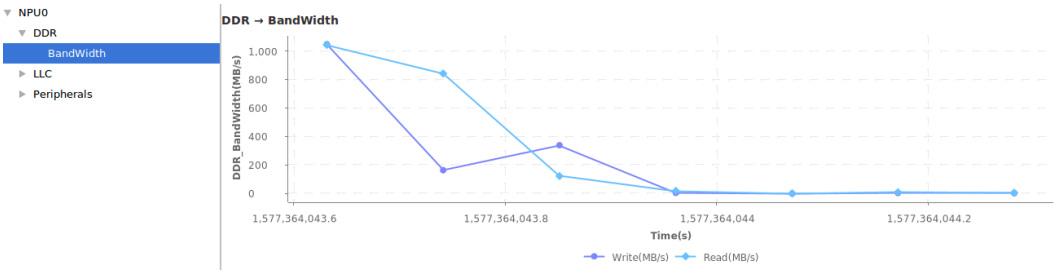
DDR和LLC数据为推理过程中的数据传输带宽信息，过低的DDR和LLC读写带宽速率有可能是算子编写的数据搬运过程存在问题。

外设数据用于观测硬件传输和DVPP处理性能。其中，NIC数据中错误率或者丢包率较高可能意味着Device侧与Host间的网络传输存在问题；通过DVPP指标查看图像处理引擎是否存在过高的利用率，过高的利用率可能意味着此模块存在数据处理瓶颈。

Hardware视图展示DDR、LLC和 Peripheral的性能数据，各个模块信息说明如下：

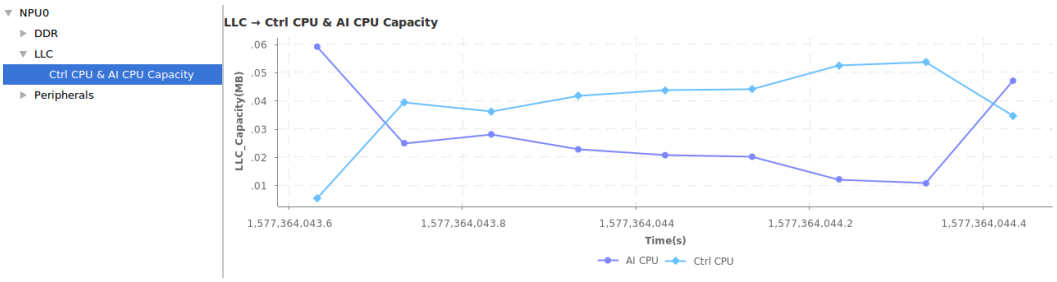
- DDR：采集带宽读写速率。

图 9-12 DDR 分析



- LLC：根据启动Profiling时选择的采集项，展示不同的数据。
  - 采集项选择“LLC capacity”：显示LLC的使用容量信息。
  - 采集项选择“LLC bandwidth”：显示LLC的读写带宽以及命中率。

图 9-13 LLC 分析

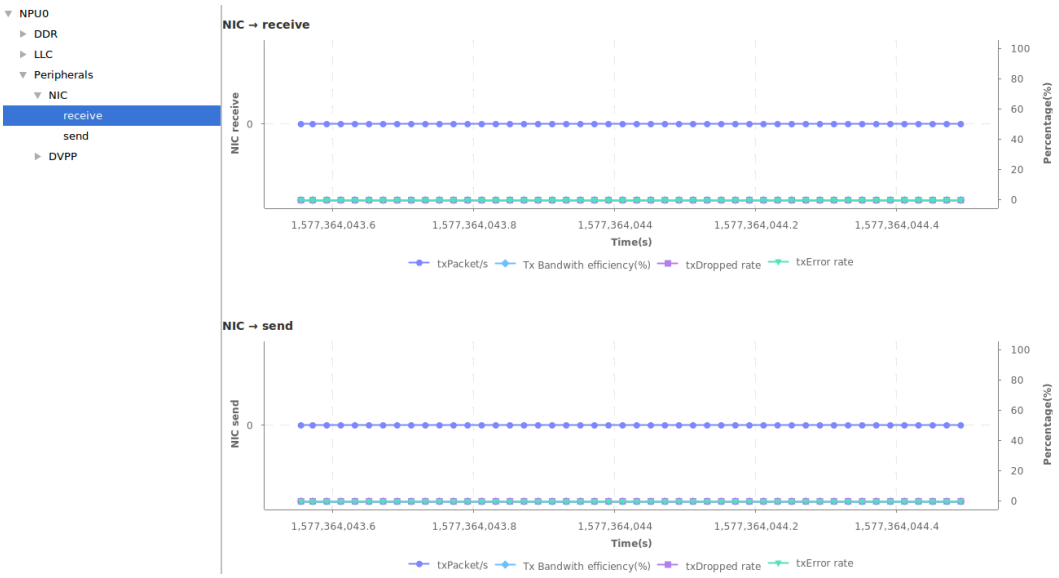


- Peripherals：外设信息，包括DVPP各个engine的任务时间及任务帧数信息，NIC代表每个时间节点上接受和发送网络信息。

  NIC图表各指标代表含义如下：

- Rx Bandwidth efficiency(%)：代表接收包带宽利用率。
- rxError rate：代表接收包错误率。
- rxPacket/s：代表每秒收包速率。
- Tx Bandwidth efficiency(%)：代表发送包带宽利用率。
- txDropped rate：代表发送包丢包率。
- txError rate：代表发送包错误率。
- txPacket/s：代表每秒发送包速率。

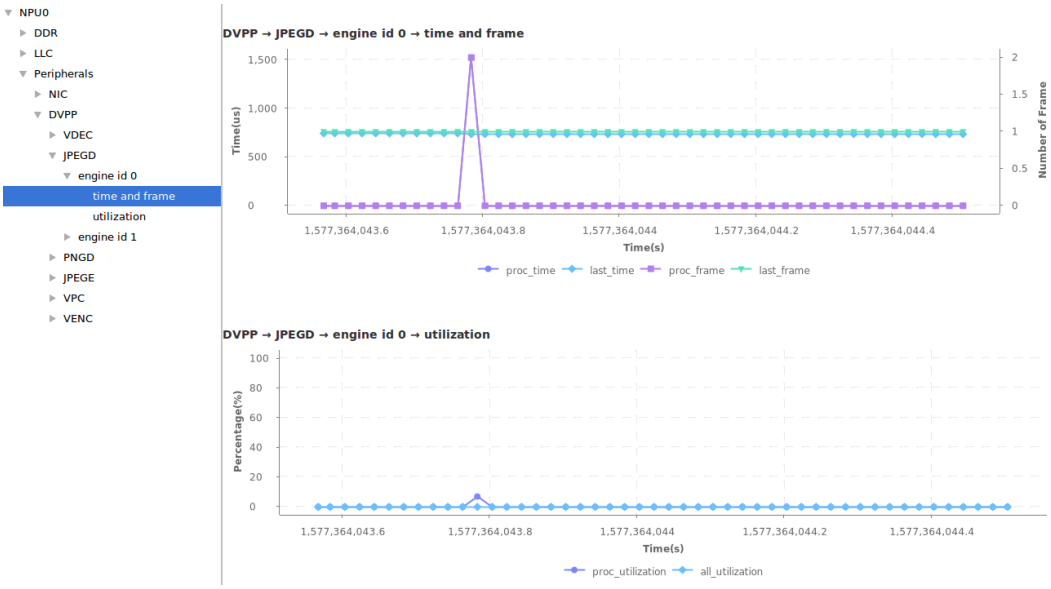
图 9-14 NIC 分析



DVPP图表各指标代表含义如下：

- proc_time ：代表每次采样间隔中引擎使用的时间。
- last_time：代表最后一次任务用的时间。
- proc_frame：代表每次采样间隔中引擎处理的帧数。
- last_frame：代表最后一次任务的帧数。
- proc_utilization：代表时间窗利用率，即时间段累加的处理时间/运行时间。
- all_utilization：代表平均利用率，即累加的处理时间/运行时间。

图 9-15 DVPP 分析



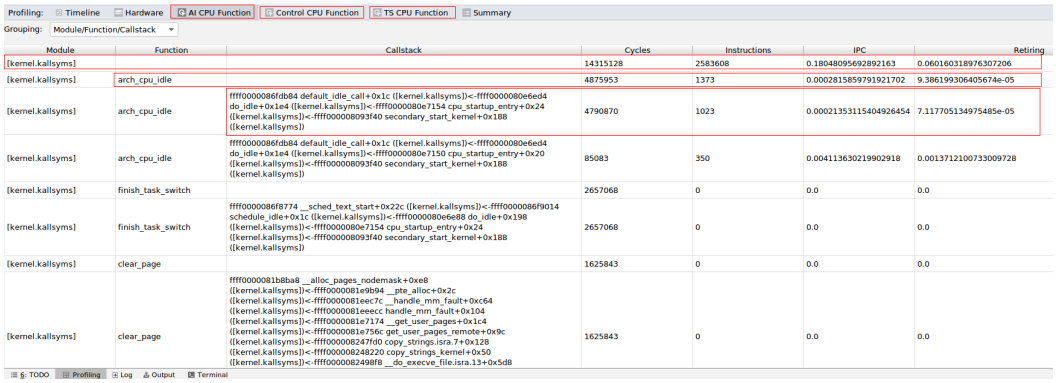
AI CPU Function、Control CPU Function、TS CPU Function 视图

在硬件采集项配置采集AI CPU Function、Control CPU Function、TS CPU Function，在分析界面会生成对应的分析数据。默认按照cycles排序显示采集过程中Al CPU、Control CPU、TS CPU 的热点函数，如图9-16所示。

图中数据部分，第一行红框里Function和Callstack列数据为空，表示该行为Module列对应模块的汇总数据；第二行红框里Callstack列数据为空，表示该行为Function列对应函数的汇总数据；第三行表示该Callstack列对应堆栈的数据。

开发者可以通过CPU Function页签获取在整个推理过程中相应CPU上时间占比较多的热点函数，并根据相应的提示信息，进行热点函数优化，以提升推理性能。

图 9-16 AI CPU Function 视图



单击表头，可以对数据进行排序；单击右上角 图标，可以将数据以.csv格式文件保存到具有写权限的指定位置，针对文件名约束：文件名只能由大小写字母、数字及特殊字符“- _ .”组成，长度为1~255字符。

视图中展示的各列信息说明：

- Module：模块名。
- Function：函数名。
- Callstack：调用堆栈。
- Cycles：对应子项的cycle数。
- Instructions：指令数。
- IPC：指令数耗时占比。
- Retiring：Pipeline输入管道利用率。

Summary 视图

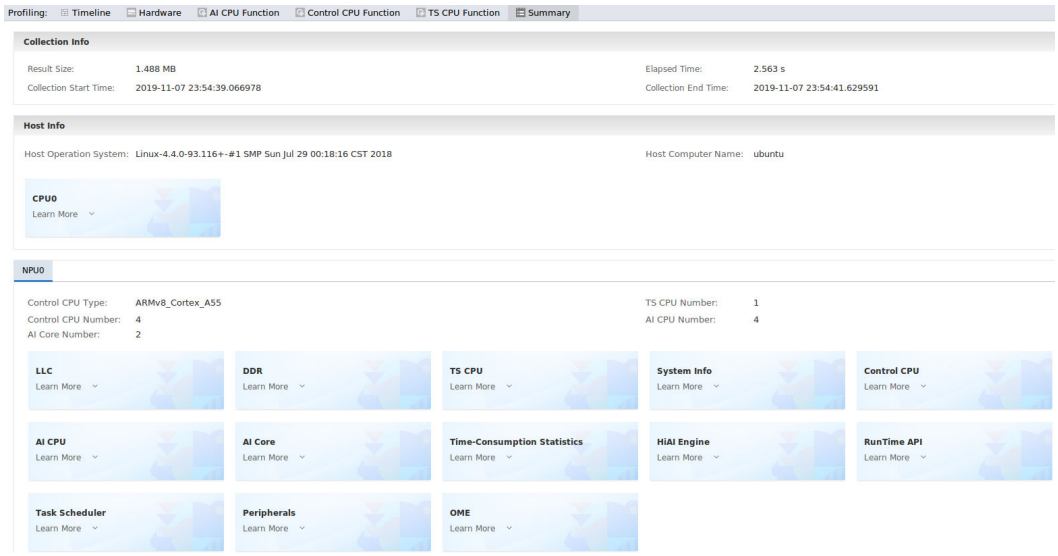
Summary分析视图对采集的性能数据以汇总的形式呈现，主要包括如下内容，如图9-17所示。

- Collection Info：呈现本次采集的性能数据文件大小、性能采集耗时、及性能采集的起止时间。
- Host Info：呈现本次性能数据采集的Host侧基本信息，包括Host侧的操作系统和CPU信息。
- NPU：呈现本次性能数据采集的NPU（Device侧）信息。

说明

- Host Info处支持多CPU信息展示，每次只展示一个CPU的信息。
- 通过单击NPU0 下的某个子模块图标，展示该模块的数据信息。

图 9-17 Summary 视图



Summary分析对采集的性能数据以表格的形式进行汇总呈现，主要包括如下内容。

1. Collection Info：呈现本次采集的性能数据文件大小、性能采集耗时、及性能采集的起止时间，帮助了解整个网络推理任务的时长信息（包含Profiling性能数据传输时间），如图9-18所示。

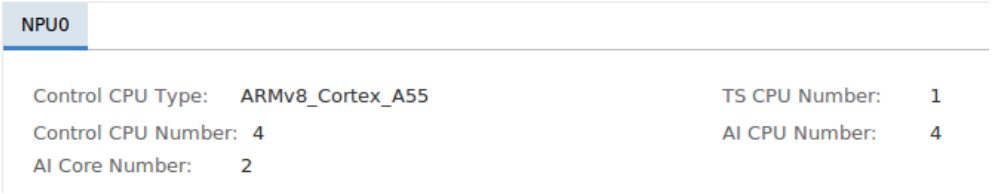
图 9-18 Collection Info 界面

2. Host Info：呈现本次性能数据采集的Host侧基本信息，包括Host侧的操作系统和CPU信息。在开发者板形态，由于Host和Device皆位于miniRC上，所以此处采集到的数据即为miniRC的基本信息，如图9-19所示。

图 9-19 Host Info 界面

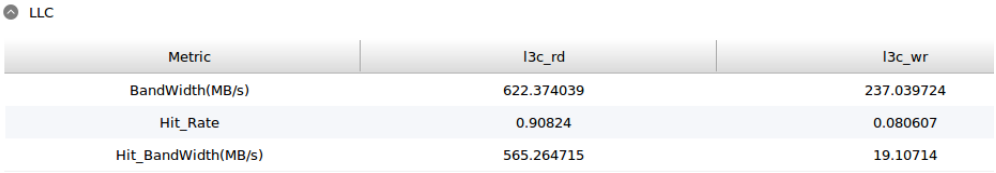
3. Device信息：呈现本次性能数据采集的Device侧信息：AI Core、AI CPU、TS CPU、Ctrl CPU个数，包括内容如图9-20所示。

图 9-20 Device 设备基本信息



4. LLC信息：LLC的读写带宽以及命中率如图9-21所示。
   采集项选择“LLC bandwidth”时显示该项目。

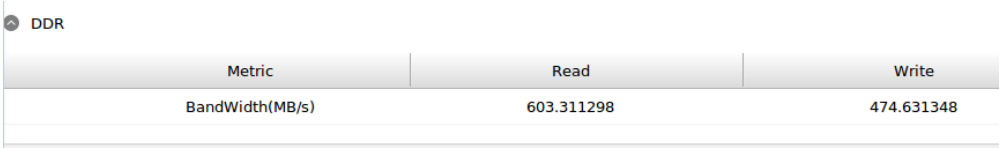
图 9-21 LLC 信息



Metric	l3c_rd	l3c_wr
BandWidth(MB/s)	622.374039	237.039724
Hit_Rate	0.90824	0.080607
Hit_BandWidth(MB/s)	565.264715	19.10714

5. DDR信息：显示整个采集过程中DDR读写带宽的平均值，如图9-22所示。通过此数据可以查看在整个采集过程中，是否存在异常读写带宽（如带宽值过低）。

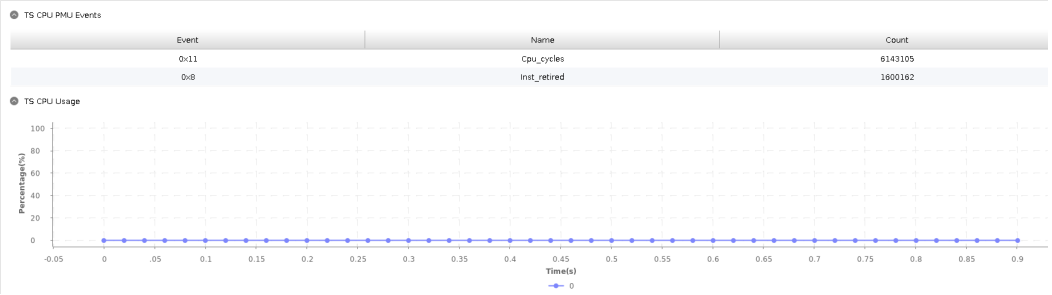
图 9-22 DDR 数据



Metric	Read	Write
BandWidth(MB/s)	603.311298	474.631348

6. TS CPU信息：显示TS CPU usage， 如图9-23所示。
   如果采集时间少于1s，则不会展示TS CPU Usage图。

图 9-23 TS CPU 信息



图中 0 表示TS CPU；横轴数字表示采集点的采集时间；纵轴表示CPU利用率。

7. System Info信息：显示系统内存平均使用情况，如图9-24所示。

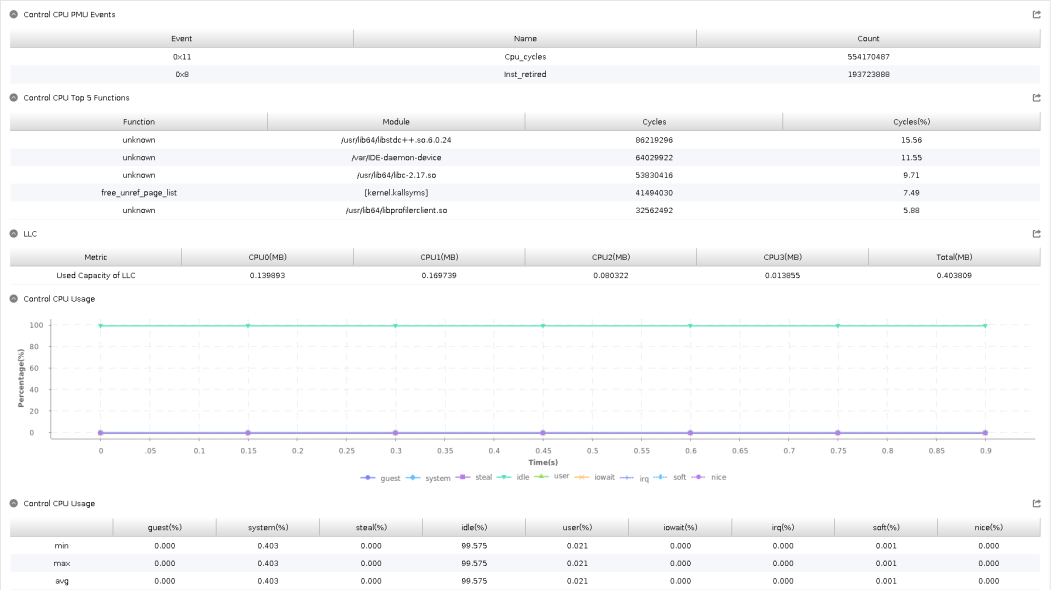
图 9-24 System Info 信息

表 9-2 System Memory 参数说明

参数名	参数含义
MemTotal	总内存。
MemFree	空闲内存。
Buffers	内存缓冲大小。
Cached	高速缓冲存储器使用大小。
Shmem	共享内存。
CommitLimit	虚拟内存限值。
Committed_AS	系统已经分配的内存。
HugePages_Total	分配的HugePages内存页面数目。
HugePages_Free	剩余的HugePages内存页面数目。

8. Control CPU信息：显示采集的Control CPU的PMU事件和热点函数，如图9-25所示。采集项选择“LLC capacity”时显示LLC信息。
   该指标用于获取Control CPU上运行的总task cycles数以及实际执行任务的cycles数（Control CPU PMU Events），当用户认为在Control CPU上有过多性能开销时，可以通过“Control CPU top 5 functions” 获取Control CPU的热点函数，以排查哪些函数导致了性能的开销。

图 9-25 Control CPU 信息



Control CPU PMU Events

Event	Name	Count
0x11	Cpu_cycles	554170487
0x8	Inst_retired	193723988

Control CPU Top 5 Functions

Function	Module	Cycles	Cycles(%)
unknown	/usr/lib64/libstdc++.so.6.0.24	86219296	15.56
unknown	/var/IDE-daemon-device	64029922	11.55
unknown	/usr/lib64/libc-2.17.so	53830416	9.71
free_unref_page_list	[kernel.kallsyms]	41494030	7.49
unknown	/usr/lib64/libprofilerclient.so	32562492	5.88

LLC

Metric	CPU0(MB)	CPU1(MB)	CPU2(MB)	CPU3(MB)	Total(MB)
Used Capacity of LLC	0.139893	0.169739	0.080322	0.013855	0.403809

Control CPU Usage

Control CPU Usage

	guest(%)	system(%)	steal(%)	idle(%)	user(%)	iowait(%)	irq(%)	soft(%)	nice(%)
min	0.000	0.403	0.000	99.575	0.021	0.000	0.000	0.001	0.000
max	0.000	0.403	0.000	99.575	0.021	0.000	0.000	0.001	0.000
avg	0.000	0.403	0.000	99.575	0.021	0.000	0.000	0.001	0.000

表 9-3 Control CPU 参数说明

参数名	参数含义
Control CPU Top5 Functions部分参数说明	
Function	运行在control CPU上的函数名称。
Module	此函数所涉及的模块名称。
Cycles	此函数运行的cycle数。
Cycles(%)	此函数在所有cycle中占比。
Control CPU Usage参数说明	
irq	硬件中断时长占比。
soft	软中断时长占比。
guest	客户模式常规进程占比。
user	用户态进程执行时长占比。
system	内核态进程执行时长占比。
iowait	IO等待状态时长占比。
nice	优先级进程在客户模式下执行的时长占比。
steal	虚拟环境中os消耗占比。
idle	空闲状态时长占比。

9. AI CPU信息：显示采集的AI CPU的PMU事件和热点函数，如图9-26所示。

采集项选择“LLC capacity”时显示LLC信息。

图 9-26 AI CPU 信息

AI CPU PMU Events

Event	Name	Count
0x11	Cpu_cycles	15503794
0x8	Inst_retired	1440455

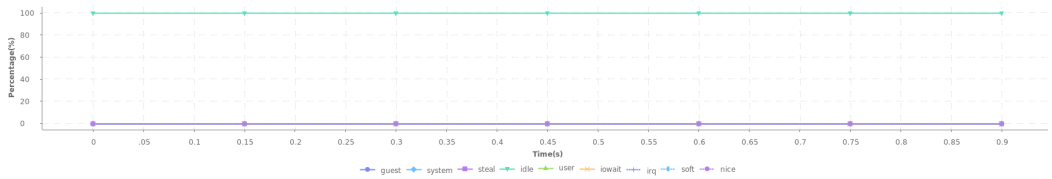
AI CPU Top 5 Functions

Function	Module	Cycles	Cycles(%)
arch_cpu_idle	[kernel.kallsyms]	9986157	64.41
unknown	/usr/lib64/ld-2.17.so	1933014	12.47
do_wp_page	[kernel.kallsyms]	989867	6.38
kmem_cache_free	[kernel.kallsyms]	874864	5.64
el0_da	[kernel.kallsyms]	859946	5.55

LLC

Metric	CPU0(MB)	CPU1(MB)	CPU2(MB)	CPU3(MB)	Total(MB)
Used Capacity of LLC	0.0	0.0	0.0	0.144531	0.144531

AI CPU Usage



AI CPU Usage

	guest(%)	system(%)	steal(%)	idle(%)	user(%)	iowait(%)	irq(%)	soft(%)	nice(%)
min	0.000	0.000	0.000	100.000	0.000	0.000	0.000	0.000	0.000
max	0.000	0.000	0.000	100.000	0.000	0.000	0.000	0.000	0.000
avg	0.000	0.000	0.000	100.000	0.000	0.000	0.000	0.000	0.000

10. AI Core Metrics信息：显示采集AI Core PMU事件计算出的指标数据以及AI Core状态信息，以“Task-based Profiling”采集模式为例，如图9-27所示。

    该指标统计整个Profiling采样周期中各个AI Core类型指令在所有类型指令中的占比以及时间，其中mac ratio，vec ratio，scalar ratio分别代表cube指令利用率、vec指令利用率、scalar指令利用率，是需要重点关注的指标。

    除此以外，在Task-based 采集场景，total_time也是重要指标，该指标表示task的实际硬件运行时间，过长的运行时间可能意味着此task所对应的算子存在设计问题。

图 9-27 AI Core 信息

AI Core Metrics

Task	total_time(ms)	mac_fp16_ratio	mac_int8_ratio	mac_ratio	vec_ratio	scalar_ratio	scalar_ld_ratio	scalar_st_ratio
total	3.423725735294118	0.3898698577803856	0	0.41718573701392564	0.11533537636355419	0.2976794339563604	0.13888174984227983	0.03524935835497111
4458	0.1469176470588235 4	0.4103139013452915	0	0.4461883408071749	0.585832399103139	0.3337303811659193	0.2146160313901345 2	0.00920884048686739 3
4458	0.1455838235294117 7	0.4140731537319312 6	0	0.4502762709981111	0.591199733325252 3	0.3372728466519187 6	0.2170671838540561 8	0.00929321090538097 1
4458	0.1454558823529411 8	0.4144373673036093 6	0	0.4506723283793347 4	0.591719745222929 9	0.3371853199878677	0.2168739257911232 5	0.00930138509756344 2
4459	0.06564117647058823	0.6355408190698091	0	0.6871583475221794	0.24572094273680437	0.4396451294918899 4	0.2706559727575141 5	0.04037100098575141
4459	0.0648014705882353	0.6437762396459775	0	0.696062634744128	0.24890502666515374	0.4452740270055599 6	0.2741631680472 03	0.04089413366617497
4459	0.06491617647058824	0.6426386969621457	0	0.694832702806787	0.24846521532292776	0.4444872346691434 7	0.2736787259588156 5	0.04082187436286614
4460	0.06362352941176 47	0.6556952662721893	0	0.7089497041420119	0.20885724852071005	0.4329927884615384 4	0.2576275887573964 4	0.04176682692307692
4460	0.06367941176470589	0.6551198558958016	0	0.7083275599279479	0.20867396425107385	0.4326128123412313 3	0.257447692947208	0.04173017412590642 6
4460	0.06313823529411765	0.6607350817533889	0	0.7143988447384357	0.21046257045698047	0.4363441561466437	0.2596077700656822	0.04208785577863697 5

AI Core Status

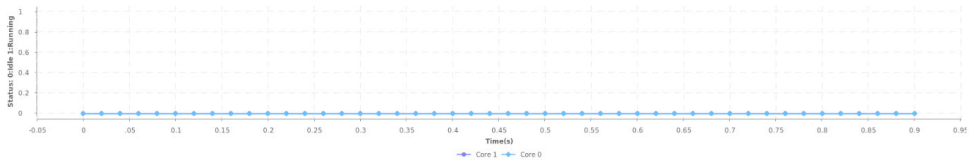


图9-27中，“Status”运行状态视图，展示各个AI Core在采样时间点的running和idle状态；如果是**“Sample-Based Profiling”**采集模式，还会展示“Percentage”利用率视图，该视图展示各个AI Core在采样时间点的利用率。

说明

当前，Mind Studio仅显示以下默认事件包括0xa、0x8、0x9、0xe、0x3a、0x3b、0x4a和0x49，0xa表示执行的cube类型指令的cycle数，0x8表示执行的vector类型指令执行的cycle数，0x9表示执行的scalar类型指令执行的cycle数，0xe表示指的是总指令cycle数，0x3a表示scalar访问UB读请求类型指令执行的cycle数，0x3b表示scalar访问UB写请求类型指令执行的cycle数，0x4a表示cube int类型指令执行的cycle数，0x49表示cube FP类型执行执行的cycle数。Profiling支持的其他事件请参考《Ascend 310 Profiling工具使用指导（命令行方式）》手册内容执行查看。

采集项选择“Task-based Profiling”模式时，统计每个Task对应事件的数据。其中，total行展示的是所有Task数据汇总后再按计算公式输出的各项采集指标，并非视图中其他各行之和。

采集项选择“Sample-based Profiling”模式时，统计每个Core对应事件的数据。

注意：由于Profiling计算的ratio存在包含关系，所以所有事件ratio相加之和不为1。

1、“total_time”通过从PMU_TASK_CYC_CNT寄存器读取task cycles数，根据运行频率计算得出耗时。

2、“mac_fp16_ratio”代表cube fp类型指令的cycle数在所有指令的cycle数中的占用比，通过公式mac_fp16_ratio=1.0*SUM(0x49)/SUM(PMU_TASK_CYC_CNT)获取。

3、“mac_int8_ratio”代表cube int类型指令的cycle数在所有指令的cycle数中的占用比，通过公式mac_int8_ratio=1.0*SUM(0x4a)/SUM(PMU_TASK_CYC_CNT)获取。

4、“vec_ratio”代表vec类型指令的cycle数在所有指令的cycle数中的占用比，通过公式vec_ratio=1.0*SUM(0x8)/SUM(PMU_TASK_CYC_CNT)获取。

5、“mac_ratio”代表cube类型指令（未包含特殊寄存器写请求）的cycle数在所有指令的cycle数中的占用比，通过公式mac_ratio=1.0*SUM(ra)/SUM(PMU_TASK_CYC_CNT)获取。

6、“scalar_ratio”代表scalar类型指令的cycle数在所有指令的cycle数中的占用比，通过公式scalar_ratio=1.0*SUM(0x9)/SUM(PMU_TASK_CYC_CNT)获取。

7、“scalar_ld_ratio”代表scalar访问UB读请求指令的cycle数在所有指令的cycle数中的占用比，通过公式scalar_ld_ratio=1.0*SUM(0x3a)/SUM(PMU_TASK_CYC_CNT)获取。

8、“scalar_st_ratio”代表scalar访问UB写请求指令的cycle数在所有指令的cycle数中的占用比，通过公式scalar_st_ratio=1.0*SUM(0x3b)/SUM(PMU_TASK_CYC_CNT)获取。

11. Time-Consumption Statistics信息：每个AICore Task上下行耗时，如图9-28所示。

图 9-28 Time-Consumption Statistics 信息

AI Core Task

task_name	AiCoreKernel_Task_Scheduler_TxQueue_time(ns)	AiCoreKernel_AI_Core_time(ns)	AiCoreKernel_Task_Scheduler_RxQueue_time(ns)
AI Core Task _Z42executor_conv_pool_c04_resnet_224x224_aipp...	27922.53	155.779	10.52
AI Core Task _Z42executor_conv_pool_c04_resnet_224x224_aipp...	31319.112	154.06	9.165
AI Core Task _Z42executor_conv_pool_c04_resnet_224x224_aipp...	34314.495	153.435	11.562
AI Core Task _Z54executor_conv_filterl1_once_reusen_cut_h_m_t...	27921.853	201.558	8.02
AI Core Task _Z54executor_conv_filterl1_once_reusen_cut_h_m_t...	31312.914	204.369	8.541
AI Core Task _Z54executor_conv_filterl1_once_reusen_cut_h_m_t...	34307.828	206.923	6.509
AI Core Task _Z54executor_conv_filterl1_once_reusen_cut_h_m_t...	27972.058	93.434	6.353
AI Core Task _Z54executor_conv_filterl1_once_reusen_cut_h_m_t...	31362.44	97.548	6.249
AI Core Task _Z54executor_conv_filterl1_once_reusen_cut_h_m_t...	34361.679	94.527	7.134
AI Core Task _Z58executor_conv_filterl1_once_reusen_1x1_cut_h...	27953.827	69.631	6.457
AI Core Task _Z58executor_conv_filterl1_once_reusen_1x1_cut_h...	31346.396	73.901	6.457
AI Core Task _Z58executor_conv_filterl1_once_reusen_1x1_cut_h...	34342.666	71.559	5.832

表 9-4 Time-Consumption Statistics 页面参数说明

参数名	参数含义
task_name	kernel函数名称。

参数名	参数含义
AiCoreKernel_Task_Scheduler_TxQueue_time	TS调度到task真正在AI Core上运行的时间。
AiCoreKernel_AI_Core_time	task在AI Core上实际运行时间。
AiCoreKernel_Task_Scheduler_RxQueue_time	Task在AI Core结束后在TS上调度的时间。

12. Matrix信息：显示每个Engine运行的时间耗时，如图9-29所示。重点关注“Engine Total Time Consumption”中的推理Engine（如MindInferenceEngine）的实际推理时长，该指标值代表的时间越长，需要关注网络设计或算子本身编写是否存在问题。

说明

Matrix数据中graph name是由graph+device/host+graphID组成，Engine Name是由Engine Name（threadID）组成。

图 9-29 Matrix 信息

Graph Information

Job ID	Graph Name	Start Time(s)	End Time(s)	Time Consumption(ms)
20190828004433097	graphhost786687806	1566924273.094513443	1566924273.79988...	705.37596
20190828004433097	graphdevice786687806	1566924273.29024288	1566924273.78689...	496.647865

Engine Total Time Consumption

Job ID	Graph Name	Engine Name	Time Consumption(ms)
20190828004433097	graphdevice786687806	ImagePreProcess(2...	6.750884
20190828004433097	graphdevice786687806	MindInferenceEngi...	18.510521
20190828004433097	graphhost786687806	DataInput(956)	7.83053
20190828004433097	graphhost786687806	SaveFilePostProces...	0.912399

Engine Concrete Information

Job ID	Graph Name	Engine Name	Start Time(s)	End Time(s)	Time Consumption(ms)
20190828004433097	graphhost786687806	SaveFilePostProcess(954)	1566924273.654715439	1566924273.654983561	0.268122
20190828004433097	graphhost786687806	SaveFilePostProcess(954)	1566924273.659547645	1566924273.659779962	0.232317
20190828004433097	graphhost786687806	SaveFilePostProcess(954)	1566924273.664438295	1566924273.66466937	0.231075
20190828004433097	graphhost786687806	SaveFilePostProcess(954)	1566924273.665214121	1566924273.665395006	0.180885
20190828004433097	graphhost786687806	DataInput(956)	1566924273.639068346	1566924273.646898876	7.83053
20190828004433097	graphdevice786687806	MindInferenceEngine(2373)	1566924273.644810536	1566924273.652700797	7.890261

表 9-5 Matrix 参数说明

参数名	参数含义
Job ID	推理任务分配的job ID。
Graph Name	推理任务分配Graph名称。
Engine Name	推理引擎的名称。
Start Time	Graph或者Engine当次调度的开始时间。
End Time	Graph或者Engine当次调度的结束时间。

参数名	参数含义
Time Consumption	Graph或者engine的耗时统计。

13. RunTime API和Task Scheduler信息：显示Runtime的API调用信息和Task Scheduler的任务调度信息，如图9-30和图9-31所示。

    Runtime API指标统计各类Runtime API的调用次数以及API的调用时长，过多的调用次数以及调用时长表示算子编写调度逻辑可能存在问题；Task Scheduler任务调度信息显示了每个task的调度时间、运行时间、task类型以及归属的API信息，若存在某些task存在耗时过长，则需要根据task的相应信息定位原因。

图 9-30 Runtime 的 API 调用信息信息

RunTime API

Name	StreamID	Time(%)	Time(ns)	Calls	Avg(ns)	Min(ns)	Max(ns)
KernelLaunch	1	12.0	9635781	57	169048	29115	7855521
KernelFusionStart	1	0.0	27344	1	27344	27344	27344
KernelFusionEnd	1	0.0	8281	1	8281	8281	8281
StreamCreate	2	0.8	657291	2	328645	280729	376562
StreamDestroy	1	0.0	32501	2	16250	7032	25469
StreamSynchronize	2	12.4	9960053	3	3320017	3255052	3447240
DevMalloc		9.3	7443812	116	64170	45833	729427
DevFree		4.3999999999999995	3569063	116	30767	18646	784322
HostMalloc		1.7999999999999998	1470253	424	3467	2240	65261
HostFree		0.7000000000000001	533273	424	1257	989	4115
ManagedMemAlloc		0.2	158698	57	2784	2135	30208
ManagedMemFree		0.1	100208	57	1758	1562	3646
MemCpy		44.800000000000004	35960300	172	209071	1927	35327656

图 9-31 Task Scheduler 的任务调度信息

Task Scheduler

Time(%)	Time(ns)	Count	Avg(ns)	Min(ns)	Max(ns)	Waiting(ns)	Running(ns)	Pending(ns)	Type	API	Task	Stream	NPU
13.3585	1355196	1	1355196	1355196	1355196	3958	1355196	0	model execute task	ModelExecute	15028	2	0
13.1094	1329928	1	1329928	1329928	1329928	6510	1329928	0	model execute task	ModelExecute	15033	2	0
13.0925	1328210	1	1328210	1328210	1328210	4427	1328210	0	model execute task	ModelExecute	15036	2	0
6.2228	631290	3	210430	208016	211715	0	631290	0	kernel AI core task	KernelLaunch	4528	1	0
4.6657	473323	3	157774	156975	158227	0	473323	0	kernel AI core task	KernelLaunch	4527	1	0
4.4074	447124	3	149041	127080	168537	0	447124	0	kernel AI core task	KernelLaunch	4547	1	0

表 9-6 RTS 页面部分参数说明

参数名	参数含义
Runtime API	
Name	调用的API名称。
StreamID	API运行的stream ID。
Calls	API被调用的次数。
Task Scheduler	
Count	对应的Task被执行的个数。
Waiting	对应Task总计waiting时间。

参数名	参数含义
Running	对应Task总计running时间。表示Task的实际运行时间，过长的Task运行时间可能意味着算子实现存在问题。
Pending	对应Task总计pending时间。
Type	对应Task 的类型。
API	对应的API名称。
Task	对应的Task名称。
Stream	Task所对应stream。

14. Peripherals信息：显示DVPP各个engine的任务时间、任务帧数及利用率信息和NIC的网络发送和接受的状态信息，如图9-32所示。

说明

pcie卡形态Device无NIC数据，采集时请关闭NIC采集开关，因为即使此时开启NIC采集开关，也不会展示图9-32中NIC表格信息。

图 9-32 Peripherals 信息

DVPP

Engine Type	Engine Id	All Time(us)	All Frame	All Utilization
VDEC	1	0.0	0.0	0.0%
VDEC	0	0.0	0.0	0.0%
JPEGD	1	1526.0	2.0	0.2%
JPEGD	0	759.0	1.0	0.1%
PNGD	5	0.0	0.0	0.0%
PNGD	4	0.0	0.0	0.0%
PNGD	3	0.0	0.0	0.0%
PNGD	2	0.0	0.0	0.0%
PNGD	1	0.0	0.0	0.0%
PNGD	0	0.0	0.0	0.0%
JPEGE	0	0.0	0.0	0.0%
VPC	3	206.0	1.0	0.0%
VPC	2	199.0	1.0	0.0%
VPC	1	201.0	1.0	0.0%
VPC	0	0.0	0.0	0.0%
VENC	0	0.0	0.0	0.0%

NIC

Duration(s)	Bandwidth(Mb)	Rx Bandwidth efficiency(%)	rxPacket/s	rxError rate	rxDropped rate	Tx Bandwidth efficiency(%)	txPacket/s	txError rate	txDropped rate
0.939	1000	0.0	0.0	0%	0%	0.0	0.0	0%	0%

15. OME信息：模型的数据输入、推理、数据输出耗时；每个op运行的详细信息，包括Op name、Op type、运行时间和内存使用情况等，如图9-33所示。

Model Statistic表格展示了每次推理数据输入时间、实际推理时间、推理结果数据输出，如果推理时间超过预期，则需要分析网络设计或算子结构是否存在问题；Op Statistic表格展示了每一层算子的运行时间、Memory信息以及算子所对应的Task信息，用户可以对结果数据进行排序，查看运行时间长的算子，依此评估算子是否为性能的瓶颈。

说明

当用户在进行Matrix代码开发时，如果在代码中新建了用于调用推理Engine的线程，那么在编译运行工程后，再执行Profiling采集OME数据时，由于系统内部的线程与Engine无法匹配，会导致采集结果中的Model Statistic表格无法展示Engine Name数据。

图 9-33 OME 信息

Model Statistic

Model Name	Model ID	Data Input Start Time(s)	Data Input End Time(s)	Inference Start Time(s)	Inference End Time(s)	Data Output Start Time(s)	Data Output End Time(s)	Engine Name
ResNet-50	1	1566924273.648051838	1566924273.648129495	1566924273.648129599	1566924273.651754547	1566924273.65176314	1566924273.651812567	MindInferenceEngine(2373)
ResNet-50	1	1566924273.653717515	1566924273.653752932	1566924273.653753245	1566924273.657105276	1566924273.657111161	1566924273.657148661	MindInferenceEngine(2373)
ResNet-50	1	1566924273.658629234	1566924273.658657255	1566924273.658657411	1566924273.662004495	1566924273.662010276	1566924273.662045536	MindInferenceEngine(2373)

Op Statistic

Op Names	Op Num	Model Name	Model ID	Stream ID	Task Start To Schedu...	Task Complete(s)	Time Consumption(ms)	Memory:input	Memory:output	Memory:weight	Memory:workspace	Memory:total	Task Num	Task IDs
conv1,pool1	2	ResNet-50	1	1	1566924273.6485233...	1566924273.64868...	0.166403	1680928	2007104	100352	0	3788384	1	598
res2a_branch2a	1	ResNet-50	1	1	1566924273.6485439...	1566924273.64871...	0.166872	401440	401440	8192	0	811072	1	599
res2a_branch2b	1	ResNet-50	1	1	1566924273.6487349...	1566924273.64881...	0.078539	401440	401440	73728	0	876608	1	600
res2a_branch2c	1	ResNet-50	1	1	1566924273.6487542...	1566924273.64885...	0.101818	401440	1605664	32768	0	2039872	1	601
res2a_branch1,res2a	2	ResNet-50	1	1	1566924273.6488390...	1566924273.64889...	0.054371	3612768	3211328	32768	0	6856864	1	602
res2b_branch2a	1	ResNet-50	1	1	1566924273.64887551	1566924273.64892...	0.054162	1605664	401440	32768	0	2039872	1	603
res2b_branch2b	1	ResNet-50	1	1	1566924273.6489164...	1566924273.64899...	0.077391	401440	401440	73728	0	876608	1	604
res2b_branch2c,res...	2	ResNet-50	1	1	1566924273.6489480...	1566924273.64903...	0.090152	3612768	3211328	32768	0	6856864	1	605
res2c_branch2a	1	ResNet-50	1	1	1566924273.6490127...	1566924273.64907...	0.065203	1605664	401440	32768	0	2039872	1	606
res2c_branch2b	1	ResNet-50	1	1	1566924273.6490611...	1566924273.64914...	0.078952	401440	401440	73728	0	876608	1	607
res2c_branch2c,res2c	2	ResNet-50	1	1	1566924273.6490972...	1566924273.64918...	0.085411	3612768	3211328	32768	0	6856864	1	608
res3a_branch2a	1	ResNet-50	1	1	1566924273.6491585...	1566924273.64922...	0.064943	1605664	200736	65536	0	1871936	1	609
res3a_branch2b	1	ResNet-50	1	1	1566924273.6492066...	1566924273.64926...	0.055723	200736	200736	294912	0	696384	1	610
res3a_branch2c	1	ResNet-50	1	1	1566924273.6492475...	1566924273.64930...	0.05463	200736	802848	131072	0	1134656	1	611

表 9-7 OME 页面部分参数说明

参数名	参数含义
Model Statistic	
Model Name/ID	模型的名称及该模型在系统运行过程中ID号。
Data Input Start/End Time	开始处理模型输入数据及处理完毕模型输入数据时间(从用户的内存拷贝到DDR上)。
Inference Start/End Time	开始模型推理及模型推理结束时间。
Data Output Start/End Time	开始处理推理结果及处理完毕推理结果时间(将推理结果数据从DDR拷贝到用户的内存中以及用户注册回调函数进行后处理操作的时间)，每次推理的实际耗时。
Op Statistic	
Op Name/Num	算子名称及该算子个数。
Time Consumption	算子的推理耗时。通过此项指标可以挖掘出单次推理的推理耗时瓶颈。
Task Start To Schedule	算子对应的Task开始调度的时间。
Task Complete	算子对应的Task完成调度的时间。 如果需要查看算子在硬件上实际的运行时间，需要查看AI Core Metrics指标中所对应的Task时间。
Memory:input	输入Tensor内存大小。
Memory:output	输出Tensor内存大小。
Memory:weight	权值内存大小。
Memory:workspace	workspace内存大小。

参数名	参数含义
Memory:total	总内存，input、output、weight、workspace 4个数据相加。
Task Num	融合算子需要执行的Task任务个数。
Task IDs	Runtime分配的Task ID。

9.3 参考

9.3.1 Profiling 数据文件说明

本节描述Mind Studio页面展示的Profiling数据对应的存储文件关系。

说明

本节描述的文件名中：

- *表示Device ID。
- ~/路径表示工程文件路径。

Timeline视图对应的Profiling数据文件如**表9-8**所示。

表 9-8 Timeline 视图对应数据文件

视图项目	数据文件
HiEngine	~/profiling/timeline_diagram.json
OME	~/profiling/timeline_diagram.json
Thread	~/profiling/timeline_diagram.json
Streams	~/profiling/timeline_diagram.json
RTStrack	~/profiling/timeline_rtstrack.json
Compute	~/profiling/timeline_diagram.json
Scheduling	~/profiling/timeline_diagram.json
Host	~/profiling/timeline_host.json

Hardware视图对应的Profiling数据文件如**表9-9**所示。

表 9-9 Hardware 视图对应数据文件

视图项目	数据文件
DDR	~/profiling/timeline_ddr.json

视图项目			数据文件
LLC			“LLC Profiling”采用“LLC Capacity”模式时，对应的数据文件为：~/profiling/timeline_llc_aicpu.json。 “LLC Profiling”采用“LLC BandWidth”模式时，对应的数据文件为：~/profiling/timeline_llc_read.json和~/profiling/timeline_llc_write.json。
peripherals	NIC	-	~/profiling/timeline_nic.json
	DVPP	VDEC	~/profiling/timeline_dvpp_VDEC_*.json
		JPEGD	~/profiling/timeline_dvpp_JPEGD_*.json
		PNGD	~/profiling/timeline_dvpp_PNGD_*.json
		JPEGE	~/profiling/timeline_dvpp_JPEGE_*.json
		VPC	~/profiling/timeline_dvpp_VPC_*.json
		VENC	~/profiling/timeline_dvpp_VENC_*.json

AI CPU Function、**Control CPU Function**、**TS CPU Function视图**对应的Profiling数据文件如**表9-10**所示。

表 9-10 Function 视图对应数据文件

视图项目	module数据文件	function数据文件	callstack数据文件
AI CPU Function	~/profiling/Function/AI_CPU_Function/module.json	~/profiling/Function/AI_CPU_Function/{function_number}/function.json	~/profiling/Function/AI_CPU_Function/{function_number}/{callstack_number}/callstack.json
Control CPU Function	~/profiling/Function/Control_CPU_Function/module.json	~/profiling/Function/Control_CPU_Function/{function_number}/function.json	~/profiling/Function/Control_CPU_Function/{function_number}/{callstack_number}/callstack.json
TS CPU Function	~/profiling/Function/TS_CPU_Function/module.json	~/profiling/Function/TS_CPU_Function/{function_number}/function.json	~/profiling/Function/TS_CPU_Function/{function_number}/{callstack_number}/callstack.json

Summary视图对应的Profiling数据文件如**表9-11**所示。

表 9-11 Summary 视图对应数据文件

视图项目		数据文件
Collection Info		~/profiling/summary_collection.json
Host Info		~/profiling/summary_host.json
DDR		~/profiling/summary_ddr_*.json
Control CPU	Control CPU PMU Events	~/profiling/summary_control_cpu_events_*.json
	Control CPU Top 5 Functions	~/profiling/summary_control_cpu_top_functions_*.json
	LLC	~/profiling/summary_control_cpu_llc_*.json
	Control CPU Usage	~/profiling/summary_control_cpu_usage_*.json ~/profiling/summary_control_cpu_usage_table_*.json
AI CPU	AI CPU PMU Events	~/profiling/summary_ai_cpu_events_*.json
	AI CPU Top 2 Functions	~/profiling/summary_ai_cpu_top_functions_*.json
	LLC	~/profiling/summary_ai_cpu_llc_*.json
	AI CPU Usage	~/profiling/summary_ai_cpu_usage_*.json ~/profiling/summary_ai_cpu_usage_table_*.json
TS CPU	TS CPU PMU Events	~/profiling/summary_ts_cpu_events_*.json
	TS CPU Usage	~/profiling/summary_ts_cpu_usage_*.json
AI Core	AI Core Metrics	~/profiling/summary_ai_core_events_*.json
LLC		~/profiling/summary_llc_*.json
HiAI Engine	Graph Information	~/profiling/summary_engine_graph_*.json
	Engine Total Time Consumption	~/profiling/summary_engine_consumption_*.json
	Engine Concrete Information	~/profiling/summary_engine_concrete_*.json
OME	Model Statistic	~/profiling/summary_ome_model_*.json

视图项目		数据文件
	Op Statistic	~/profiling/summary_ome_op_*.json
Runtime API		~/profiling/summary_runtime_api_*.json
Task Scheduler		~/profiling/summary_task_scheduler_*.json
Peripherals	NIC	~/profiling/summary_peripherals_nic_*.json
	DVPP	~/profiling/summary_peripherals_dvpp_*.json
Time-consumption Statistics	AI Core Task	~/profiling/summary_rtstrack_ai_core_task_*.json
System Info	System Memory Info	~/profiling/summary_memory_*.json

9.3.2 脚本清单

表9-12中脚本的执行方法如下：

1. 以Mind Studio安装用户登录服务器。
2. 切换到“~/.mindstudio/huawei/ddk/1.3*.T*.B***/ddk/toolchains/profiler/analysis/msvp/host/bin64”路径下。

   说明

   - “~/”是Mind Studio安装用户家目录路径。
   - “1.3*.T*.B***”为DDK版本信息，根据实际版本情况选择。
   - 执行以下脚本命令时，不支持以下特殊字符：[';*?`!#$%^&+=<>{}]|"
3. 执行表9-12中的命令调用工具。

表 9-12 脚本列表

脚本	作用	参数及示例
get_env_info.pyc 校验文件夹或删除文件/文件夹	校验指定工程采集结果的文件夹是否符合导入规范。	--verify/-v <指定工程采集结果的文件夹> **示例：** python get_env_info.pyc -v /home/ascend/.mindstudio/profiler/data/profiling_out **说明** “~/.mindstudio/profiler/data/profiling_out”为Mind Studio界面执行Profiling采集的工程结果路径。

脚本	作用	参数及示例
	删除指定的文件或文件夹。	--remove/-r <*指定要删除的采集结果文件*> **示例：**python get_env_info.pyc -r /home/ascend/info.xml **说明** 仅支持删除“~/.mindstudio/profiler”目录下的采集结果文件。
	查看帮助信息	-h/--help
get_msvp_function.pyc 根据指定的project、Module名称、Function名称获取Function页签的数据	指定工程采集结果的文件夹，获取Function页签下所有数据类型的数据	● --project <*指定工程采集结果的文件夹*>：必选参数 ● --target <*指定数据类型*>：可选值包括total、core、thread、module、function、callstack、export、class、classmethod、exportfunc、exportmodule、exportcore、exporttid ● --deviceid=<*指定设备id*> ● --type <*指定cpu类型*>：可选值包括ctrlcpu、aicpu ● --order/-o <*指定按照什么指标来排序*>：可选值是cycles ● --core/-c <*指定cpu核数*>：通过core来过滤结果 ● --pid/-p <*指定pid*>：通过进程ID进行过滤结果 ● --tid/-t <*指定tid*>：通过线程ID进行过滤结果 ● --sort/-s <*指定排序*>：可选值包括升序ASC、降序DESC ● --limit/-l <*指定限制结果条数*> ● --export/-e：添加此选项表示要导出结果 **示例：**python get_msvp_function.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --target total --deviceid=0 --type=ctrlcpu -o cycles -c 1 -p 1 -t 1 -s DESC -l 1 -e
	指定工程采集结果的文件夹和Module名称，获取Function页签下数据类型为function的数据	--module/-m <*指定Module名称*> **示例：**python get_msvp_function.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --target function -m /usr/lib64/libc-2.17.so --type=ctrlcpu --deviceid=0

脚本	作用	参数及示例
	指定工程采集结果的文件夹和Function名称，获取Function页签下数据类型为callstack的数据	--function/-f <*指定Function名称*> **示例：**python get_msvp_function.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --target callstack -f _dl_relocate_object --type=ctrlcpu --deviceid=0
	查看帮助信息	-h/--help
get_msvp_info.pyc 获取指定工程的Summary页签的数据	获取Summary页签指定CPU的CPU Usage信息	● --project<*指定工程采集结果的文件夹*>：必选参数 ● --deviceid=<*指定设备id*> ● --cpuusage：获取特定类型的CPU的CPU Usage数据，与“--type”一起使用 ● --type <*指定CPU类型*>：可选值包括aicpu、ctrlcpu ● --start_time=<*指定开始时间*>：缩放时指定开始时间，若采集时间为10s，指定“--start_time=0”，表示从0s开始。 ● --end_time=<*指定结束时间*>：缩放时指定结束时间，若采集时间为10s，指定“--end_time=6”，表示到6s结束。 ● --number <*指定页面最多显示的数据条数*> **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --cpuusage --type=ctrlcpu --start_time=0 --end_time=6 --number 1000
	获取Summary页签Collection Info信息	--collection_info **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --type=aicpu --collection_info
	获取Summary页签Host Info信息	--host_info **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --type=aicpu --host_info
	获取设备id	--msvp_device **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --msvp_device

脚本	作用	参数及示例
	获取指定设备的设备信息	--msvp_deviceinfo **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --msvp_deviceinfo --deviceid=0
	获取Runtime API数据	--runtime_api **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --runtime_api --deviceid=0
	获取Framework的算子数据	● --framework_op ● --page_length <*指定每页显示多少条数据*> ● --page_index <*指定要获取第几页的数据*> ● --sort_column <*指定要排序的列名*>：可选参数，列包括op_names、fusion_op_nums、modelname、model_id、stream_id、op_start、op_end、op_en、memory_input、memory_output、memory_weight、memory_workspace、memory_total、task_num、task_ids ● --sort_type <*指定按照什么方式来排序*>：可选参数，排查方式包括desc（降序）和asc（升序） **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --framework_op --page_length 50 --page_index 1 --sort_column op_names --sort_type desc

脚本	作用	参数及示例
	获取Framework的模型数据	● --framework_model ● --page_length <*指定每页显示多少条数据*> ● --page_index <*指定要获取第几页的数据*> ● --sort_column <指定要排序的列名>：可选参数，列包括modelname、model_id、input_start、input_end、infer_start、infer_end、output_start、output_end、thread_id ● --sort_type <*指定按照什么方式来排序*>：可选参数，排查方式包括desc（降序）和asc（升序） **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --framework_model --page_length 50 --page_index 1 --sort_column modelname --sort_type desc
	获取CCE数据	--cce **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --cce
	获取Task Scheduler数据	● --task_scheduler ● --page_length <*指定每页显示多少条数据*> ● --page_index <*指定要获取第几页的数据*> ● --sort_column <指定要排序的列名>：可选参数，列包括TimeRatio、Time、Count、Avg、Min、Max、Waiting、Running、Pending、Type、API、taskID、streamID ● --sort_type <*指定按照什么方式来排序*>：可选参数，排查方式包括desc（降序）和asc（升序） **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --type=aicpu --task_scheduler --page_length 100 --page_index 1 --sort_column TimeRatio --sort_type desc

脚本	作用	参数及示例
	获取Control CPU PMU Events数据	--control_cpu_pmu_events **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --control_cpu_pmu_events
	获取Control CPU Top 5 Functions	--control_cpu_top_functions **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --control_cpu_top_functions
	获取TS CPU PMU Events数据	--ts_cpu_pmu_events **示例：** python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --ts_cpu_pmu_events
	获取TS CPU Top 5 Functions数据	--ts_cpu_top_functions **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --ts_cpu_top_functions
	获取AI CPU PMU Events数据	--ai_cpu_pmu_events **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --ai_cpu_pmu_events
	获取AI CPU Top 5 Functions	--ai_cpu_functions **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --ai_cpu_functions

脚本	作用	参数及示例
	获取AI core CPU PMU Events数据	● --ai_core_pmu_events ● --page_length <*指定每页显示多少条数据*> ● --page_index <*指定要获取第几页的数据*> ● --sort_column <指定要排序的列名>：可选参数，列包括total_time、mac_fp16_ratio、mac_int8_ratio、mac_ratio、vec_ratio、scalar_ratio、scalar_ld_ratio、scalar_st_ratio ● --sort_type<*指定按照什么方式来排序*>：可选参数，排查方式包括desc（降序）和asc（升序） **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --ai_core_pmu_events --page_length 50 --page_index 1 --sort_column total_time --sort_type desc
	获取指定事件类型的AI core CPU PMU Events数据	--event_type <指定事件类型>：可选值包括“0”和“5”，“0”表示“AI Core”，“5”表示“DMA” **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --type=aicpu --event_type=0 --ai_core_pmu_events
	获取nic外设数据	--nic **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --nic
	获取dvpp外设信息	--dvpp **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --dvpp
	获取llc数据	● --llc ● --type <*指定CPU类型*>：可选参数，可选值包括aicpu、ctrlcpu **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --llc --type=ctrlcpu

脚本	作用	参数及示例
	获取ddr数据	--ddr **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --ddr
	获取TS CPU Usage数据	● --ts_cpu_usage ● --start_time=<*指定开始时间*>：可选参数，缩放时指定开始时间，若采集时间为10s，指定“--start_time=0”，表示从0s开始。 ● --end_time=<*指定结束时间*>：可选参数，缩放时指定结束时间，若采集时间为10s，指定“--end_time=6”，表示到6s结束。 ● --number <*指定页面最多显示的数据条数*>：必选参数 **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --ts_cpu_usage --number 100
	获取AI Core Status数据	● --ai_core_status ● --number <*指定页面最多显示的数据条数*>：必选参数 **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --ai_core_status -number 100

脚本	作用	参数及示例
	获取RTS track数据	● --autotuning：同时需指定“--event_type”参数，“--event_type”参数可选值包括“0”和“5”，“0”表示“AI Core”，“5”表示“DMA” ● --page_length <*指定每页显示多少条数据*> ● --page_index <*指定要获取第几页的数据*> ● --sort_column <指定要排序的列名>：可选参数，列包括app_runtime、task_duration、run_tx_time、run_rx_time、runtime_app、sched_tx_time、core_time、sched_rx_time ● --sort_type<*指定按照什么方式来排序*>：可选参数，排查方式包括desc（降序）和asc（升序） **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --autotuning --event_type=0 --page_length 100 --page_index 1 --sort_column app_runtime --sort_type desc
	获取每个core的使用率以及所有core平均使用率的数据	--utilization **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --utilization --start_time=0 --end_time=6 --number 1000
	获取HiAI concrete的数据	--hiai_engine_concrete **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --hiai_engine_concrete --limitPage=100 --page=1
	获取HiAI Graph的数据	--hiai_engine_graph **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --hiai_engine_graph --limitPage=100 --page=1

脚本	作用	参数及示例
	获取HiAI consumption的数据	--hiai_engine_consumption **示例：**python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --hiai_engine_consumption --limitPage=100 --page=1
	获取系统内存占用system memory数据	--sys_mem **示例**：python get_msvp_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --sys_mem
	查看帮助信息	-h/--help
get_msvp_instruction.pyc 实现代码跳转的功能，获取code页签的数据	指定源码路径，获取code页签的数据	● --project <*指定工程采集结果的文件夹*>：必选参数 ● --ddk_dir <*指定ddk目录*>：必选参数 ● --module/-m <*指定Module名称*> ● --function/-f <*指定Function名称*> ● --source <*指定要配置的源码路径*> ● --type <*指定cpu类型*>：可选值包括ctrlcpu、aicpu ● --deviceid=<*指定设备id*> ● --core/-c <*指定cpu核数*>：通过core来过滤结果 ● --pid/-p <*指定pid*>：通过进程ID进行过滤结果 ● --tid/-t <*指定tid*>：通过线程ID进行过滤结果 ● --field/-fld <*指定code页签的field字段*>：可选值包括cycles(default)、r8，其中cycles是默认值 **示例：**python get_msvp_instruction.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --ddk_dir /home/tools/che/ddk/ddk --source /home/ascend/MIND0214001 -m /usr/lib64/libpthread-2.17.so -f __errno_location --deviceid=0 --type=ctrlcpu --field cycles -c 1 -p 1 -t 1

脚本	作用	参数及示例
	指定二进制符号表文件，获取code页签的数据	--symtab <*指定要配置的二进制符号表文件*> **示例：**python get_msvp_instruction.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --ddk_dir /home/tools/che/ddk/ddk --symtab /home/ascend/MIND0214001 -m /usr/lib64/libpthread-2.17.so -f __errno_location --deviceid=0 --type=ctrlcpu
	查看帮助信息	-h/--help
get_msvp_timeline.pyc 获取Timeline页签的数据	获取时序图数据	● --project <*指定工程采集结果的文件夹*>：必选参数 ● --start_time=<*指定起始时间*> ● --end_time=<*指定结束时间*> ● --deviceid=<*指定设备id*> ● --timeline_diagram：指定获取时序图的数据，不包含左侧数据的树形结构。 ● --replayid=<指定replayid>：预留参数。 **示例：**python get_msvp_timeline.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --start_time=1550759017.8505 --end_time=1550759018.9249 --deviceid=0 --timeline_diagram
	获取llc的timeline数据	● --type <*指定CPU类型*>：可选值包括aicpu、ctrlcpu ● --timeline_llc：获取llc的timeline数据 **示例：**python get_msvp_timeline.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --start_time=1550759017.8505 --end_time=1550759018.3505 --type=aicpu --timeline_llc
	获取ddr的timeline数据	● --master_id <*指定DDR通道*>：可选值包括0、1、2、3、4、5、6、7 ● --timeline_ddr：获取ddr的timeline数据 **示例：**python get_msvp_timeline.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --start_time=1550759017.8505 --end_time=1550759018.3505 --master_id 0 --timeline_ddr

脚本	作用	参数及示例
	获取timeline页签Host侧的数据	--timeline_host **示例：**python get_msvp_timeline.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --start_time=1550813088.0822 --end_time=1550813089.9895 --deviceid=0 --timeline_host
	获取左侧外设数据列表	--timeline_list_peri **示例：**python get_msvp_timeline.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --timeline_list_peri
	获取Timeline的开始时间、结束时间、Duration参数值（大数据量情况下用于展示一次接受多长时间的数据）	● --timeline_maxtime ● --deviceid=<*指定设备id*>：必选参数 **示例：**python get_msvp_timeline.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --timeline_maxtime --deviceid=0
	获取dvpp的timeline数据	● --timeline_dvpp ● --enginetype <*指定enginetype*>：可选值VDEC、JPEGD、PNGD、JPEGE、VPC、VENC。 ● --engineid=<*指定engineid*> ● --number <*指定页面最多显示的数据条数*> **示例：**python get_msvp_timeline.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --start_time=1550813088.0822 --end_time=1550813089.9895 --deviceid=0 --timeline_dvpp --enginetype VDEC --engineid=1 --number 10000
	获取timeline_apitask数据	● --timeline_apitask：获取timeline_apitask数据（必须指定rowid） ● --row_id=<*指定rowid*> **示例：**python get_msvp_timeline.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 -- row_id=0 --timeline_apitask
	查看帮助信息	-h/--help

脚本	作用	参数及示例
get_rstrack_info.pyc 获取Timeline页签的RTStrack数据	获取rtstrack数据	● --project <*指定工程采集结果的文件夹*>：必选参数 ● --start_time=<*指定开始时间*> ● --end_time=<*指定结束时间*> ● --deviceid=<*指定设备id*> ● --rtstrack **示例：**python get_rstrack_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --start_time=1550759017.8505 --end_time=1550759018.9249 --deviceid=0 --rtstrack
	获取开始时间和结束时间	--timerange **示例：**python get_rstrack_info.pyc --project /home/ascend/.mindstudio/profiler/data/profiling_out --deviceid=0 --timerange
	查看帮助信息	-h/--help
msvp_import.pyc	将采集结果导入数据库	● --target <*指定要导入的目标文件夹*>：必选参数 ● --ddk_dir <*指定ddk目录*> **示例：**python msvp_import.pyc --target /home/ascend/tools/profiler/projects/68:05:ca:83:ad:57/projects_20190221/rts_test_2019022109234475 --ddk_dir /home/ascend/tools/che/ddk/ddk **说明** 在执行该脚本前，需要先清理“--target”指定目录下的“sqlite”目录下的所有*.db文件以及“data/log”目录下的*.db文件。
	查看帮助信息	-h/--help

10 日志工具

10.1 工具简介
10.2 日志操作
10.3 日志参考

10.1 工具简介

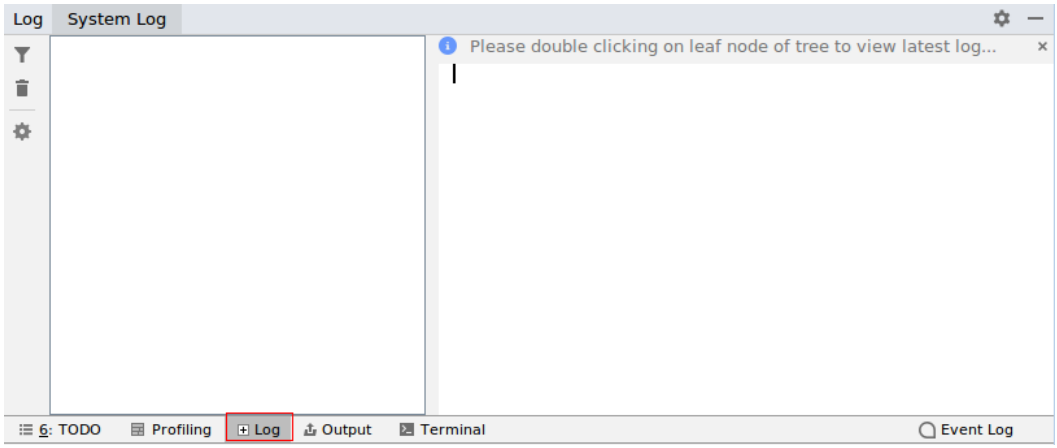
Mind Studio为昇腾AI处理器提供覆盖全系统的日志收集与日志分析解决方案，提升运行时算法问题定位效率。Mind Studio提供全系统统一的日志格式，并以图形化的形式提供跨平台日志可视化分析能力及运行时诊断能力，提升日志分析系统的易用性。

通过Log工具，您可以进行以下操作：

- 管理Device设备
- 设置日志级别
- 查看系统日志

在Mind Studio窗口底部单击“+Log”标签，显示“Log”窗口，您可以通过该窗口查看日志信息，如**图10-1**所示。

图 10-1 log 界面



说明

在Ubuntu系统下使用Log工具打开文件夹和文件功能，需要使用nautilus、gedit软件；如果系统未安装此软件，请使用Mind Studio安装用户执行命令**sudo apt-get install nautilus gedit**进行安装。

10.2 日志操作

10.2.1 管理 Device 设备

进行系统日志查看前，需要添加Device设备才能获取到相关日志信息。本文仅描述Device设备管理的增加、删除、修改功能，方便使用者对Device设备信息管理。

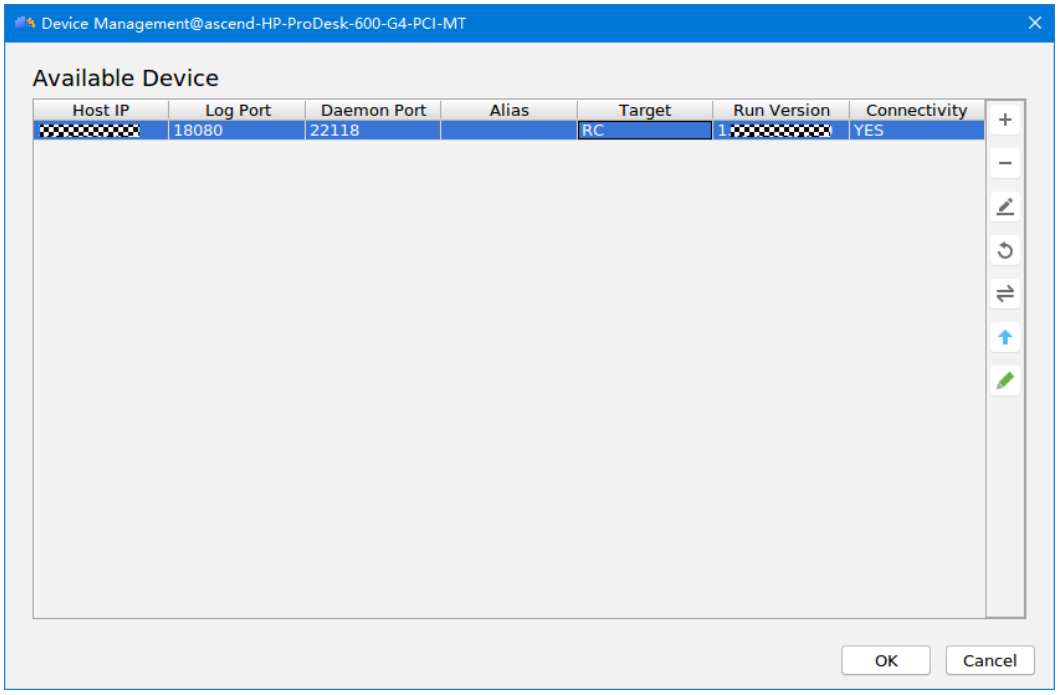
您可以通过以下方式进行设备管理：

步骤1 在Mind Studio主界面，单击菜单栏“Tools > Device Management”,弹出设备维护窗口，如图10-2所示。

说明

在Mind Studio界面的“Log”窗口，单击图标，也可以打开Device Management窗口。

图 10-2 Device Management 窗口



步骤2 在Device Management页面，可以进行以下设备管理操作。

- 增加Device设备

  单击 + 按钮，输入Device设备Host IP、Log Port、Daemon Port、Alias信息，单击“OK”按钮。添加设备后需要选择设备并单击 ⇌ 按钮同步安装lib文件。参数说明如表10-1所示。

表 10-1 增加设备界面参数

参数	说明
Host IP	host侧IP地址。
Log Port	Log端口号，取值范围为[18000~18080]，默认为18080。
Daemon Port	IDE_Daemon_client与IDE_Daemon_host通信使用的端口号，取值范围为[20000~25000]，默认为22118。
Alias	host侧设备的别名，如果有多个设备时，方便用户管理设备信息。

- 修改Device设备

  选择已添加设备后，单击 按钮，修改Device设备信息，单击“OK”按钮。

- 删除Device设备

  选择已添加设备，单击 按钮。

步骤3 在Device Management页面上执行**步骤 2**操作后，需要单击“OK”按钮才能生效。

----结束

10.2.2 System Log 操作

在System Log页签，可以查看系统运行日志，操作步骤如下：

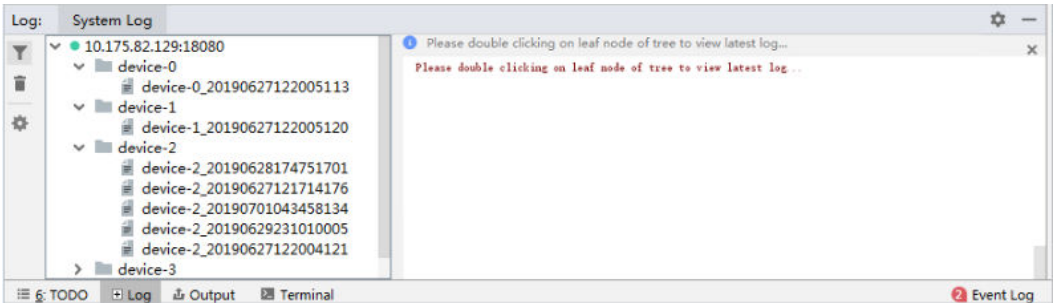
说明

Mind Studio不支持通过界面方式删除设备上的system log日志。

如需对日志进行管理，请使用root或HwHiAiUser用户登录Mind Studio所在服务器操作系统后对/var/dlog目录下日志进行管理操作；如果Mind Studio安装用户是非root的普通用户，则Mind Studio安装用户没有查看、删除等日志管理权限。

步骤1 在Mind Studio界面的“Log”窗口，单击“System Log”页签，在显示的设备列表中，选中某个设备，展开该设备的日志列表，如**图10-3**所示。

图 10-3 日志列表

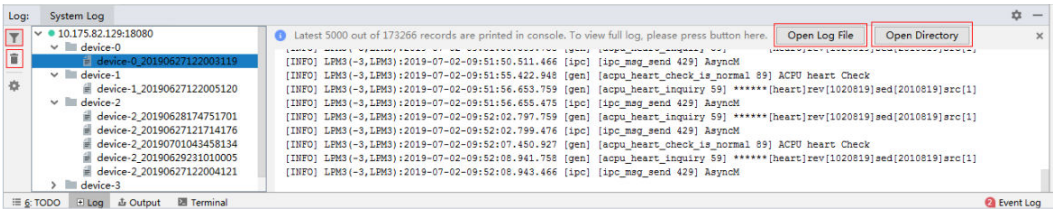


步骤2 双击日志列表中某日志文件，展示日志内容。

说明

本处仅展示该日志文件中最新的5000条日志。

图 10-4 System log 页签



步骤3 在System Log页签，您可以进行如下操作。

- **单击**图标： 提供查询功能，按关键字搜索日志内容。
- **单击**图标：清除显示的日志。
- **单击** Open Log File 图标： 打开日志文件，展示该日志文件的全部日志内容。
- **单击** Open Directory 图标：打开日志文件下载到的IDE所在机器的本地存储路径。

----结束

10.3 日志参考

10.3.1 日志处理机制介绍

图 10-5 日志处理机制

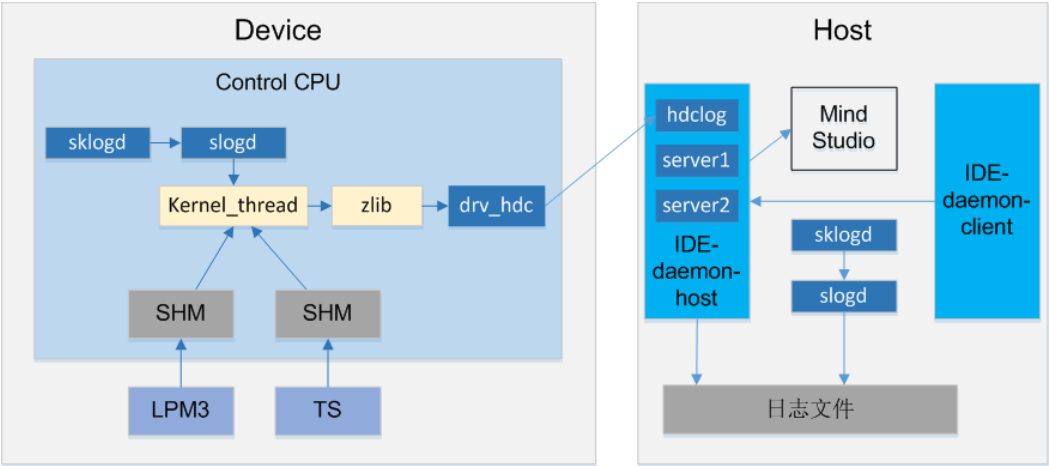


图10-5中的日志数据的流向解释如下：

1. 采集日志

   在Device侧，调用log驱动采集非Control CPU上的日志，sklogd和slogd进程采集Control CPU上的日志。

   在Host侧，sklogd和slogd进程采集用户态和内核态日志。

2. 传输日志

   Device侧的日志可通过驱动从Device侧传输给Host侧，在Host侧通过IDE-daemon-host模块接收日志。

3. 将日志记录到文件中

Device侧的日志被IDE-daemon-host记录在文件名称以device-*id*开头的日志文件中，Host侧的日志被slogd记录在文件名称以host-0开头的日志文件中。IDE-daemon-host中拉起TCP/IP线程，将host侧日志发送给Mind Studio，即完成日志的传输。

Host将日志内容记录到日志文件中，您可以在Mind Studio工具上查看日志信息。如果需要调整日志级别，可通过Mind Studio界面操作实现，详情请参见**10.3.6 设置日志级别** 。

10.3.2 日志文件介绍

在Mind Studio中，日志文件区分Device侧和Host侧，本节介绍各侧日志的存储路径、文件名称及记录的主要信息。

表 10-2 日志分类介绍

日志文件	存储路径	说明
device-*id*_*.log	“/var/dlog/device-id”	• Slog采集Device侧Control CPU上的日志，主要采集以下模块的日志： – Slog – IDE-daemon-device – Matrix – Driver – HDC • 采集Device侧非Control CPU上的日志，主要采集以下模块的日志： – Task Scheduler CPU – LPM3
host-0_*.log	“/var/dlog/host-0”	Slog采集Host侧用户态和内核态日志，主要采集以下模块的日志： • Slog • Matrix • Framework • Runtime • CCE • IDE-daemon-host • Driver • HCCL • DVPP • HDC • MDC • MLL • Kernel

说明

日志工具自身的日志文件存储在“/usr/slog/slogd.log”目录下，日志文件名为slogd.log，记录日志工具自身的运行信息，用于日志工具自身问题定位。日志具备老化策略，当slogd.log文件达到规定大小后，名称变更为slogd.log.old进行备份。

其中，日志文件中记录的error信息为linux系统自带错误号。

10.3.3 日志级别介绍

各日志级别介绍如表10-3所示。

表 10-3 日志级别

日志级别	定义
ERROR	一般错误级别，该级别的日志记录了如下错误： ● 非预期的数据或者事件。 ● 影响面较大且模块内部能够处理的错误。 ● 限制在模块内的错误。 ● 对其他模块有轻微影响的错误，例如统计任务创建失败。 ● 引起调用失败的错误。 ● 在业务逻辑错误的情况下记录错误状态的信息及造成错误的可能原因。
WARNING	警告级别，系统和预期的状态不一致，但不影响整个系统的运行。
INFO	正常级别，系统正常运行的信息。
DEBUG	调试级别，该级别的日志记录了调试信息，便于开发人员或维护人员定位问题。
NULL	表示NULL级别，不输出日志。
EVENT	输出全系统最关键日志，例如：整网运算启动\完成\异常终止， 内存不够，单板温度过高等。

10.3.4 日志记录格式介绍

本节介绍系统记录日志的格式以及各字段的含义，便于理解日志所记录的信息。

日志样例如下：

```
[INFO] KERNEL(1080,sklogd):1970-01-01-08:00:04.495.604 sklogd started
```

日志格式如下：

```
[Level] ModuleName(PID,PName):DateTimeMS LogContent
```

表 10-4 日志字段说明

字段	说明
Level	日志级别。运行日志存在5种日志级别：ERROR、WARNING、INFO、DEBUG、EVENT。
ModuleName	产生日志的模块的名称。
PID	进程ID。
PName	进程名称。
DateTimeMS	日志打印时间，格式为：yyyy-mm-dd-hh:mm:ss.SSS。
LogContent	各模块具体的日志内容。

10.3.5 日志配置介绍

系统运行过程中需要记录的日志级别、日志输出路径、日志文件名、单个日志文件大小等信息可以配置，本节对控制日志生成的配置进行介绍。

说明

用户在修改日志配置项时，请严格按照**表10-5**中的要求，根据建议值或取值范围进行配置，若不按照配置项配置要求，可能会导致系统异常。

手动修改Host侧配置文件后，需执行reboot命令重启系统，使配置在Host侧生效；如果仅涉及日志级别修改，请参考**10.3.6 设置日志级别**说明。

/etc/slog.conf文件用于控制**Slog**采集日志时的配置，该文件中的配置段样例，如下所示。

```
# Global log level
global_level=1

# User
user=HwHiAiUser

# log-agent-host #
logAgentMaxFileNum=8
# set host one log file max size, range is [1048576,104857600]
logAgentMaxFileSize=10485760
# set host log dir
logAgentFileDir=/var/dlog

#log server send log to IDE
#IP_address to bind: if you want to set ip_address manually, please open the item and add your ip.
#log_server_ip = 127.0.0.1

# Port: Please select a port number between 18000 and 18080([18000, 18080]).
daemon_socket_port=18080

#enable print event log, 0:disable, 1:enable
enableEvent=1

#zip switch, default : not zip
#0 : not zip
#1 : zip
zip_switch = 1
```

```
#zip level: 0 ~ 9, others err and exit, default:0
#zip_level=0
#############dlog level configure###############
TS=5
TSDUMP=5
AICPU=5
LPM3=5
##################################################


##############begin module log level##############################
##############please don't change this items #####################
DLOG=5                          # Dlog
SLOG=5                          # Slog
IDEDD=5                         # IDE daemon device
IDEDH=5                         # IDE daemon host
LOGAGTH=5                         # log agent host
HCCL=5                          # HCCL
```

相关配置项说明详情见**表10-5**。

表 10-5 相关配置项说明

配置项	说明
Global_level	配置slog日志级别。 ● 0：表示DEBUG级别。 ● 1：表示INFO级别。 ● 2：表示WARNING级别。 ● 3：表示ERROR级别。 ● 4：表示NULL级别，不输出日志。
user	后台服务器上直接查看日志的用户，默认是HwHiAiUser用户。
LogAgentMaxFileNum	Host侧“/var/dlog”下保存文件的数量，Host侧日志文件数目大于该数目时发生滚动，新日志覆盖最早的日志。 该参数在Device侧无效。
logAgentMaxFileSize	单个日志文件大小，该数值为最大值，如果日志文件大小超过该最大值，则生成新的日志文件。当前默认值是10MB，您可以根据实际情况调整大小。最大不超过100MB。 该参数在Device侧无效。 **说明** 如果设置值小于1MB，系统默认为1MB。
logAgentFileDir	日志文件路径。如果修改该路径，需确保该路径存在且属主为运行日志进程的用户。 该参数在Device侧无效。
daemon_socket_port	发送日志给IDE（Mind Studio）的端口号。 该参数在Device侧无效。
enableEvent	是否开启event日志级别。 ● 1：开启event日志级别。 ● 0：不开启event日志级别。

配置项	说明
zip_switch	是否可以直接在后台服务器查看日志文件。 ● 0：可以直接在后台服务器查看日志文件。 ● 1：不可以直接在后台服务器查看日志文件，需通过Mind Studio前台的Log窗口查看日志。 该参数在Device侧无效。
TS、TSDUMP、AICPU、LPM3	配置各模块级别。这4个参数仅Device侧有效。 当参数值为5时，模块级别取Global_level设置的级别；其他值时，模板级别取该参数设置级别。
DLOG、SLOG、IDEDD、IDEDH……	配置各模块级别。 当参数值为5时，模块级别取Global_level设置的级别；其他值时，模板级别取该参数设置级别。

10.3.6 设置日志级别

日志记录了系统的运行情况和功能流程的处理情况，是维护人员查看系统状态、进行问题定位的重要工具和手段。日志模块根据系统设置的日志级别，记录不同详细程度的内容，满足不同系统维护需求。

支持通过以下三种方式来设置日志级别：

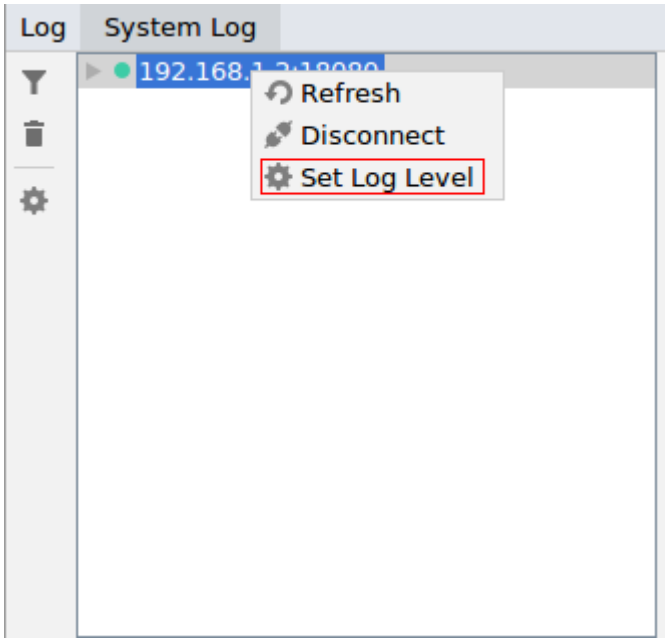
- 通过Mind Studio界面修改
- 通过IDE-daemon-client命令行修改
- 通过/etc/slog.conf配置文件修改

通过 Mind Studio 界面修改日志级别

Log工具提供界面方式修改系统的日志级别，设置日志级别方法：

步骤1 在Mind Studio界面的“Log”窗口，单击“System Log”页签，在显示的Device设备列表中，选中某个设备，单击右键，选择“Set Log Level”。如图10-6所示。

图 10-6 选择 Set Log Level



步骤2 在弹出“Set Log Level”窗口，您可以设置日志级别。如图10-7所示。

“Global Log Level”区域的日志级别设置是全局的，“Module Log Level”区域的日志级别设置是针对每个模块。

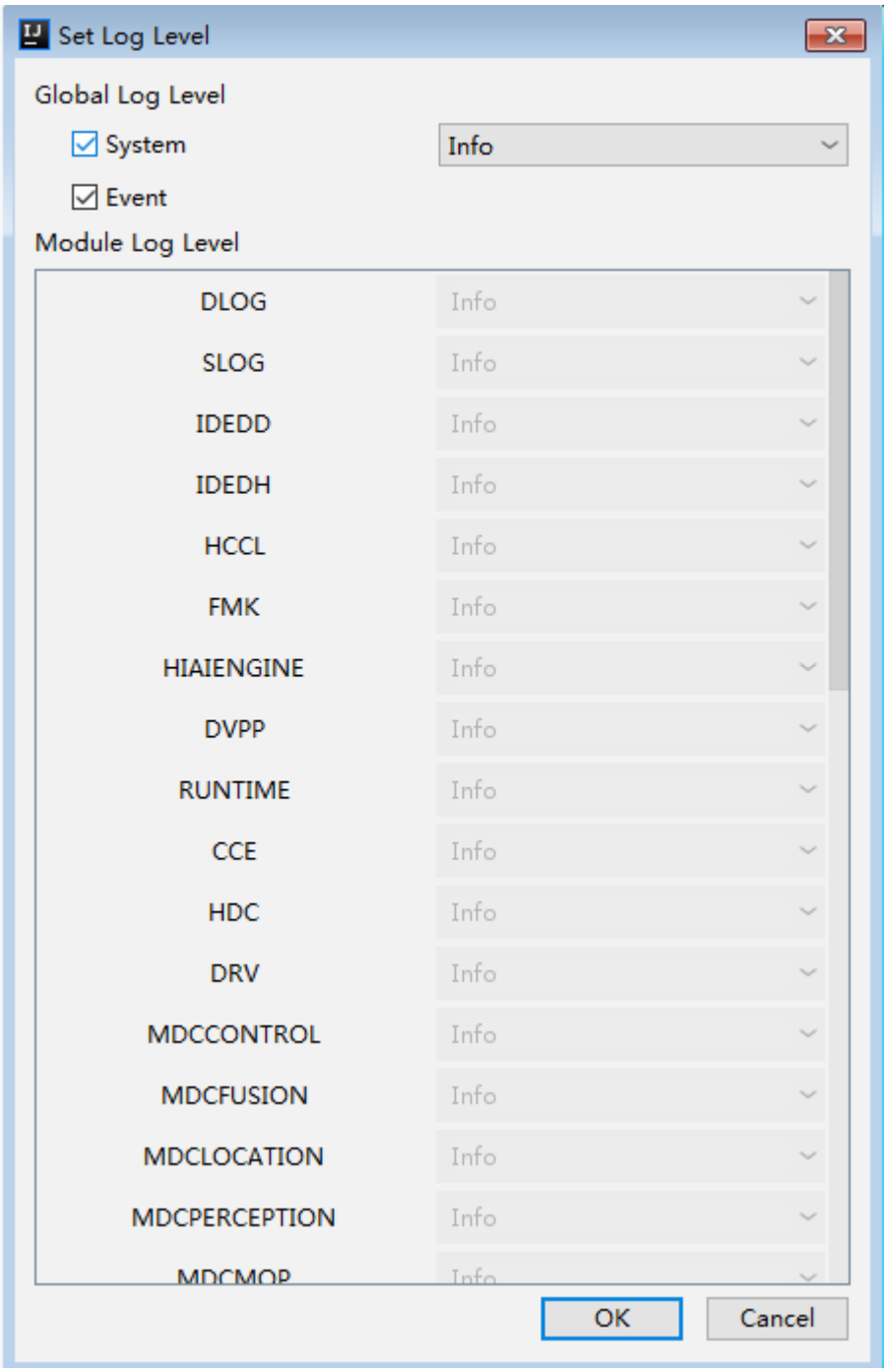
如果设置了全局级的日志级别就不能再设置模块级的日志级别；如果需要设置模块级的日志级别，需要不勾选“System”复选框的日志级别设置。

说明

日志级别说明请参考10.3.3 日志级别介绍。

执行修改日志级别后，针对PCIE场景，如果重启昇腾AI处理器，Device侧日志级别将恢复为系统默认级别。

图 10-7 Set Log Level 窗口



步骤3 设置日志级别后，单击“OK”。

----结束

通过 IDE-daemon-client 命令行修改日志级别

您可以用Mind Studio安装用户登录服务器，依次执行**export LD_LIBRARY_PATH=~/.mindstudio/huawei/ddk/1.3*.T*.B***/ddk/uihost/lib**和**export PATH=~/.mindstudio/huawei/ddk/1.3*.T*.B***/ddk/uihost/bin**设置环境变量后，执行全局级或模块级的日志级别设置命令。

说明

- “~/”是Mind Studio安装用户家目录路径；1.3*.T*.B***为DDK的版本号。如果使用DDK安装用户执行，请替换示例命令中环境变量路径为实际路径。
- 您也可以用HwHiAiUser用户登录Host侧服务器，执行全局级或模块级的日志级别设置命令（目前仅支持PCIE形态）。
- 执行命令修改日志级别，同时在Host侧和Device侧生效。
  - 针对非Atlas 200 DK场景，如果重启昇腾AI处理器，Device侧日志级别将恢复为系统默认级别。
  - 针对非Atlas 200 DK场景，一卡多芯片形态，通过命令修改指定ID的任意一Device日志级别，均会修改整个Host侧日志级别和指定Device的日志级别 。
- 修改日志级别命令参数说明：
  - “服务器的IP地址”：指Host侧IP地址。
  - “端口号”：默认值为22118。
  - “level”：可设置的日志级别名称，参数范围：error、info、warning、debug、null（表示不打印日志）中的一个。
  - “id”：指待修改日志级别的Device的ID号。

- 设置全局级的日志级别的命令如下：

  IDE-daemon-client --host 服务器的IP地址:端口号 --log 'SetLogLevel(0)[level]' --device id

  举例：IDE-daemon-client --host 192.168.1.2:22118 --log 'SetLogLevel(0)[info]' --device 0

- 设置模块级的日志级别的命令如下：

  IDE-daemon-client --host 服务器的IP地址:端口号 --log 'SetLogLevel(1)[moduleName:level]' --device id

  举例：IDE-daemon-client --host 192.168.1.2:22118 --log 'SetLogLevel(1)[TS:info]' --device 0

  “moduleName”可设置为TS、TSDUMP、AICPU、LPM3、DLOG、SLOG、IDEDD、IDEDH、HCCL、FMK、HIAIENGINE、DVPP、RUNTIME、CCE、HDC、DRV、MDCFUSION、MDCLOCATION、MDCPERCEPTION、MDCFSM、MDCCOMMON、MDCMONITOR、MDCBSWP、MDCDEFAULT、MDCSC、MDCPNC、MLL、DEVMM、KERNEL、LIBMEDIA、CCECPU、ASCENDDK、ROS、HCCP、RoCE、TEFUSION、PROFILING、DP、APP、TDT、FE、MD、MB、ME等模块中的一个。

- 设置event的日志级别的命令如下：

  IDE-daemon-client --host 服务器的IP地址:端口号 --log 'SetLogLevel(2)[enable/disable]' --device id

  举例：IDE-daemon-client --host 192.168.1.2:22118 --log 'SetLogLevel(2)[enable]' --device 0

  “enable”：开启event日志级别。

  “disable”：不开启event日志级别。

执行命令修改日志级别后，可以分别查看Host侧和Device侧/etc/slog.conf配置文件是否修改成功，如果修改成功，则修改日志级别生效。

通过/etc/slog.conf 配置文件修改日志级别

- Atlas 200 DK场景

  参考**10.3.5 日志配置介绍**章节内容，修改Host侧/etc/slog.conf配置文件中全局日志级别、模块日志级别和是否开启Event日志。

  修改配置文件中日志级别后，除常驻进程对应的模块不能即时生效外，其他模块的日志级别对新执行的任务即时生效。

说明

常驻进程包括：sklogd、IDE-daemon-host、IDE-daemon-device、matrixdaemon、dmp_daemon、aicpufw_monitor、hdcd。

如果需要使日志修改全部生效，可以在修改配置文件后，执行reboot命令重启处理器，使配置侧生效；也可以依次重启常驻进程，使配置生效。

a. 以**HwHiAiUser**用户登录Host侧，进入/var目录。
b. 重启sklogd进程：pkill sklogd; ./sklogd &
c. 重启IDE-daemon-host进程：pkill IDE-daemon-host; ./IDE-daemon-host &
d. 重启IDE-daemon-device进程：pkill IDE-daemon-device; ./IDE-daemon-device &
e. 依次执行其他常驻进程重启。

   hdcd需要在/sbin目录下执行。

- 非Atlas 200 DK场景

  参考**10.3.5 日志配置介绍**章节内容，分别修改Host侧和Device侧/etc/slog.conf配置文件中全局日志级别、模块日志级别和是否开启Event日志。

  修改配置文件中日志级别后，除常驻进程对应的模块不能及时生效外，其他模块的日志级别对新执行的任务及时生效。

  说明

  Host侧常驻进程：sklogd、IDE-daemon-host、hdcd。

  Device侧常驻进程：sklogd、IDE-daemon-device、matrixdaemon、dmp_daemon、aicpufw_monitor、hdcd。

  修改日志级别后，可以参考**•Atlas 200 DK场景**中的重启进程命令，分别登陆Host侧（/usr/local/HiAI/driver/tools）和Device侧（/var）依次重启常驻进程，使日志修改对常驻进程生效；也可以在Host侧执行reboot命令，但配置仅在Host侧生效，且Device侧日志级别将恢复为系统默认级别。

11 算子比对

11.1 简介

11.2 数据准备

11.3 Lower Bound比对

11.4 Vector比对

11.5 FAQ

11.1 简介

Mind Studio提供算子比对工具是为了定位模型的精度问题，比对华为自有模型算子的运算结果与Caffe标准算子的运算结果，以便确认误差发生的原因。

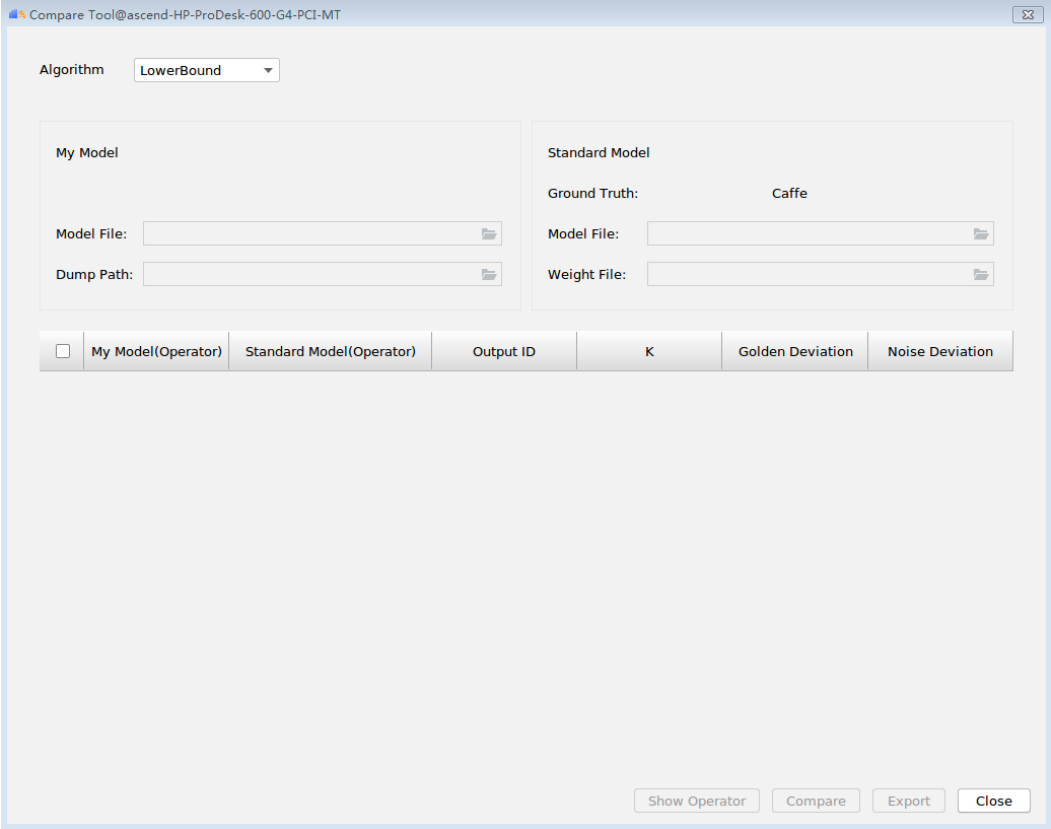
Mind Studio提供以下比对方法：

- Lower Bound比对，华为自主研发的算法比对。
- Vector比对，包含余弦相似度、最大绝对误差、累积相对误差、欧氏相对距离的算法比对。

Mind Studio为比对工具提供菜单入口：

在Mind Studio界面，选中某个工程，在菜单栏选择“Tools > Compare Tool”菜单，进入比对工具界面，如**图11-1**所示。

图 11-1 比对工具界面



说明

算子比对工具不支持以下场景：

- 单个Dump文件的大小超过1GB。
- 算子在转换过程中输入/输出维度发生变化，例如不支持FastRCNN的proposal，roi_pool5，relu6， relu7，cls_score，bbox_pred，cls_prob算子。

  如果需要比对包含上述类型算子的模型，需要删除Dump数据中这些不支持的算子Dump文件后进行比对。
- 模型batch数大于1。
- 多网串联。

11.2 数据准备

准备离线模型 Dump 数据

已存在完成编译的APP应用工程。

打开该工程的graph.config文件，在推理Engine节点增加如下oam_config代码后再次进行Build和Run。

```
engines {
  id: 226
  engine_name: "MindInferenceEngine"
  side: DEVICE
  thread_num: 1
  so_name: "./libDevice.so"
```

```
  ai_config {

    items {
      name: "model_path"
      value: "/home/wshide/MindStudio-ubuntu/samples/modelfile/resnet18.om"
    }
  }
  oam_config{
    dump_list{
       model_name: "ResNet-18"
       is_dump_all: "true"
    }
    dump_path: "/home/ascend/.mindstudio/huawei/ddk/dump"
  }
}
```

说明

- model_name：模型名称。通过以下方式获取：在Mind Studio界面选择菜单栏“Tools > View Model”，选择.om模型文件，在显示该模型内容的面板空白处单击左键，此时右侧属性面板中显示模型名称属性“Model Name”，取该项对应值。
- dump_path：Dump文件生成路径，请根据实际进行替换，Mind Studio安装用户需要有dump_path指定的路径读写权限。

准备 Caffe 模型 Dump 数据

本版本Mind Studio不提供Caffe模型Dump数据生成功能，请自行安装Caffe环境并提前准备Dump数据。我们仅提供Caffe原始数据转换成protobuffer格式Dump文件的python脚本（generateDumpFile.pyc），以及生成numpy格式Caffe原始数据“*.npy”文件的样例要求方法。

说明

generateDumpFile.pyc脚本存放在“/home/ascend/.mindstudio/huawei/ddk/1.3*.T*.B***/ddk/toolchains/operator_cmp/compare”路径下。

- “/home/ascend”为Mind Studio安装用户的家目录。
- “1.3*.T*.B***”为DDK版本信息，根据实际版本情况选择。

该脚本执行需依赖numpy、protobuffer库。需要执行命令**pip install numpy protobuf --user**进行安装。

Caffe原始数据文件准备要求：

- 文件内容以 numpy格式保存（用来获取数据和shape）。
- 文件命名：以“层名”+后缀“.npy”形式命名。

输出“*.npy”数据文件处理方式示例说明如下和**图11-2**所示。

在Caffe工程的每层推理结果代码处，增加以下类似代码：

- line 157 && line 151：设置layer_out_numpy是裸数据，从推理中获取。
- line 158：设置layer_out_numpy的shape值，blob.data.shape。
- line 156：设置layer_npy_name 为文件名， 结构是“写入文件的路径 + 层名 + npy格式”。
- line 159：数据保存 np.save(layer_npy_name,layer_out_numpy)，这里的np是numpy库。

图 11-2 Caffe 推理结果处理代码示例

```
    idx = 0
    for layer, blob in net.blobs.iteritems():
        layer = re.sub('/', '_', layer)
        if "all" not in real_layer_list:
            if layer not in real_layer_list:
                continue

        layer_out = blob.data.astype(np.float32, copy=False).copy()
        op_id = str(idx)
        dump_file_name = data_index + '.' + net_param.name + '.' + model_id + '.' + op_id
        dump_file_name += '.' + layer
        dump_path = os.path.join(dump_path_tmp3, dump_file_name)
        layer_npy_name = '/home/studio/webdump_caffe/' + layer + '.npy'
        layer_out_numpy = layer_out
        layer_out_numpy.shape = blob.data.shape
        np.save(layer_npy_name,layer_out_numpy)
        # print 'layer/shape: %-40s =>  %s' % (layer + str(blob.data.shape), dump_path)

        idx += 1
        data = _combine_data(layer, blob.data.shape, layer_out)
        with open(dump_path, 'w') as fp:
            fp.write(data)
            os.chmod(dump_path, 0o400)
```

获取到Caffe原始数据后，通过generateDumpFile.pyc脚本进行数据转换，输出protobuffer格式的Dump文件。命令行格式如下：

```
python generateDumpFile.pyc -in INPUT_DATA -out OUTPUT_PATH -name LAYER_NAME
```

- -in：数据文件路径（可以是文件夹或者文件），必选参数 。

  如果in 参数输入是文件夹，则格式要求如下：文件夹里面内容必须是“层名.npy”文件。

  如果in 输入参数是文件名，则输入以算子“层名”+后缀“.npy”组成的“层名.npy”形式文件名；如果输入的是“非层名”+后缀“.npy”形式的文件名，则需要“-name” 参数传入层名。每次仅支持输入1个文件转换。
- -out：转换后输出文件路径，必选参数。
- -name：层名，可选参数。仅在“-in”参数输入文件名是“非层名.npy”形式时生效。

11.3 Lower Bound 比对

比对步骤

说明

Lower Bound比对需要使用到Caffe环境，请参考Caffe官方指导准备Caffe环境，华为不提供相关资料。另，本次准备的Caffe环境需要与您的Caffe模型训练时使用的Caffe环境一致，确保算子和算子属性一致。

因需要使用离线模型算子的上一层Dump数据作为Caffe的输入来计算Caffe的标准输出结果，所以比对单个算子的时候，需要上传待比对这一层及其上一层的离线模型Dump数据。

Lower Bound比对约束：

- 开源Caffe不支持的层（如新增算子或修改的算子等），不可用于Lower Bound比对。
- 仅适合离线模型的算子，不可用于Lower Bound比对，如SSD和Fasterrcnn的detection_out层。

  如果比对SSD和Fasterrcnn网络，则需要在推理运行SSD和Fasterrcnn网络时删除prototxt中的detection_out层。

Lower Bound比对操作步骤：

步骤1 在Mind Studio界面，在菜单栏选择“Tools > Compare Tool”菜单，进入比对工具界面。

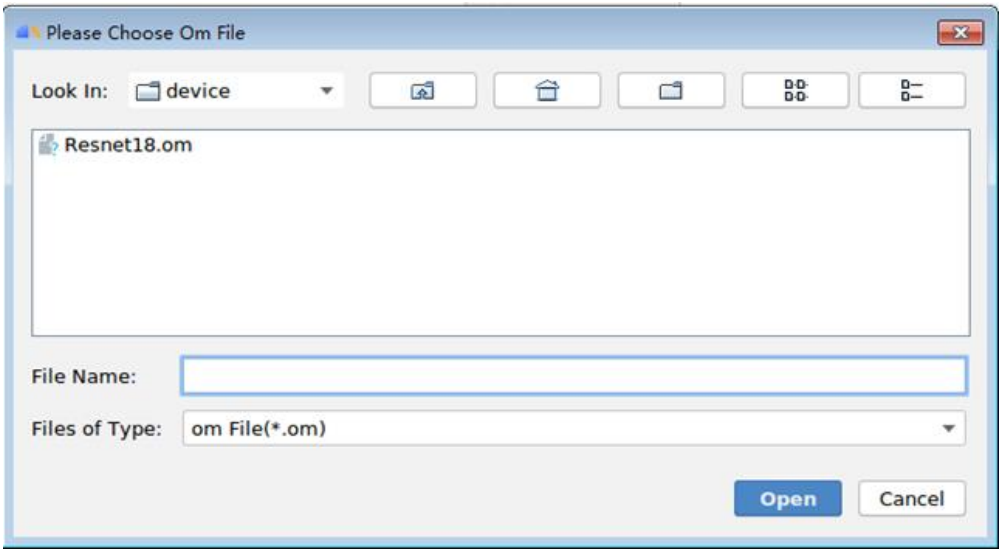
步骤2 “Algorithm”选择“LowerBound”。

步骤3 设置“My Model”信息。

1. 单击“Model File”右侧的，设置离线模型文件，选择一个.om文件，如图11-3和图11-4所示。

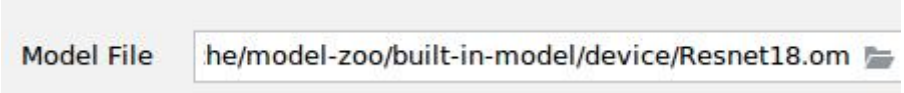
图 11-3 配置 My Model 的 Model File 参数

图 11-4 选择离线模型



选择完成后，在文本框中显示选择的离线模型文件，如图11-5所示。

图 11-5 Model File 参数配置结果



2. 单击“Dump Path”右侧的，设置离线模型Dump数据路径，如图11-6和图11-7所示。

图 11-6 配置 My Model 的 Dump Path 参数

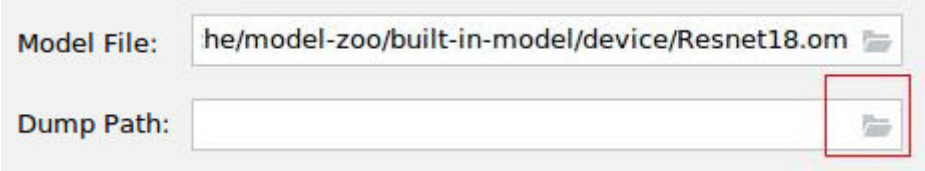
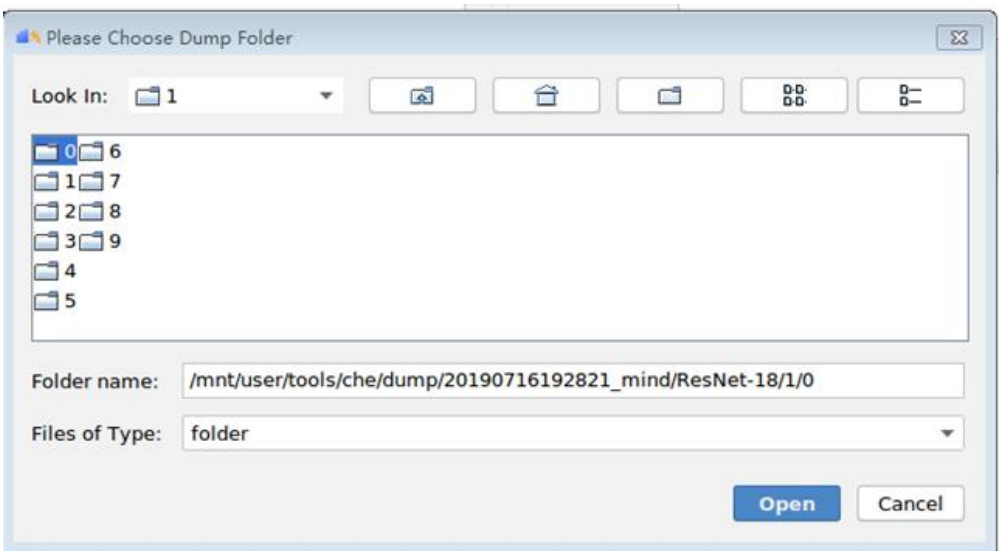


图 11-7 选择 My Model 的 Dump 数据



选择完成后，在文本框中显示选择的Dump数据路径图11-8所示。

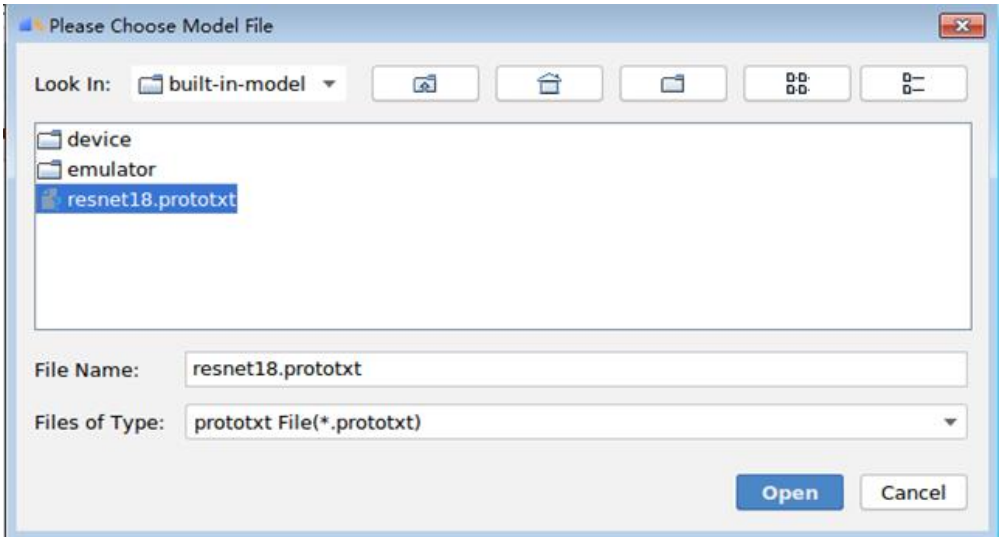
图 11-8 My Model 的 Dump Path 参数配置结果

步骤4 设置“Standard Model”信息。

1. 单击“Model File”右侧的 ，设置Caffe模型文件，选择一个.prototxt文件，如图11-9和图11-10所示。

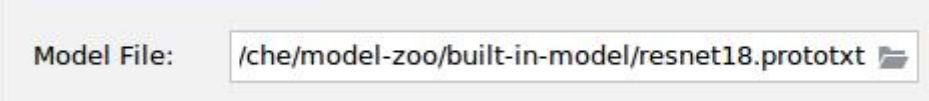
图 11-9 配置 Standard Model 的 Model File 参数

图 11-10 选择 Caffe 模型文件



选择完成后，在文本框中显示选择的Caffe模型文件，如图11-11所示。

图 11-11 Standard Model 的 Model File 参数配置结果



2. 单击“Weight File”右侧的 ，设置Caffe权重文件，选择一个.caffemodel文件，如图11-12和图11-13所示。

说明

如果在选择.prototxt文件时，该目录下同时存在对应的.caffemode文件，则默认直接填充到“Weight File”，无需再次选择文件。

图 11-12 配置 Standard Model 的 Weight File 参数

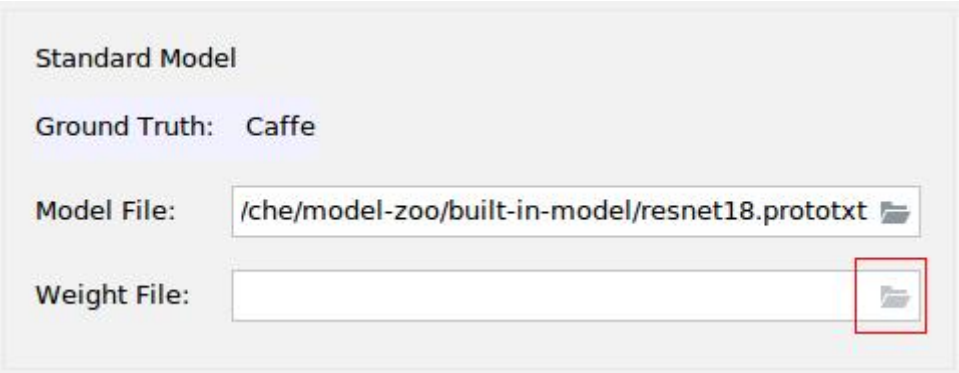
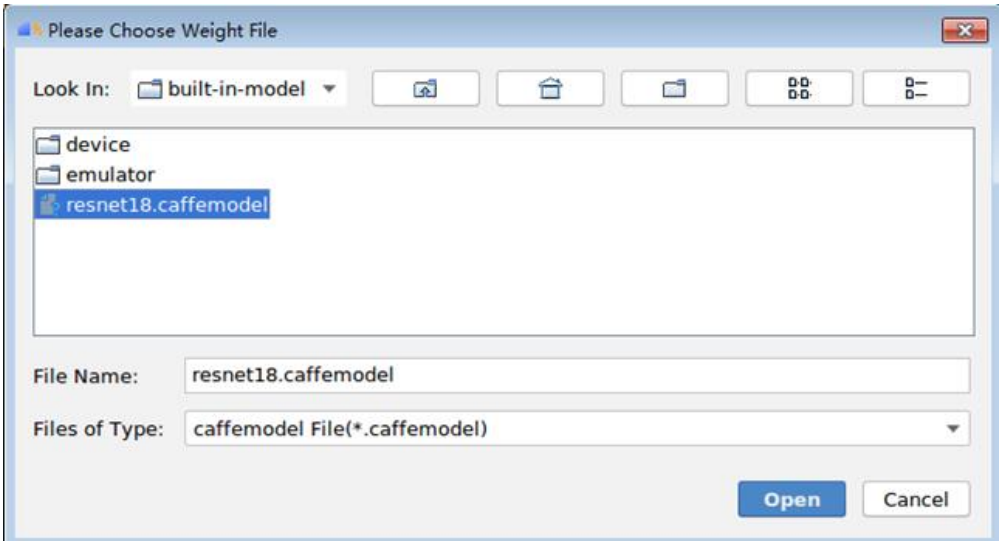


图 11-13 选择 Caffe 权重文件

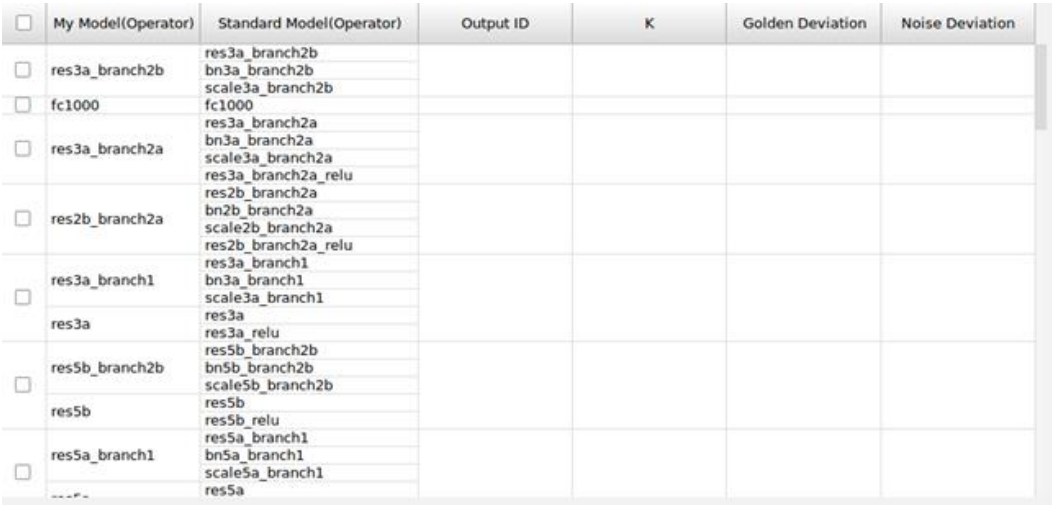


选择完成后，在文本框中显示选择的权重文件，如图11-14所示。

图 11-14 Standard Model 的 Weight File 参数配置结果

步骤5 单击“Show Operator”按钮，获取要比对的算子信息，如图11-15所示。

图 11-15 获取算子信息



My Model(Operator)	Standard Model(Operator)	Output ID	K	Golden Deviation	Noise Deviation
res3a_branch2b	res3a_branch2b bn3a_branch2b scale3a_branch2b				
fc1000	fc1000				
res3a_branch2a	res3a_branch2a bn3a_branch2a scale3a_branch2a res3a_branch2a_relu				
res2b_branch2a	res2b_branch2a bn2b_branch2a scale2b_branch2a res2b_branch2a_relu				
res3a_branch1	res3a_branch1 bn3a_branch1 scale3a_branch1				
res3a	res3a res3a_relu				
res5b_branch2b	res5b_branch2b bn5b_branch2b scale5b_branch2b				
res5b	res5b res5b_relu				
res5a_branch1	res5a_branch1 bn5a_branch1 scale5a_branch1				
	res5a				

图11-15展示算子比对信息说明：

- 表示该层是否可以被比对。
- My Model(Operator)表示离线模型的算子名。
- Standard Model（Operator）表示Caffe的算子名。

融合算子的场景分类说明：

- 算子消除：OMG在构建图的过程中会将一些小算子融合为一个大算子。

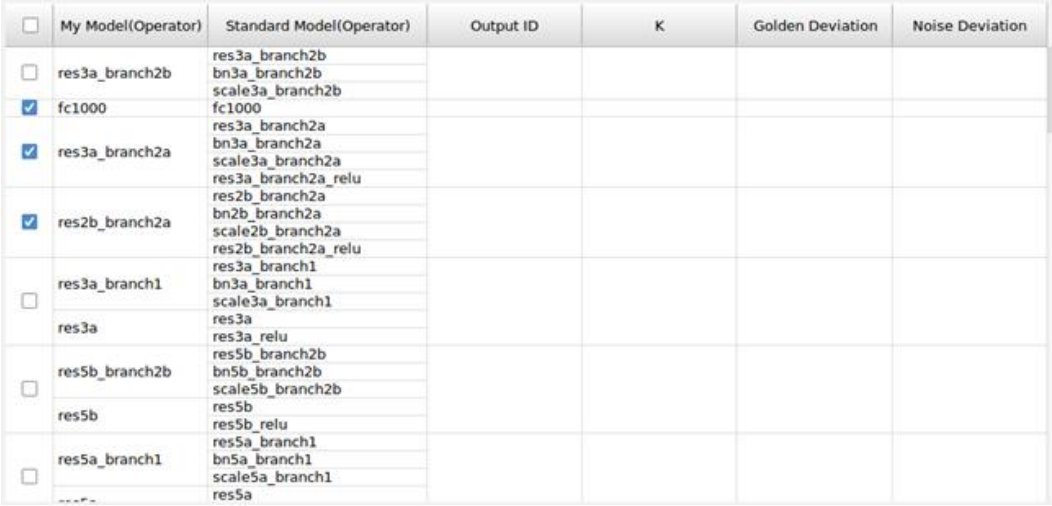
  如图中res3a_branch2b，res3a_branch2a，res2b_branch2a，res3a_branch1，res3a，res5b_branch2b等，为融合后的大算子。
- 算子融合：OMG支持UB融合、L1融合和L2融合，这些算子在模型中还存在，但是执行的时候作为一个整体执行，因此比对的时候也只能作为一个整体比对。

  如图中的res3a_branch1和res3a，res5b_branch2b和res5b。
- 算子新增：在模型转换过程中，OMG会增加一些算子，该算子没有对应右侧算子。
- 算子一致：算子不变，左侧算子和右侧算子的算子名一致。

  如果该算子为新增算子，或没有输入，那么该算子不能进行比对，即没有复选框可勾选。

步骤6 选择要比对的算子，如图11-16所示。

图 11-16 选择待比对层级算子



	My Model(Operator)	Standard Model(Operator)	Output ID	K	Golden Deviation	Noise Deviation
☐	res3a_branch2b	res3a_branch2b bn3a_branch2b scale3a_branch2b				
☑	fc1000	fc1000				
☑	res3a_branch2a	res3a_branch2a bn3a_branch2a scale3a_branch2a res3a_branch2a_relu				
☑	res2b_branch2a	res2b_branch2a bn2b_branch2a scale2b_branch2a res2b_branch2a_relu				
☐	res3a_branch1	res3a_branch1 bn3a_branch1 scale3a_branch1				
	res3a	res3a res3a_relu				
☐	res5b_branch2b	res5b_branch2b bn5b_branch2b scale5b_branch2b				
	res5b	res5b res5b_relu				
☐	res5a_branch1	res5a_branch1 bn5a_branch1 scale5a_branch1				
		res5a				

步骤7 单击“Compare”按钮，进行比对。

由于层级较多，需要等待一段时间后出比对结果。结果说明请参考结果展示。

----结束

结果展示

Lower Bound比对仅用户选择的要比对的层有结果，其他层级没有结果，且展示结果按照K值从大到小排序。

- My Model(Operator)表示离线模型的算子名。
- Standard Model（Operator）表示Caffe的算子名。
- Output ID表示离线模型算子的Output Id，一般从0开始。
- K表示几倍的扰动，范围在1.0到8.0之间，数值越大表明算子越差。
- Golden Deviation表示当前算子的离线模型结果跟Caffe结果的标准方差。
- Noise Deviation表示添加K倍扰动后的Caffe结果与标准Caffe结果的标准方差。

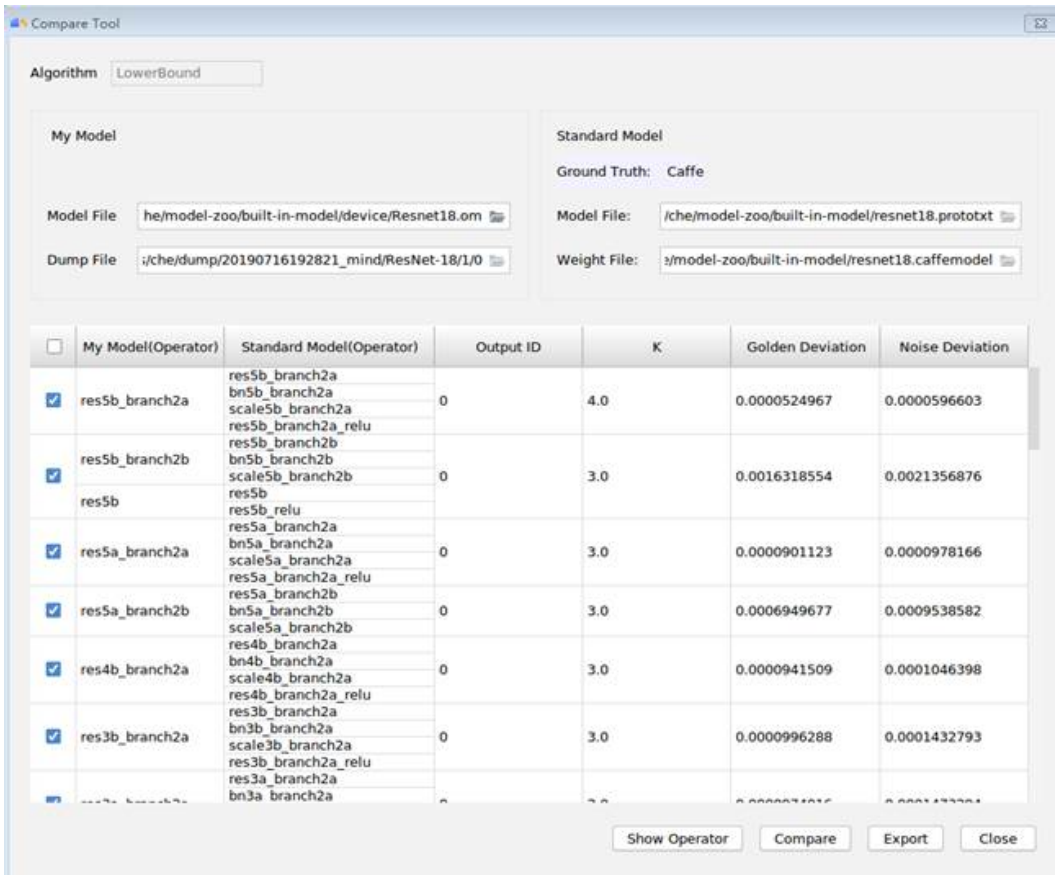
随着K的值变大，扰动倍数增加，直至Noise Deviation的值大于Golden Deviation值停止扰动，或者K达到最大值8的时候Noise Deviation的值仍然小于Golden Deviation值也停止扰动。K越大，扰动越大，说明当前算子效果越差。

说明

因安装Mind Studio的服务器操作系统环境不同（如Python版本及依赖库不同），可能会造成生成的同一个工程的部分算子的Golden Deviation与Noise Deviation值略微不同。

Lower Bound比对结果可能会随输入数据不同而波动，建议用户进行多次数据比对取均值。

图 11-17 Lower Bound 比对结果



11.4 Vector 比对

比对步骤

说明

- 约束：需要确保离线模型、离线模型的Dump文件与Caffe模型的Dump文件为相同模型的数据。如果不是相同模型的数据，但模型中有相同的算子名称，也可以进行比较，但结果数据只显示匹配到的相同算子的比对结果。
- Vector比对仅展示离线模型Dump数据与Caffe模型Dump数据两者相同的算子比对结果，两者交集外的算子不展示，如用户自定义算子。

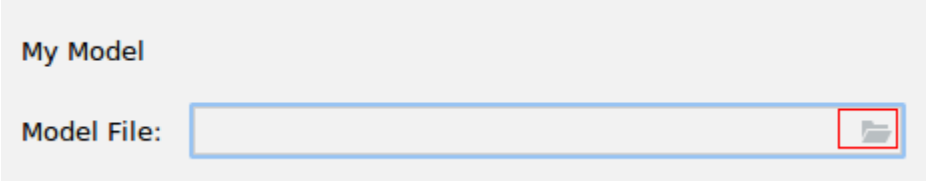
Vector比对操作步骤：

步骤1 在Mind Studio界面，在菜单栏选择“Tools > Compare Tool”菜单，进入比对工具界面。

步骤2 “Algorithm”选择“Vector”。

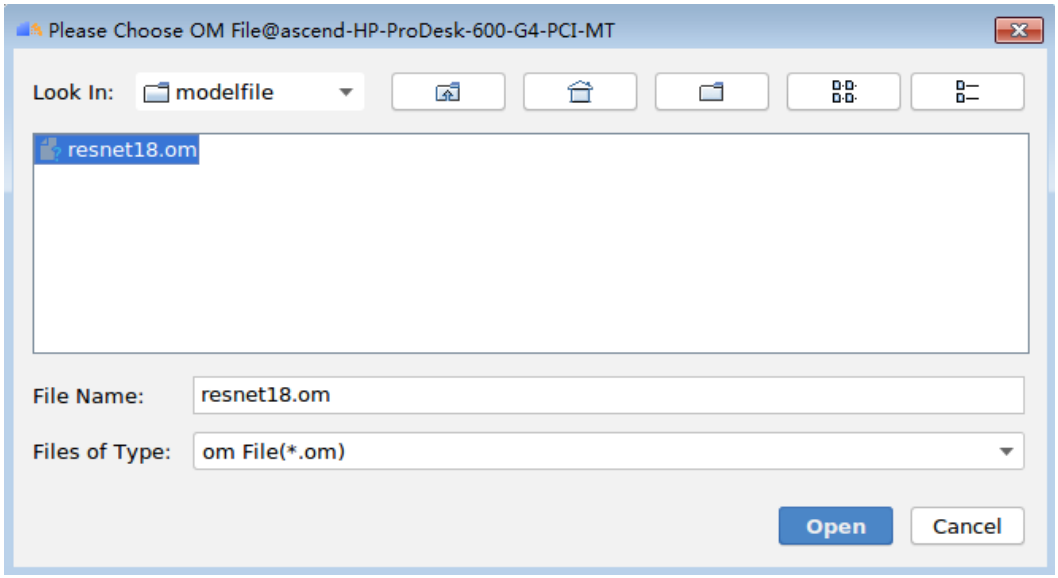
步骤3 单击My Model的“Model File”右侧的，设置离线模型文件，选择一个.om文件。如图11-18所示。

图 11-18 配置 My Model 的 Model File 参数



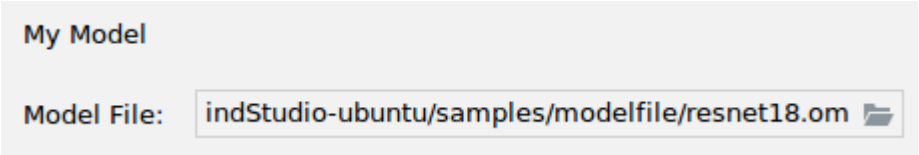
选择需要比对离线模型，然后单击“Open”，如图11-19所示。

图 11-19 选择离线模型



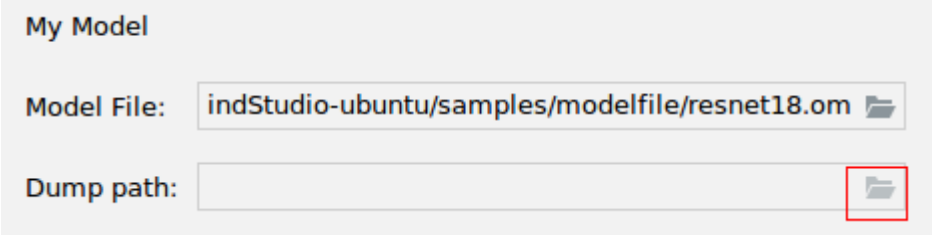
选择完成后，在文本框中显示选择的离线模型文件，如图11-20所示。

图 11-20 My Model 的 Model File 参数配置结果



步骤4 单击My Model的“Dump Path”右侧的，设置离线模型Dump数据路径，选择一个Dump数据文件夹。如图11-21所示。

图 11-21 配置 My Model 的 Dump Path 参数

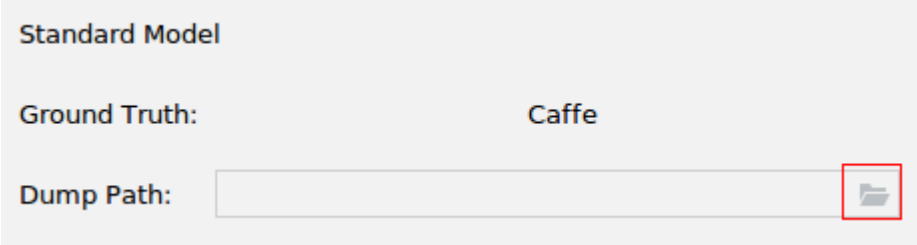


选择完成后，在文本框中显示选择的离线模型文件，如图11-22所示。

图 11-22 My Model 的 Dump Path 参数配置结果

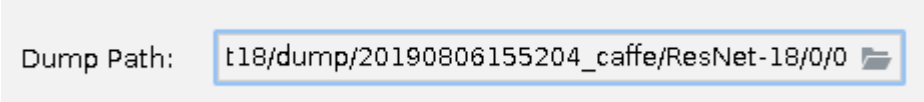
步骤5 单击Standard Model的“Dump Path”右侧的 ，设置Caffe模型Dump数据路径，选择一个Dump数据文件夹。如图11-23所示。

图 11-23 配置 Standard Model 的 Dump Path 参数



选择完成后，在文本框中显示选择的Dump数据路径如图11-24所示。

图 11-24 Right 的 Dump Path 参数配置结果



步骤6 单击“Compare”按钮。

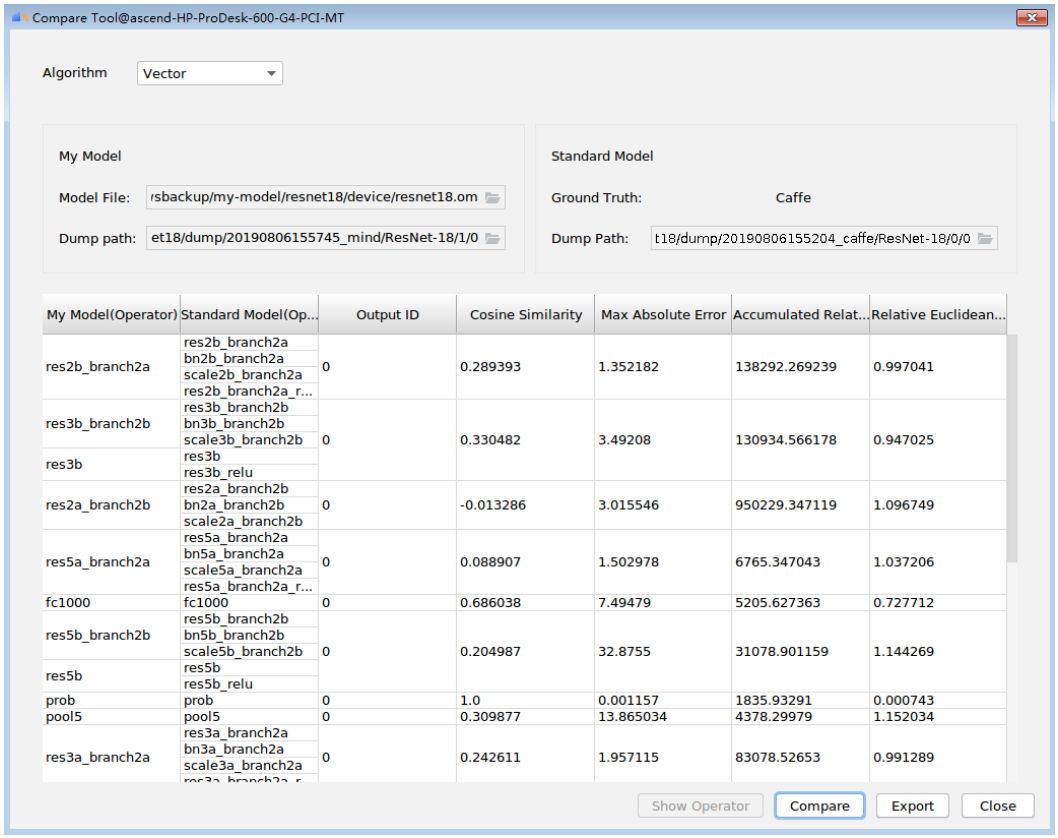
结果说明请参考结果展示。

----结束

结果展示

Vector比对结果及说明。

图 11-25 Vector 比对结果



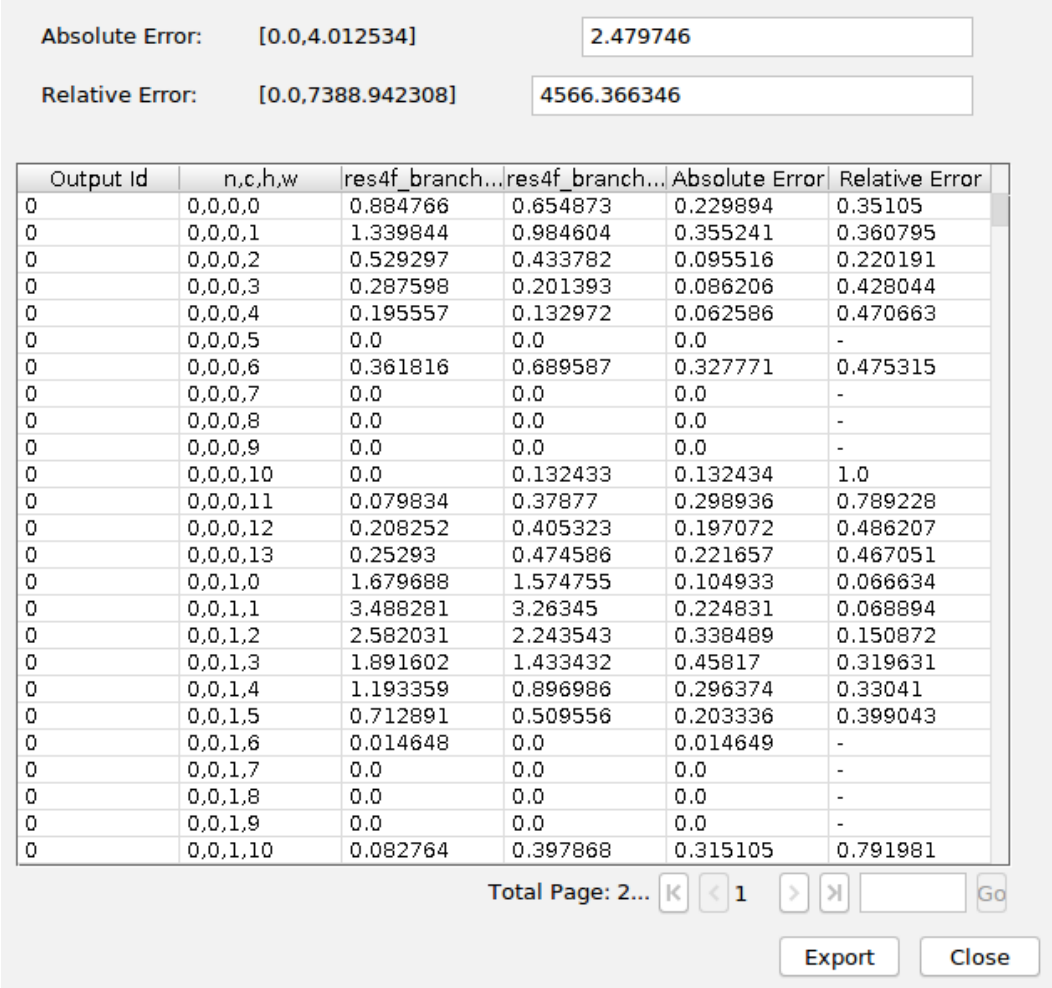
仅显示有结果的算子。可单击有 标识的列名，进行排序。

- My Model(Operator)：表示离线模型的算子名。
- Standard Model(Operator)：表示Caffe的算子名。
- Output ID：表示离线模型算子的Output ID，一般从0开始。
- Cosine Similarity：进行余弦相似度算法比对出来的结果，范围是[-1,1]，比对的结果如果越接近1，表示两者的值越相近，越接近-1意味着两者的值越相反。
- Max Absolute Error：进行最大绝对误差算法比对出来的结果，值越接近于0，表明越相近，值越大，表明差距越大。
- Accumulated Relative Error：进行累积相对误差算法比对出来的结果，值越接近于0，表明越相近，值越大，表明差距越大。
- Relative Euclidean Distance：进行欧氏相对距离算法比对出来的结果，值越接近于0，表明越相近，值越大，表明差距越大。

详细比对结果页面

单击Vector比对结果页面某个算子，弹出详细比对结果窗口，如**图11-26**所示。

图 11-26 详细比对结果



Output Id	n,c,h,w	res4f_branch...	res4f_branch...	Absolute Error	Relative Error
0	0,0,0,0	0.884766	0.654873	0.229894	0.35105
0	0,0,0,1	1.339844	0.984604	0.355241	0.360795
0	0,0,0,2	0.529297	0.433782	0.095516	0.220191
0	0,0,0,3	0.287598	0.201393	0.086206	0.428044
0	0,0,0,4	0.195557	0.132972	0.062586	0.470663
0	0,0,0,5	0.0	0.0	0.0	-
0	0,0,0,6	0.361816	0.689587	0.327771	0.475315
0	0,0,0,7	0.0	0.0	0.0	-
0	0,0,0,8	0.0	0.0	0.0	-
0	0,0,0,9	0.0	0.0	0.0	-
0	0,0,0,10	0.0	0.132433	0.132434	1.0
0	0,0,0,11	0.079834	0.37877	0.298936	0.789228
0	0,0,0,12	0.208252	0.405323	0.197072	0.486207
0	0,0,0,13	0.25293	0.474586	0.221657	0.467051
0	0,0,1,0	1.679688	1.574755	0.104933	0.066634
0	0,0,1,1	3.488281	3.26345	0.224831	0.068894
0	0,0,1,2	2.582031	2.243543	0.338489	0.150872
0	0,0,1,3	1.891602	1.433432	0.45817	0.319631
0	0,0,1,4	1.193359	0.896986	0.296374	0.33041
0	0,0,1,5	0.712891	0.509556	0.203336	0.399043
0	0,0,1,6	0.014648	0.0	0.014649	-
0	0,0,1,7	0.0	0.0	0.0	-
0	0,0,1,8	0.0	0.0	0.0	-
0	0,0,1,9	0.0	0.0	0.0	-
0	0,0,1,10	0.082764	0.397868	0.315105	0.791981

详细比对结果页面数值说明：

- Absolute Error：绝对误差；[0.0, 4.012534]表示绝对误差的范围；2.479746表示绝对误差的黄金分割点，若下表中的Absolute Error结果大于该值，显示为红色，用户可以进行编辑，输入的是整数或float，小数点后最多6位。
- Relative Error：相对误差；[0.0, 7388.942308]表示相对误差的范围；4566.366346表示相对误差的黄金分割点，若下表中的Relative Error结果大于该值，显示为红色，用户可以进行编辑，输入的是整数或float，小数点后最多6位。
- Output Id：output的index。
- n,c,h,w：数据格式。
- 第三列：My Model算子的Dump值。res4f_branch2b, …：My Model算子名。
- 第四列：Standard Model算子的Dump值。res4f_branch2b，...：Standard Model算子名。
- Absolute Error：绝对误差，My Model算子的Dump值减Standard Model算子的Dump值取绝对值比对出来的结果。
- Relative Error：相对误差，Absolute Error值除以Standard Model算子的Dump值比对出来的结果。

11.5 FAQ

11.5.1 Dump 数据丢失

故障现象

生成的Dump数据存在丢失，如图11-27所示，查看日志，有如图11-28所示的异常信息。

图 11-27 Dump 数据丢失

```
_home_qiong_modelzoo_dump_20191225220704_20191225220722_mind_VGG_ILSVRC_16_layers_1_2_2.VGG_ILSVRC_16_layers.1.33.conv1_1
_home_qiong_modelzoo_dump_20191225220704_20191225220722_mind_VGG_ILSVRC_16_layers_1_2_2.VGG_ILSVRC_16_layers.1.34.conv1_2
_home_qiong_modelzoo_dump_20191225220704_20191225220722_mind_VGG_ILSVRC_16_layers_1_2_2.VGG_ILSVRC_16_layers.1.35.pool1
_home_qiong_modelzoo_dump_20191225220704_20191225220722_mind_VGG_ILSVRC_16_layers_1_2_2.VGG_ILSVRC_16_layers.1.36.conv2_1
_home_qiong_modelzoo_dump_20191225220704_20191225220722_mind_VGG_ILSVRC_16_layers_1_2_2.VGG_ILSVRC_16_layers.1.37.conv2_2
_home_qiong_modelzoo_dump_20191225220704_20191225220722_mind_VGG_ILSVRC_16_layers_1_2_2.VGG_ILSVRC_16_layers.1.40.conv3_2
_home_qiong_modelzoo_dump_20191225220704_20191225220722_mind_VGG_ILSVRC_16_layers_1_2_2.VGG_ILSVRC_16_layers.1.43.conv4_1
_home_qiong_modelzoo_dump_20191225220704_20191225220722_mind_VGG_ILSVRC_16_layers_1_2_2.VGG_ILSVRC_16_layers.1.46.pool4
_home_qiong_modelzoo_dump_20191225220704_20191225220722_mind_VGG_ILSVRC_16_layers_1_2_2.VGG_ILSVRC_16_layers.1.48.conv5_2
_home_qiong_modelzoo_dump_20191225220704_20191225220722_mind_VGG_ILSVRC_16_layers_1_2_2.VGG_ILSVRC_16_layers.1.50.pool5
```

图 11-28 数据队列满

```
toolchain/ide/ide-daemon/IDE-daemon-host/ide_daemon_host.cc:826 >>> Recevie new data
-13:57:17.705.844 DoDump:framework/domi/ome/model_manager/model_dump_manager.cpp:701:"dumpCostTime:1"
-13:57:17.705.958 [drvHdcClientSessionClose:530] >>> destroy client session(sock: 5)
toolchain/ide/ide-daemon/IDE-daemon-host/ide_daemon_host.cc:840 >>> received ctrl msg: end
04 toolchain/ide/ide-daemon/IDE-daemon-host/ide_daemon_host.cc:629 >>> the dump data queue is full. please reduce model batch, images or dump layers.
-13:57:17.706.017 DoDump:framework/domi/ome/model_manager/model_dump_manager.cpp:711:"dump file /home/qiong/modelzoo/dump/20191219135417/20191219135434_minc
drvHdcSocketSend:545] >>> send (sockfd 27)error: Broken pipe(errno: 32) -1 276 4
toolchain/ide/ide-daemon/IDE-daemon-host/../hdc-common/hdc_api.cc:345 >>> drvHdcSend, error: 27
toolchain/ide/ide-daemon/IDE-daemon-host/../hdc-common/ide_daemon_msg.cc:136 >>> IdeWrite failed, err: -1
toolchain/ide/ide-daemon/IDE-daemon-host/ide_daemon_host.cc:871 >>> IdeSendResponse failed, err: 10
```

故障处理

原因分析：开发板场景下，因写磁盘速度较慢导致在Dump文件时堵塞在写盘队列中，导致Dump队列满，数据丢失。

处理方法：降低Dump的batch数，或减少算子比对的图片数，或减少Dump数据算子层数。

12 术语和缩略语

12.1 术语/缩略语

12.1 术语/缩略语

表 12-1 术语/缩略语

缩略语	全称	含义
A		
AI	人工智能（artificial intelligence）	研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新的技术科学。
AIPP	AI预处理（artificial intelligence pre-processing）	支持格式转换、Padding/Crop等处理操作；CSC色域转换：YUV2RGB或者RBG2YUV；padding；抠图；Scale UP/Down；通道数据交换等
ACL	Ascend Computing Language	提供Device管理、Context管理、Stream管理、内存管理、模型加载与执行、算子加载与执行、媒体数据处理等C++ API库供用户开发深度神经网络应用，用于实现目标识别、图像分类等功能。
ADA	Ascend Debug Agent	守护进程主机。
ADC	Ascend Debug Client	守护进程客户端。
ADX	Ascend Debug For X	守护进程设备。
ATC	Ascend Tensor Compiler	用户使用Caffe等框架训练好的模型，通过ATC将其转换为昇腾AI处理器海思SOC支持的离线模型，实现算子调度的优化，权值数据重排，内存使用优化等可以脱离设备完成的预处理功能。

缩略语	全称	含义
B		
BIOS	基本输入输出系统（basic input output system）	存于计算机主板上的一种固件。包括基本输入输出控制程序、上电自检程序、系统启动自举程序、系统设置信息，为计算机提供底层的硬件设置和控制功能
BIU	总线接口单元（bus interface unit）	记录AICore和DDR/L2之间的内存访问情况
BTB	板对板连接器（board-to-board connector）	BTB连接器用于连接印刷电路板（PCB）。
C		
CAN	控制区域网络（Controller Area Network）	CAN是一种串行通讯总线，能有效地支持具有很高安全等级的分布式实时控制。
CCE	基于Cube的计算引擎（cube-based computing engine）	CCE加速库通过API的方式，为上层应用（为机器学习提供的各种framework或者Application）提供加速。
CCE-GDB	CCE调试器（CCE-GNU debugger）	CCE-GDB 用来调试AI应用程序，同时支持控制CPU、AICore及AICPU侧的代码调试（AICPU的调试还在开发过程中）。
CCEC	CCE编译器（CCE compiler）	CCE Compiler是一个异构系统编译器，是CCE异构编程语言的编译工具，它编译CCE混合代码：CCE Host代码和CCE AICPU|AICORE设备代码，产生可以在CCE系统上运行的可执行文件。
CFM	立方英尺每分钟（cubic feet per minute）	测量气体流速的单位。
CNN	卷积神经网络（convolutional neural network）	卷积神经网络，是一种前馈神经网络，人工神经元可以响应周围单元，可以进行大型图像处理。
CPU	中央处理单元（central processing unit）	中央处理器是计算机的主要设备之一，其功能是解释计算机指令以及处理计算机软件中的数据，与内部存储器、输入及输出设备成为现代电脑的三大部件。
CUBE	-	CUBE 是AI Core中的运算单元，主要处理矩阵乘、加运算。
D		

缩略语	全称	含义
DDK	设备开发套件（device development kit）	DDK是Mind解决方案提供的开发者套件包，通过安装DDK后获得Mind开发必需的API、库、工具链等开发组件。
DDR	双倍数据速率（double data rate）	与传统的单数据速率相比，DDR技术实现了一个时钟周期内进行两次读/写操作，即在时钟的上升沿和下降沿分别执行一次读/写操作。
DL	深度学习（deep learning）	是机器学习的分支，是一种试图使用包含复杂结构或由多重非线性变换构成的多个处理层对数据进行高层抽象的算法。
E		
ECC	错误检查和纠错技术（error checking and correction）	该技术在原来的数据位中增加校验位，通过检验位来检测数据错误，并具备错误纠正能力。
EMMC	多媒体存储卡（embedded multimedia card）	嵌入式多媒体卡，是一种新的存储技术，外部提供的接口和SD卡接口类似，内部存储介质为Flash，并且内建坏块管理系统。
F		
FLOPS	每秒所执行的浮点运算次数（floating-point operations per second）	常被用来估算电脑的执行效能，尤其是在使用到大量浮点运算的科学计算领域中。正因为FLOPS字尾的那个S，代表秒，而不是复数，所以不能省略掉。
G		
GDB	GNU调试器（GNU debugger）	UNIX及UNIX-like下的命令行调试工具，可以执行程序、管理断点、检查变量赋值、调用函数等。
GE	Graph Engine	Graph Engine提供了Graph/Operator IR作为安全易用的构图接口集合，用户可以调用这些接口构建网络模型，设置模型所包含的图、图内的算子、以及模型和算子的属性。
GPU	图形处理器（graphics processing unit）	GPU又称视觉处理器、显示芯片等，更多资源用于数据计算，适合具备可预测、针对数组的计算模式。
H		
HCCL	华为集合通信库（Huawei Collective Communication Library）	HCCL提供了深度学习训练场景中服务器间高性能集合通信的功能。

缩略语	全称	含义
HDC	主机设备通信（host device communication）	用于Host和Device之间通信模块，在Host和Device里面均有部署。
HDR	高动态范围（high dynamic range）	摄影术语，用来描述媒体应用，如数字影像和数字音频。
I		
I2C	集成电路总线（inter-integrated circuit）	I2C总线允许在同一电路板上的组件之间轻松通信。
IDE	集成开发环境（integrated development environment）	集成开发环境。
IFU	取指单元（instruction fetch unit）	记录每一次icache访问情况
IPC	IP摄像机（IP camera）	IP摄像机
IR	中间表示（intermediate representation）	IR是一种数据结构，可将输入的资料建构为一个计算机程序，也可以将一部分或是所有输出的程式反推回输入资料。
ISP	图像信号处理（image signal processing）	主要用来对前端图像传感器输出信号处理的单元，以匹配不同厂商的图象传感器。
IVS	智能视频监控系统（intelligent video surveillance）	华为公司提供的集管理、存储、智能分析及应用、编解码为一体的智能视频监控系统。
IMU	IO板管理单元（I/O board management unit）	-
J		
JPEGD	JPEG图像解码器（JPEG decoder）	提供对JPEG格式的图像进行解码的能力。
JPEGE	JPEG图像编码器（JPEG encoder）	提供对图像进行编码输出为JPEG格式的能力。
L		
LAN	局域网（local area network）	由处于同一建筑或方圆几公里范围内的个人计算机和工作站相连接而组成的网络，具有高速和低错误率的特点。
LLC	最后一级Cache（last level cache）	在访问内存之前调用的共享最高级别缓存通常称为最后一级缓存（LLC）。
LPDDR4x	低功耗内存技术（Low-Power DDR4x）	面向低功耗内存而制定的通信标准，以低功耗和小体积著称，专门用于移动式电子产品。

缩略语	全称	含义
M		
Matrix	-	Matrix是一个通用业务流程执行引擎，主要包含Agent（运行在Host侧）和Manger（运行在Device侧）两个部分。
MDC	移动数据中心（mobile data center）	用于自动驾驶的移动数据中心。
Mic	麦克风（microphone）	麦克风
ML	机器学习（machine learning）	机器学习是实现人工智能的一个途径，即以机器学习为手段解决人工智能中的问题。
MLL	机器学习库（machine learning library）	基于opencv算子通过算法优化、neon指令等方式大幅提升opencv算子性能的机械学习库。
MTE1	内存传输引擎1（memory transfer engine 1）	从L1 BUFFER拷贝内存
MTE2	内存传输引擎2（memory transfer engine 2）	从DDR或者L2 BUFFER拷贝内存
MTE3	内存传输引擎3（memory transfer engine 3）	从UB拷贝内存
N		
NIC	网络接口控制器（network interface controller）	网络接口控制器（NIC，也称为网络接口卡，网络适配器，LAN适配器，以及类似术语）是将计算机连接到计算机网络的计算机硬件组件。
NN	神经网络（neural network）	神经网络在机器学习和认知科学领域，是一种模仿生物神经网络的结构和功能的数学模型或计算模型。
NPU	神经网络处理器单元（neural-network processing unit）	采用“数据驱动并行计算”的架构，特别擅长处理视频、图像类的海量多媒体业数据，专门用于处理人工智能应用中的大量计算任务。
NV	永久性（nonvolatile）	数据一旦写入NV，即使掉电也不会丢掉，下次重启，仍然会保留原有设置。
O		
OP	算子（Operator）	操作运算，比如AI的Relu、Conv、fc、pooling、scale、softmax等。

缩略语	全称	含义
OS	操作系统（operating system）	-
OTG	On-The-Go	主要应用于各种不同的设备或移动设备间的联接，进行数据交换。
P		
PCB	印刷电路板（printed circuit board）	含有按预先设计形成的印制元件或印制线路以及两者结合的导电图形的印制板。
PCIe	快捷外围部件互连标准（PCI Express，peripheral component interconnect express）	PCIe属于高速串行点对点双通道高带宽传输，所连接的设备分配独享通道带宽，不共享总线带宽，主要支持主动电源管理，错误报告，端对端的可靠性传输，热插拔以及服务质量(QOS)等功能。
PMU	性能监视单元（performance monitor unit）	PMU是CPU提供的一个单元，属于硬件的范畴。PMU通过访问相关的寄存器能读取到CPU的一些性能数据。
PNGD	PNG图像解码器（PNG decoder）	提供对PNG格式的图像进行解码的能力。
PWM	脉冲宽度调制（pulse width modulation）	脉冲载波的脉冲持续时间脉宽随调制波的样值而变的脉冲调制方式。
R		
RAM	随机存储器（random access memory）	基于半导体的可被CPU或者其他硬件设备读写的内存。可以任何顺序访问存储位置。
RC	根组件（root complex）	在PCI Express（PCIe）系统中，根组件设备将处理器和存储器子系统连接到由一个或多个交换设备组成的PCI Express交换结构。类似于PCI系统中的主机桥，根组件代表处理器生成事务请求，处理器通过本地总线互连。根组件功能可以实现为分立设备，或者可以与处理器集成。
RGMII	精简的千兆比媒介独立接口（reduced gigabit media independent interface）	-
Runtime	-	Runtime运行于APP进程空间，为APP提供了Ascend 310设备的Memory管理、Device管理、Stream管理、Event管理、Kernel执行等功能。
S		

缩略语	全称	含义
Scalar	-	标量，一般表示一个常数。
SDK	软件开发工具包（software development kit）	一般都是一些软件工程师为特定的软件包、软件框架、硬件平台、操作系统等建立应用软件时的开发工具的集合。
SoC	片上系统（system on chip）	这是ENP成本低的关键技术，华为通过强大的芯片技术能力，把一个交换机单板的包转发功能全部集成到了一个芯片中，获得了软件灵活性+交换机的低成本。
SPI	串行外设接口（serial peripheral interface）	SPI总线系统是一种同步串行外设接口，它可以使MCU（Microcontroller Unit，微控制单元）与各种外围设备以串行方式进行通信以交换信息。
T		
TBE	张量加速引擎（Tensor Boost Engine）	提供通过Python语言实现算子的接口，能够编译生成CCE算子。
TEE	可信执行环境（trusted execution environment）	在ARM Trustzone的硬件隔离环境基础上，结合硬件可信根设计，实现安全启动、安全存储、安全升级、安全运行等功能，为系统提供可信的基础运行环境。
Tensor	-	张量是TensorFlow程序中的主要数据结构。张量是N维（其中N可能非常大）数据结构，最常见的是标量、向量或矩阵。张量的元素可以包含整数值、浮点值或字符串值。
Tops	每秒万亿次的运算（trillion operations per second）	用于衡量CPU、GPU、NPU的计算能力。
TS	任务调度（task scheduler）	通过Task Schedule分发不同的kernel到AI CPU/AI Core执行。
TVM	张量虚拟机（tensor virtual machine）	TVM提供内置算子和自定义算子扩展。支持Caffe、Tensorflow等开源框架。
U		
USB	通用串行总线（Universal Serial Bus）	带接口设备的串行总线的标准。

缩略语	全称	含义
UART	通用异步收发传输器（universal asynchronous receiver/transmitter）	UART是用于控制计算机与串行设备的芯片。它提供了RS-232C数据终端设备接口，这样计算机就可以和调制解调器或其它使用RS-232C接口的串行设备通信。
V		
VCM	视频内容管理系统（video content management）	华为公司提供的视频内容管理平台，依托于华为领先的智能图像处理及大数据分析技术，定位与视频结构化分析能力平台，提供高性能硬件、丰富的算法集成接口，有效提升图像侦查效率及业务协同能力，协助案件快速侦破，助力全球城市安全。
VDEC	视频解码器（video decoder）	提供对特定格式的视频进行解码的能力。
VENC	视频编码器（video encoder）	提供对特定格式的视频进行编码的能力。
VECTOR	-	向量运算
VPC	视觉预处理核心（vision preprocessing core）	提供对图像进行缩放、色域转换、降bit数处理、存储格式转换、区块切割转换等能力。
Y		
YUV	度-带宽-色度（luminance-bandwidth-chrominance）	根据三基色原理(RGB介绍)，利用R(红)、G(绿)、B(蓝)三色不同比例的混合可以表示各种色彩，极大地加宽视频信号带宽，增加相关设备成本。

13 附录

13.1 软件包说明
13.2 caffe.proto文件算子定义
13.3 设置Mind Studio支持自动解析TBE Python依赖库
13.4 开源第三方库说明
13.5 修订记录

13.1 软件包说明

Mind Studio以及DDK软件包解压后包的详细内容请参见**表13-1**。

表 13-1 解压后软件包说明

软件包	包内容	使用场景
MindStudio-ubuntu.tar.gz	MindStudio-ubuntu： ● bin：Mind Studio的执行目录及依赖的二进制文件 ● build.txt：软件包构建信息 ● config：Mind Studio配置文件 ● Install-Linux-tar.txt：Mind Studio安装说明 ● jre64：64位系统的依赖库以及文件 ● lib：JAVA依赖库 ● license：所用的第三方依赖专利文档 ● LICENSE.txt：Apache Licence说明文档 ● NOTICE.txt：注意事项 ● plugins：Mind Studio所用到的基本插件及JAVA库 ● product-info.json：Mind Studio版本号以及执行文档路径 ● redist：Java辅助注解 ● samples：算子样例工程模板文件以及典型网络模型的模型文件 ● tools：工具库，例如ide_daemon，profiler等	Mind Studio工具软件包
Ascend_DDK-{software version}-{interface version}-<uihost arch.os>.tar.gz 各参数解释请参见**表13-2**。	● ddk.tar.gz：DDK软件包。 ● ddk_setup.py：安装脚本。 ● check_sha.sh：校验以上两个文件的完整性，安装Mind Studio时，安装脚本会自动加载DDK软件包，加载过程中会自动调用该脚本进行完整性校验。	DDK软件包

表 13-2 DDK 软件包命名规范

参数	说明
{software version}	软件版本号。
{interface version}	DDK内部版本号。
<uihost arch.os>	Mind Studio安装服务器CPU架构和操作系统及版本，本手册使用的为x86_64.ubuntu16.04。

表 13-3 lib 包说明

软件包	包内容	使用场景
Ascend_lib-{software version}-<host arch.os>-<device arch.os>.zip 各参数解释请参见表13-4。	EP\host-host arch.os	Atlas 300形态下存在： host侧lib库。
	EP\device-device arch.os	Atlas 300形态下存在： device侧lib库。
	RC\host-aarch64_ubuntu16.04	Atlas 200 DK形态下存在： host侧（host侧和device侧合并）lib库。
	check_sha.sh	校验以上文件的完整性。

表 13-4 lib 包命名规范

参数	说明
{software version}	软件版本号，需要和DDK包中的该参数版本号保持一致。
<host arch.os>	host侧操作系统架构。
<device arch.os>	device侧操作系统架构。

表 13-5 Host 侧操作系统和架构对应的 lib 包

host侧操作系统以及架构	lib包
Ubuntu系统	Ascend_lib-{software version}-x86_64.ubuntu16.04-aarch64.miniOS.zip：Atlas 300形态场景对应的lib包。 Ascend_lib-{software version}-aarch64.ubuntu16.04-aarch64.ubuntu16.04.zip：Atlas 200 DK形态场景对应的lib包。
CentOS x86系统	Ascend_lib-{software version}-x86_64.centOS7.4-aarch64.miniOS.zip
CentOS Arm系统	Ascend_lib-{software version}-aarch64.centOS7.6-aarch64.miniOS.zip

13.2 caffe.proto 文件算子定义

DDK中内置的caffe.proto文件的路径为DDK安装目录下的：“/ddk/include/inc/custom/proto/caffe/caffe.proto”，用户可获取此文件进行修改，添加自定义算子的定义。

此处以添加Reduction算子的定义为例。

步骤1 在LayerParameter中添加Reduction算子的定义。

如下所示，在LayerParameter中加入Reduction的定义，并设置其ID。

```
message LayerParameter {
......
optional ReductionParameter reduction_param = 136;   //此ID需要保持唯一
......
}
```

步骤2 在caffe.proto文件中增加Reduction的参数定义。

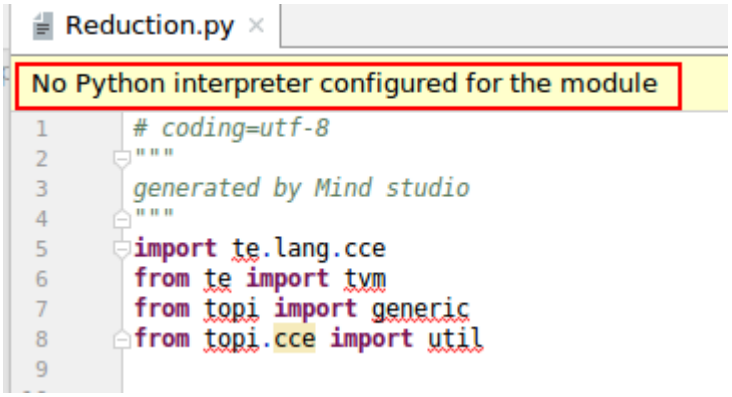
```
message ReductionParameter {
  enum ReductionOp {          //算子支持的操作类型
    SUM = 1;                    //对被reduce的所有轴求和
    ASUM = 2;                   //对被reduce的所有轴求绝对值后求和
    SUMSQ = 3;                  //对被reduce的所有轴求平方后再求和
    MEAN = 4;                   //对被reduce的所有轴求均值
  }
  optional ReductionOp operation = 1 [default = SUM];   // 定义算子的操作
  optional int32 axis = 2 [default = 0];                //定义reduce的轴
  optional float coeff = 3 [default = 1.0]; // coefficient for output  //标量，对结果进行缩放的倍数
}
```

----结束

13.3 设置 Mind Studio 支持自动解析 TBE Python 依赖库

由于TBE算子工程依赖TBE的Python插件，但Mind Studio的Python解析器无法获取这些Python插件的安装路径，导致工程依赖的Python库解析失败如下图所示。

图 13-1 Mind Studio 无法解析 TBE 的 Python 库



所以开发者需要手工设置TBE的Python库的环境变量，并在Mind Studio中加载这些Python库。

若开发者已经按照**首次设置**配置好了环境变量及Python解析器，当再次新建TBE工程后，请参照**非首次设置**进行配置。

首次设置

步骤1 以Mind Studio安装用户登录Mind Studio安装服务器，并设置环境变量。

执行如下命令在文件最后添加如下环境变量。

```
export DDK_PATH=$HOME/.mindstudio/huawei/ddk/1.31.T8.B080/ddk
export PYTHONPATH=$DDK_PATH/site-packages/te-0.4.0.egg:$DDK_PATH/site-packages/
topi-0.4.0.egg
export LD_LIBRARY_PATH=$DDK_PATH/uihost/lib
```

说明

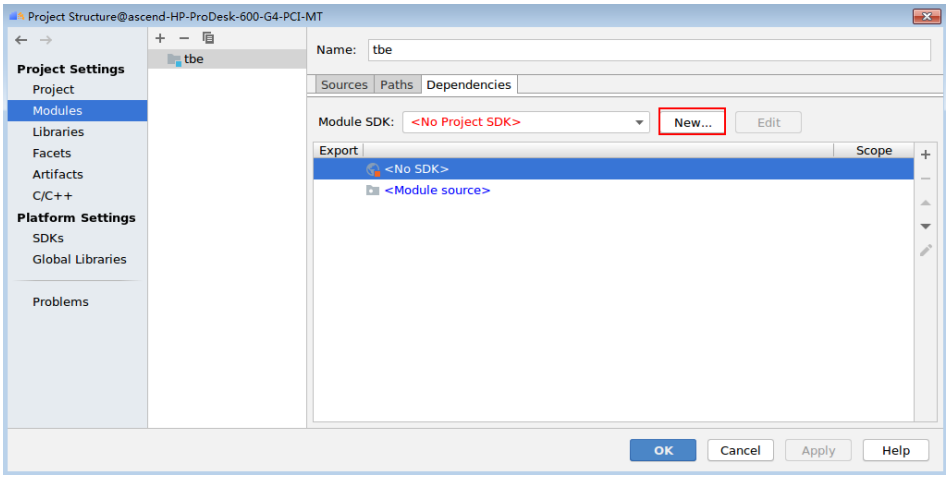
DDK_PATH请替换为实际的DDK部署路径。

步骤2 请在执行**步骤1**的命令窗口中重新启动Mind Studio，确保Mind Studio可以获取到**步骤1**设置的环境变量。

步骤3 在TBE工程中设置依赖的Python SDK。

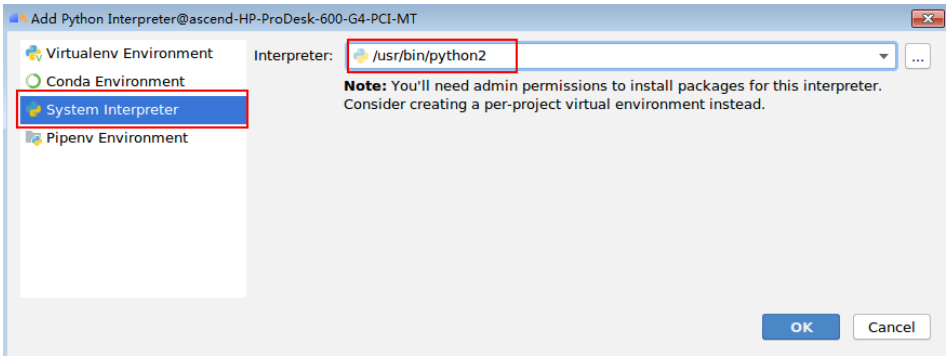
1. 打开TBE工程。
2. 选中工程界面中，单击菜单栏中的“File > Project Structure”，进入**Project Structure**设置页面。
3. 在左侧菜单栏中选择“Project Settings > Modules”，进行Modules设置。
4. 在Modules配置界面中，选择**Dependencies**页签，如**图13-2**所示。

图 13-2 Modules 设置界面



5. 在**图13-2**所示界面中单击**New**，在下拉菜单中选择**Python SDK**，进入**Add Python Interpreter**窗口。
6. 在**Add Python Interpreter**窗口左侧导航栏选择**System Interpreter**，右侧窗口中**Interpreter**下拉选择**/usr/bin/python2**，如下图所示。

图 13-3 Add Python Interprete

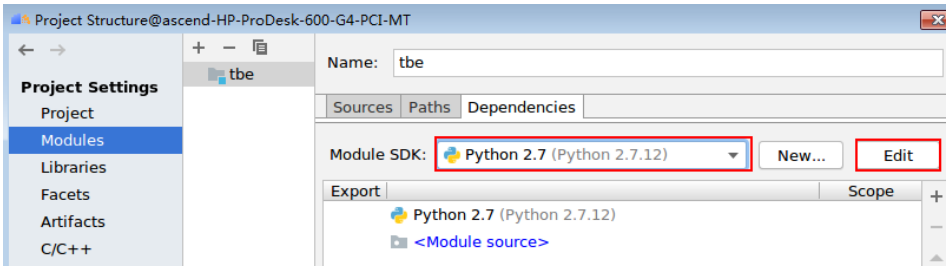


说明

- System Interpreter：使用本地解释器（请使用此方式）。
- 若选用以下三种方式，您需要提前安装依赖并配置python镜像，安装依赖包括conda、pipenv等。
  1. Virtualenv Environment：使用虚拟管理环境，在创建Python虚拟环境的目录下创建独立的Python库和解释器，创建好后，会在创建目录下重新根据当前用户的~/.pip/pip.conf文件中配置的PIP源通过PIP工具进行库的下载。
  2. Conda Environment：选择Anaconda python解释器并利用conda创建虚拟环境。
  3. Pipenv Environment：整合了virtualenv与pip，可同时创建虚拟环境并进行库管理功能。

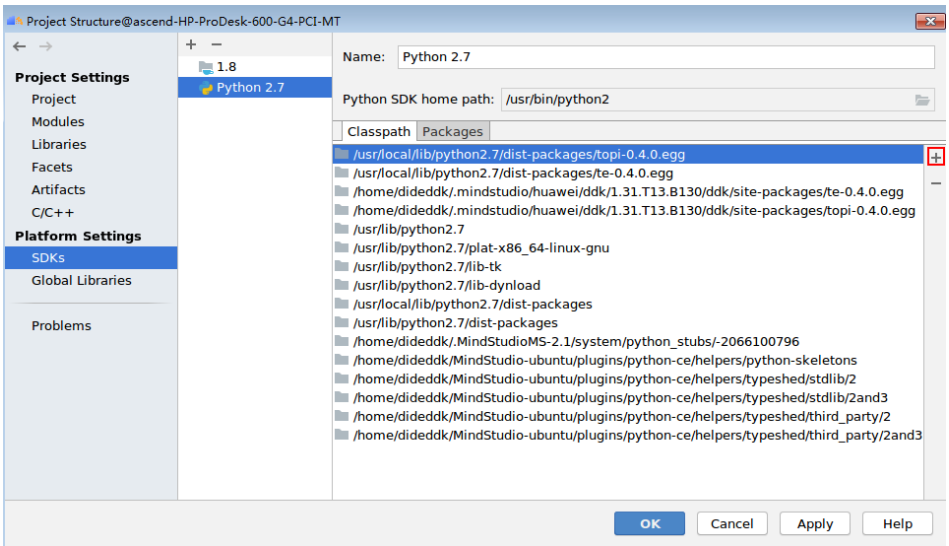
7. 单击**OK**，弹出**Set up created SDK on project?**提示信息，单击**Yes**。
8. 在**Modules > Dependencies**中选择创建的**Python 2.7**，单击**Edit**。

图 13-4 编辑 Module SDK



9. 在进入的SDKs编辑界面中， 选中Classpath页签，单击 + 。

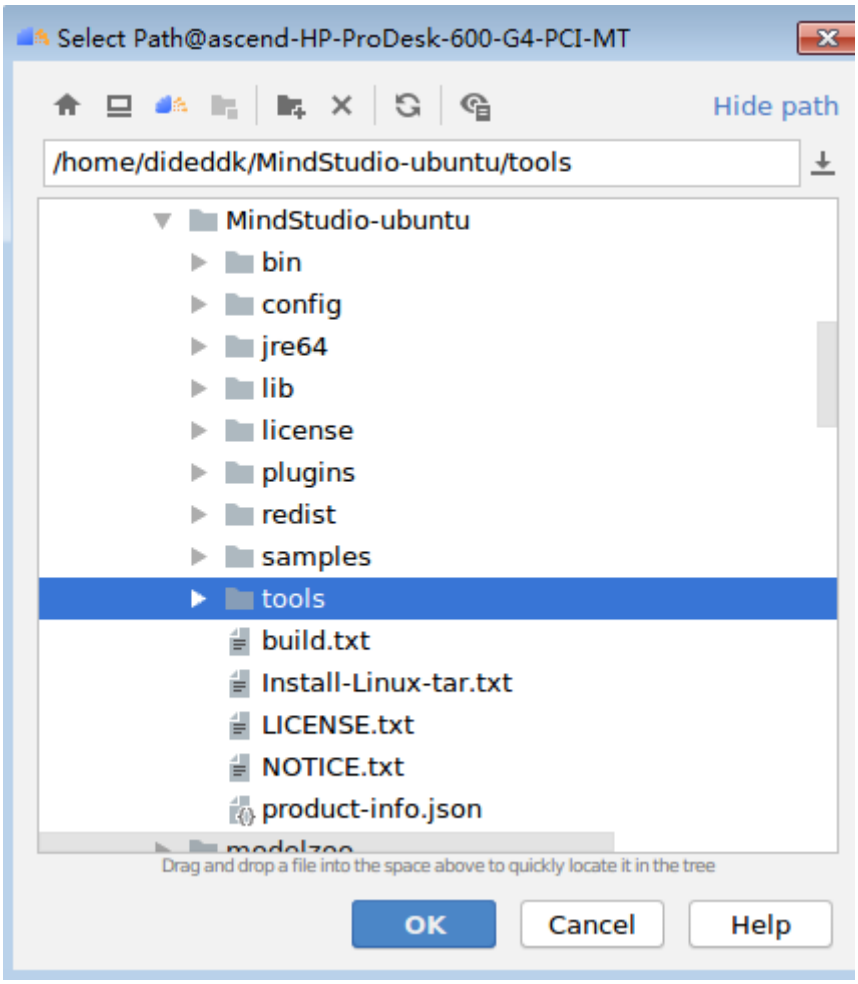
图 13-5 编辑 SDKs



10. 在弹出的Select Path界面中，选择Mind Studio安装路径下的**MindStudio-ubuntu/tools**目录，然后单击**OK**。

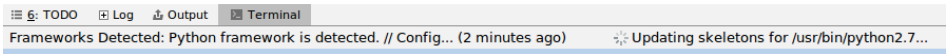
    算子工程的**test/testcase.py**文件中导入的python库在**MindStudio-ubuntu/tools**目录下，所以需要在SDKs中添加此依赖库，若开发者有自己开发的第三方依赖Python库，也可按照此方式添加。

图 13-6 选择依赖的 Python 库所在路径



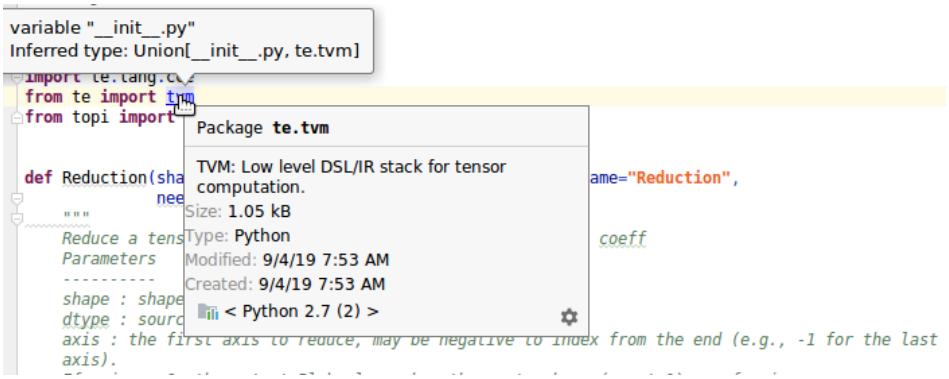
11. 在SDKs编辑界面单击**Apply**，单击**OK**，关闭**Project Structure**窗口。
此时，在工程底部菜单栏中会显示如图13-7所示提示。

图 13-7 更在更新界面



12. 更新成功后，如图13-8所示，单击导入的Python模块代码，然后按**Ctrl+Q**，会出现相应的Tips。按住Ctrl建，单击还可进入对应的模块定义文件。

图 13-8 Python 代码自动解析示例



说明

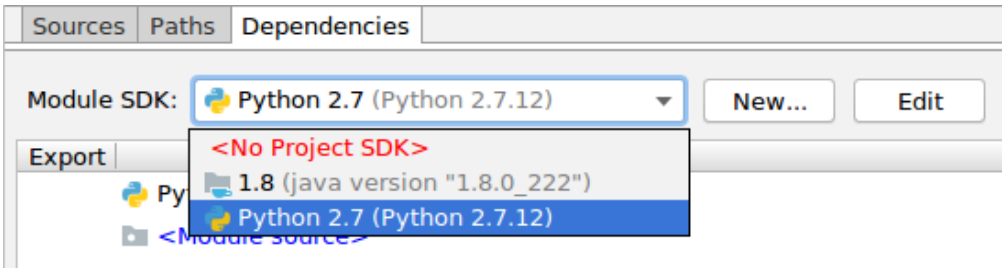
若仍然无法自动解析，请关闭工程后重新打开。

----结束

非首次设置

1. 选中工程界面中，单击菜单栏中的“File > Project Structure”，进入**Project Structure**设置页面。
2. 在左侧菜单栏中选择“Project Settings > Modules”，进行Modules设置。
3. 在图13-9所示界面中Module SDK处下拉选择Python 2.7。

图 13-9 选择 Module SDK



4. 在**Project Structure**窗口中单击**Apply**。
5. 单击**OK**，关闭**Project Structure**窗口。

   此时Python代码可以实现自动解析，如图13-8所示。

   说明

   若仍然无法自动解析，请关闭工程后重新打开。

13.4 开源第三方库说明

表 13-6 开源软件列表

开源软件列表	当前使用版本	官方地址
cereal	1.2.2	http://uscilab.github.io/cereal/
gflags	2.2.1	https://github.com/gflags/gflags
glog	0.3.5	https://github.com/google/glog
opencv	3.4.2	https://opencv.org/
Protobuf	3.5.1	https://developers.google.com/protocol-buffers/

开源软件列表	当前使用版本	官方地址
IntelliJ IDEA - Community Edition	2019.1.2	https://www.jetbrains.com/idea/
gson	2.8.5	https://github.com/google/gson
FindBugs-jsr305	3.0.2	http://findbugs.sourceforge.net
Xchart	3.5.4	http://knowm.org/open-source/xchart
fastjson	1.2.56	https://github.com/alibaba/fastjson/
jgraphx	3.4.1.3	https://github.com/jgraph/jgraphx
guava	26.0	https://code.google.com/p/guava-libraries/@https://github.com/google/guava
zip4j	2.1.0	http://www.lingala.net/zip4j/index.php
SQLite JDBC	3.23.1	https://github.com/xerial/sqlite-jdbc/
google-guice	4.2	https://github.com/google/guice
animal-sniffer-annotations	1.17	http://www.mojohaus.org/animal-sniffer/
llvm	8.0	http://clang.llvm.org/
flexmark-java	0.34.58	https://github.com/vsch/flexmark-java
antlr	4.7.2	https://www.antlr.org/
Eclipse Xtext	2.19.0.M2	https://www.eclipse.org/Xtext/
eclipse lsp4j	0.6.0	https://github.com/eclipse/lsp4j
lsp4intellij	0.91.0	https://github.com/ballerina-platform/lsp4intellij
cmake	3.14.1	https://cmake.org/

13.5 修订记录

发布日期	修订记录
2020-05-12	第一次正式发布。