

华为认证计算系列教程

HClE-Computing

智算环境建设及模型运行实验指导手册

版本：2.0



华为技术有限公司

版权所有 © 华为技术有限公司 2025。保留一切权利。

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

商标声明



和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

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

华为技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<https://e.huawei.com>

华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构，打造的覆盖ICT（Information and Communications Technology，信息技术）全技术领域的认证体系，包含ICT基础设施认证、基础软硬件认证、云平台及云服务认证三类认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

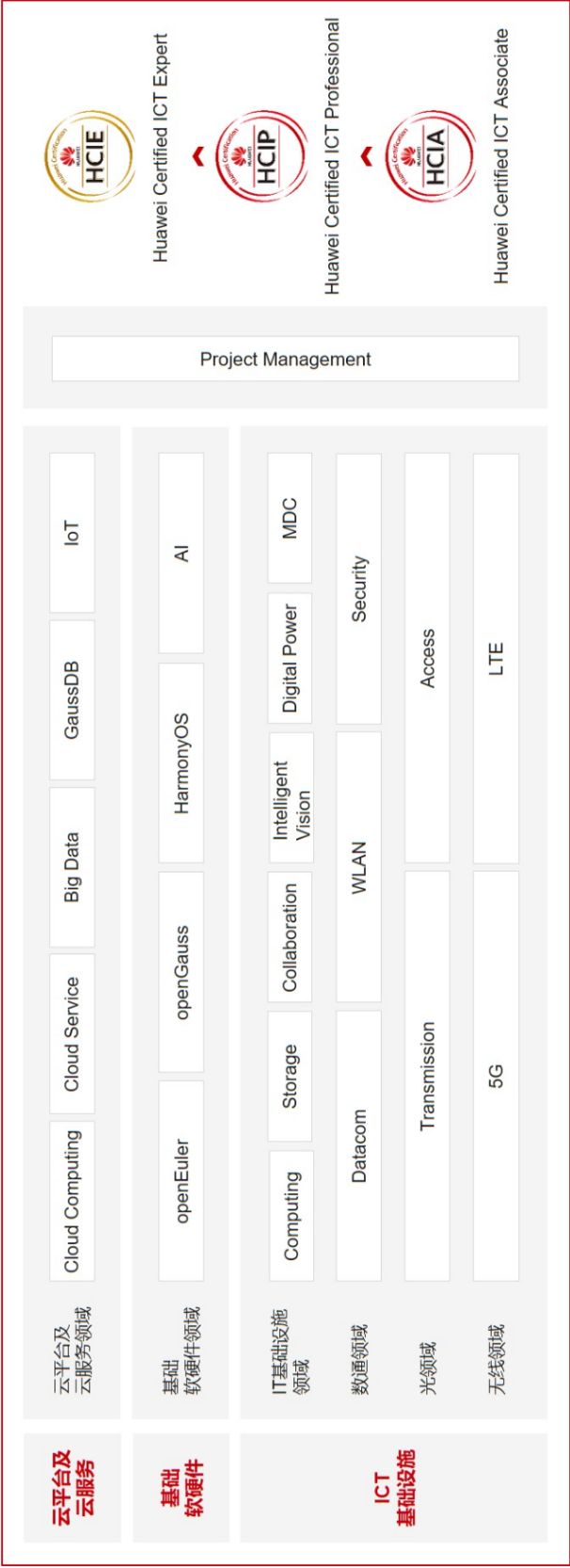
华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

华为认证HCIE-Computing V2.0定位于培养与认证具备人工智能计算中心规划设计、安装部署、性能优化与运维能力的计算专家。

通过HCIE-Computing V2.0认证，您将了解人工智能的发展历程及大模型相关原理，理解人工智能计算中心面临的挑战与解决方案，掌握数据中心的设计方法及人工智能计算中心训练、推理场景的方案，并能进行人工智能数据中心部署，掌握计算设备的性能调优及维护方法。

华为职业认证全景图

华为职业认证覆盖ICT全领域，致力于提供领先的人才培养体系和认证标准，培养数字化时代的新型ICT人才，构建良性的ICT人才生态。



简介

本书为 HCIE-Computing 认证培训教程，适用于准备参加 HCIE-Computing 考试的学员或者希望了解昇腾解决方案的读者。

内容描述

本实验指导书共包含 4 个实验，从基础环境部署、昇腾软件安装，到大模型训练及推理体验和昇腾环境下基础工具的应用操作。具体如下：

- 实验一为基础环境部署，包括使用 SmartKit 工具为 Atlas 800T A2 算力服务器批量进行固件升级、OS 安装等内容。
- 实验二为昇腾软件安装，包括昇腾软件包的制作，以及各类软件包的批量部署操作，这在工程实施过程中非常适用。
- 实验三为大模型体验，主要介绍了昇腾环境下，如何进行模型的推理操作并体验 NPU 数量对推理性能的影响。
- 实验四为基础工具运用及环境验收测试，主要介绍在昇腾环境下，有哪些工具可以针对环境相关的组件进行测试，并介绍在验收之前，需要对环境做哪些测试验收动作。

读者知识背景

本课程为华为认证专家级别课程，为了更好地掌握本书内容，阅读本书的读者应首先具备以下基本条件：

- 已经具备同方向 HCIA 及 HCIP 同等知识技能。
- 了解操作系统、容器等基础知识。

实验环境说明

本实验环境面向准备 HCIE-Computing 考试的工程师。每套实验环境包括 Atlas 800T A2 服务器 1 台，每套实验环境适用于 1~8 名学员同时上机操作。

实验组网：

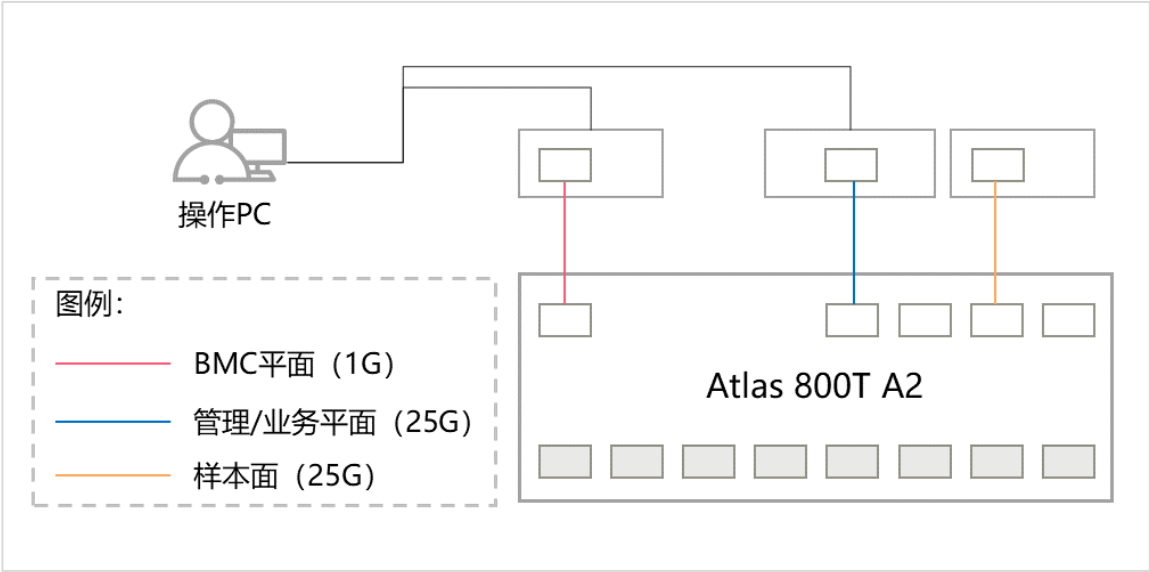


图1-1 实验组网

组网说明：

- 使用 1 台 Atlas 800T A2 服务器，3 台交换机，1 台用于 BMC，1 台用于管理和业务网络，1 台用于样本面网络，连接存储侧。
- 操作 PC 和 BMC 及管理业务网络互通。

准备实验环境

检查设备

实验开始之前请每组学员检查自己的实验设备是否齐全，实验清单如下：

设备名称	数量	备注
Atlas 800T A2	1台	所有实验组共用

目录

前 言	3
简介	3
内容描述	3
读者知识背景	3
实验环境说明	3
准备实验环境	4
1 实验环境介绍	7
1.1 实验介绍	7
1.1.1 关于本实验	7
1.1.2 实验设备及软件介绍	7
1.1.3 网络信息规划	9
1.1.4 账号密码规划	9
2 基础环境部署	10
2.1 SmartKit 工具安装配置	10
2.2 Atlas 服务器固件升级	20
2.3 Atlas 服务器操作系统安装	52
3 昇腾软件部署	66
3.1 昇腾软件包制作	66
3.1.1 Python 环境安装	66
3.1.2 昇腾软件包制作	69
3.2 昇腾软件包部署	71
3.3 NPU 网络配置	75
3.4 单机综合检测	77
4 大模型体验	86
4.1 ChatGLM3-6B 模型推理体验	86
4.2 RAG 模型推理体验（可选）	92
4.3 DeepSeek 推理性能测试	97
5 基础工具运用	101
5.1 HCCN Tool 工具使用	101
5.2 ascend-dmi 工具使用	102

1

实验环境介绍

1.1 实验介绍

1.1.1 关于本实验

本实验基于 Atlas 800T A2 服务器，完成基于昇腾解决方案的人工智能计算中心训练服务器的安装部署和测试验收实操工作。本实验会涉及操作系统安装、昇腾软件部署以及在昇腾环境下的模型测试体验操作等过程。

1.1.2 实验设备及软件介绍

设备名称、型号与版本的对应关系如下：

组件	软件名称	下载链接
系统镜像	CentOS-7-aarch64-Everything-1810.iso	https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/CentOS-7-aarch64-Everything-1810.iso
SmartKit Computing	SmartKit_24.0.0_zh.exe 及其他插件包	https://support.huawei.com/enterprise/zh/scale-out-storage/smartkit-pid-8576706/software/261974137
SmartProvisioning	SmartProvisioning_1.15.0_aarch64.zip	https://support.huawei.com/enterprise/zh/management-software/smart-provisioning-pid-23143793/software/261980112
SP686 驱动	ComputingComponenti Driver-CentOS7.6- Driver-ARM_2.1.0.zip	https://support.huawei.com/enterprise/zh/management-software/computing-component-idriver-pid-259488843/software/262226508
SP686 驱动	ComputingComponenti Driver-openEuler22.03- Driver-ARM_2.1.3.zip	https://support.huawei.com/enterprise/zh/management-software/computing-component-idriver-pid-259488843/software/262226508
Atlas 800T A2 1.0.0.7	iBMC_3.10.02.65.zip HAM- Module(IT21HMDC08)- CPLD_3.06.zip	https://www.hiascend.com/hardware/firmware-drivers/community?product=4&model=26&cann=8.0.0.beta1&driver=1.0.28.alpha

	HDD-Backplane(IT21BP8E01)-CPLD_1.01.zip Mainboard(IT21SK4D03)-CPLD_1.03.zip NPU-Board(IT21SD8B01)-CPLD_1.00.zip BIOS_7.37.zip	
NPU MCU	Ascend-hdk-910b-mcu_23.3.13.zip	
NPU Driver	Ascend-hdk-910b-npu-driver_23.0.7_linux-aarch64.run	
NPU Firmware	Ascend-hdk-910b-npu-firmware_7.1.0.11.220.run	
昇腾 Toolkits	Ascend-cann-toolkit_8.0.0_linux-aarch64.run	https://www.hiascend.com/developer/download/community/result?module=cann&cann=8.0.0.beta1
昇腾 kernels	Ascend-cann-kernels-910b_8.0.0_linux-aarch64.run	
昇腾容器运行时	Ascend-docker-runtime_5.0.0_linux-aarch64.run	https://gitee.com/ascend/mind-cluster/releases/download/v6.0.0/Ascend-docker-runtime_6.0.0_linux-aarch64.run
昇腾 Toolbox	Ascend-mindx-toolbox_6.0.0_linux-aarch64.run	https://www.hiascend.com/developer/download/community/result?module=cluster+cann&cluster=6.0.0&cann=8.0.0.beta1
Ascend Deployer 工具	ascend-deployer-dev.zip	https://gitee.com/nanchengfeinan/ascend-deployer/repository/archive/dev.zip
Python	python-3.7.5-amd64.exe	https://www.python.org/ftp/python/3.7.5/python-3.7.5-amd64.exe
deepseek 模型数据	deepseek-R1-bf16.zip	https://modelscope.cn/models/deepseek-ai/DeepSeek-R1-Distill-Llama-70B
MindIE 镜像	2.0.T3.1-800I-A2-py311-openeuler24.03-lts	https://www.hiascend.com/developer/ascendhub/detail/af85b724a7e5469ebd7ea13c3439d48f
ChatGLM3-6B 模型镜像	chatglm.tar	请前往 HCIE-Computing V2.0 认证页面下载

ChatGLM3-6B 模型权重	ChatGLM-6B.tar.gz	
RAG 模型镜像及数据	rag_hyde.zip	
相关测试脚本及示例文件	HCIE_Computing_file.zip	

1.1.3 网络信息规划

组件	IP 地址	网关
BMC	10.190.40.17/24	10.190.40.1
管理/业务地址	10.190.41.17/24	10.190.41.1
存储地址	10.190.44.17/24	10.190.44.1
NPU 地址	192.168.27.11-18/24	192.168.27.1

1.1.4 账号密码规划

组件	账号	密码
iBMC	Administrator	Admin@9000
OS	root	HCIETraining@123

2 基础环境部署

完成相关硬件上架后，进入软件安装环境。本章节指导学员完成 SmartKit 工具的安装配置，以及训练服务器的操作系统安装。

2.1 SmartKit 工具安装配置

步骤 1 获取 SmartKit 工具及插件包

根据[实验设备及软件介绍](#)小节获取 SmartKit 工具以及相关的插件包。

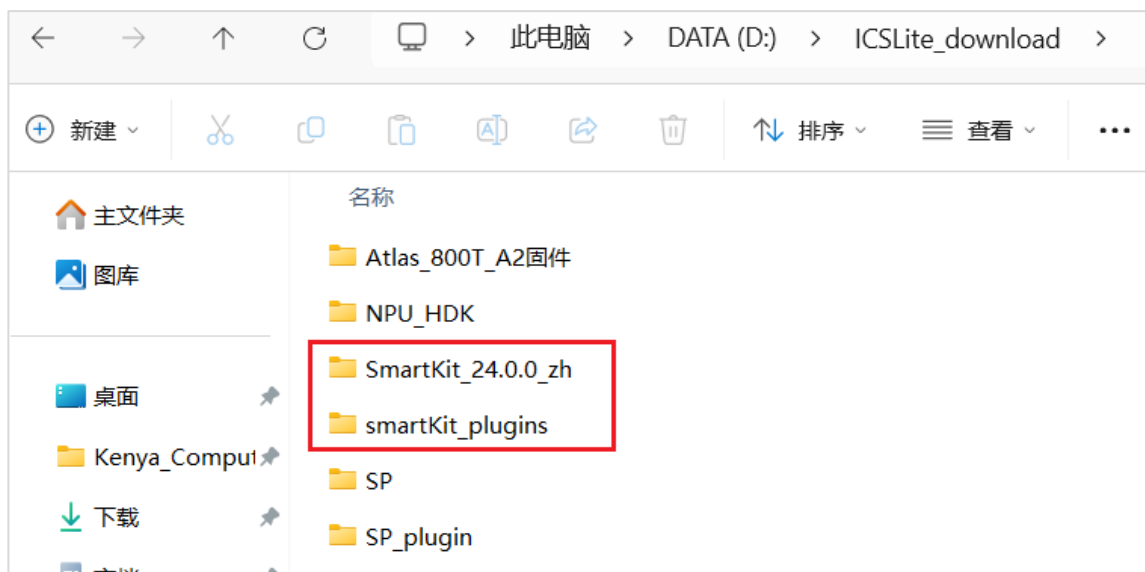


图2-1 SmartKit 及相关插件

步骤 2 安装 SmartKit 工具包

进入 SmartKit 软件包目录，双击安装文件，打开安装界面。如遇到用户账户控制弹窗，单击“是”。



图2-2 打开 SmartKit 安装界面

根据界面提示，单击“next”。

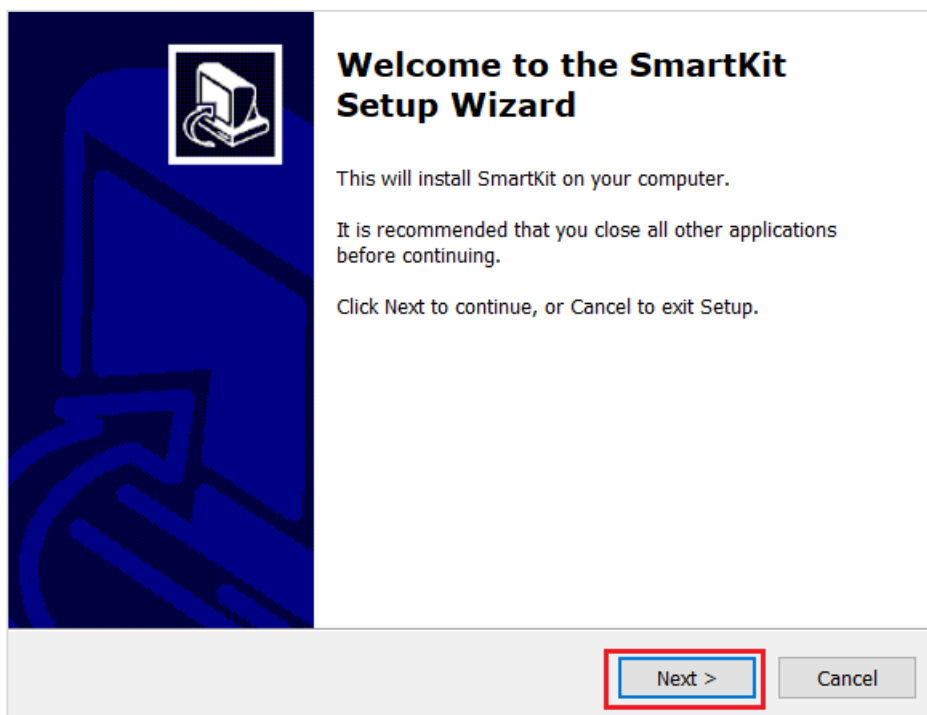


图2-3 继续安装 SmartKit

在授权页面，选择接受相关协议。

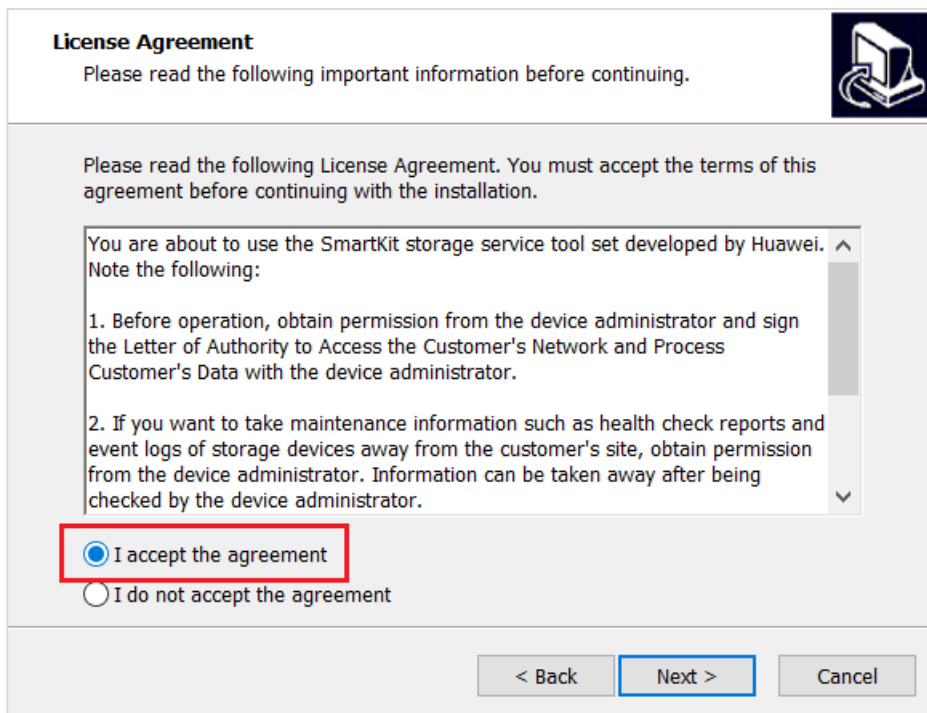


图2-4 接受许可协议

安装位置根据实际情况调整，无特殊要求时，保持默认。单击“next”。

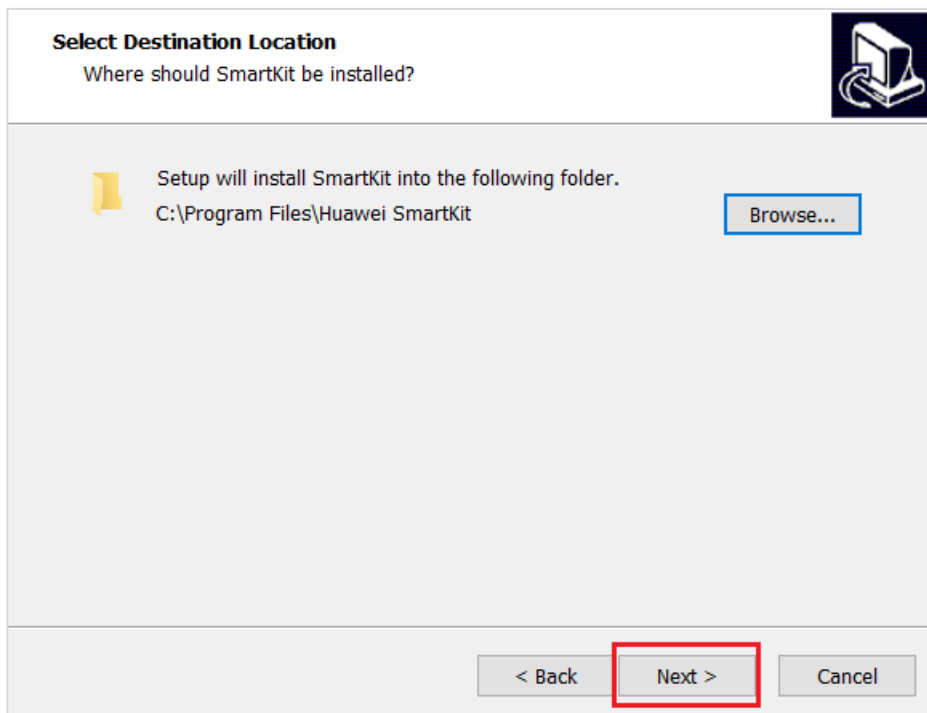


图2-5 配置安装位置

角色选项页面，选择所有用户可用。单击“Next”。

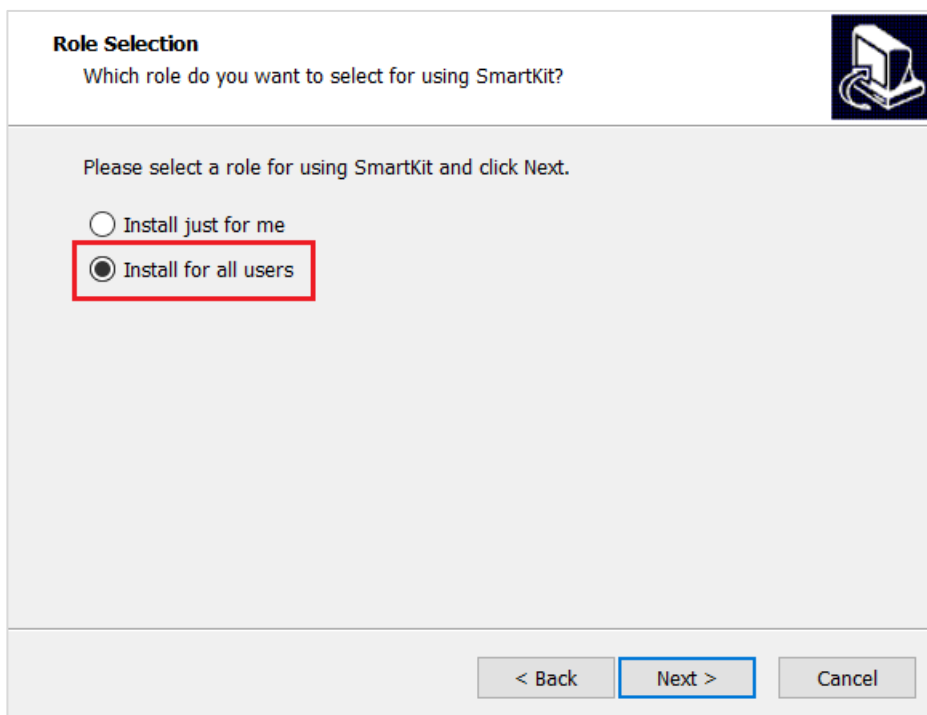


图2-6 选择安装角色

配置完成后，单击“Install”开始安装。

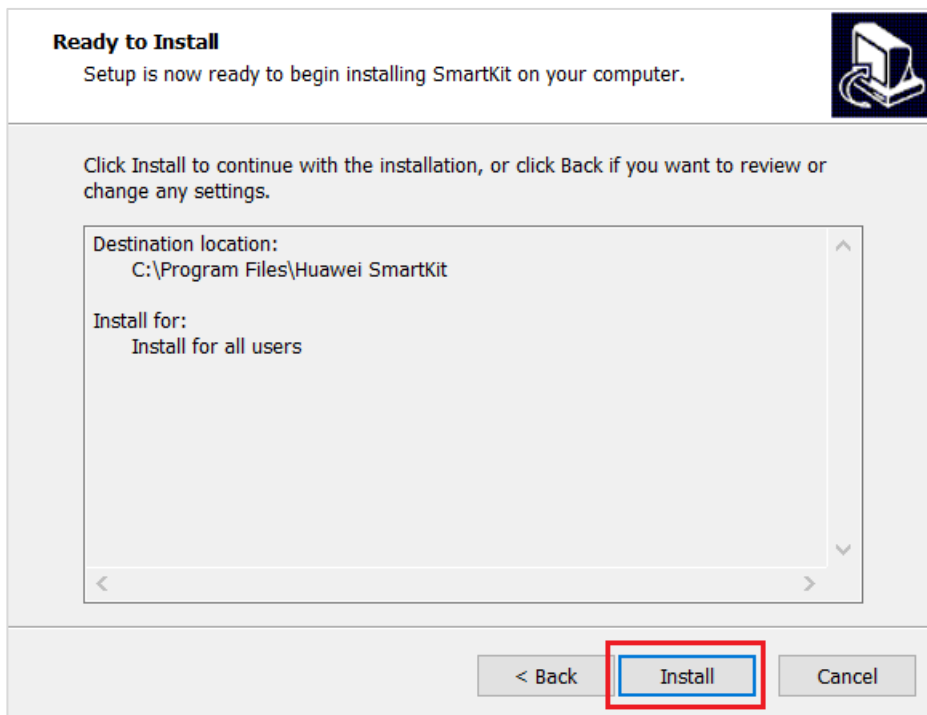


图2-7 开始安装

安装完成后，单击“Finish”，自动打开 SmartKit 工具。

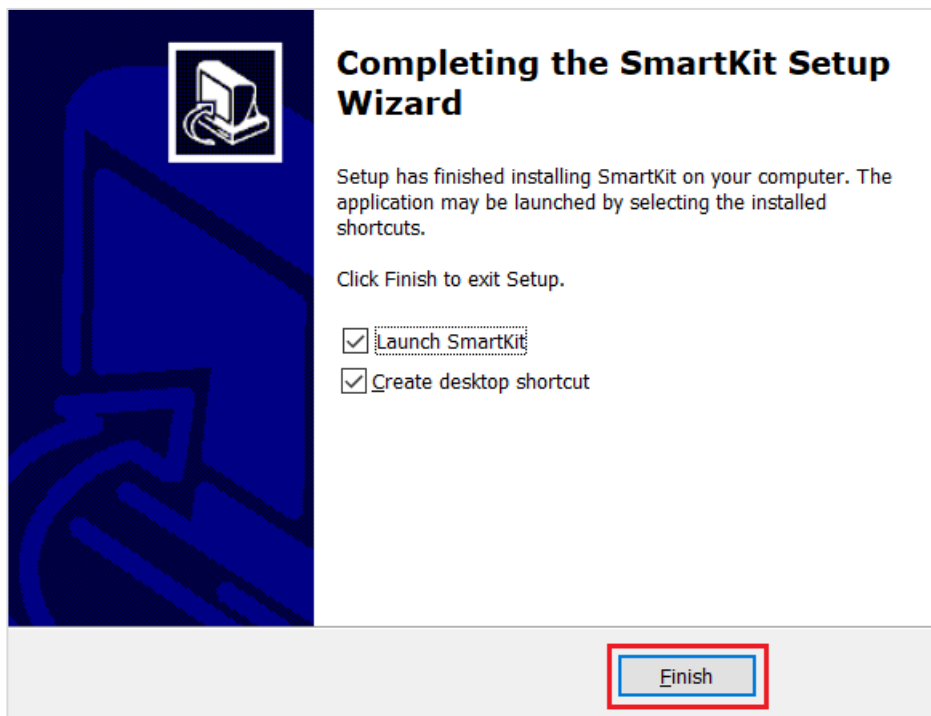


图2-8 安装完成

在登录认证界面，根据提示完成认证（后面批量分发功能需要完成认证才能使用）。

登录认证

1 步骤一 选择区域

省份

2 步骤二 扫描二维码



⚠ 二维码有效期为5分钟，可点击二维码进行刷新

3 步骤三 输入认证码

* 认证码:

登录认证

☒ 阅读并同意 [华为隐私声明](#)

暂不认证

图2-9 完成认证

根据提示勾选仅在中国区使用，并同意安全声明。



图2-10 同意安全声明

根据提示输入“我确认只在中国区使用”，并单击“继续”。

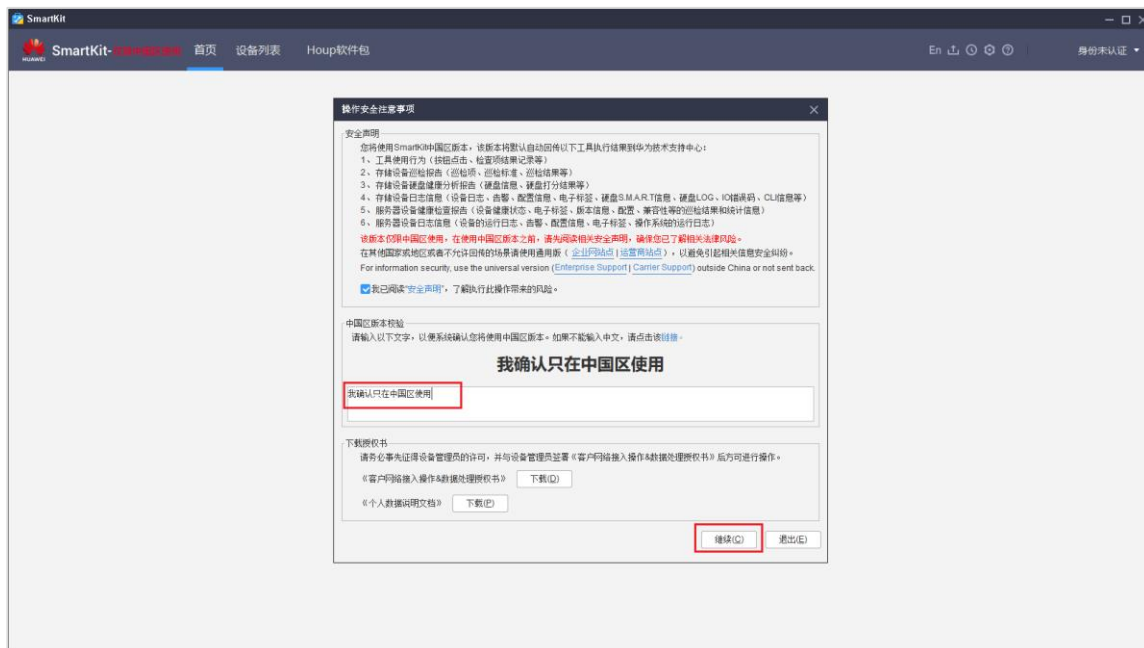


图2-11 确认只在中国区使用

正常进入 SmartKit 界面。

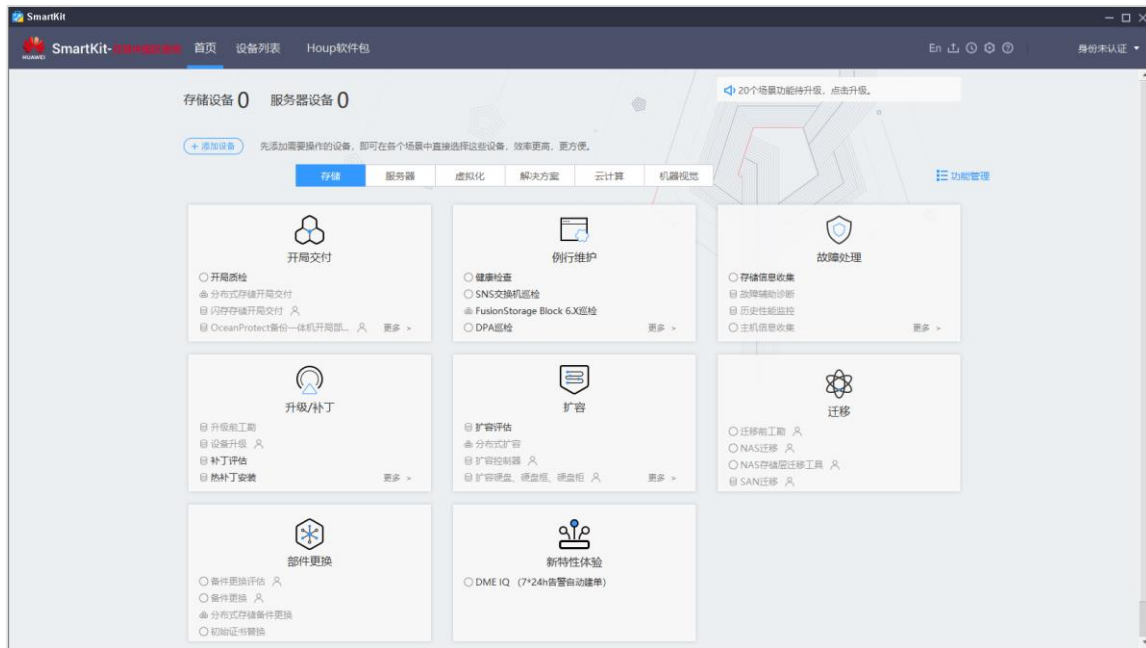


图2-12 进入 SmartKit 首页

步骤 3 加载 SmartKit 插件

确保已获取 SmartKit 插件后，在 SmartKit 首页，单击“功能管理”。

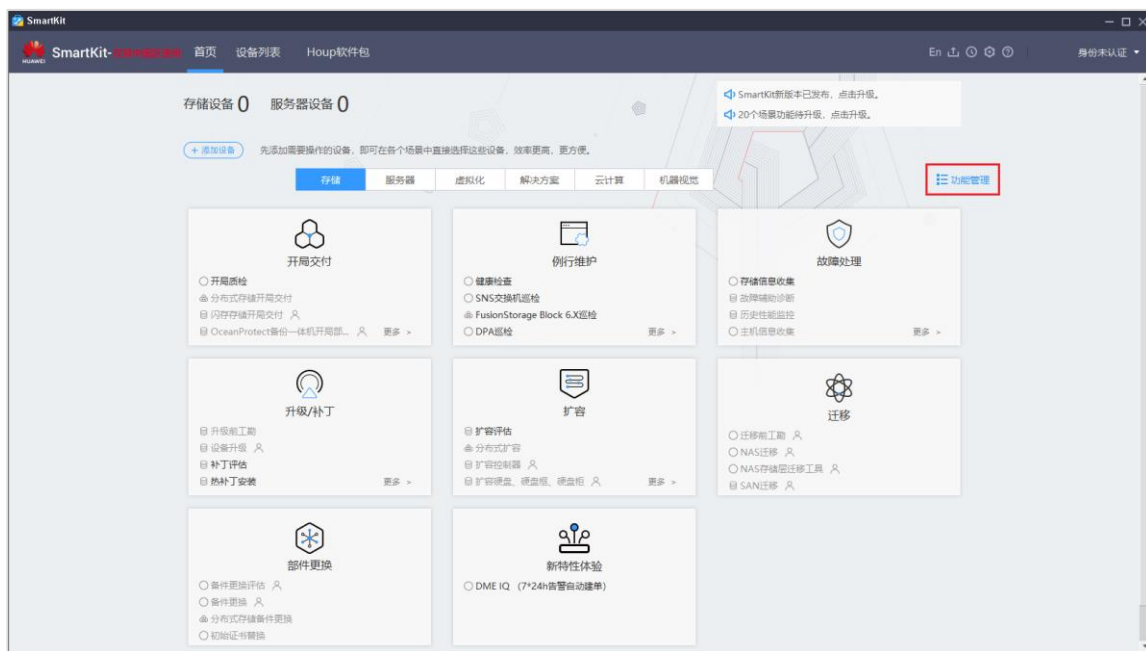


图2-13 进入功能管理

在功能管理页面，单击“导入”。

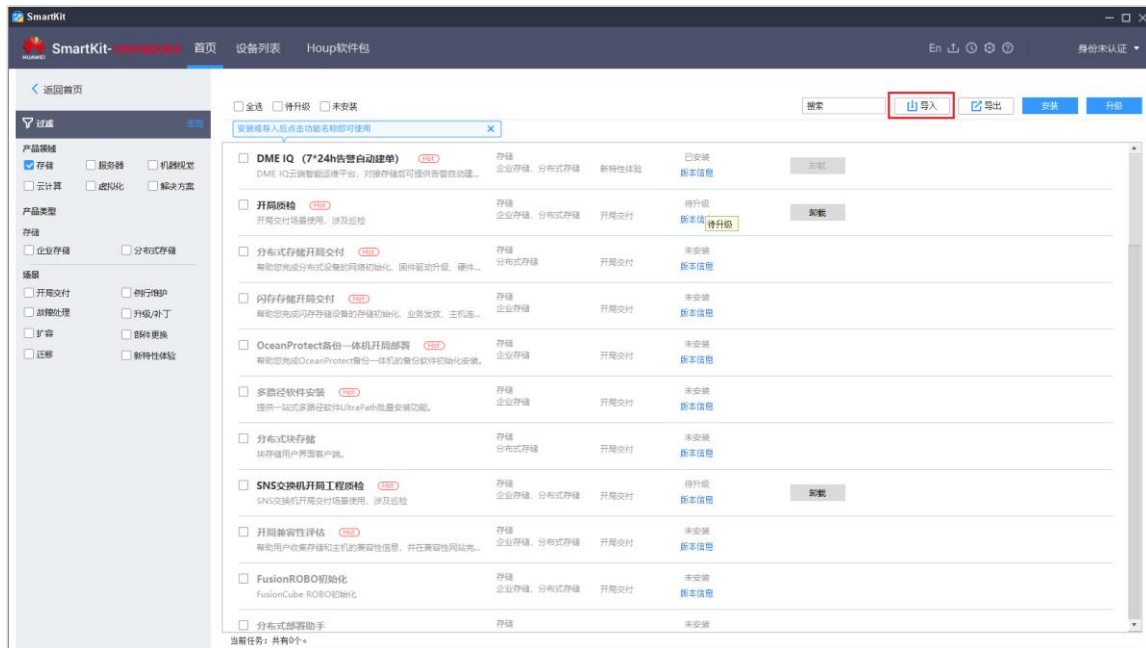


图2-14 导入插件

在弹出的插件选择框中，选择**实验设备及软件介绍**小节提前准备好的所有插件后，单击“确定”。（如果出现提示没有校验文件时，单击“确定”。）

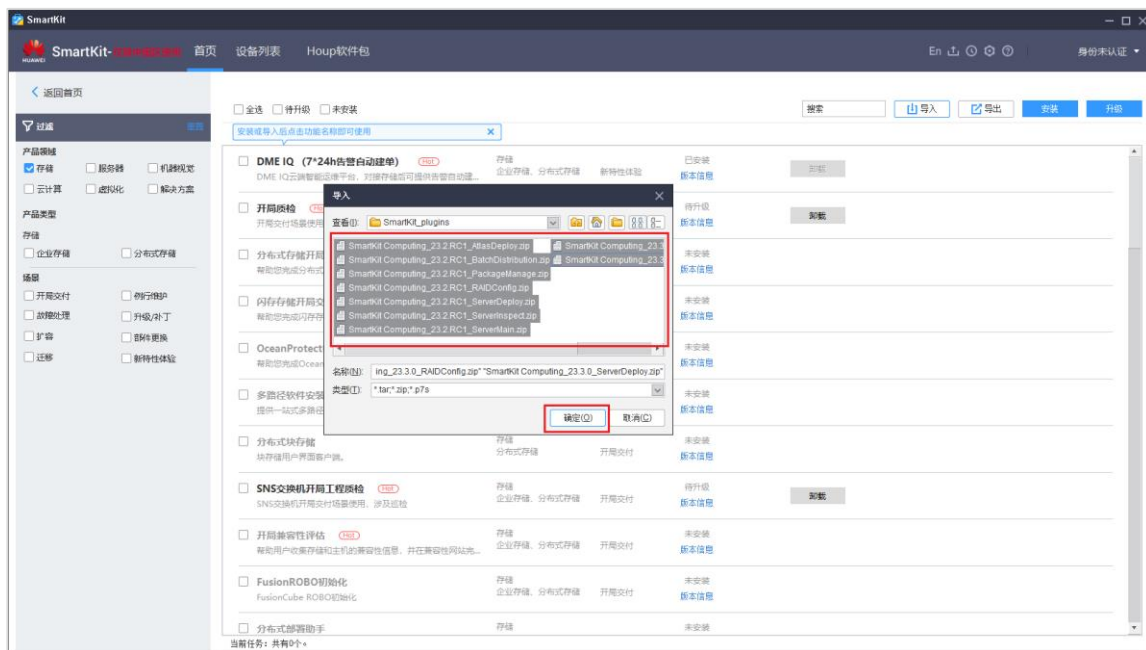


图2-15 选择插件

忽略校验文件的提示，单击“安装”。等待插件导入安装完成。

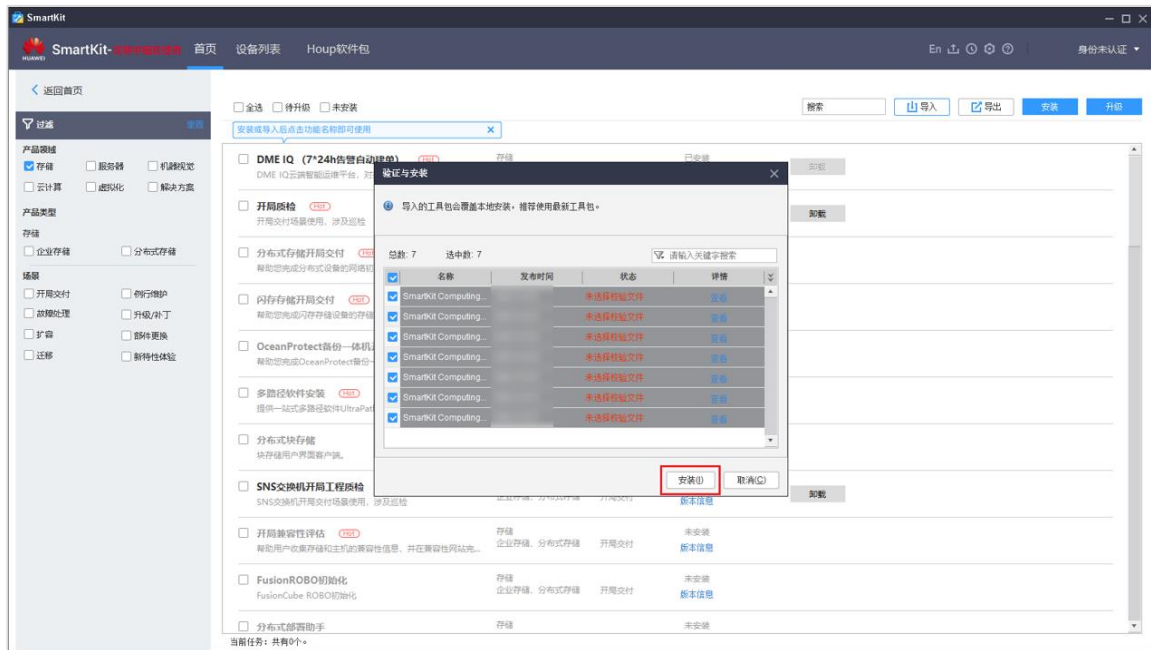


图2-16 安装插件

导入成功后，单击“确定”。

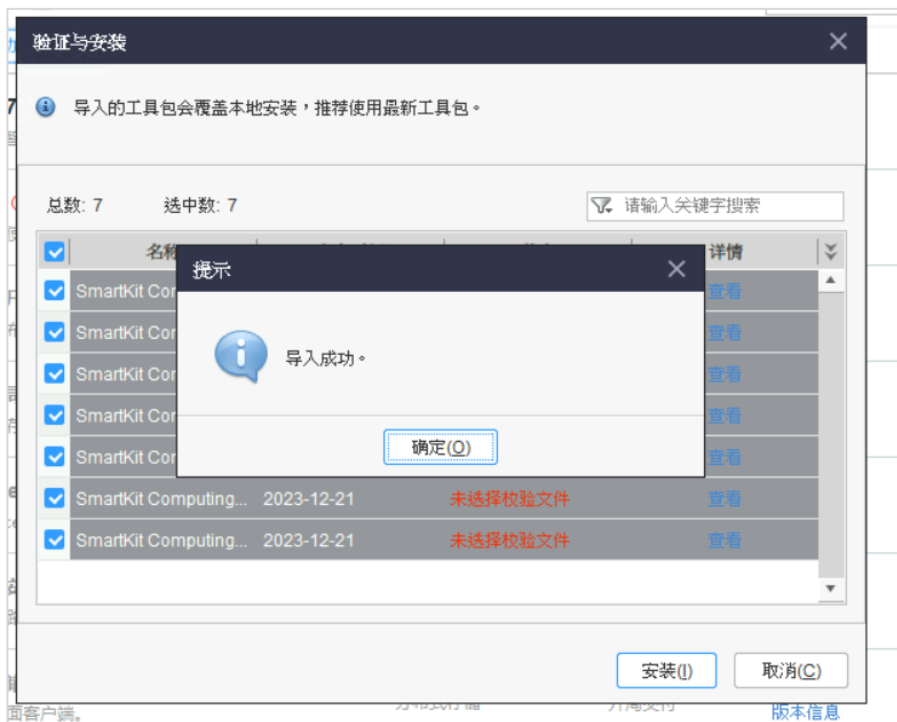


图2-17 插件导入成功

返回首页。

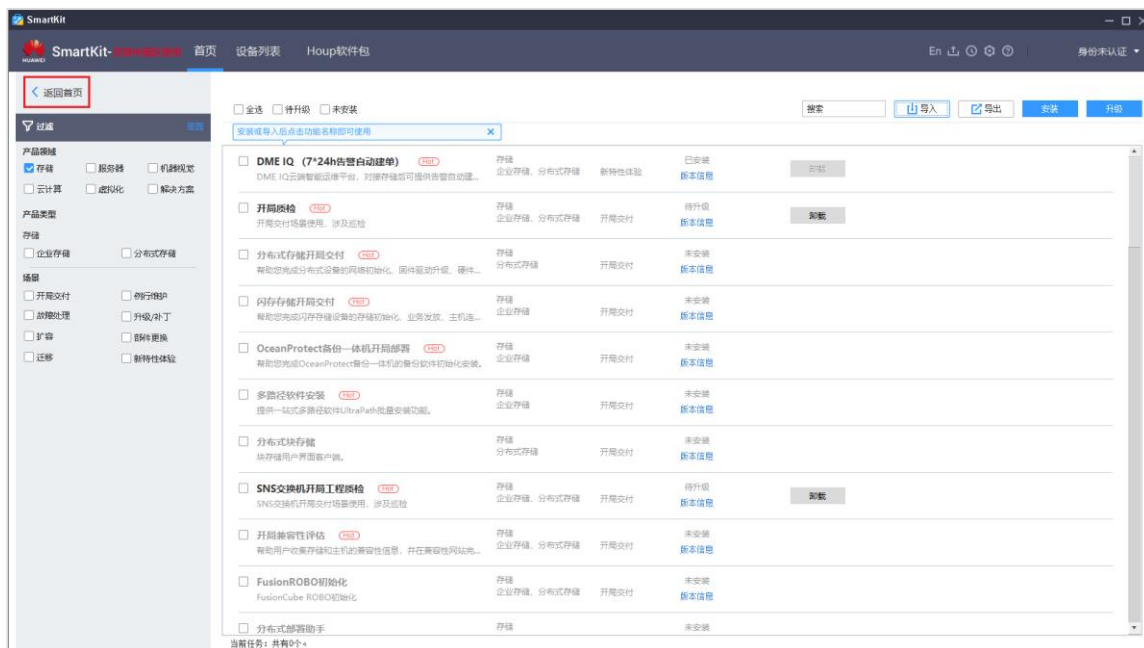


图2-18 返回 SmartKit 首页

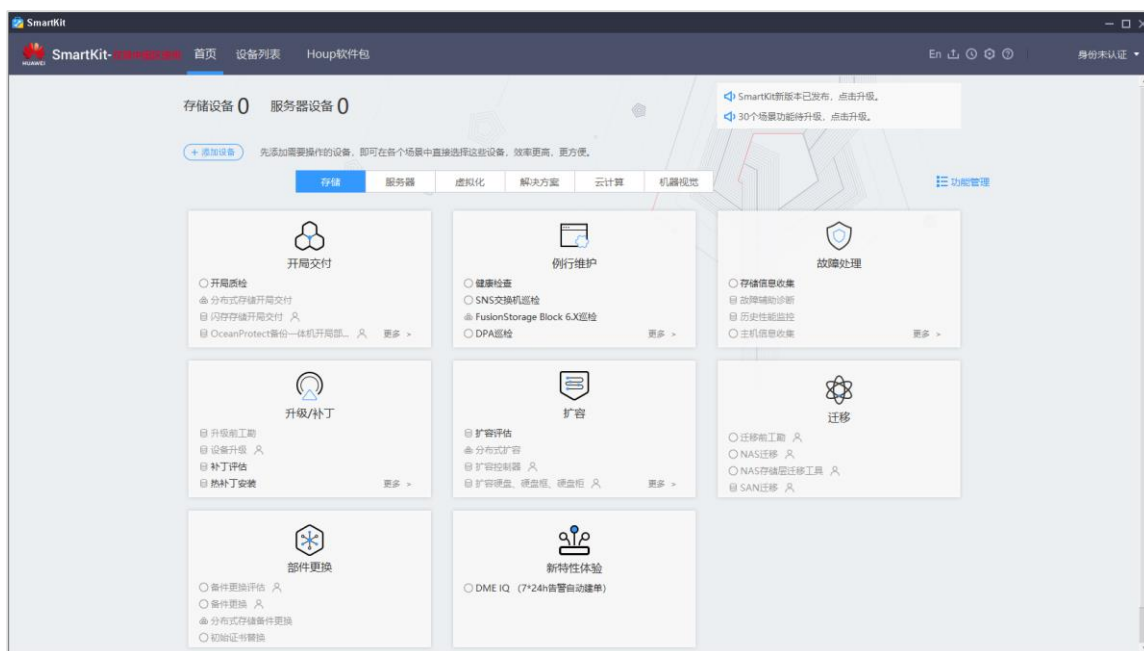


图2-19 SmartKit 首页

SmartKit 安装及插件加载完成。

2.2 Atlas 服务器固件升级

服务器固件升级可通过 SmartKit 批量完成。

步骤 1 待安装服务器的 iBMC 确认

根据网络信息规划章节的服务器 iBMC 信息，确保通过操作 PC 能正常访问服务器的 iBMC。

例如，某服务器的 iBMC 信息如下：

- iBMC 地址：10.190.40.11
- 用户：Administrator
- 密码：Admin@9000

则通过浏览器登录服务器 iBMC，链接为 <https://10.190.40.11/UI/Static/#/login> 。

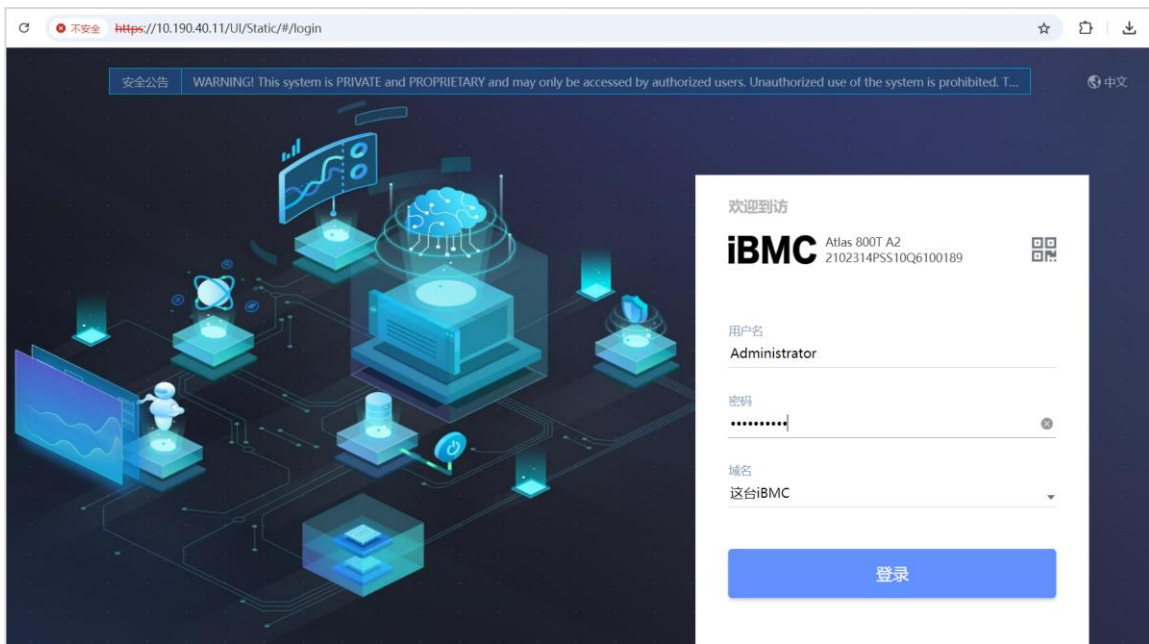


图2-20 服务器 iBMC 登录界面

可正常登录，服务器 iBMC 信息检查完成。



图2-21 服务器 iBMC 登录成功

步骤 2 SmartKit 添加服务器设备

在 SmartKit 首页单击“添加设备”。

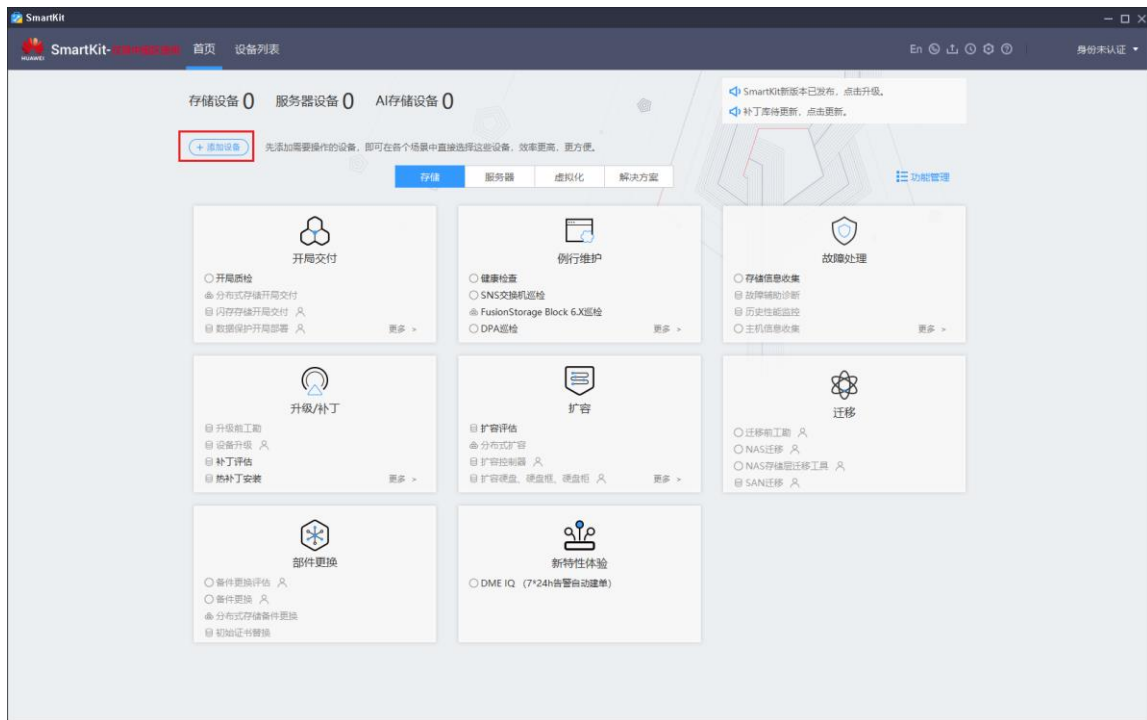


图2-22 SmartKit 添加设备

选择“服务器”，默认通过 iBMC 添加，并单击“增加”。



图2-23 根据 iBMC 增加设备

输入服务器 iBMC 连接信息后，单击“确定”（注意：这里起始 IP 和结束 IP 通过配置后可批量添加多台设备，在此以 1 台设备为例）。

增加设备

* 起始IP:

10.190.40.17

* 用户名:

Administrator

SNMPv3加密密码:

高级模式

* SNMP端口:

161

* SSH / SFTP端口:

22

* HTTPS端口:

443

通道加密算法:

开启兼容性算法

OS/IES选项

root 密码:

* 结束IP:

10.190.40.17

* 密码:

(如果不填则使用上面的用户密码, 如果SNMPv3加密密码和用户密码不一致, 则SNMPv3加密...

* IPM端口:

623

* VMM端口:

8208

是否支持帐号解耦:

是否为代理场景:

√ 交换模块选项 (E9000 健康检查、日志收集、升级涉及到交换模块的场景需要)

√ 计算节点信息 (E9000 MM920/MM921或E9000H SMM 涉及到计算节点的场景需要)

确定

取消

端口检测

图2-24 填写设备信息

进入等待执行阶段。

SmartKit

SmartKit-设备管理工具

首页 设备列表

存储

服务器

AI存储

iBMC

HMM

OS

CMC

IES

E9000H(TCE)

SWITCH

修改

移除

更新

端口检测

总数: 0

选中数: 0

☐

序列号

BMC IP

产品名

iBMC

CPU

增加设备

执行中... 请稍候。

10%

总数: 1 选中数: 0

对象

状态

进度

处理建议

10.190.40.17

 执行中

10 %

取消

图2-25 执行设备增加

执行成功后，设备添加完成，返回 SmartKit 首页。

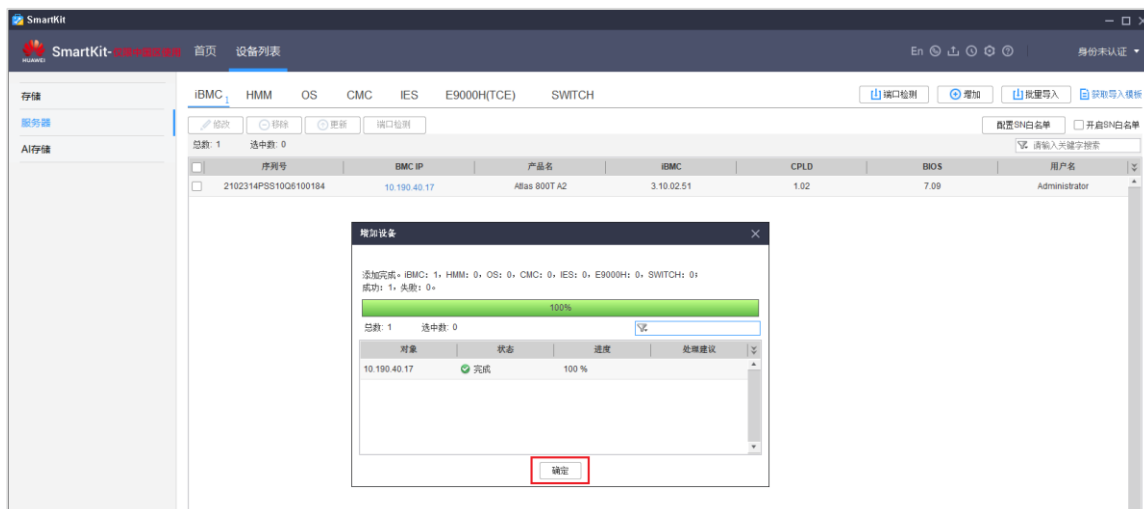


图2-26 设备增加完成



图2-27 返回 SmartKit 首页

步骤 3 升级服务器固件

在 SmartKit 首页，选择服务器页签，并单击“固件升级”。（如果出现 SmartKit 工具升级提示，请忽略，单击“继续”）

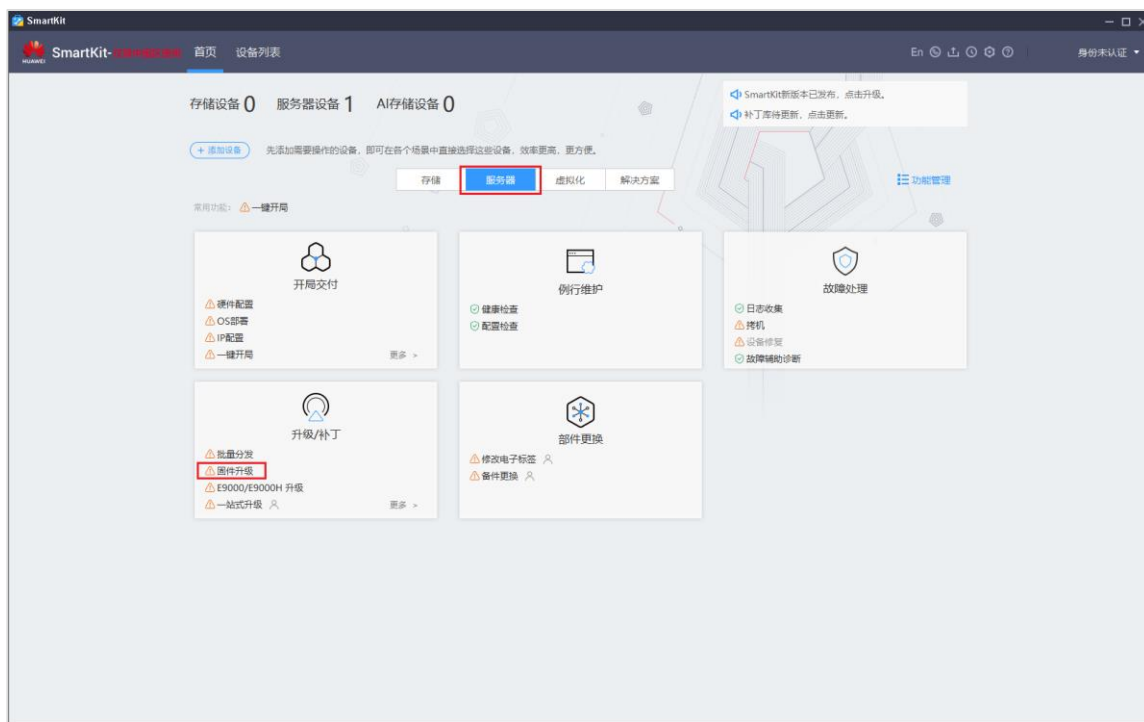


图2-28 选择固件升级

选择设备，选择刚刚通过 iBMC 增加的服务器。

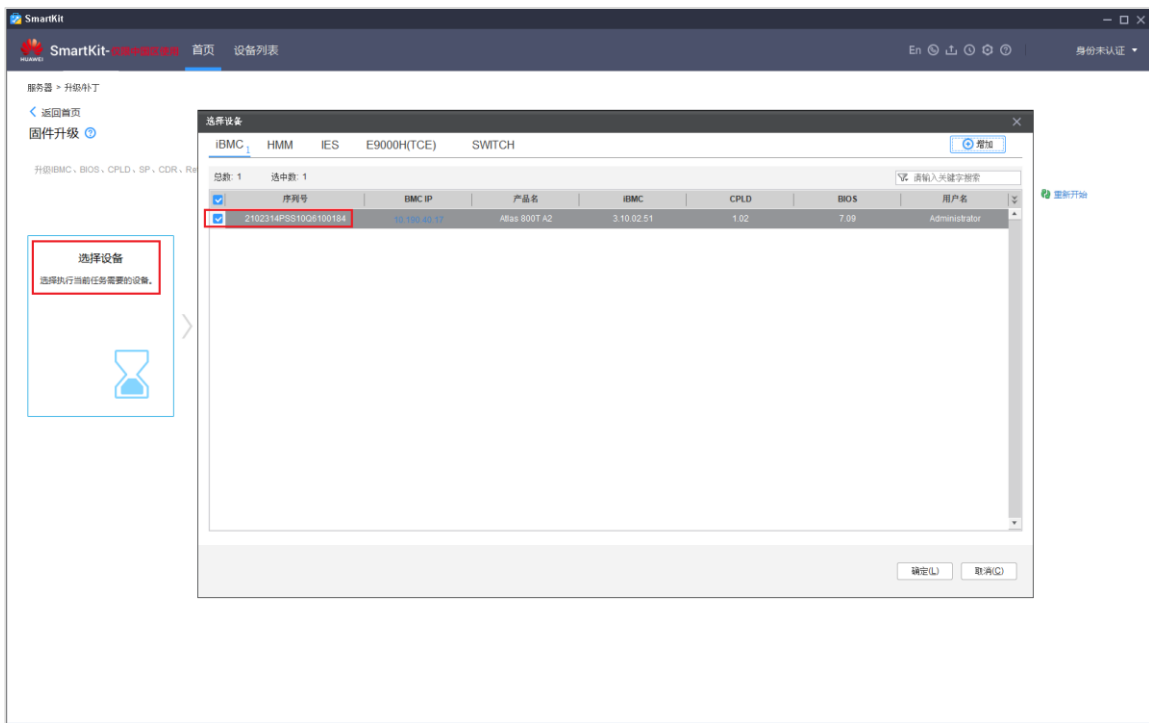


图2-29 选择待升级固件的设备

进行升级评估，该阶段也可以直接在后面的“固件升级”阶段完成。

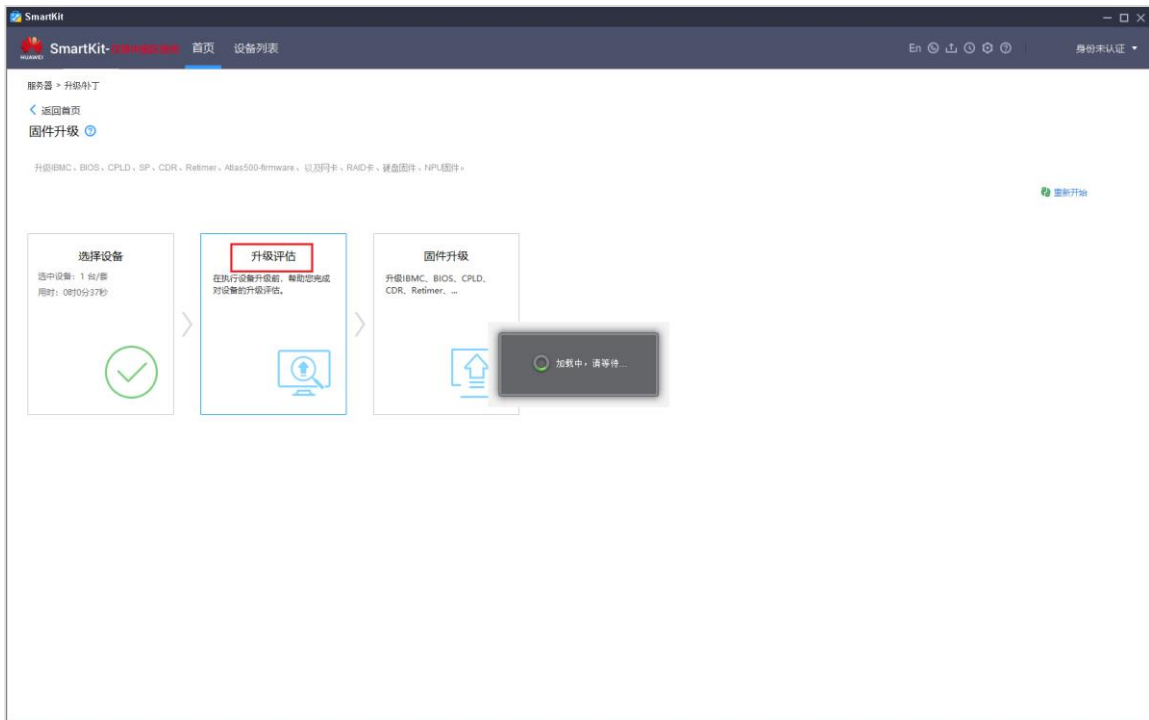


图2-30 进行升级评估



图2-31 升级评估欢迎页面

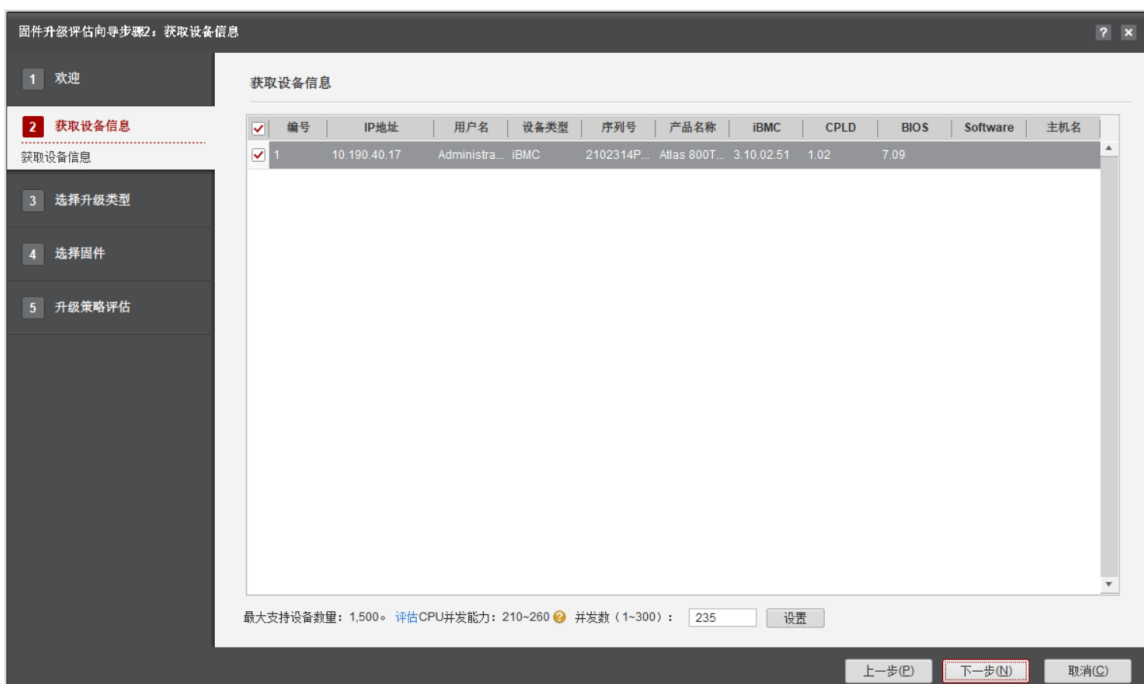


图2-32 升级评估-获取设备信息



图2-33 升级评估-选择升级类型

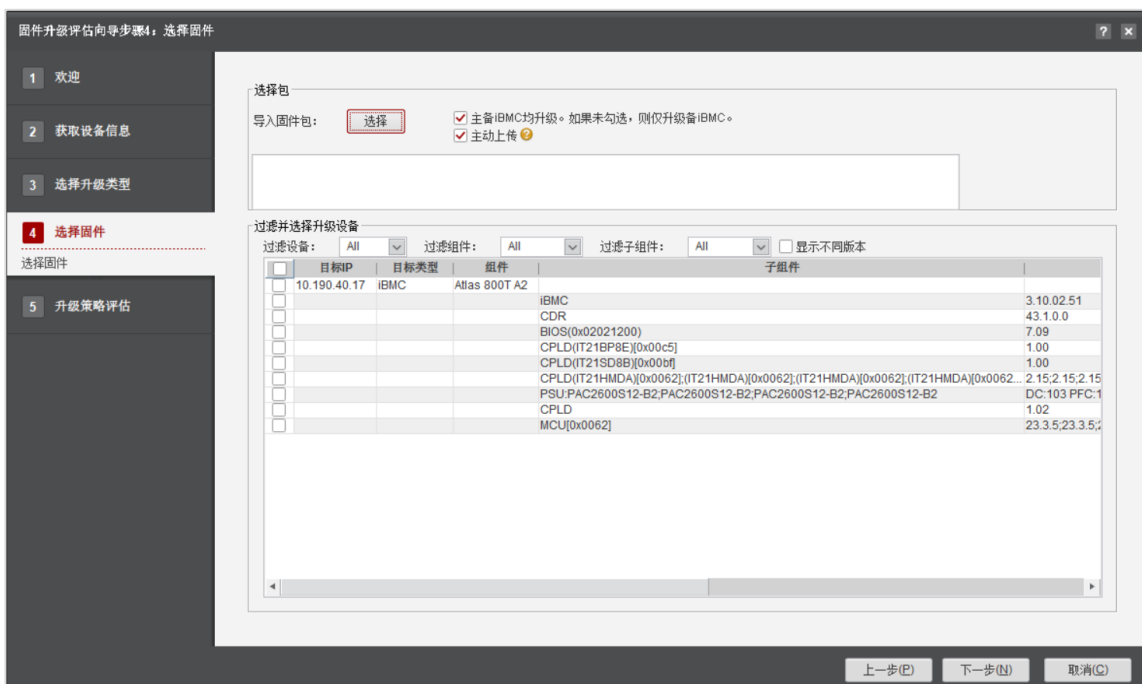


图2-34 升级评估-选择相关固件包

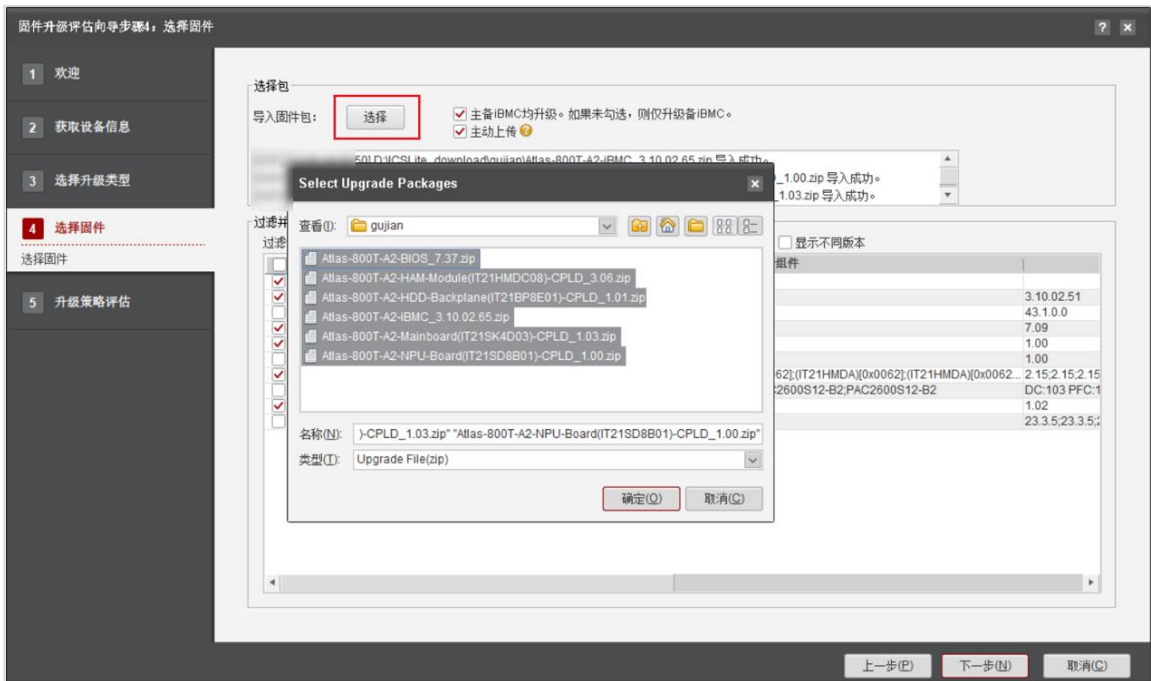


图2-35 升级评估-确定固件包

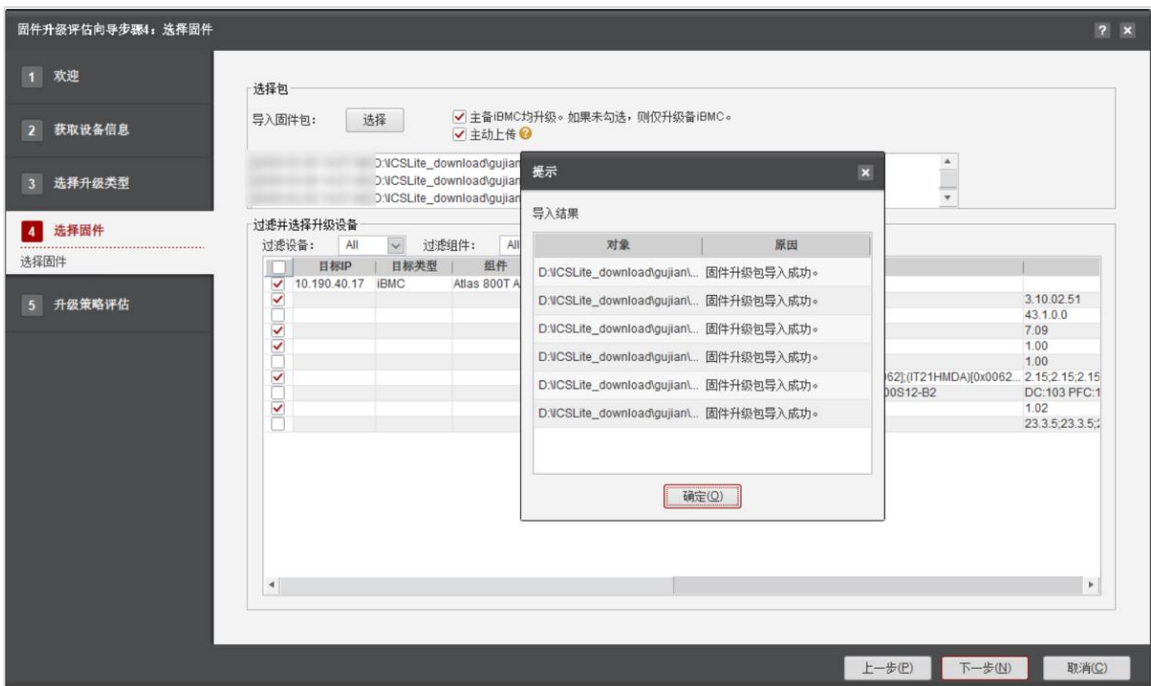


图2-36 导入固件包成功



图2-37 执行升级评估



图2-38 升级评估完成

评估完成后，单击“固件升级”，在弹窗中选择“升级固件”。

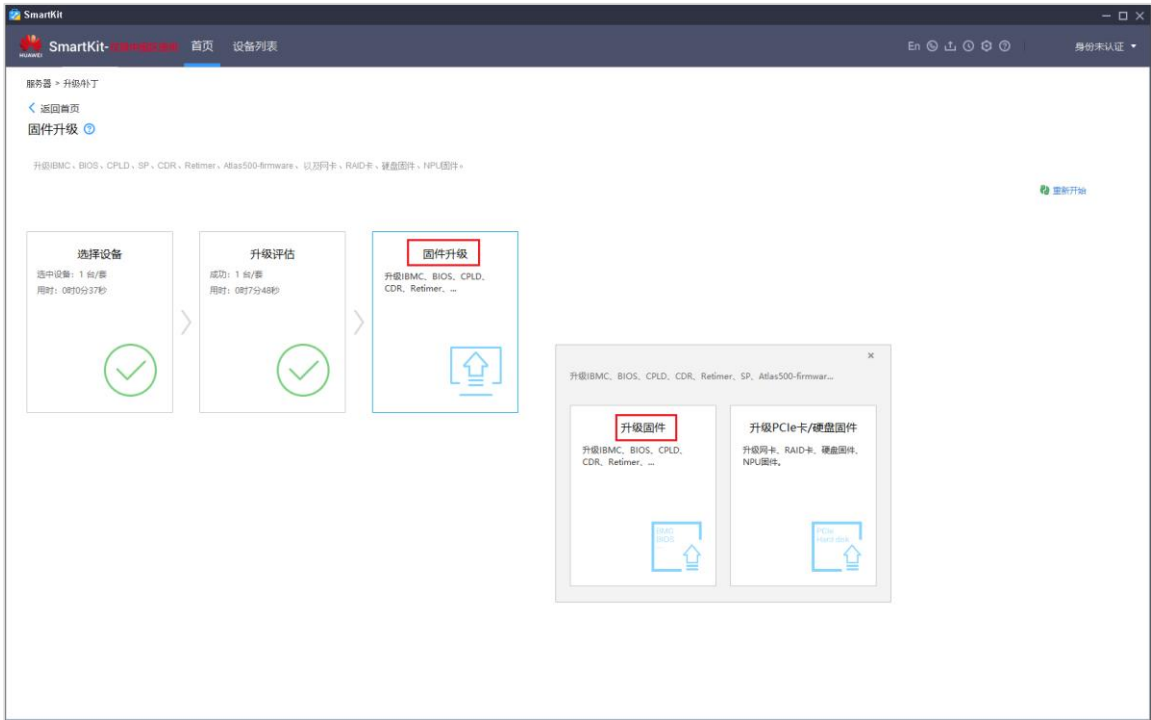


图2-39 进入升级固件页面

单击“下一步”。



图2-40 附件升级欢迎页面

设备信息获取页面，单击“下一步”。

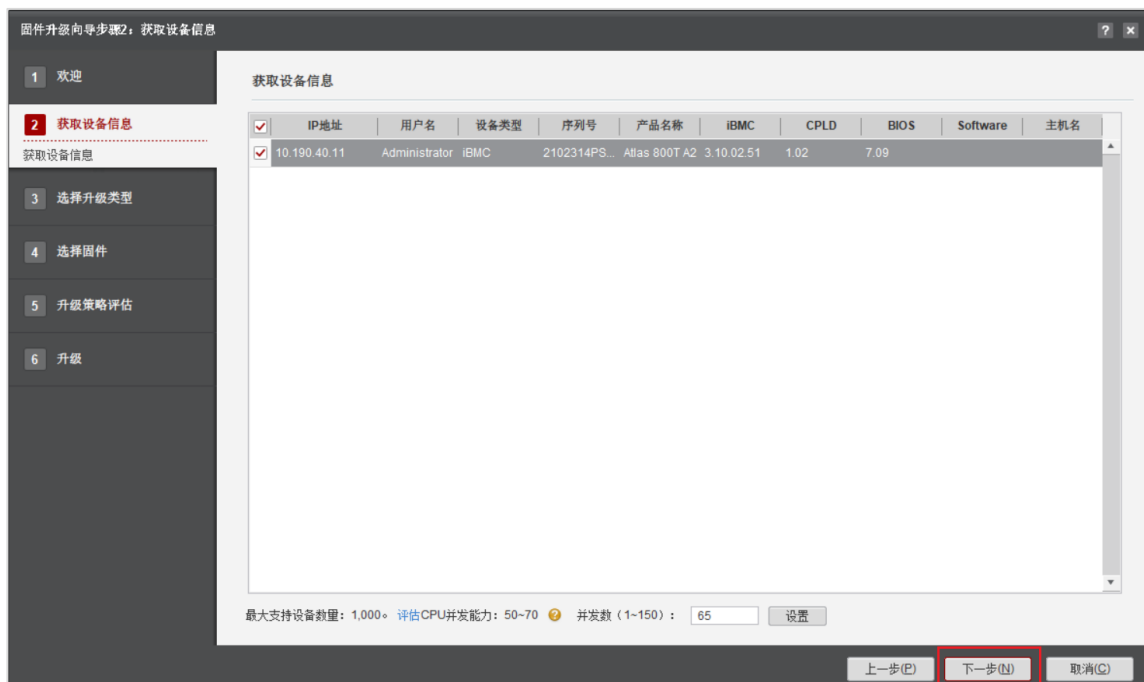


图2-41 获取设备信息

等待设备发现服务完成并检查通过，单击“下一步”。

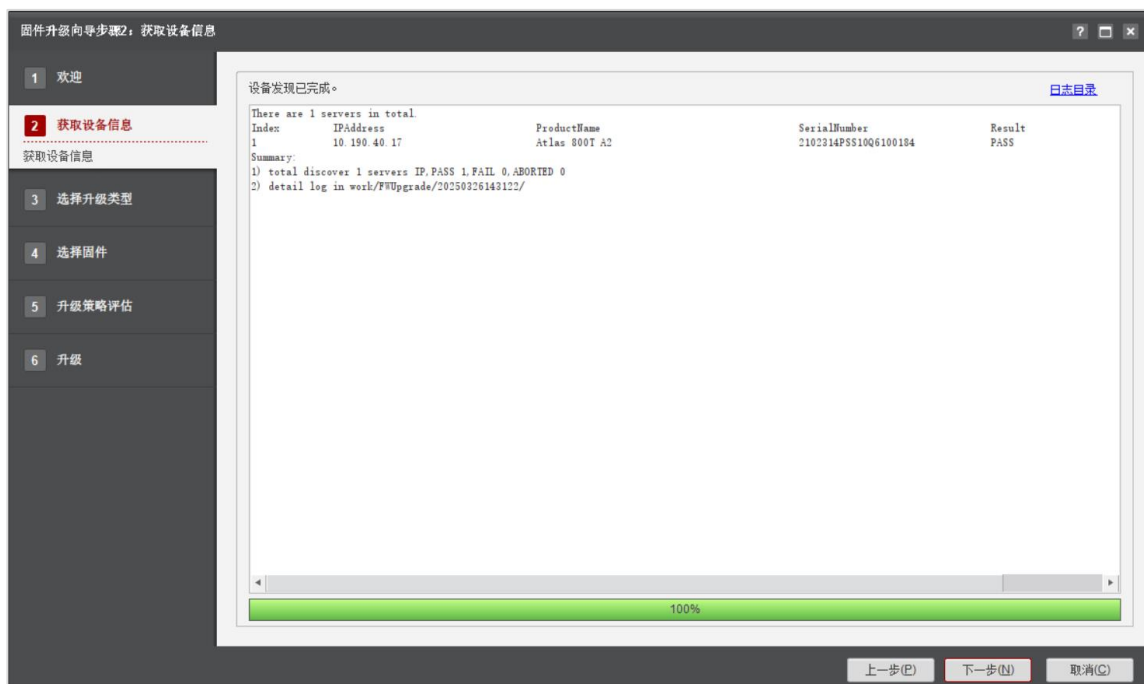


图2-42 设备信息获取完成

在选择升级类型页面，选择“多部件升级”，单击“下一步”。



图2-43 选择升级类型

在选择固件页面，单击“选择”，并在弹窗中选择相应固件，固件根据“[实验设备及软件介绍](#)”小节提供的链接获取。（需要的固件如下：Atlas-800T-A2-BIOS_7.37.zip、Atlas-800T-A2-HAM-Module(IT21HMDC08)-CPLD_3.06.zip、Atlas-800T-A2-HDD-Backplane(IT21BP8E01)-CPLD_1.01.zip、Atlas-800T-A2-iBMC_3.10.02.65.zip、Atlas-800T-A2-Mainboard(IT21SK4D03)-CPLD_1.03.zip、Atlas-800T-A2-NPU-Board(IT21SD8B01)-CPLD_1.00.zip）

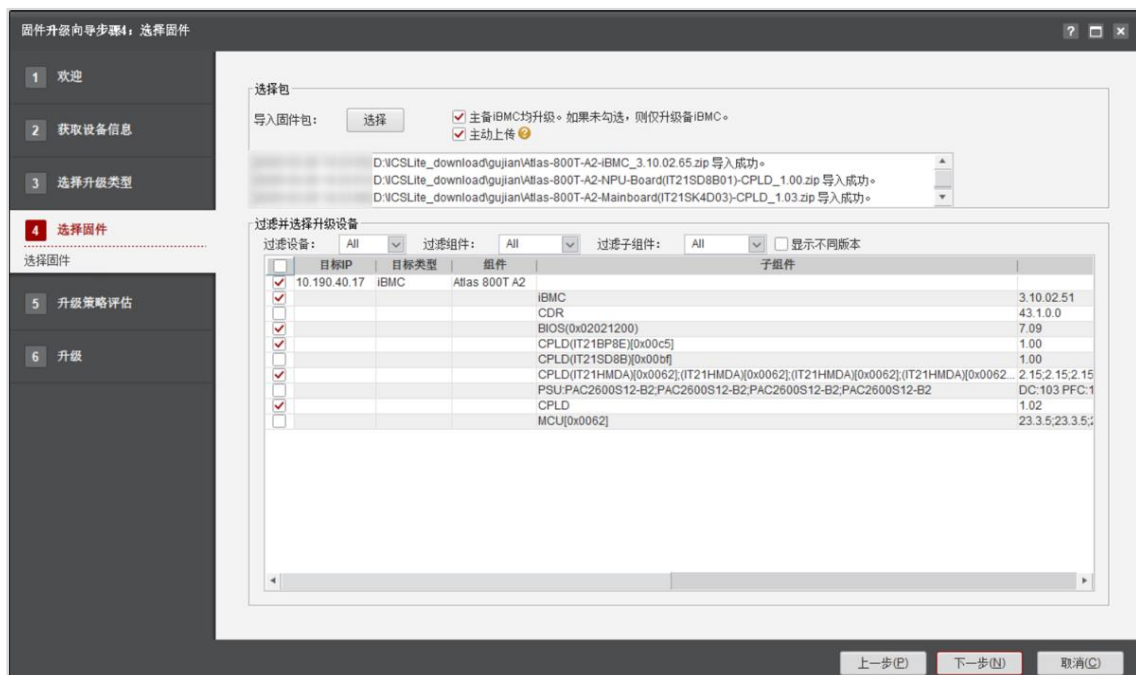


图2-44 选择待升级固件包

在策略评估页面，选中服务器后，单击“评估”，并等待评估完成（如果提前完成了评估，可忽略该步骤）。



图2-45 升级评估

评估通过后，单击“下一步”。



图2-46 评估完成

进入升级页面，等待升级完成，根据不同的业务场景，选择生效方式，因为本实验属于初始部署，可选择“强制下电再上电生效”。

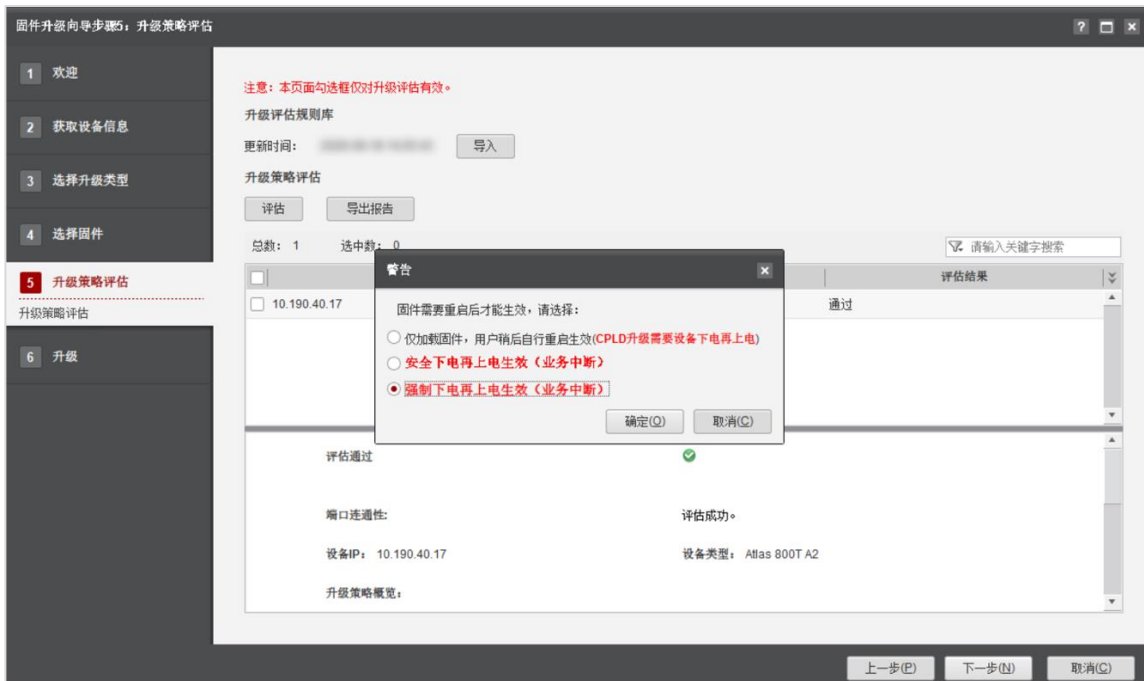


图2-47 确认警告

弹出升级会重启的提示，输入“yes”后确定。

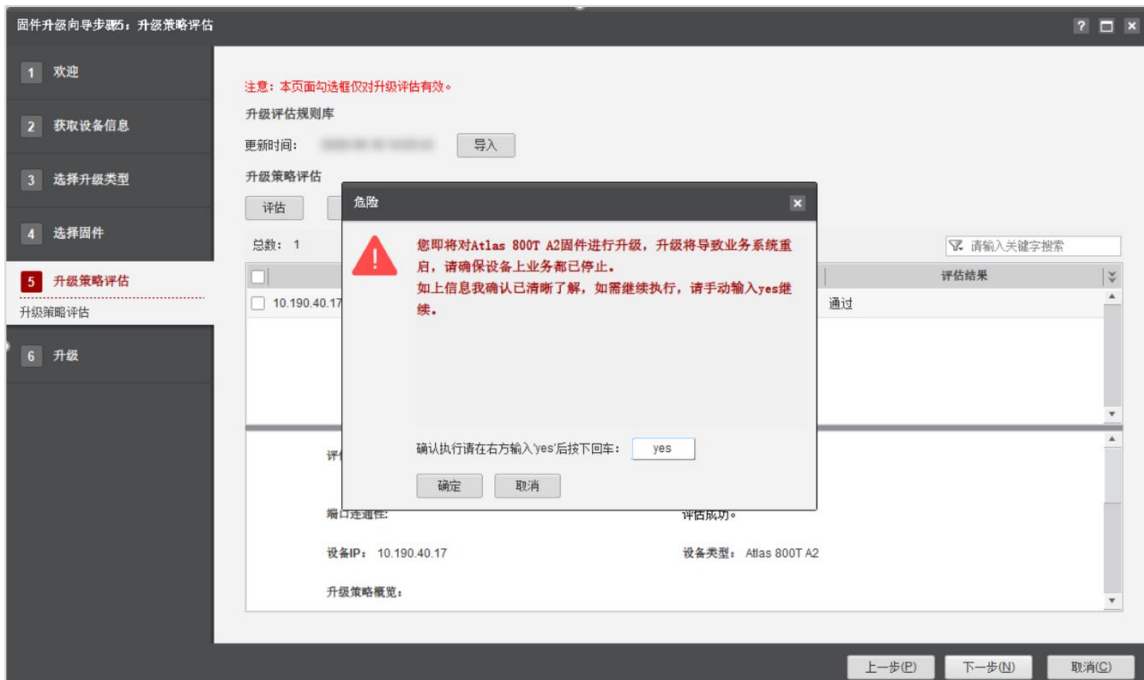


图2-48 二次升级确认

进入升级页面，等待升级完成。

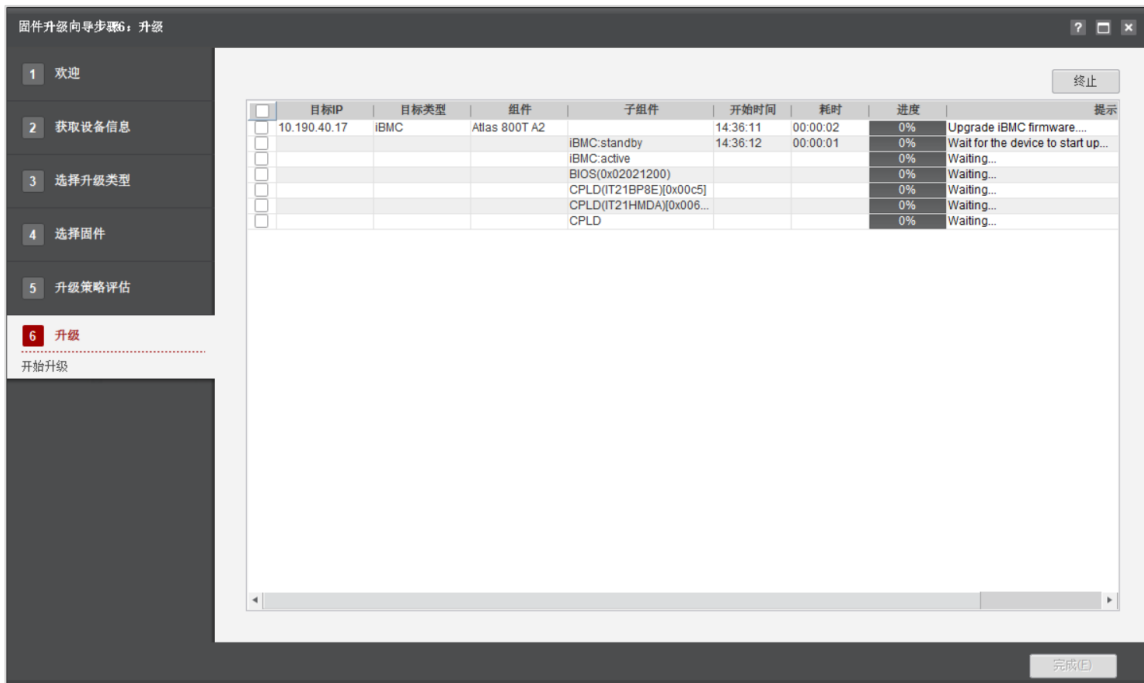


图2-49 升级固件中

大约 20 分钟后，升级完成，单击“完成”后，自动返回 SmartKit 首页。

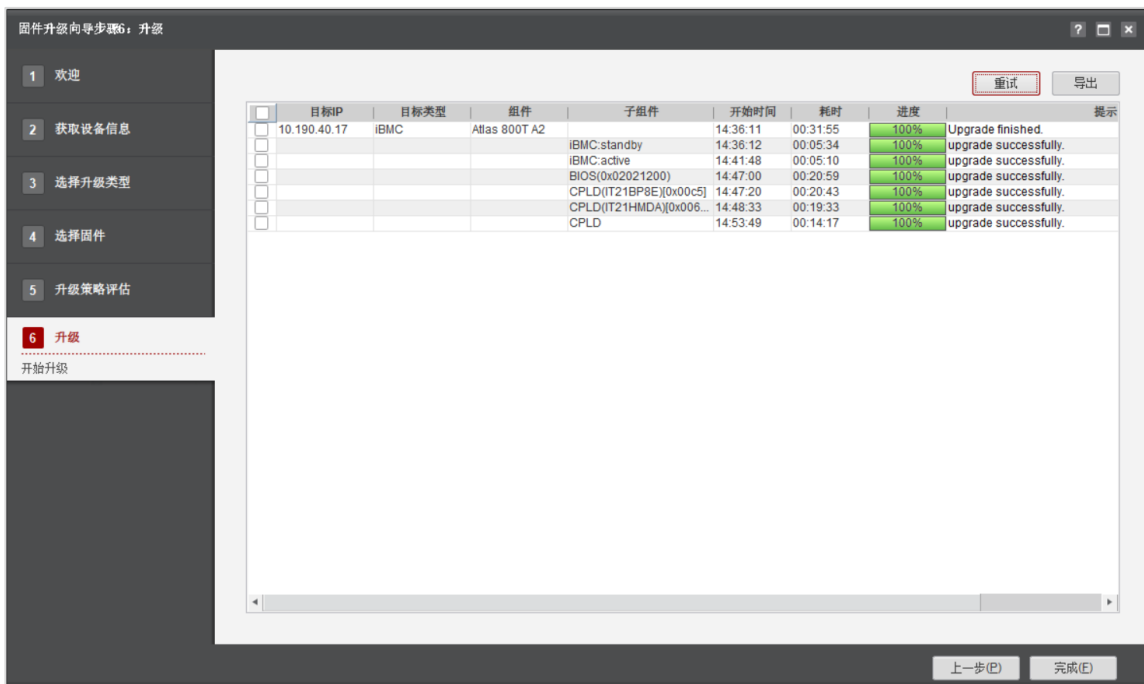


图2-50 固件升级完成

步骤 4 升级 Smart Provisioning

在首页选择固件升级。

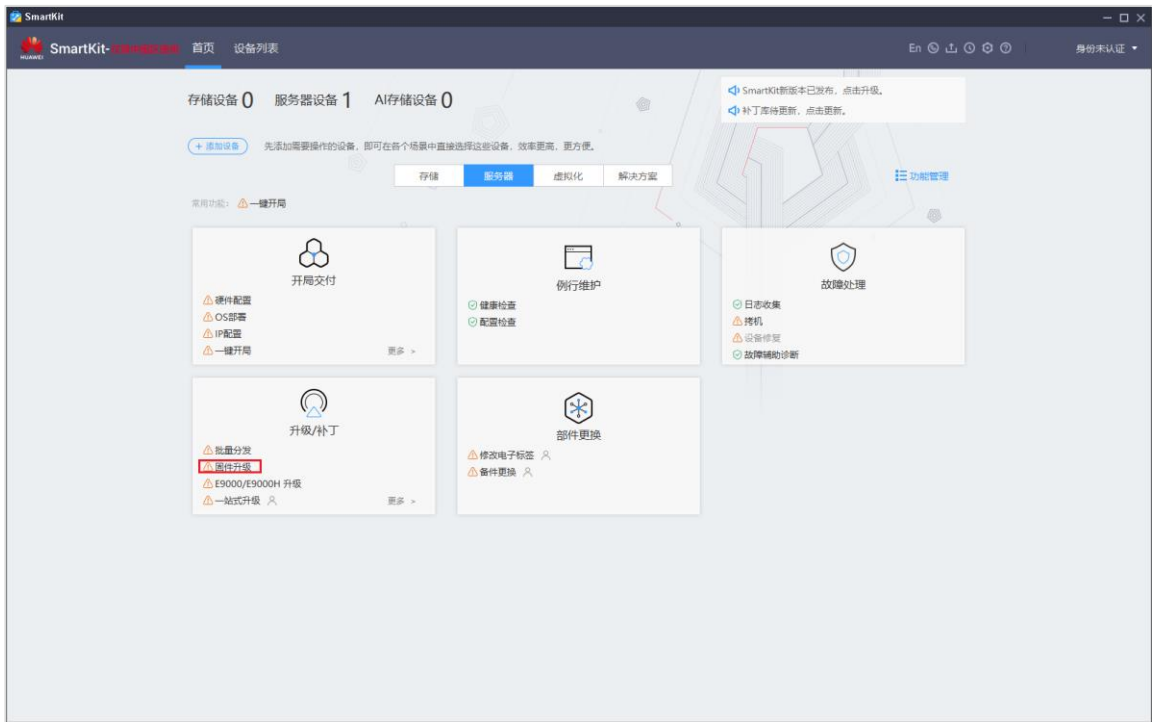


图2-51 进入固件升级

选择设备，勾选需要升级的设备。

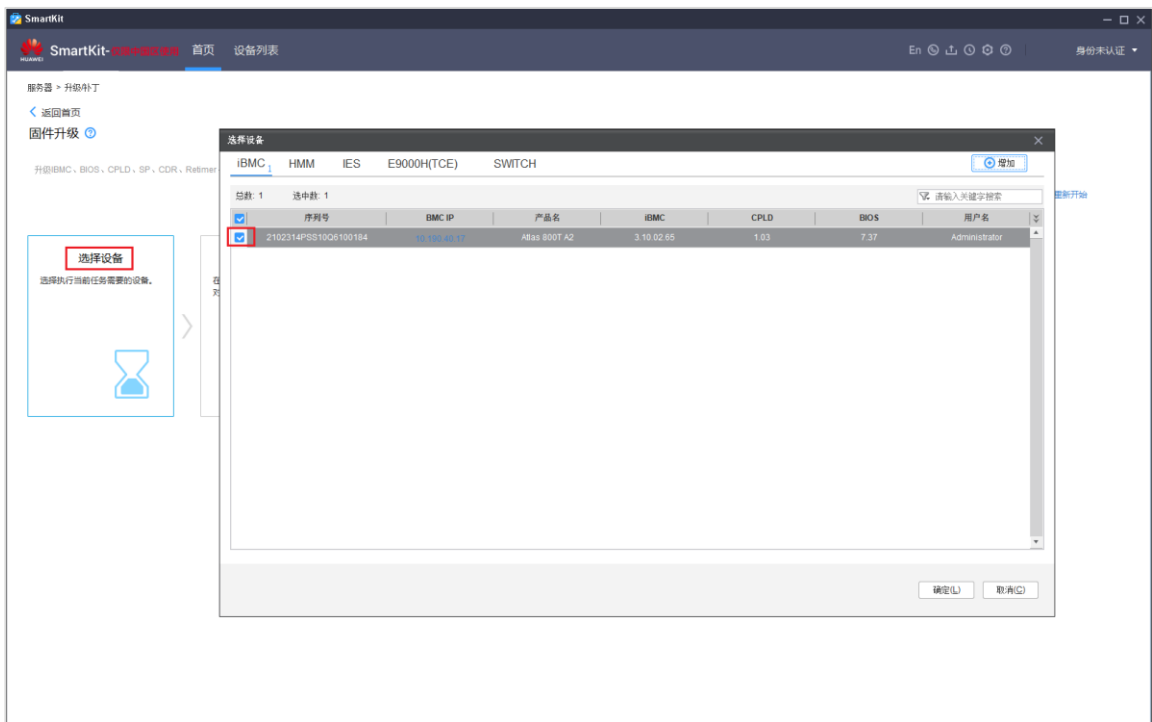


图2-52 选择待升级固件

直接选择固件升级。（这里跳过了升级评估，因为固件升级阶段可以执行评估）

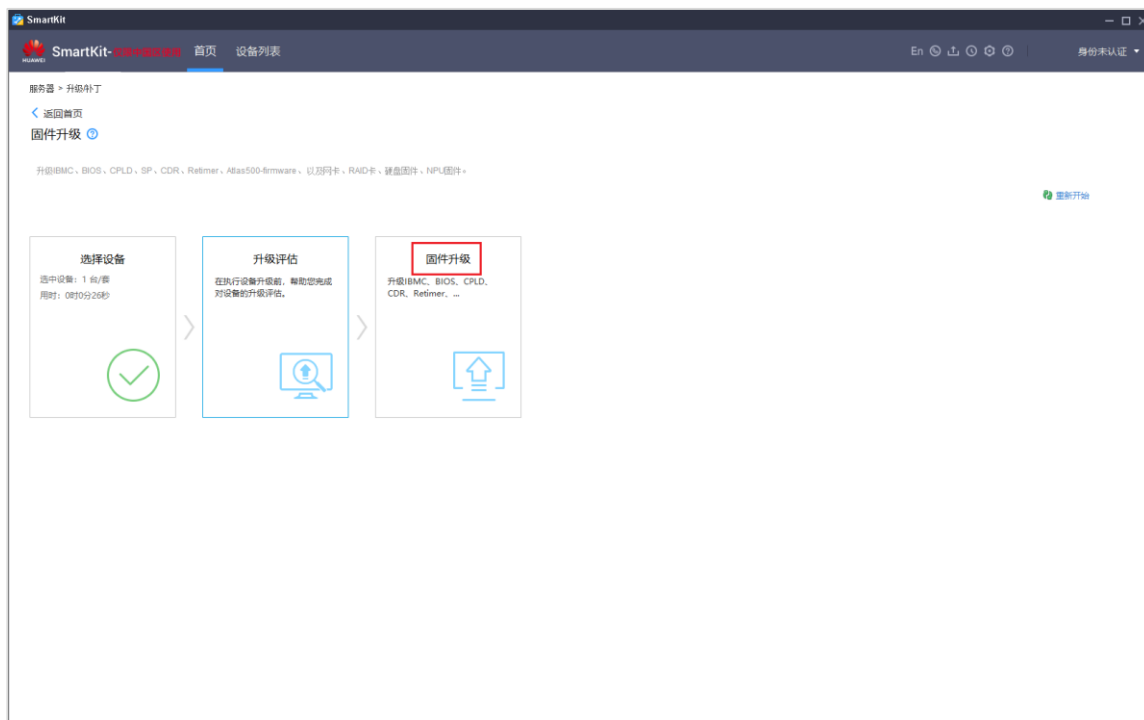


图2-53 直接进行固件升级

选择升级固件。

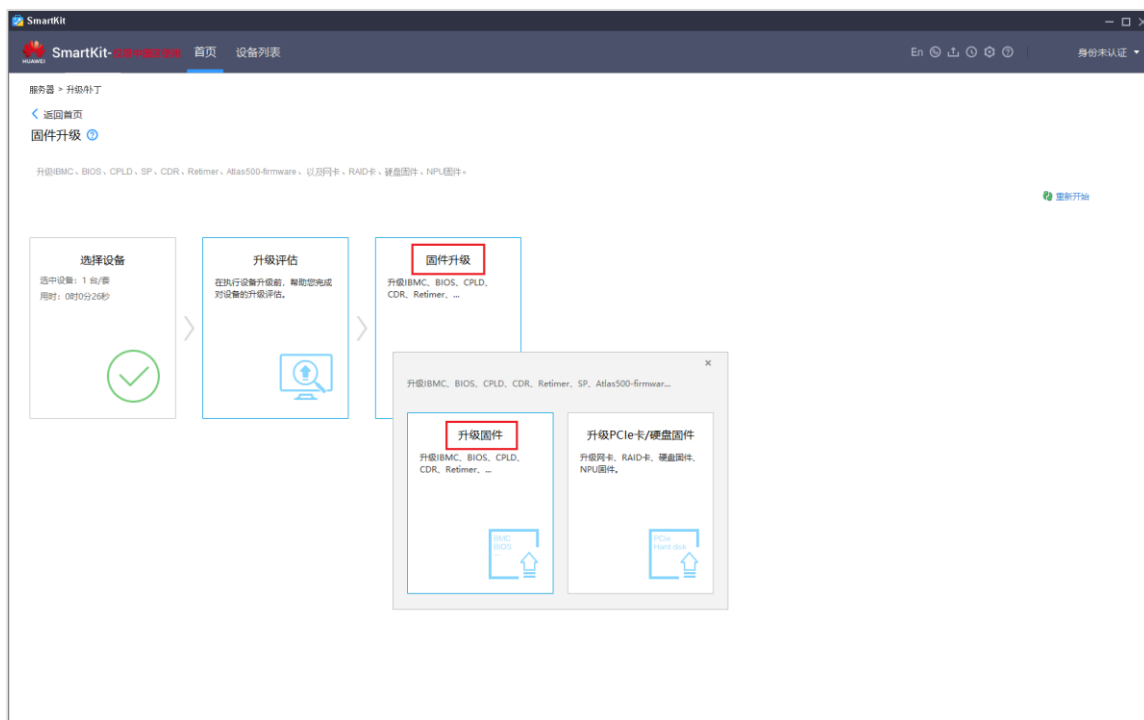


图2-54 进入升级固件窗口

单击“下一步”。



图2-55 固件升级欢迎页面

勾选需要升级的设备。

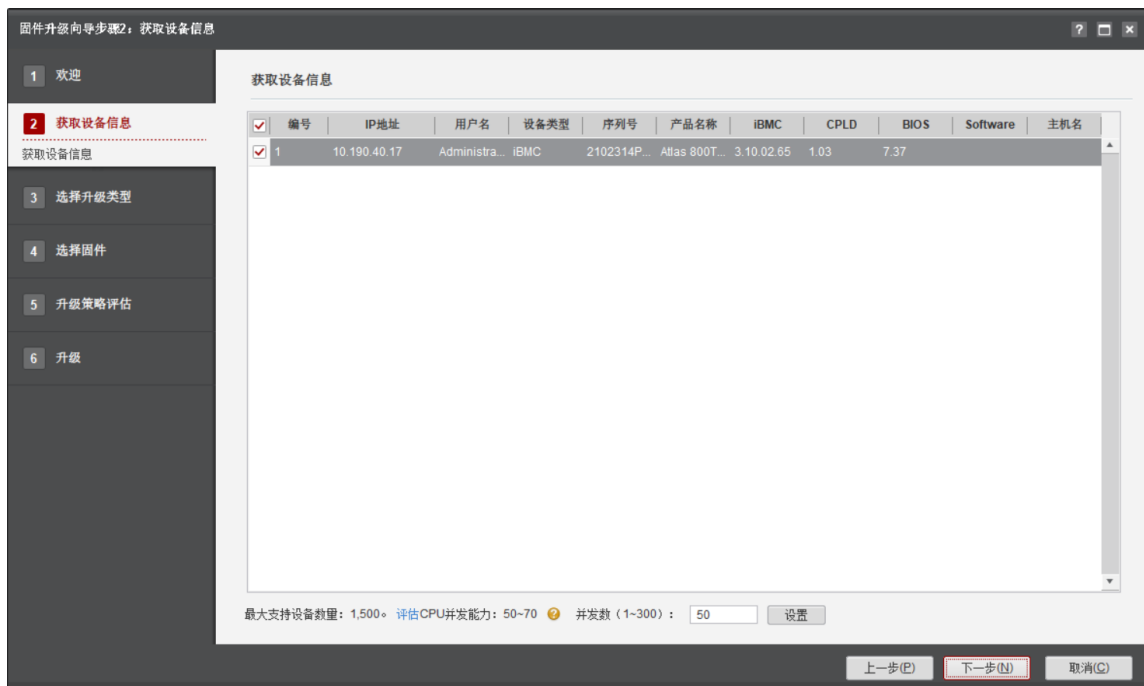


图2-56 获取设备信息

获取设备信息。

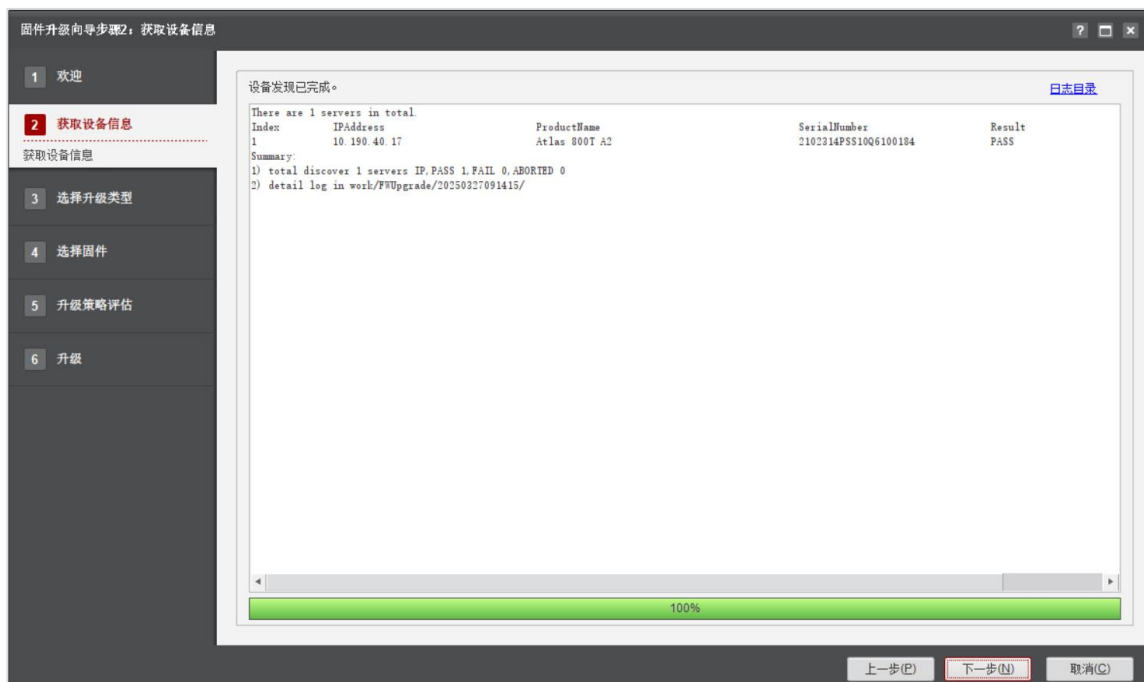


图2-57 设备信息获取成功

选择“单部件升级”。



图2-58 选择单部件升级

选择“升级 SmartProvisioning”，并选择本地的“SmartProvisioning_1.15.0_aarch64.iso”文件，该文件通过“[实验设备及软件介绍](#)”小节获取名为“SmartProvisioning_1.15.0_aarch64.zip”的软件包后解压而来。

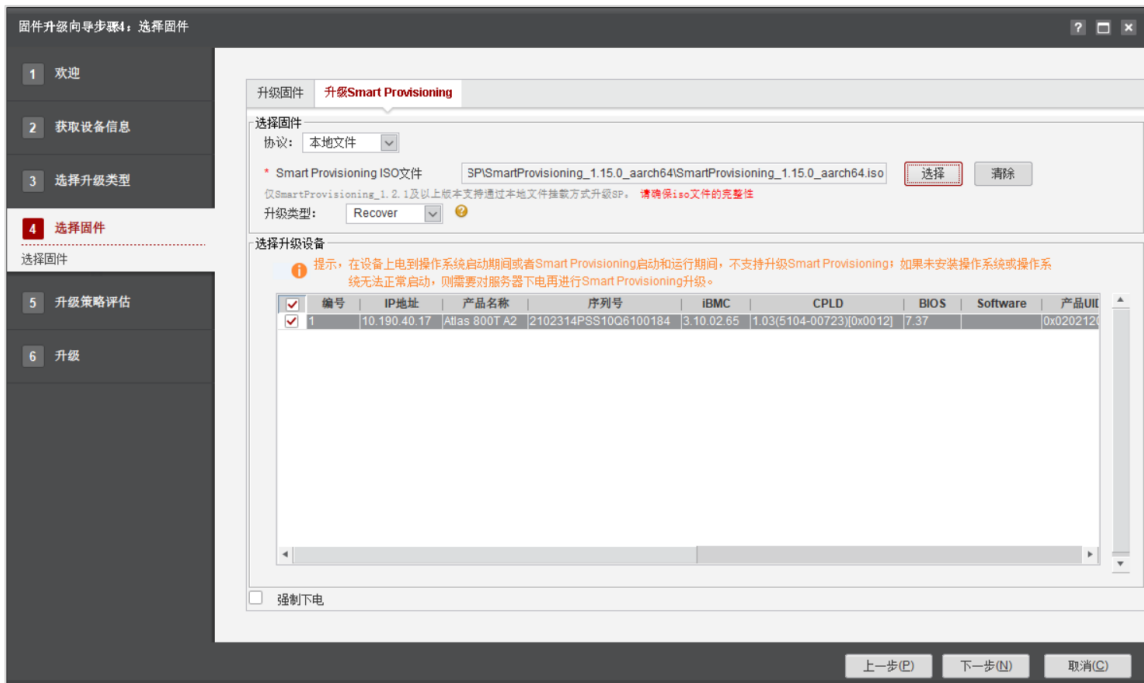


图2-59 选择 SP 固件包

升级评估。



图2-60 完成升级评估

按照要求输入 “yes” 后确定。

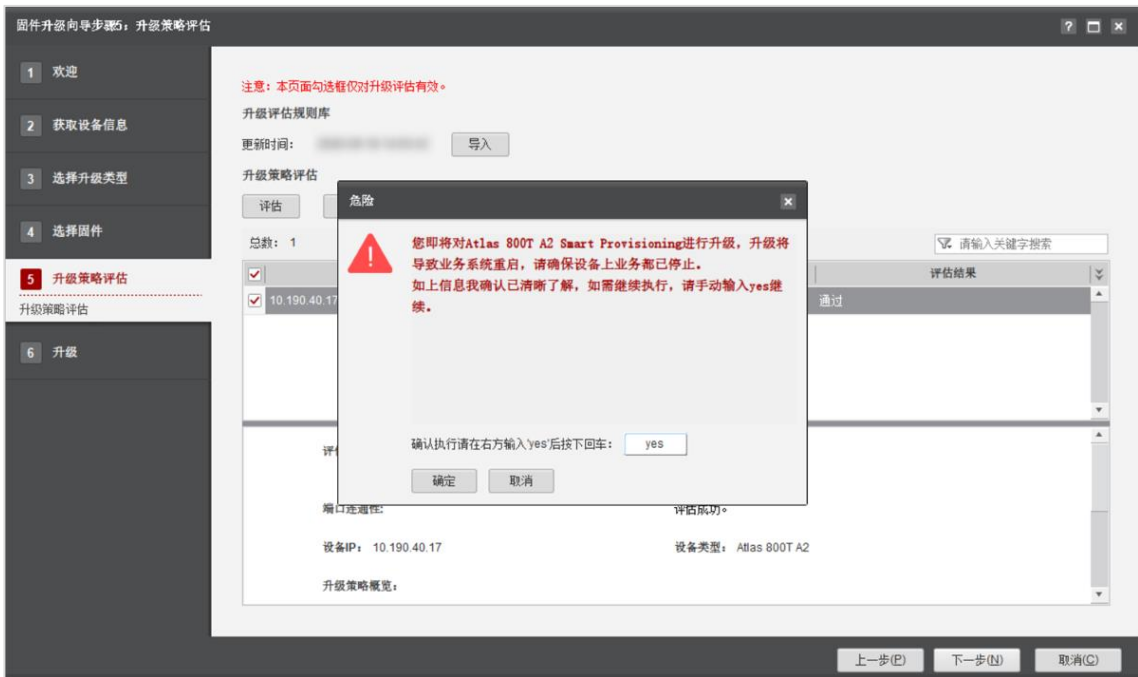


图2-61 确认警告

开始升级。

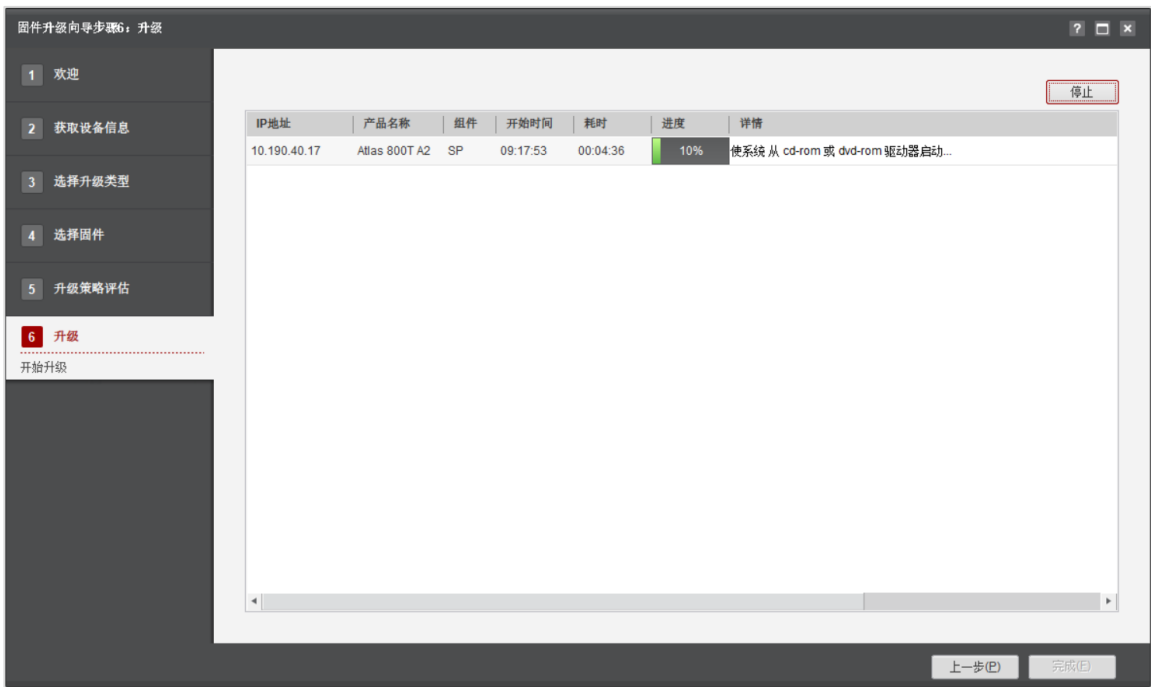


图2-62 开始升级

升级完成后单击“完成”。

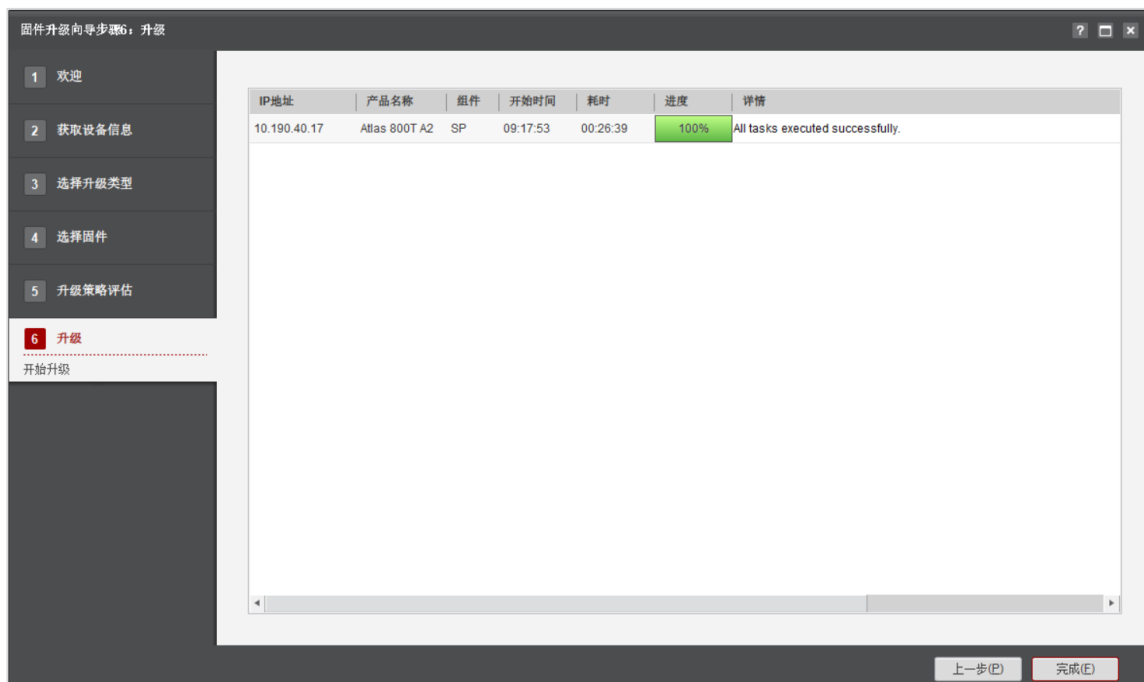


图2-63 升级完成

步骤 5 版本检查

在 SmartKit 首页，选择服务器页签下的健康检查。

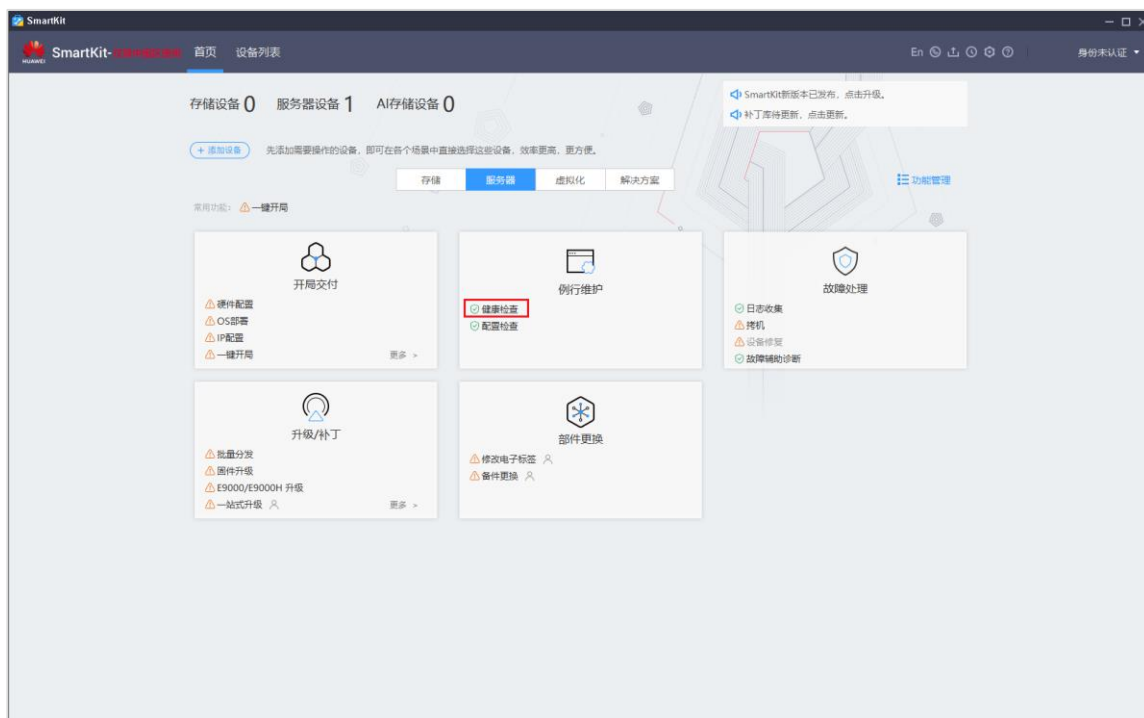


图2-64 选择健康检查

选择待进行版本检查的设备。

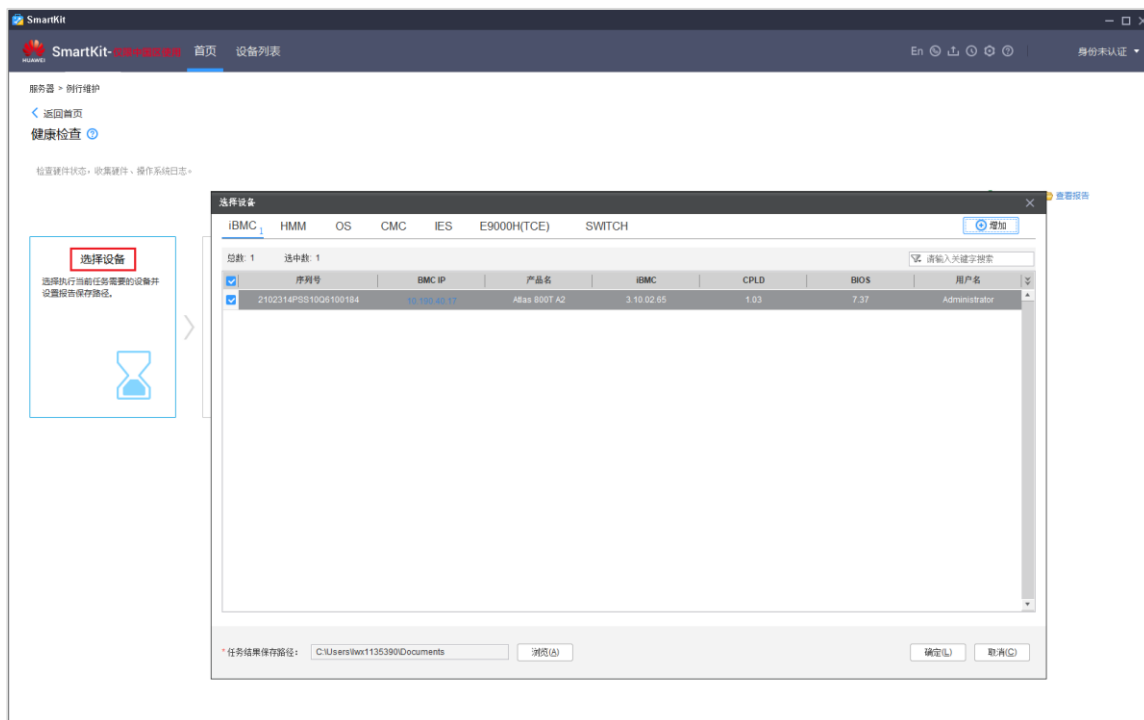


图2-65 选择设备

执行健康检查。



图2-66 选择健康检查

进入健康检查欢迎页面。



图2-67 健康检查欢迎页面

单击“下一步”，进入获取设备信息页面。



图2-68 获取设备信息

选择检查项目，在此，我们只需要检查配置信息、资产信息和日志信息，获取对应的固件版本即可。



图2-69 选择检查项目

检查项目选好后，执行健康检查。



图2-70 执行健康检查

健康检查完成，单击“下一步”。

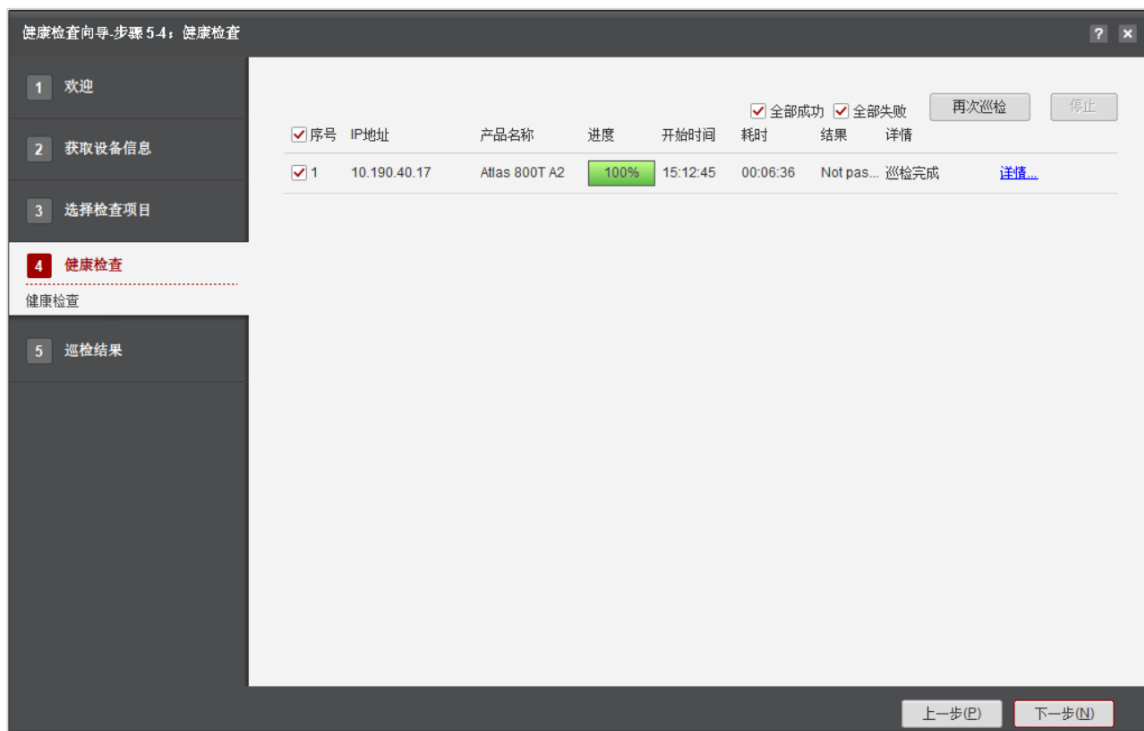


图2-71 健康检查完成

填写检查报告封面信息后，单击“生成报告”。

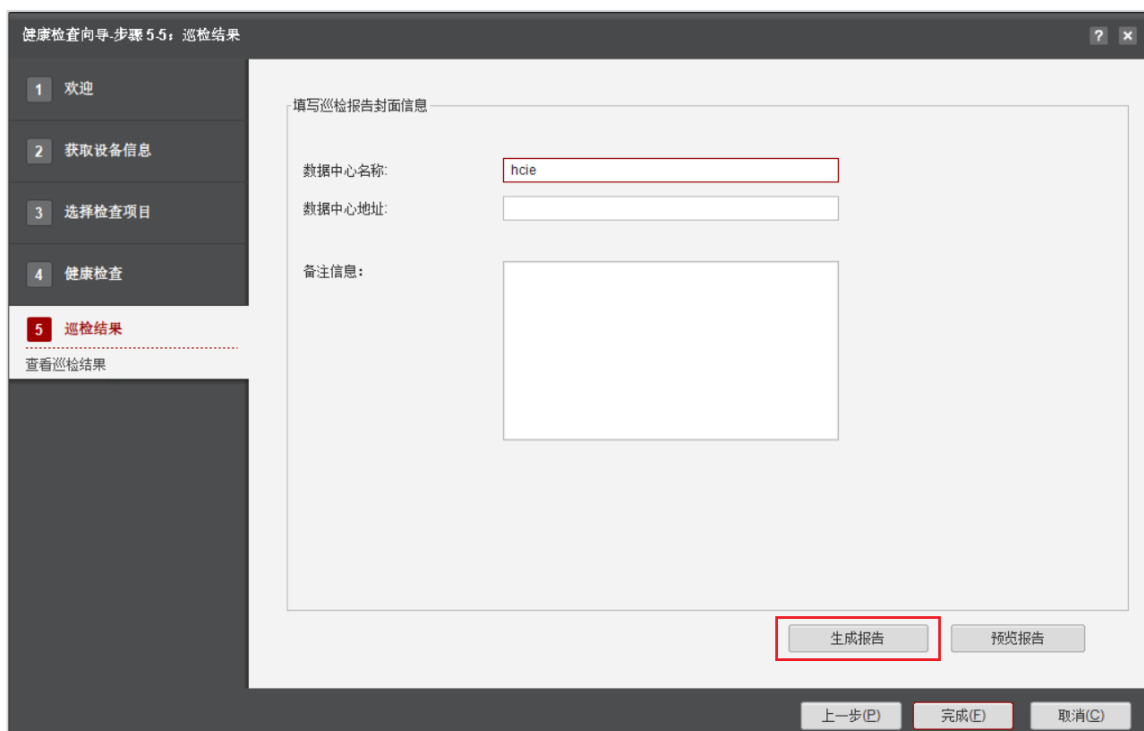


图2-72 生成健康检查报告

根据提示进入报告目录，查看报告。

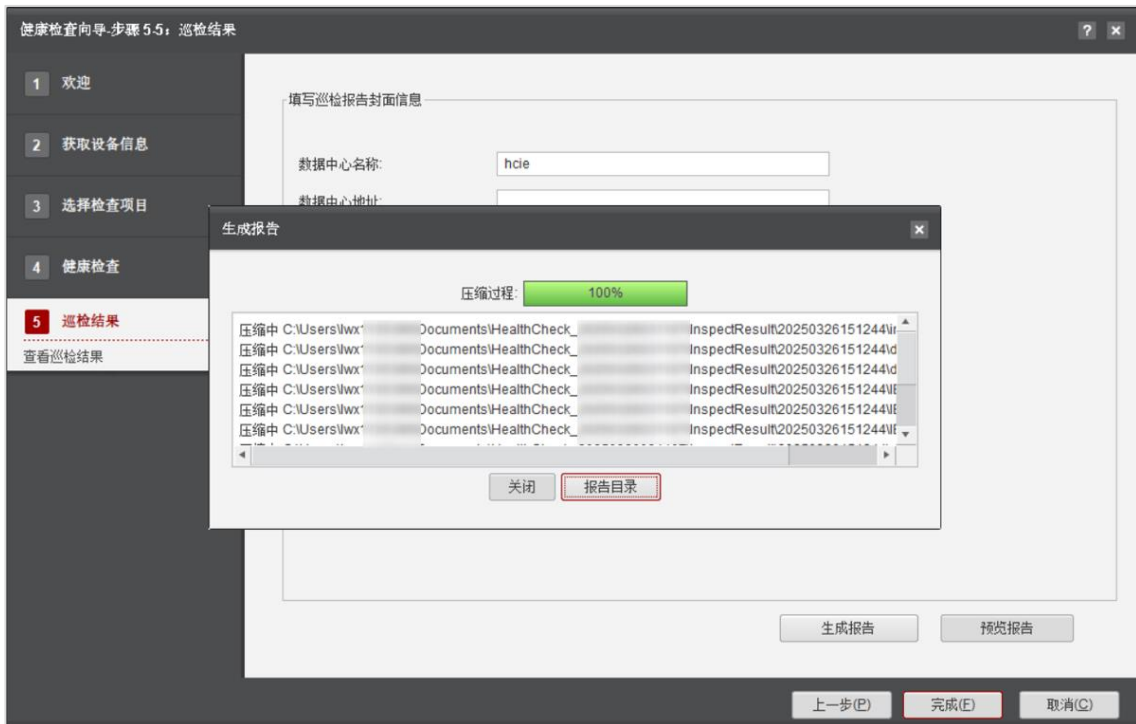


图2-73 进入报告目录

打开“.html”格式的报告文件，查看健康检查报告，是否存在版本风险。

Common							
产品名称	IP地址	主机名称	部件名称	部件类型	版本	版本风险	
Atlas 800T A2	10.190.40.17		IBMC Active	Active	3.10.02.65	✓	
			IBMC Backup	Backup	3.10.02.65	✓	
			BIOS		7.37	✓	
			MainBoardCPLD	Mainboard	1.03	✓	
			chassisDiskBP1CPLD	IT21BPBE	1.01	✓	
			mainboardNPUBoardCPLD	IT21SD8B	1.00	✓	
			NpuBoardHIAM1CPLD	HIAM 1	3.06	✓	
			NpuBoardHIAM2CPLD	HIAM 2	3.06	✓	
			NpuBoardHIAM3CPLD	HIAM 3	3.06	✓	
			NpuBoardHIAM4CPLD	HIAM 4	3.06	✓	
			NpuBoardHIAM5CPLD	HIAM 5	3.06	✓	
			NpuBoardHIAM6CPLD	HIAM 6	3.06	✓	
			NpuBoardHIAM7CPLD	HIAM 7	3.06	✓	
			NpuBoardHIAM8CPLD	HIAM 8	3.06	✓	
			NpuBoardHAMMCU1	HAMMCU1	23.3.5	✓	
			NpuBoardHAMMCU2	HAMMCU2	23.3.5	✓	
			NpuBoardHAMMCU3	HAMMCU3	23.3.5	✓	
			NpuBoardHAMMCU4	HAMMCU4	23.3.5	✓	
			NpuBoardHAMMCU5	HAMMCU5	23.3.5	✓	
			NpuBoardHAMMCU6	HAMMCU6	23.3.5	✓	
			NpuBoardHAMMCU7	HAMMCU7	23.3.5	✓	
			NpuBoardHAMMCU8	HAMMCU8	23.3.5	✓	
			DISK0	HW625T31T9L003N	1086	✓	
			DISK1	HW625T31T9L003N	1086	✓	
			DISK4	HW625T31T9L003N	1086	✓	
			DISK5	HW625T31T9L003N	1086	✓	
			PCIe Card 1 (SP686C-M-16i) Controller	SP686C-M-16i 4G	1.3.11.22	✓	
			mainboardTM210	TM210		✓	
			mainboardFLEXIOA1	TM280		✓	
			NPUBoardFLEXIOB1	FLEX IO B1		✓	
			NPUBoardFLEXIOB2	FLEX IO B2		✓	
			NPUBoardFLEXIOB3	FLEX IO B3		✓	
			NPUBoardFLEXIOB4	FLEX IO B4		✓	

步骤 6 重启服务器

相关固件升级成功后，对服务器执行重启操作。进入 SmartKit 首页，并选择服务器页签下的“硬件配置”。

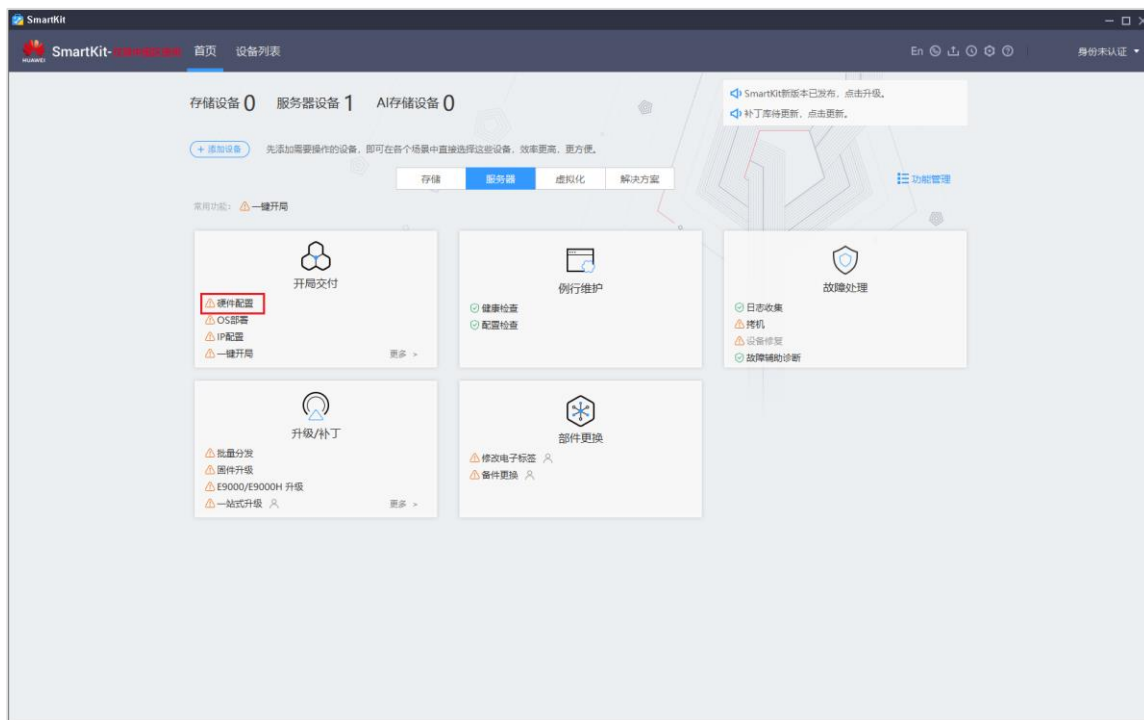


图2-74 选择硬件配置

选择待重启的设备。

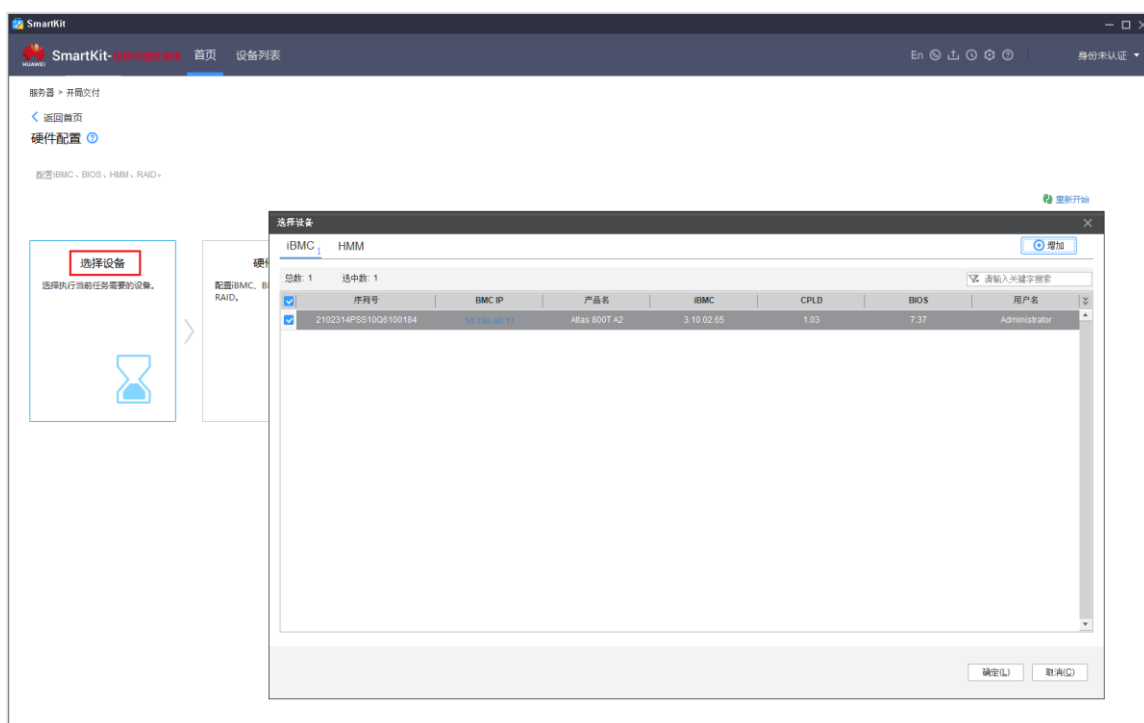


图2-75 选择设备

选择设备后，选择“电源控制”选项。



图2-76 选择电源控制

进入电源控制向导欢迎页面。



图2-77 电源控制欢迎页面

获取待重启设备的信息。

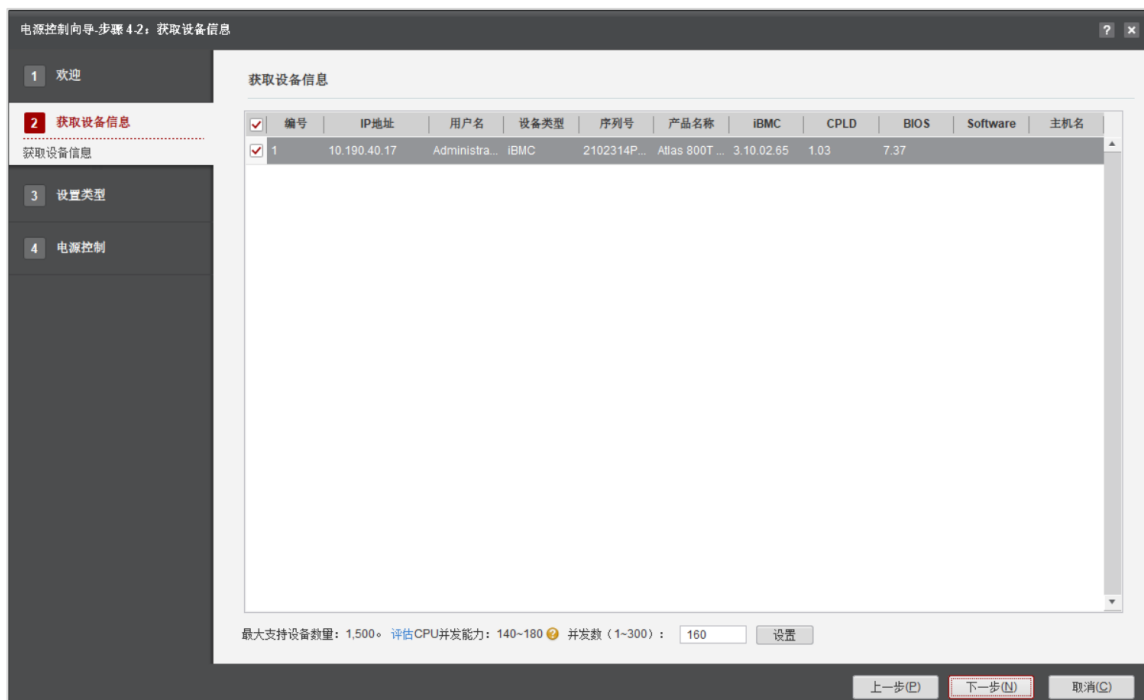


图2-78 获取设备信息

设备信息获取检查通过后，单击“下一步”。

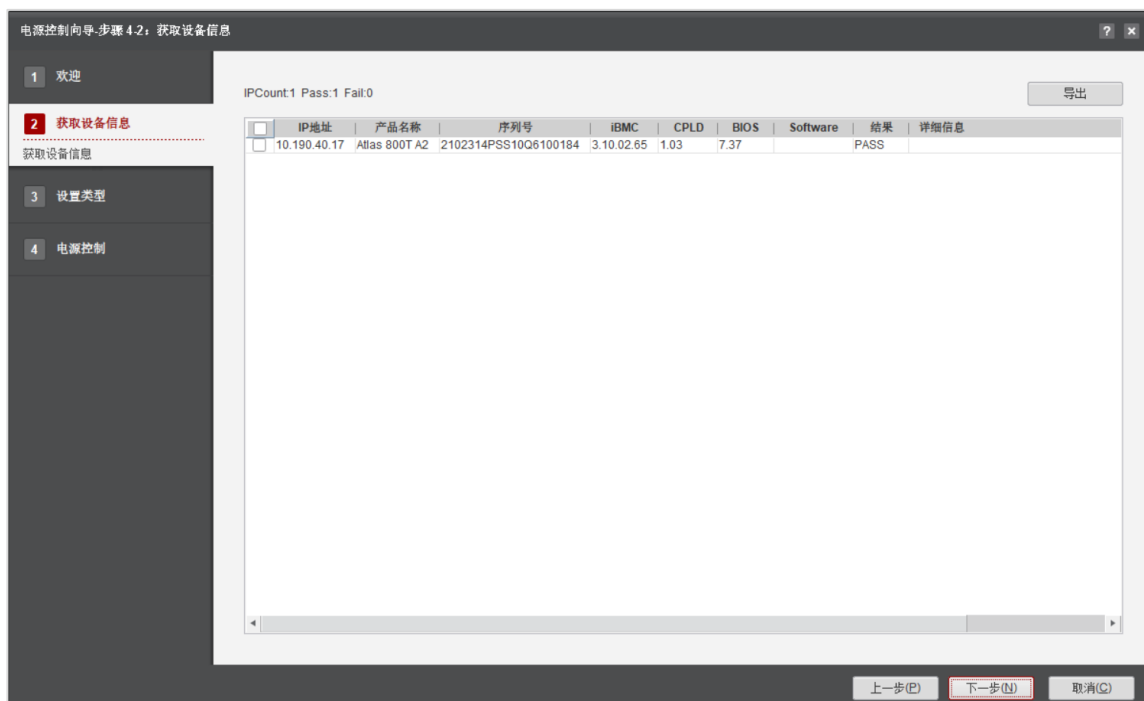


图2-79 设备信息获取成功

在设置类型页面，选择“强制系统重启”，并单击“下一步”。



图2-80 强制系统重启

根据提示，输入“yes”后，单击“确定”。

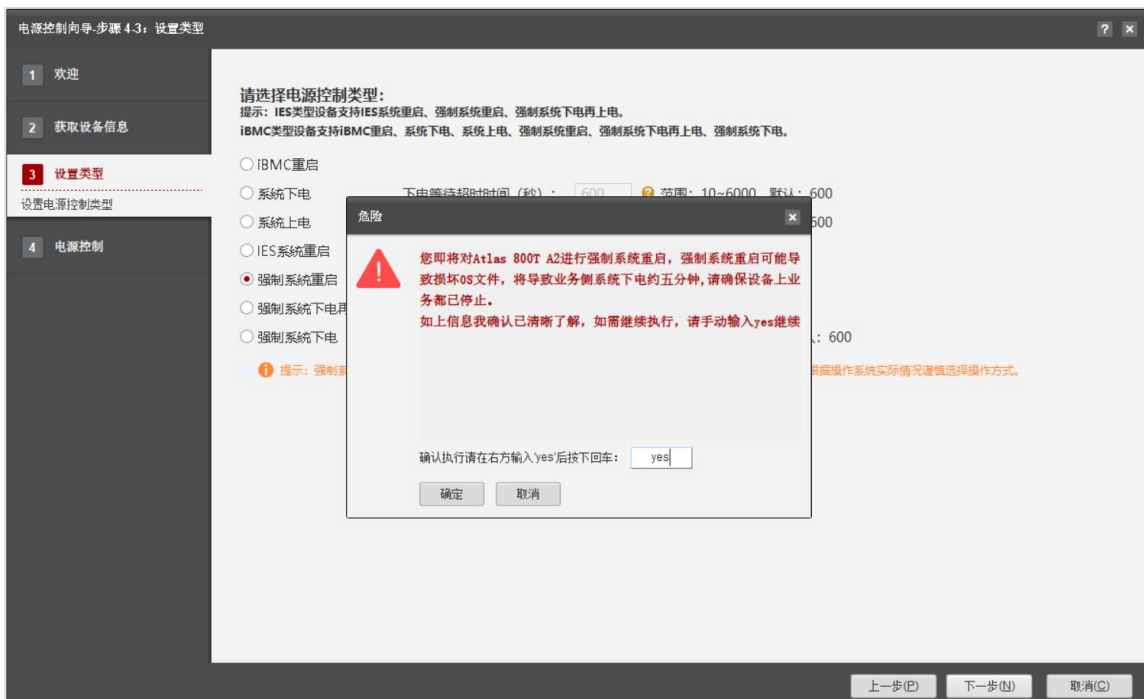


图2-81 确认警告

系统重启命令下发成功，单击“完成”，并等待系统重启完成。

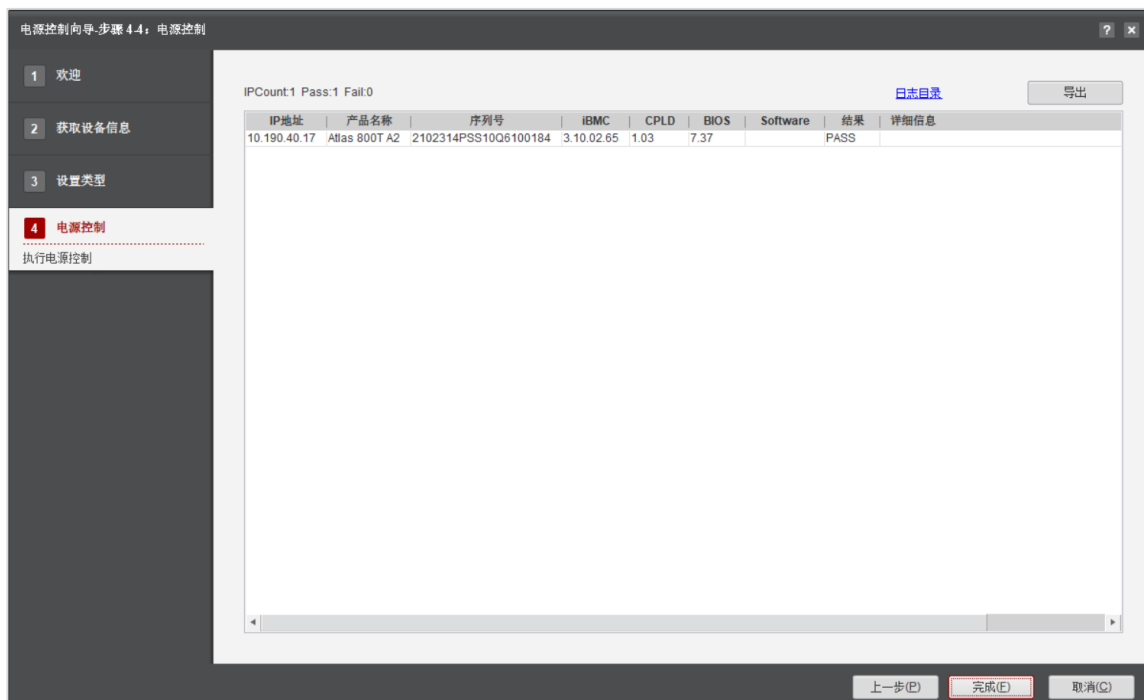


图2-82 系统重启配置完成

2.3 Atlas 服务器操作系统安装

步骤 1 镜像准备

根据[实验设备及软件介绍](#)小节获取名为“CentOS-7-aarch64-Everything-1810.iso”的操作系统镜像包。

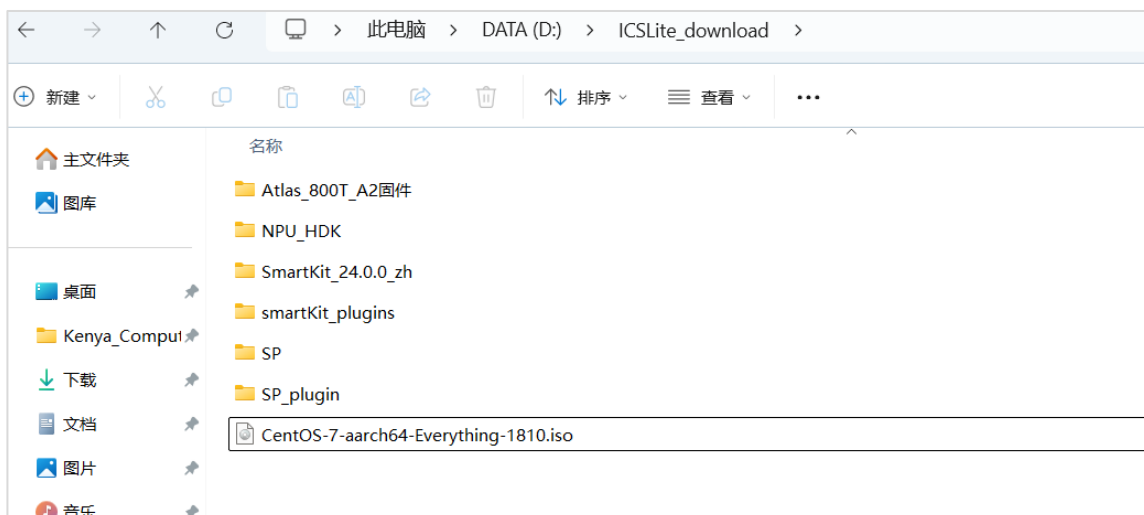


图2-83 OS 安装镜像

步骤 2 安装 OS

在 SmartKit 首页，选择“服务器”，并单击“一键开局”（如弹出升级提示，不升级，单击继续使用）。

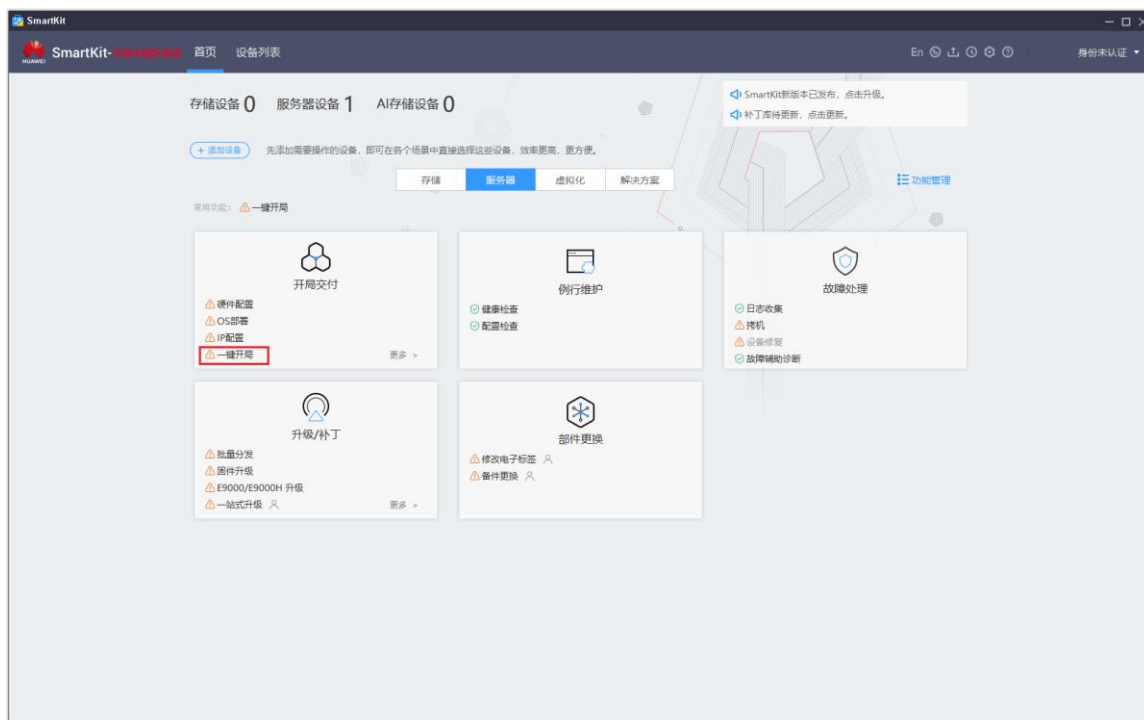


图2-84 选择一键开局

选择设备，选择“Atlas 服务器固件升级小节步骤 2”添加的服务器，单击“确定”。

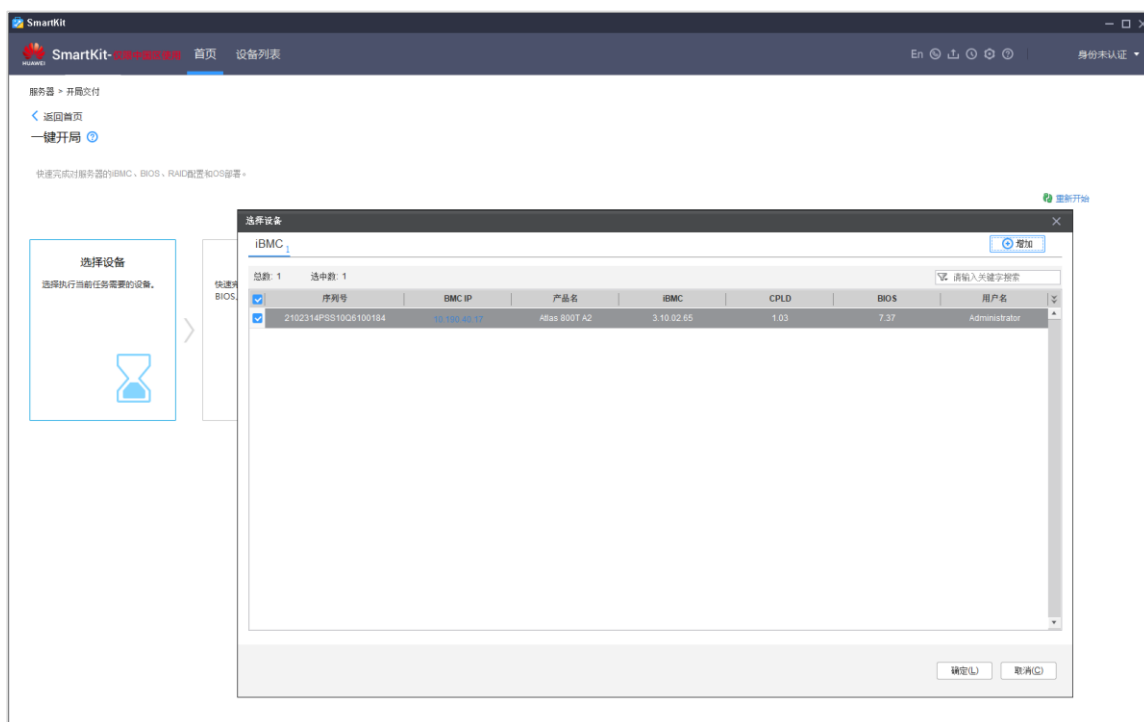


图2-85 选择待开局设备

单击“一键开局”。

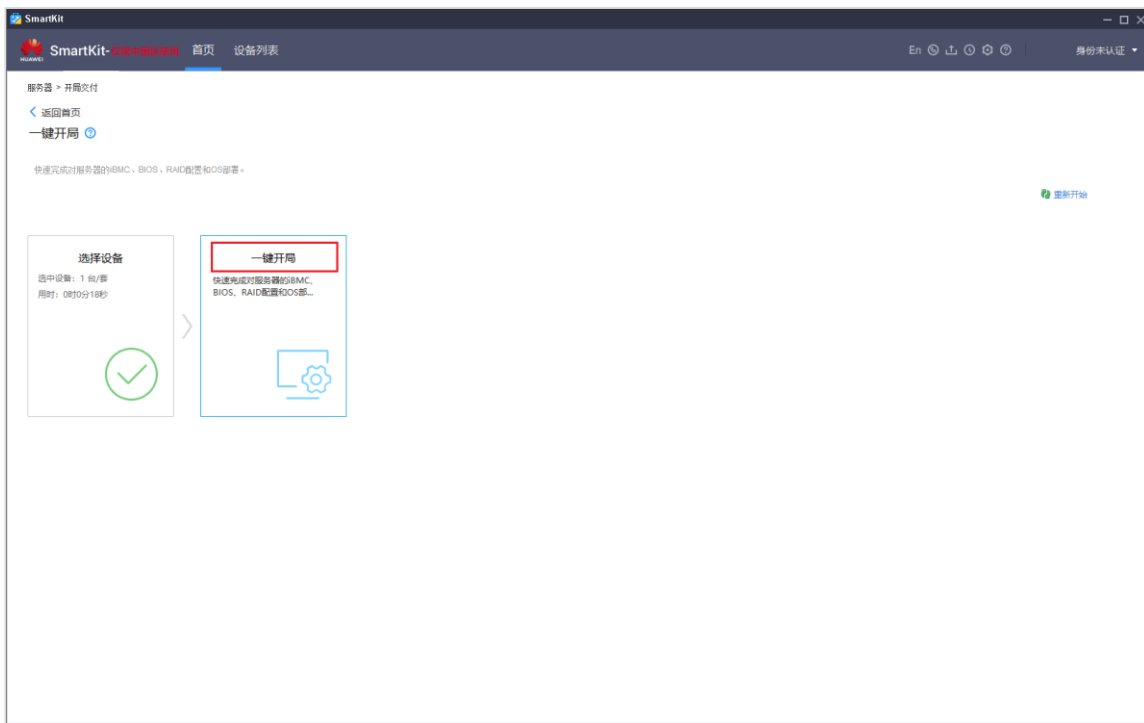


图2-86 选择“一键开局”

等待服务器状态变为“未设置参数”后，勾选服务器，并单击“设置参数”。



图2-87 设置参数

在参数设置页面，勾选“配置 BIOS”、“配置 RAID”、“部署 OS”后，单击“下一步”。



图2-88 进入配置向导

获取设备信息页面，单击“获取”。信息获取完成后，单击“下一步”，进入“BIOS 配置”页面。



图2-89 获取设备信息



图2-90 设备信息获取成功

在 BIOS 配置页面，单击“MISC config”，勾选“Support GOP FB for SM750”（确保值为“Enable”），勾选“Support SPCR”（确保值为“Disabled”）

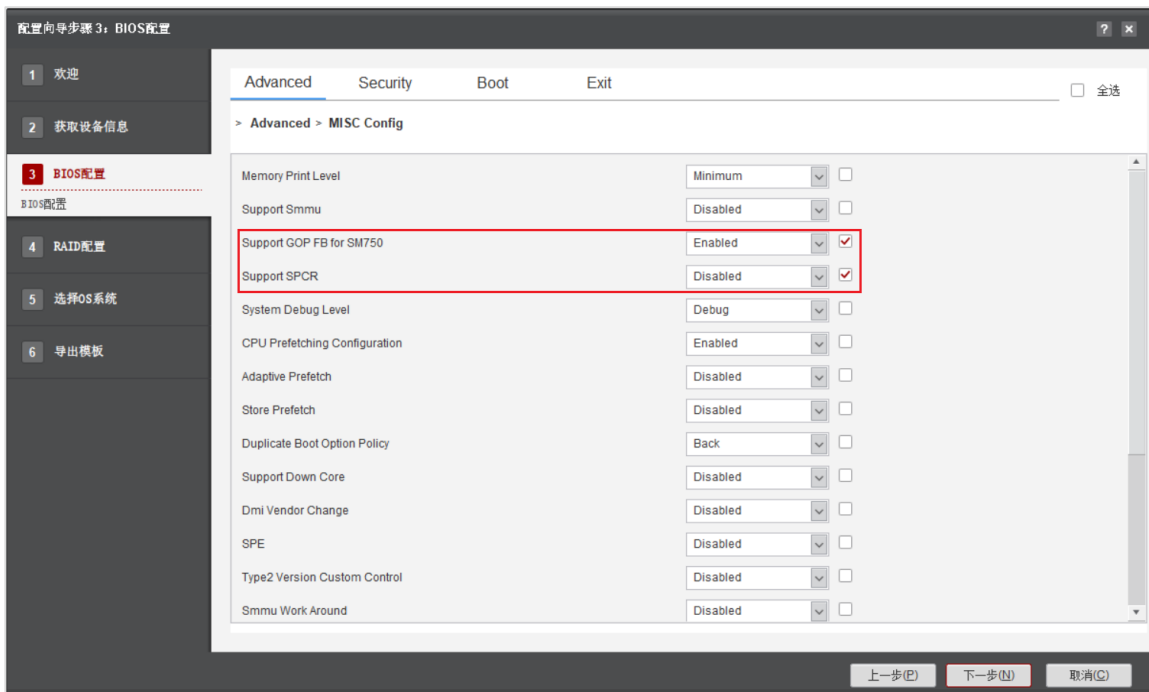


图2-91 配置 BIOS1

返回 Advanced，选择 Memory Config。勾选 One Numa Per Socket（确保值为“Enable”）。

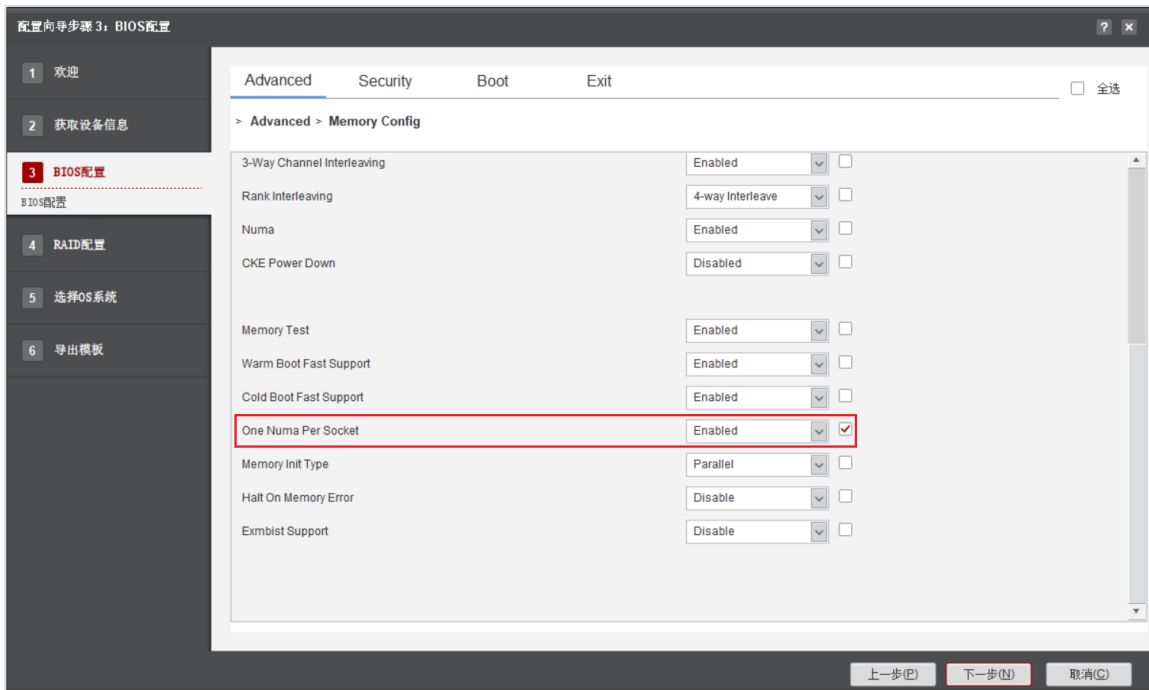


图2-92 配置 BIOS2

返回 Advanced，选择 IPMI iBMC Configuraion。如图所示都进行勾选，并确保值为“Disable”。

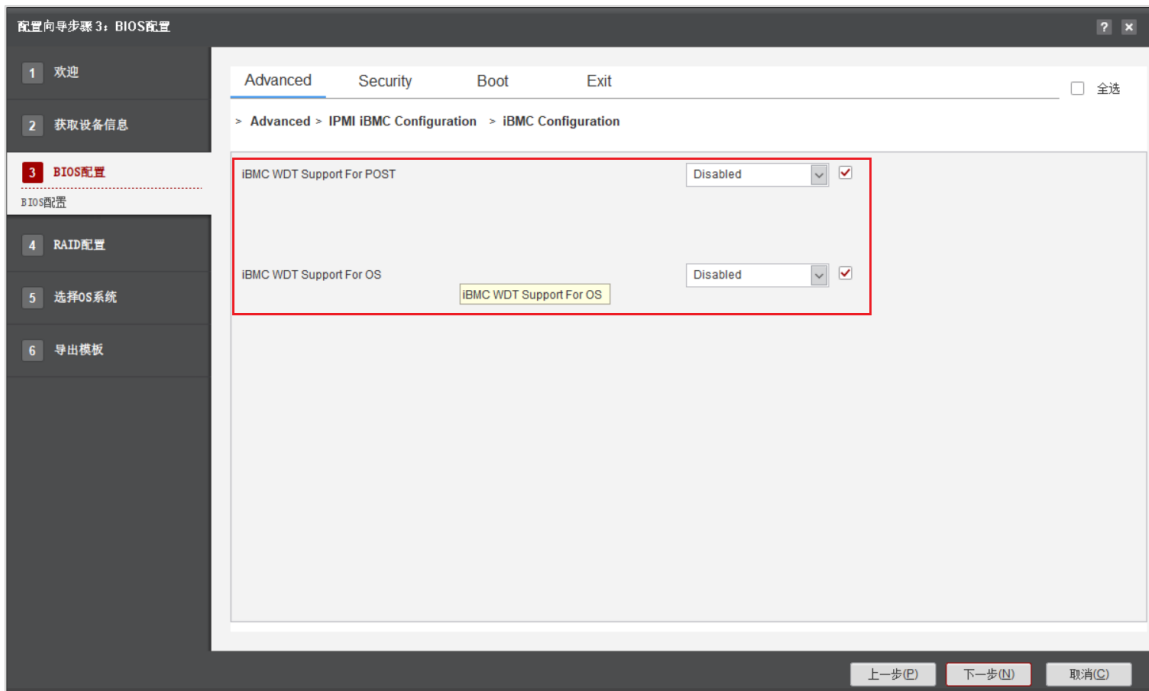


图2-93 配置 BIOS3

返回 Advanced，选择 Performance Config。修改 Power Policy 策略为 Performance，并勾选。

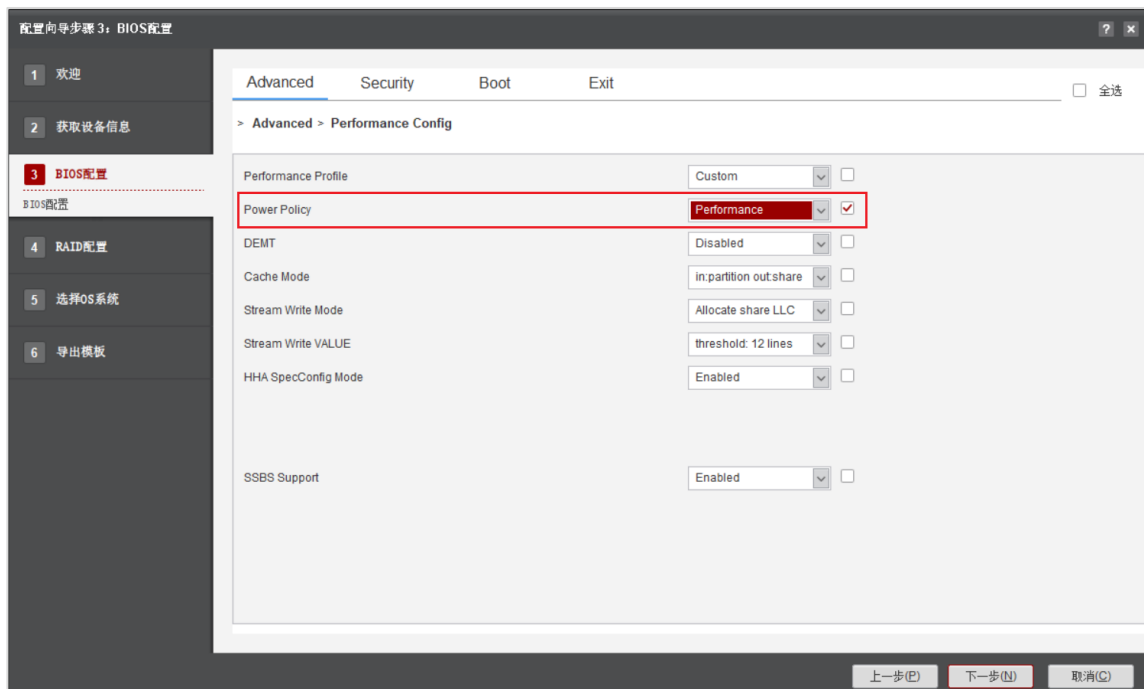


图2-94 配置 BIOS4

选择 Boot，勾选 SP Boot，确保值为“Enabled”。单击“下一步”，进入 RAID 配置页面。

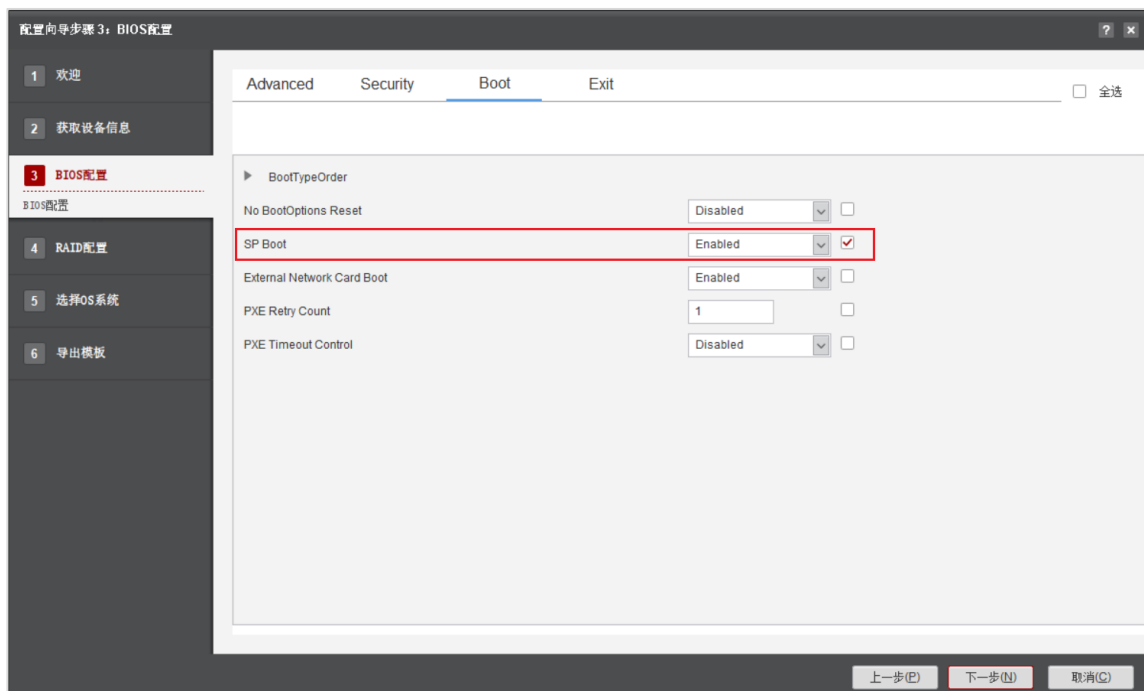


图2-95 配置 BIOS5

在 RAID 配置页面，单击“创建”，打开 RAID 配置窗口。

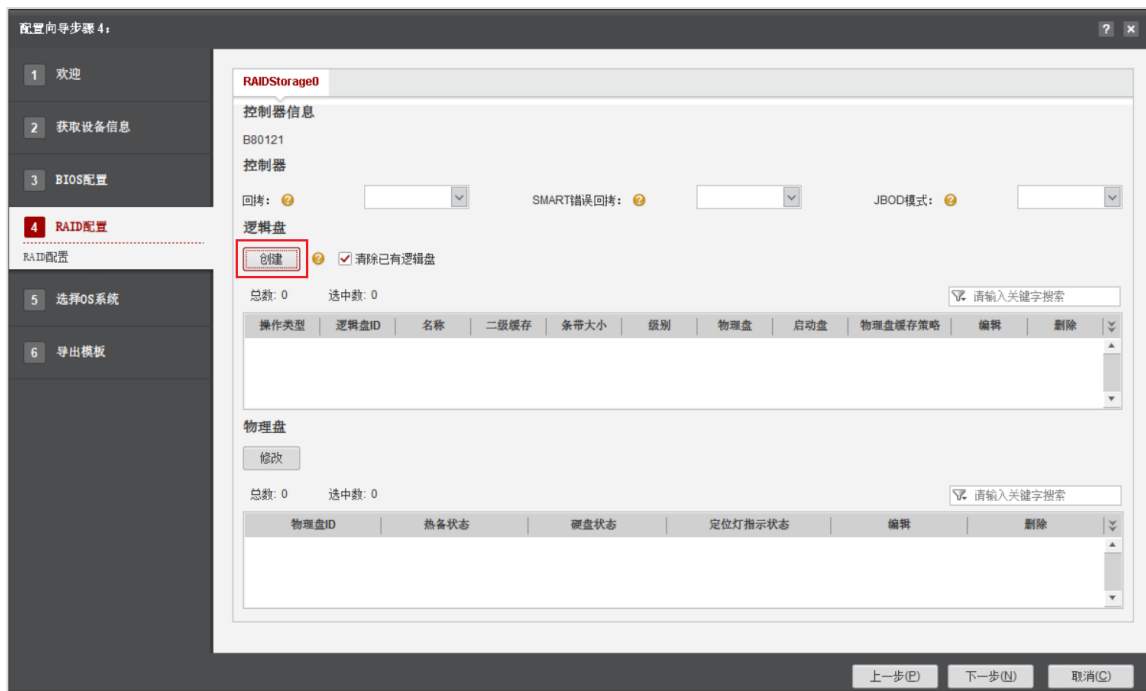


图2-96 创建逻辑盘

在创建逻辑盘页面，选择前两个物理盘 Disk0 和 Disk1，单击“保存”。

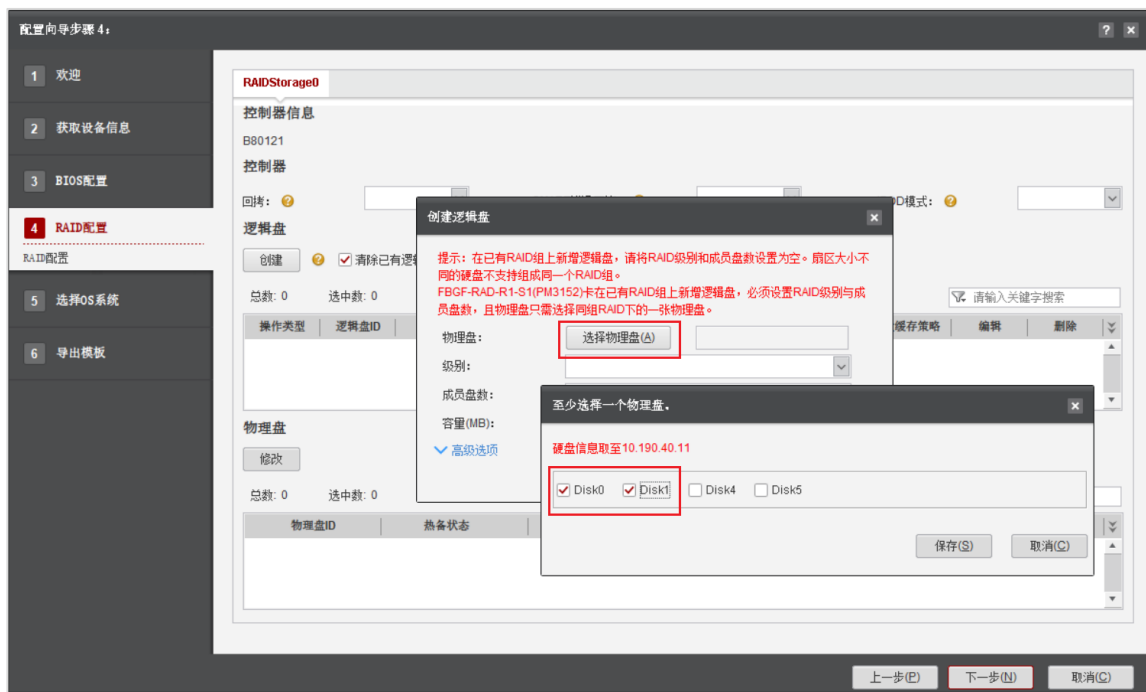


图2-97 选择物理盘

级别选择“RAID 1”，打开高级选项，启动盘选择“yes”。单击“保存”。

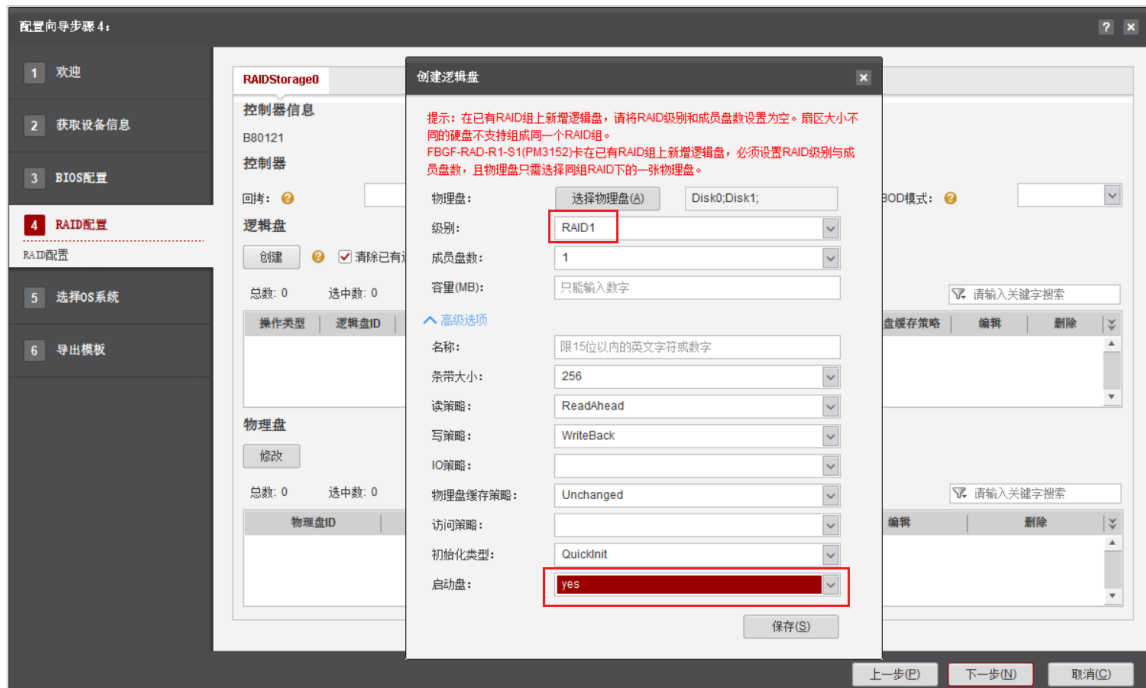


图2-98 RAID 级别配置

单击“下一步”后，确认勾选警告，单击“确定”，进入选择 OS 系统页面。

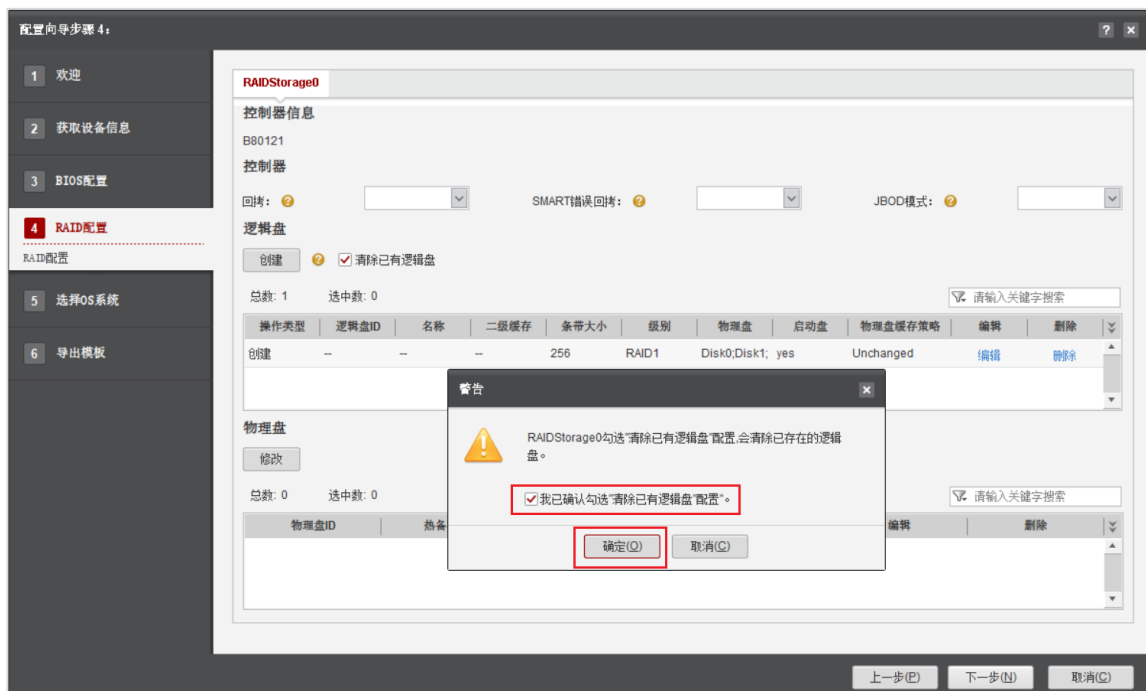


图2-99 确认警告

在选择 OS 系统页面，部署方式选择 SmartProvisioning，系统类型选择 CentOS，系统版本选择 CentOS Linux 7U6 aarch64，并选择本地的 OS 镜像文件“CentOS-7-aarch64-Everything-1810.iso”，单击“下一步”。

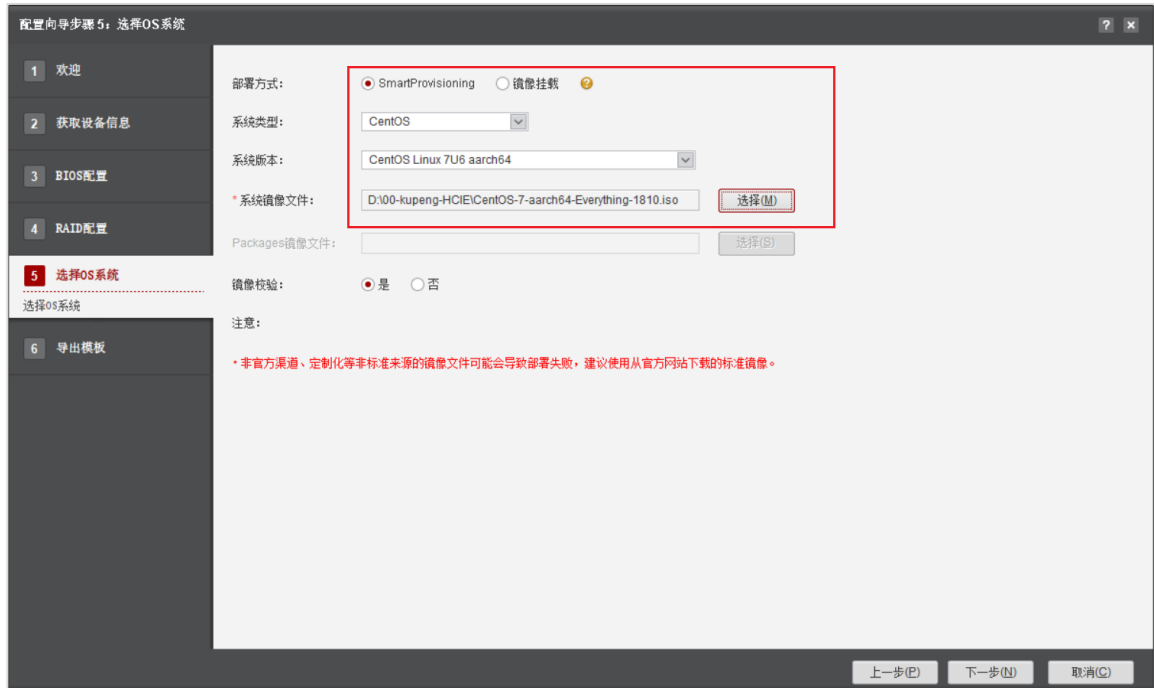


图2-100 选择操作系统类型

在设置 OS 参数页面，配置相关参数，相关参数及配置注意如下：

- 管理员密码：安装 OS 的时候设置的 root 用户密码
- 时区：根据实际情况选择，中国区选择 Asia/Shanghai
- 驱动固件包路径配置：在本地新建一个文件夹，比如“SP_plugin”，在此文件夹下，新建两个文件夹“driver”和“firmware”，firmware 文件夹保持空目录，将“[实验设备及软件介绍](#)”小节获取的名为“ComputingComponentiDriver-CentOS7.6-Driver-ARM_2.1.0.zip”的软件包及相关校验文件放到 driver 文件夹下。根据提示启动 SFTP 服务并配置可用网卡。（注意：需要确保当前操作 PC 的防火墙为关闭状态）

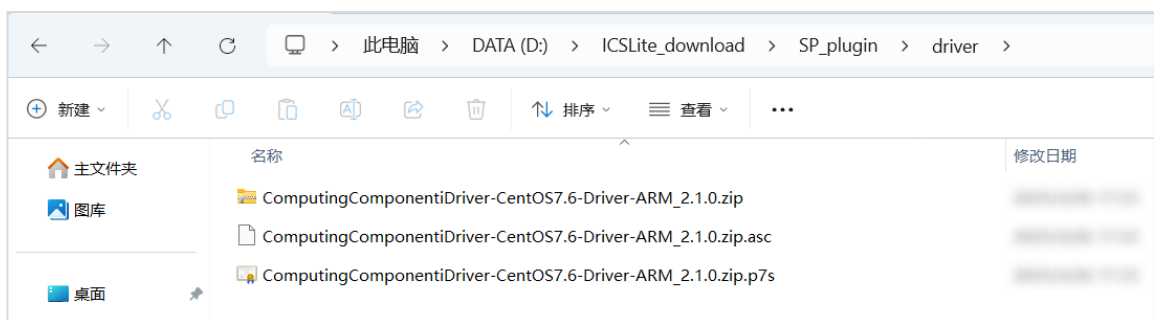


图2-101 准备驱动包

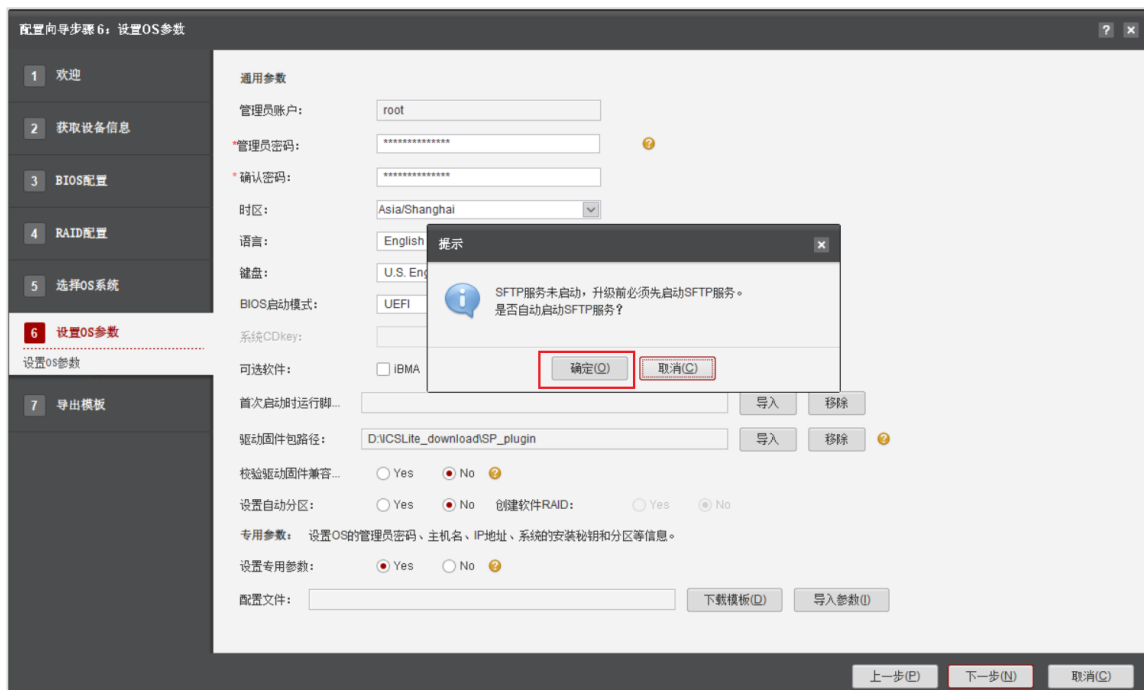


图2-102 启动 SFTP

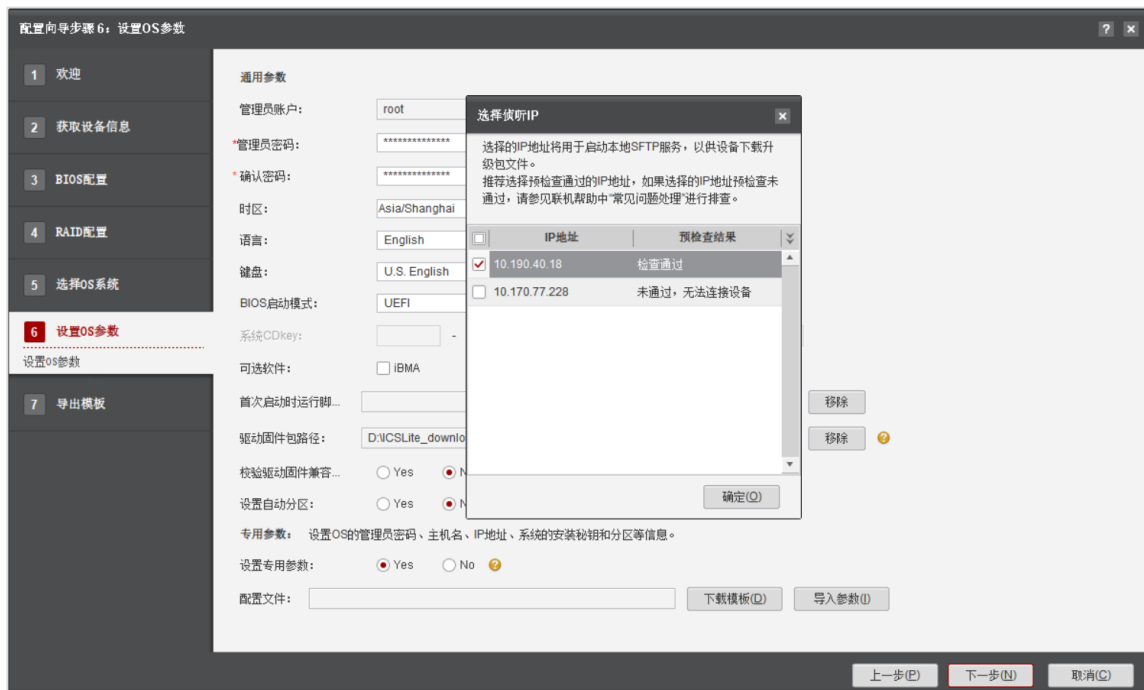


图2-103 选择可用网卡

- 校验驱动固件包：选择 “no”
- 设置自动分区：选择 “no”
- 设置专用参数：选择 “Yes”

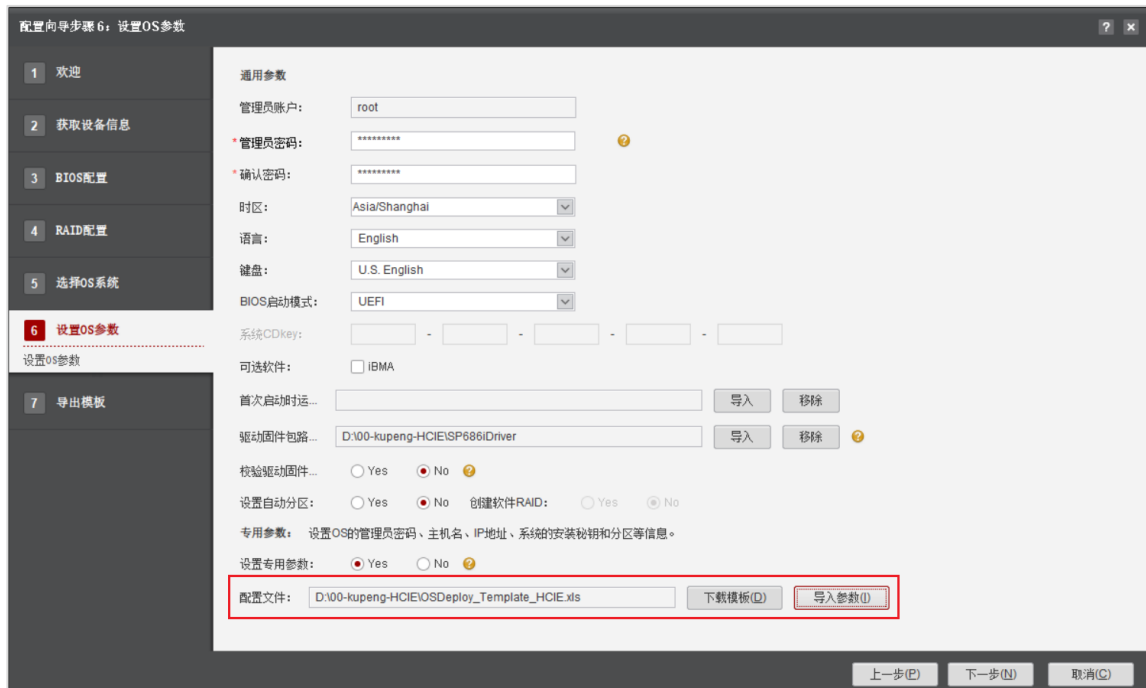


图2-104 上传专用参数

在设置专用参数选项下，选择“**Yes**”，并下载模板。使用本地 Excel 打开模板后，填写如下信息，这些信息可从[网络信息规划](#)及[账号密码规划](#)小节获取（参考附件

“OSDeploy_Template_Computing_HCIE.xls”）：

- DeviceIP：服务器的 iBMC 地址，下载模板后会自动填充
- HostName：OS 的主机名
- Partition_Name：系统分区
- Partition_FileSystem：系统分区的文件系统
- Partition_Size：系统各个分区的大小
- Device_MAC：安装 OS 系统的过程中配置的 MAC 地址，可通过 iBMC 登录 Web 页面，获取具体网卡的 MAC 地址
- VLAN_ID：规划的 VLAN
- IPv4_AddressOrigin：IP 地址配置方式，建议为 Static
- IPv4_Address：规划的 IP 地址
- IPv4_SubnetMask：掩码
- IPv4_Gateway：网关地址
- IPv4_DNS：DNS 服务器地址，可填写 8.8.8.8

单击“下一步”后，根据提示关闭设置参数弹窗，回到一键开局页面，单击“执行部署”，开始部署 OS。（此过程会持续约 1 小时）

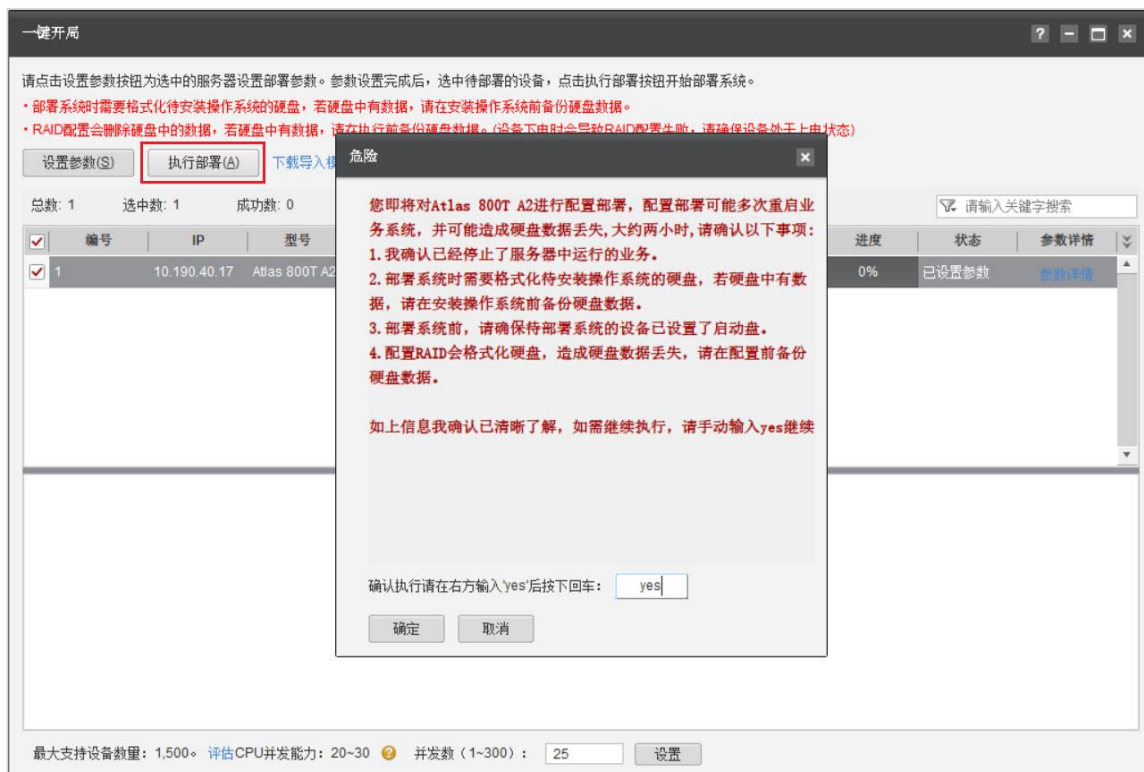


图2-105 确认执行部署

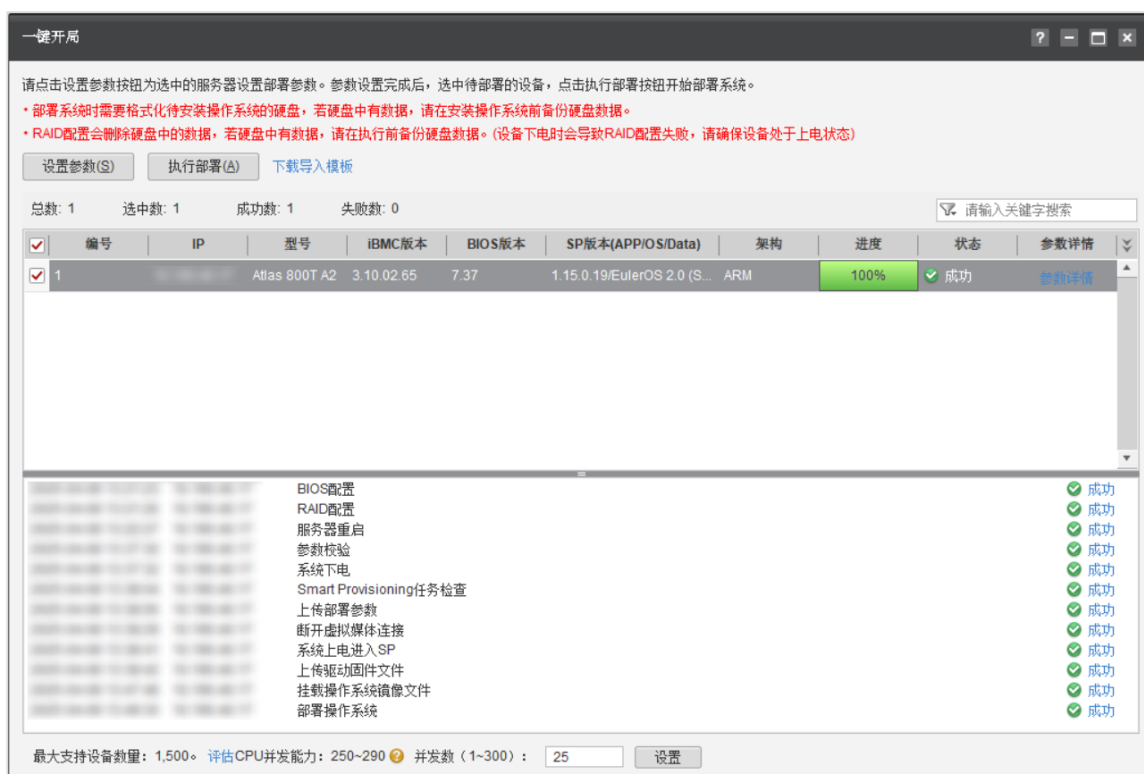


图2-106 部署中

部署完成后，SmartKit 显示如下：

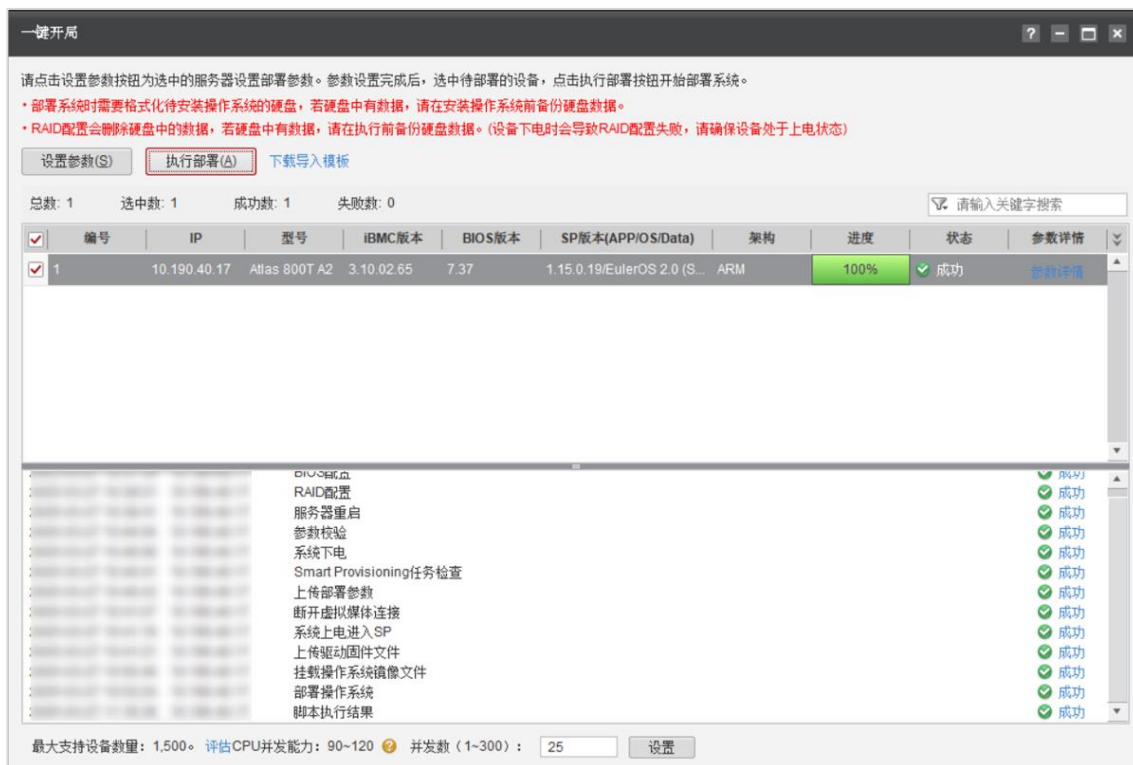


图2-107 部署完成

步骤 3 登录验证

OS 部署成功后，通过 SSH 终端工具登录到服务器，可正常登录表示 OS 部署成功。

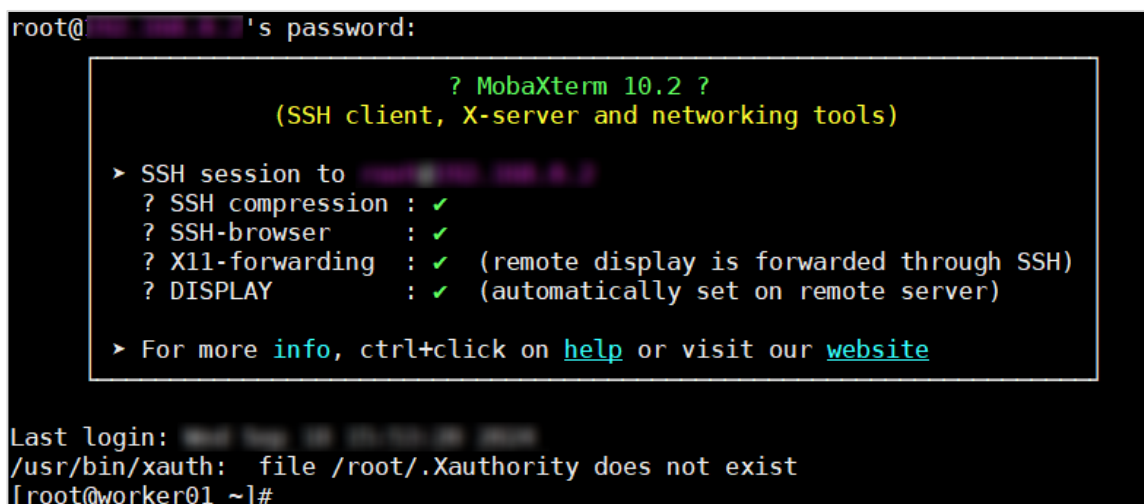


图2-108 OS 登录验证

3 昇腾软件部署

本章节指导学员完成昇腾软件包制作及部署到服务器。

在安装昇腾软件前，需要在训练服务器安装相关的系统依赖，可以根据训练服务器是否能够连接外部网络选择如下安装依赖方式。

- 离线安装：服务器未连通外部网络的，需要在 PC 下载 Ascend Deployer 工具，并通过 Ascend Deployer 工具下载安装依赖和 MindCluster 集群调度软件。
- 在线安装：在能连通外部网络的通用服务器或者训练服务器上安装 Ascend Deployer 工具，并通过 Ascend Deployer 工具在线拉取安装依赖和 MindCluster 集群调度软件。

本实验以离线安装方式为例介绍软件包的制作以及安装。离线安装指待安装昇腾软件的服务器没有连接互联网，在一台能连接互联网的 PC 上制作好离线软件包，上传到服务器完成软件安装。

3.1 昇腾软件包制作

3.1.1 Python 环境安装

根据[实验设备及软件介绍](#)小节获取名为“python-3.7.5-amd64.exe”的安装包，按照以下步骤完成安装。

步骤 1 打开安装包

双击 python-3.7.5-amd64.exe 安装包，进入安装界面。勾选“Add Python3.7 to PATH”（添加 Python 环境变量），单击“Customize installation”（自定义安装）。

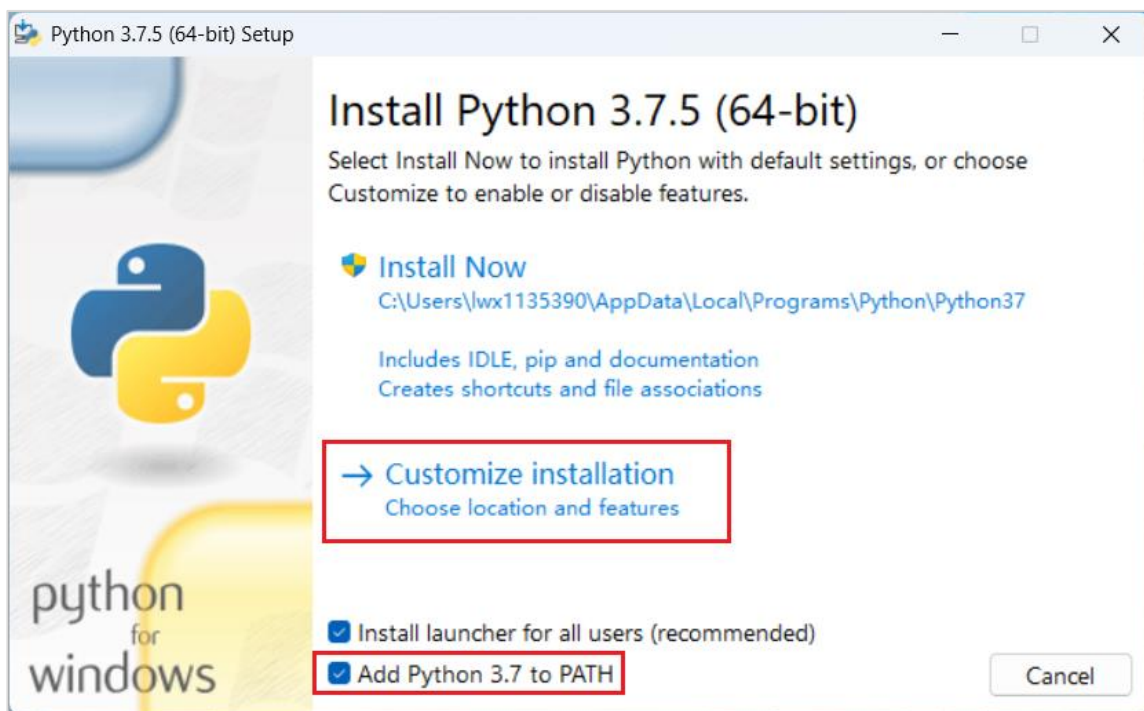


图3-1 配置安装环境变量

步骤 2 自定义安装配置

保持默认勾选，包括文档、pip、开发环境及测试套件，单击“Next”。

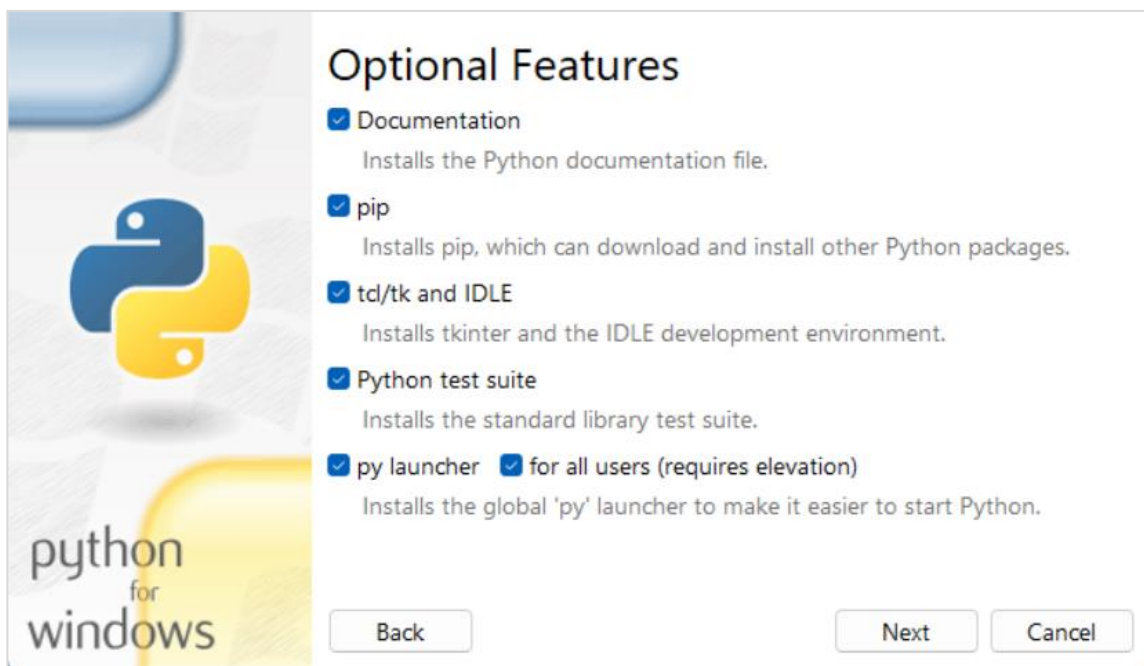


图3-2 安装选项选择

步骤 3 添加环境变量

勾选“Add Python to environment variables”，并自定义安装路径后，单击“Install”。

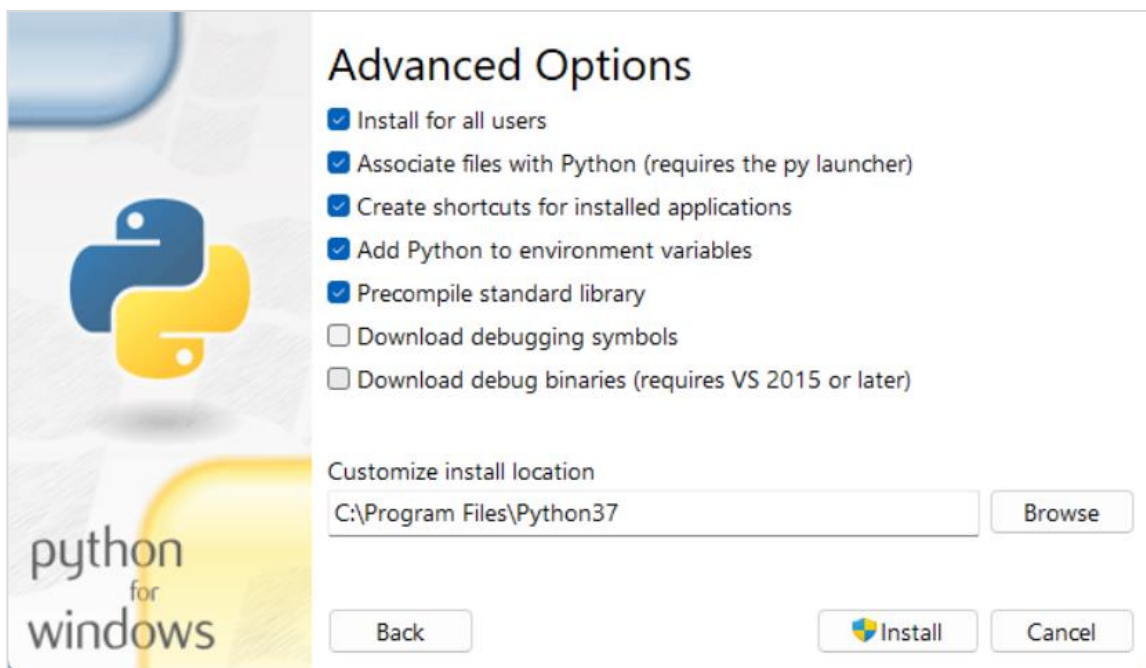


图3-3 添加变量及安装路径选择

步骤 4 完成安装

安装成功后，单击“Disable path length limit”，禁用系统的 Path 长度限制后，再单击“Close”关闭安装界面。

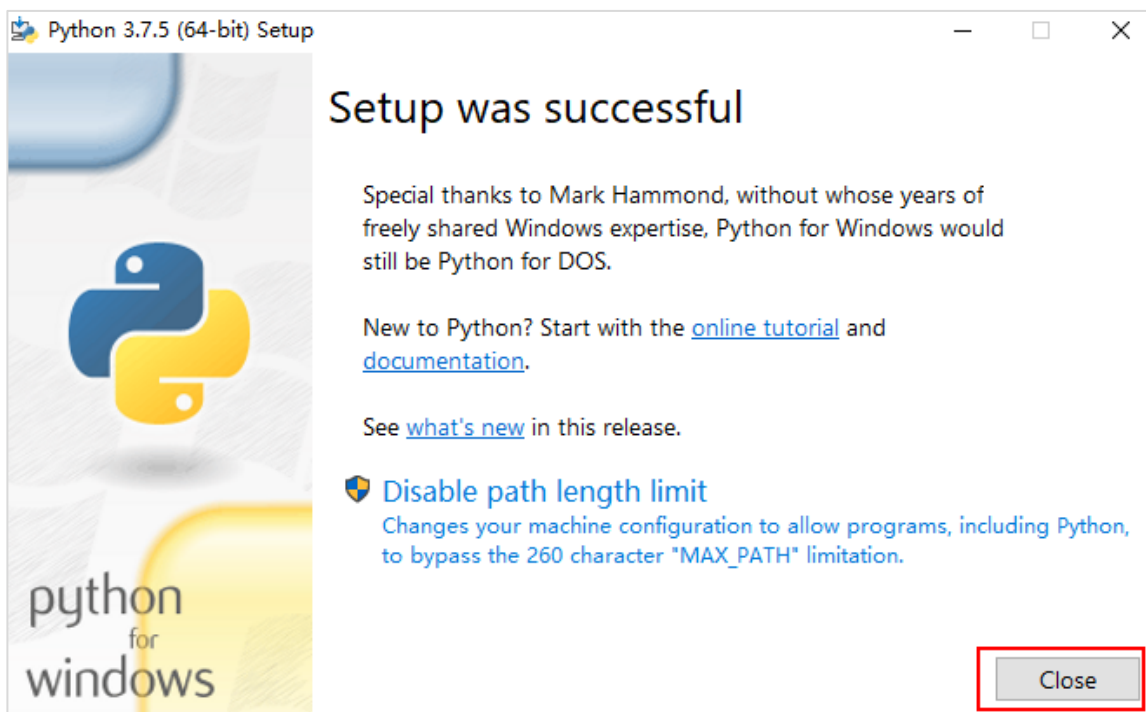


图3-4 安装完成

步骤 5 安装检查

输入“Win+R”组合键打开运行窗口，在运行窗口中输入“cmd”打开 Windows 命令提示符窗口，在命令提示符窗口下输入如下命令：

```
python -V
```

正常输入 Python 的版本号，说明安装成功。

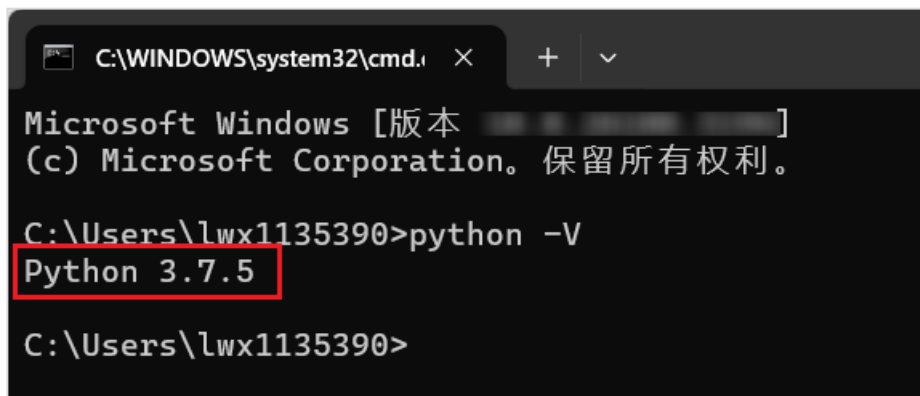


图3-5 安装检查

3.1.2 昇腾软件包制作

通过 Ascend Deployer 工具完成昇腾软件包制作。

步骤 1 Ascend Deployer 工具准备

根据[实验设备及软件介绍](#)小节获取名为“ascend-deployer-dev.zip”的软件包，并完成解压，得到“ascend-deployer-dev”文件夹。

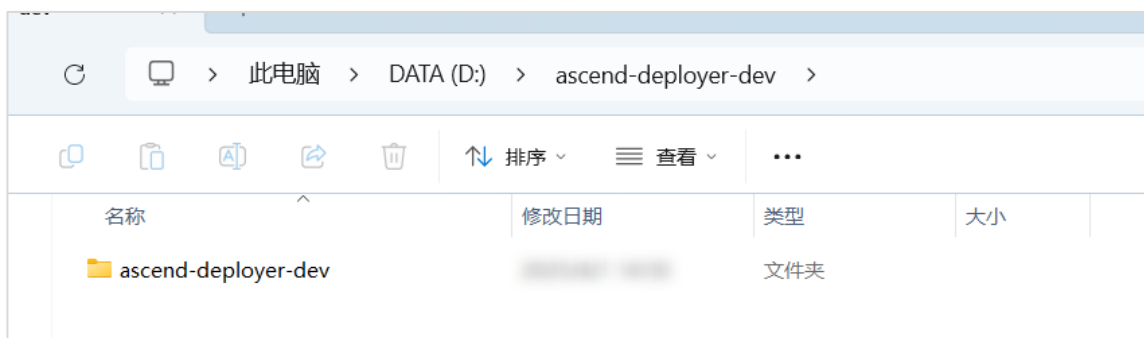


图3-6 Ascend 工具准备

步骤 2 运行 Ascend Deployer 工具

双击运行“ascend-deployer-dev\ascend-deployer-dev\ascend_deployer”目录下的“start_download_ui.bat”。在弹出的简易 UI 界面，勾选待安装设备的 OS（本实验以 CentOS 7.6_aarch64 为例）下载 Docker 软件和系统依赖，并勾选界面右侧的 DL 软件（本实验配套 6.0.0），单击“开始下载”。

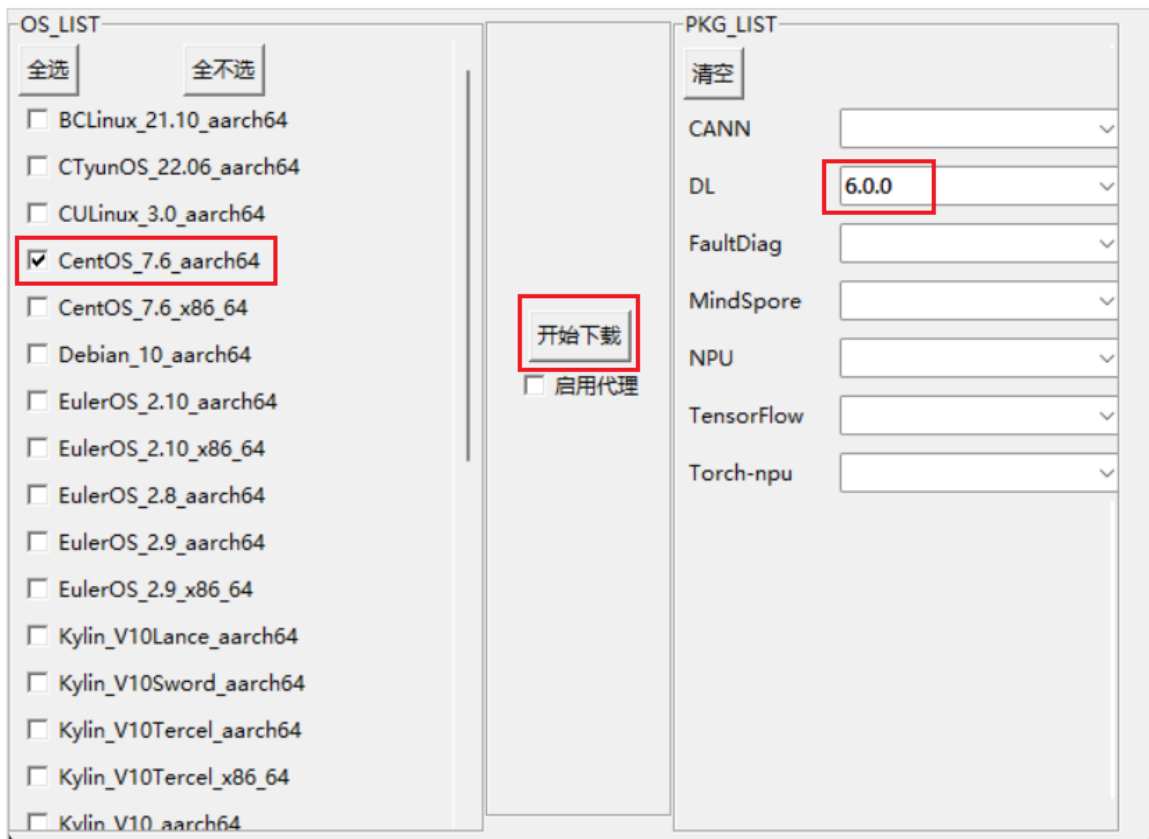


图3-7 下载依赖包

出现类似如下回显，表示下载完成。

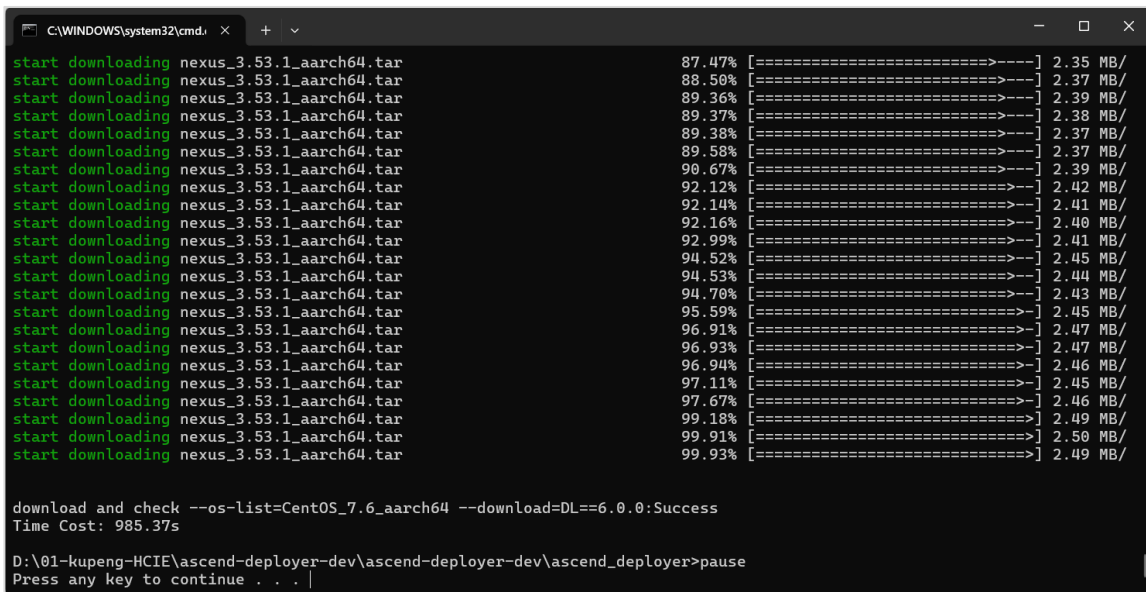


图3-8 依赖包下载完成

步骤 3 准备其他软件

将[实验设备及软件介绍](#)小节获取的 kernels、toolkit、NPU 驱动和固件、Toolbox 软件包放置于“ascend-deployer-dev\ascend_deployer\resources”目录。resources 文件夹内容如下（以 CentOS_7.6 为例）：

```
resources
├─ mindxdl
├─ nexus
├─ CentOS_7.6_aarch64
├─ pylibs
├─ sources
├─ Ascend-cann-kernels-910b_8.0.0_linux-aarch64.run
├─ Ascend-cann-toolkit_8.0.0_linux-aarch64.run
├─ Ascend-docker-runtime_6.0.0_linux-aarch64.run
├─ Ascend-hdk-910b-npu-driver_23.0.7_linux-aarch64.run
├─ Ascend-hdk-910b-npu-firmware_7.1.0.11.220.run
├─ Ascend-mindx-toolbox_6.0.0_linux-aarch64.run
└─ version.json
```

因部署软件时会校验安装包，在文件夹下不要添加其它的软件安装包。

步骤 4 形成软件包

将“ascend-deployer-dev”整个目录压缩为“.zip”文件。该文件用于上传到服务器完成昇腾软件的部署。



图3-9 形成软件包

3.2 昇腾软件包部署

步骤 1 上传软件包

将制作好的离线软件包“ascend-deployer-dev.zip”上传到待部署服务器的“/home”目录，并将软件包解压。

```
unzip ascend-deployer-hcie.zip
ll
```

```
[root@worker07 home]# ll
total 5253172
drwxr-xr-x. 9 root root      4096  2023-08-14 14:13 ascend-deployer-dev
-rw-r--r--. 1 root root 5379237815  2023-08-14 14:13 ascend-deployer-dev.zip
drwxr-xr-x. 2 root root      4096  2023-08-14 14:13
[root@worker07 home]#
```

图3-10 上传软件包并解压

步骤 2 配置安装文件

Ascend-deployer 工具使用 Ansible 工具来完成批量部署操作，需要批量操作的服务器都放在一个 inventory_file 文件里，在正式部署前，需要根据待部署的服务器信息，配置 inventory_file 文件。

进入配置路径，打开 inventory_file 文件：

```
cd /home/ascend-deployer-hcie/ascend_deployer/
vim inventory_file
```

参考[网络信息规划](#)规划小节配置网络信息，文件内容示例如下：

```
[hccn]
10.190.41.17 ansible_ssh_user='root' ansible_ssh_pass='aiTraining@123'
deviceip=192.168.27.10,192.168.27.11,192.168.27.12,192.168.27.13,192.168.27.14,192.168.27.15,192.168.27.16,192.168.27.17
detectip=192.168.27.1,192.168.27.1,192.168.27.1,192.168.27.1,192.168.27.1,192.168.27.1,192.168.27.1,192.168.27.1
[hccn:vars]
gateways="192.168.27.1"
netmask="255.255.255.0"
roce_port=4791
bitmap=""
dscp_tc=""
common_network="0.0.0.0/0"

[master]
#10.10.10.10 ansible_ssh_user="root" ansible_ssh_pass="test1234" k8s_api_server_ip=10.10.10.10

[worker]
10.190.41.17 ansible_ssh_user='root' ansible_ssh_pass='aiTraining@123'

[other_build_image]

[all:vars]
POD_NETWORK_CIDR="192.168.0.0/16"
KUBE_SERVICE_CIDR="10.96.0.0/12"
KUBE_VIP=""
HARBOR_SERVER=""
HARBOR_ADMIN_USER=""
HARBOR_ADMIN_PASSWORD=""
```



```
HARBOR_PUBLIC_PROJECT="false"
HARBOR_CA_FILE=""
```

文件说明：

- [hccn]字段表示为 NPU 配置的 IP 地址信息，每台服务器 1 行（注意 deviceip 和 detectip 都没有换行）
- [hccn:vars]字段表示 NPU 网络的通用信息
- [master]表示 K8S 的管理节点信息
- [worker]表示待部署的训练服务器，每台服务器 1 行
- 其他配置默认即可。

步骤 3 执行批量部署

inventory_file 文件准备完成后，执行部署命令，这个命令会依次在待部署服务器后安装部署 --install 后面列出的软件。

```
bash install.sh --install=sys_pkg,python,npu,toolkit,kernels,toolbox
```

根据提示输入“y”。

```
policy), and is the entire agreement between you and Huawei regarding your use of the
representations, communications, understandings and agreements between You and us,
ject to additional terms and conditions that may apply when you use or purchase open
9.6 Changes and Modifications. Huawei may modify this Agreement, including the docum
posting a revised version on Huawei's official Website or by otherwise notifying yo
cies, the modified terms will come into effect upon posting or notification. You wil
s use of the Huawei Software after the effectiveness of such modification will be de
Release date: November,9th 2023

Do you accept the EULA to use Ascend-deployer?[y/N]y
openssl11-devel-1.1.1w-1.el7 #####
container-selinux-2:2.107-3.el7 #####
```

图3-11 昇腾软件安装

安装过程大约需要 2 小时，安装完成出现类似回显表示安装成功：

```
TASK [output to csv] *****
changed: [localhost]

TASK [print the error] *****
skipping: [localhost]

PLAY RECAP *****
10.190.41.11 : ok=240 changed=91 unreachable=0 failed=0 skipped=229 rescued=0 ignored=2
localhost : ok=61 changed=25 unreachable=0 failed=0 skipped=36 rescued=0 ignored=1

run cmd: --install=sys_pkg,python,npu,toolkit,kernels,toolbox successfully
[root@worker01 ascend_deployer]#
```

图3-12 昇腾软件安装成功

步骤 4 安装 ascend-docker-runtime

进入如下路径执行手动安装昇腾容器运行时：

```
cd /home/ascend-deployer-dev/ascend_deployer/resourcesd
chmod +x Ascend-docker-runtime_6.0.0_linux-aarch64.run
./Ascend-docker-runtime_6.0.0_linux-aarch64.run --install
```

```
[root@worker07 resources]# pwd
/home/ascend-deployer-dev/ascend_deployer/resources
[root@worker07 resources]# chmod +x Ascend-docker-runtime_6.0.0_linux-aarch64.run
[root@worker07 resources]# ./Ascend-docker-runtime_6.0.0_linux-aarch64.run --install
Uncompressing ascend-docker-runtime 100%
[INFO] installing ascend docker runtime
[INFO] platform(aarch64) matched!
[INFO] install executable files success
[INFO] install scene is 'docker'.
[INFO] /etc/docker/daemon.json modify success
[INFO] Ascend Docker Runtime has been installed in: /usr/local/Ascend/Ascend-Docker-Runtime
[INFO] The version of Ascend Docker Runtime is: 6.0.0
[INFO] please reboot daemon and container engine to take effect
[INFO] Ascend Docker Runtime install success
[root@worker07 resources]#
```

图3-13 安装容器运行时

步骤 5 重启服务器

配置 ansible 环境变量。

```
export PATH=$PATH:/root/.local/python3.7.5/bin
export LD_LIBRARY_PATH=/root/.local/python3.7.5/lib:$LD_LIBRARY_PATH
```

批量重启服务器。

```
cd /home/ascend-deployer-dev/ascend_deployer
ansible -i inventory_file worker -m shell -a 'reboot'
```

```
[root@worker07 ascend_deployer]# pwd
/home/ascend-deployer-dev/ascend_deployer
[root@worker07 ascend_deployer]# ls
ansible.cfg          downloader.log        install_operation.log  large_scale_deploy
ansible_plugin       downloader_operation.log  install.sh             large_scale_deploy.py
ascend_deployer.py   downloader_ui.py       inventory_file          large_scale_inventory.in
ascend_deployer.pyc  group_vars            inventory_template.xls  library
ascend_download.py   __init__.py           jobs.py                 module_utils
downloader           install.log            jobs.pyc                parsed_inventory_file
[root@worker07 ascend_deployer]# ansible -i inventory_file worker -m shell -a 'reboot'
```

图3-14 执行批量重启

步骤 6 配置环境变量

配置 Toolkit 环境变量

```
source /usr/local/Ascend/ascend-toolkit/set_env.sh
```

配置 Ascend Deployer 工具安装的 3.7.5 版本 Python 环境变量

```
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
export PATH=/usr/local/python3.7.5/bin:$PATH
```

查询 python3 环境变量，确认安装结果。

```
python3 -V
```

配置 gcc 环境变量

```
export LD_LIBRARY_PATH=/usr/local/gcc7.3.0/lib64:${LD_LIBRARY_PATH}
export CC=/usr/local/gcc7.3.0/bin/gcc
export CXX=/usr/local/gcc7.3.0/bin/g++
```



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

最后检查 gcc 版本确认安装成功。

```
gcc --version
```

```
[root@worker07 ~]# source /usr/local/Ascend/ascend-toolkit/set_env.sh
[root@worker07 ~]#
[root@worker07 ~]# export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:${LD_LIBRARY_PATH}
[root@worker07 ~]# export PATH=/usr/local/python3.7.5/bin:${PATH}
[root@worker07 ~]# python3 -V
Python 3.7.5
[root@worker07 ~]# export LD_LIBRARY_PATH=/usr/local/gcc7.3.0/lib64:${LD_LIBRARY_PATH}
[root@worker07 ~]# export CC=/usr/local/gcc7.3.0/bin/gcc
[root@worker07 ~]# export CXX=/usr/local/gcc7.3.0/bin/g++
[root@worker07 ~]# export PATH=/usr/local/gcc7.3.0/bin:${PATH}
[root@worker07 ~]# gcc --version
gcc (GCC) 7.3.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[root@worker07 ~]#
```

图3-15 环境变量配置与检查

设置开机自启动，并重启 docker。

```
systemctl enable docker
```

```
systemctl restart docker
```

```
systemctl status docker
```

```
[root@worker07 ~]# systemctl enable docker
[root@worker07 ~]# systemctl restart docker
[root@worker07 ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2023-08-01 11:17:17; 10ms ago
     Docs: https://docs.docker.com
    Main PID: 24017 (dockerd)
      Tasks: 47
     Memory: 129.8M
    CGroup: /system.slice/docker.service
            └─24017 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

worker07 dockerd[24017]: time="2023-08-01T11:17:17.945578480+08:00" level=info msg="Firewall
worker07 dockerd[24017]: time="2023-08-01T11:17:18.170433824+08:00" level=info msg="Firewall
worker07 dockerd[24017]: time="2023-08-01T11:17:18.197180074+08:00" level=info msg="Firewall
worker07 dockerd[24017]: time="2023-08-01T11:17:18.404337422+08:00" level=info msg="Default
worker07 dockerd[24017]: time="2023-08-01T11:17:18.509405835+08:00" level=info msg="Firewall
worker07 dockerd[24017]: time="2023-08-01T11:17:18.614052298+08:00" level=info msg="Loading
worker07 dockerd[24017]: time="2023-08-01T11:17:18.692267576+08:00" level=info msg="Docker c
worker07 dockerd[24017]: time="2023-08-01T11:17:18.692340193+08:00" level=info msg="Daemon h
worker07 systemd[1]: Started Docker Application Container Engine.
worker07 dockerd[24017]: time="2023-08-01T11:17:17.71752721+08:00" level=info msg="API list
Hint: Some lines were ellipsized, use -l to show in full.
[root@worker07 ~]#
```

图3-16 Docker 服务重启

3.3 NPU 网络配置

步骤 1 配置网络

为实现后续分布式训练时各训练服务器上模型的参数同步，需要为每台训练服务器的 NPU 网口配置参数面网络。

```
cd /home/ascend-deployer-dev/ascend_deployer/
python ascend_deployer.py --hccn
```

```
[root@worker07 ascend_deployer]# python ascend_deployer.py --hccn
PLAY [localhost] *****
TASK [check user] *****
changed: [localhost]
PLAY [hccn] *****
TASK [check hccn config] *****
changed: [10.190.41.17]
TASK [check npu] *****
changed: [10.190.41.17]
TASK [handle npu check error] *****
skipping: [10.190.41.17]
PLAY RECAP *****
10.190.41.17 : ok=2 changed=2 unreachable=0 failed=0 skipped=1 rescued=0 ignored=0
localhost : ok=1 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

图3-17 配置 NPU 地址

等待几分钟，回显如下信息，表示配置成功。

```
PLAY RECAP *****
10.190.41.17 : ok=2 changed=2 unreachable=0 failed=0 skipped=1
rescued=0 ignored=0
localhost : ok=1 changed=1 unreachable=0 failed=0 skipped=0
rescued=0 ignored=0

PLAY [hccn conf] *****
TASK [Gathering Facts] *****
ok: [10.190.41.17]

TASK [include_tasks] *****
included: /home/ascend-deployer-6.0.0/ascend_deployer/playbooks/roles/mindx.hccn/tasks/hccn_conf.yaml for 10.190.41.17

TASK [hccn_tool configures IP address] *****
changed: [10.190.41.17]

PLAY RECAP *****
10.190.41.17 : ok=3 changed=1 unreachable=0 failed=0 skipped=0
rescued=0 ignored=0

run cmd: --hccn successfully
[root@worker07 ascend_deployer]#
```

图3-18 NPU 地址配置完成

步骤 2 网络检查

执行命令对所有 NPU 网络信息进行检查，并对照[网络信息规划](#)小节查看是否配置正确。

```
for i in {0..7}; do hccn_tool -i $i -ip -g; done
```

```
[root@worker07 ascend_deployer]# for i in {0..7}; do hccn_tool -i $i -ip -g; done
ipaddr:192.168.27.10
netmask:255.255.255.0
ipaddr:192.168.27.11
netmask:255.255.255.0
ipaddr:192.168.27.12
netmask:255.255.255.0
ipaddr:192.168.27.13
netmask:255.255.255.0
ipaddr:192.168.27.14
netmask:255.255.255.0
ipaddr:192.168.27.15
netmask:255.255.255.0
ipaddr:192.168.27.16
netmask:255.255.255.0
ipaddr:192.168.27.17
netmask:255.255.255.0
[root@worker07 ascend_deployer]#
```

图3-19 NPU 信息检查

3.4 单机综合检测

环境部署完成后，在进行正式的训练或推理前，需要对服务器所部署的环境进行单机综合检测，主要对相关软件版本及 NPU 状态进行检查。

步骤 1 获取检测脚本

根据[实验设备及软件介绍](#)小节获取单机检测的脚本“ser_check.sh”和工作流文件“Workflow_check.json”。

Data (D:) > 00-kupeng-HCIE > scripts > danji_check			
名称	修改日期	类型	大小
 server_check.sh		SH 源文件	8 KB
 Workflow_check.json		JSON File	3 KB

图3-20 准备检查脚本

步骤 2 添加服务器到 SmartKit

在 SmartKit 首页选择“添加设备”，并选择“OS”后，单击“增加”，在弹窗中输入待添加设备的 IP 地址及登录信息后，单击“确定”。

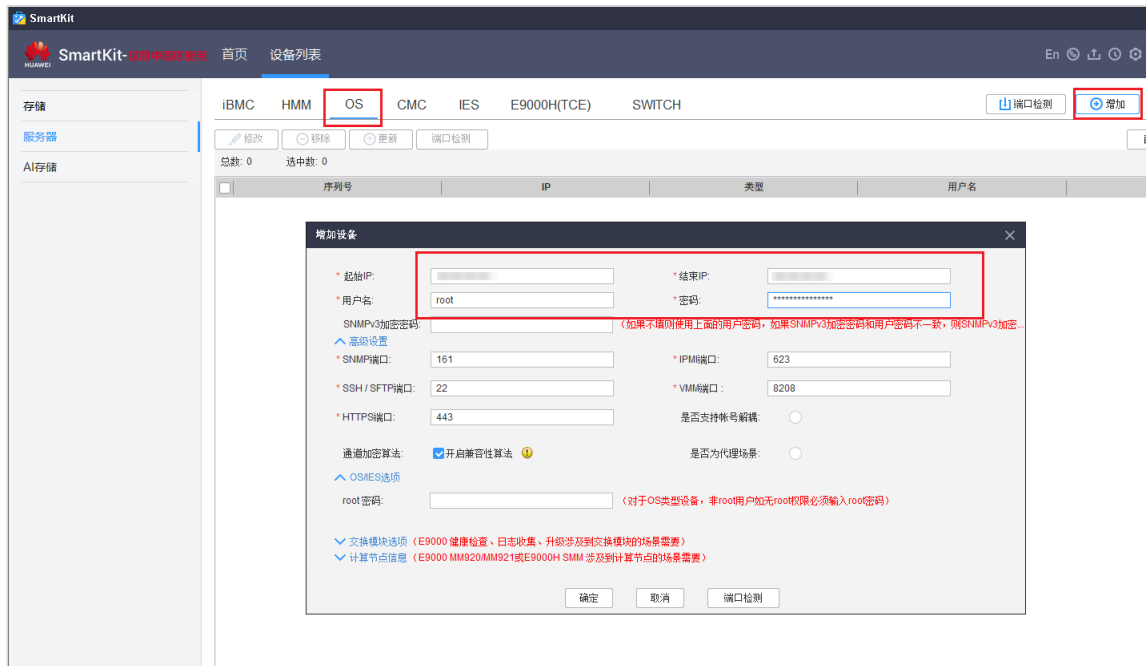


图3-21 以 OS 方式添加设备

步骤 3 执行检测

在 SmartKit 首页服务器页签下，单击“批量分发”。（此处需要根据 SmartKit 产品文档进行认证后操作）

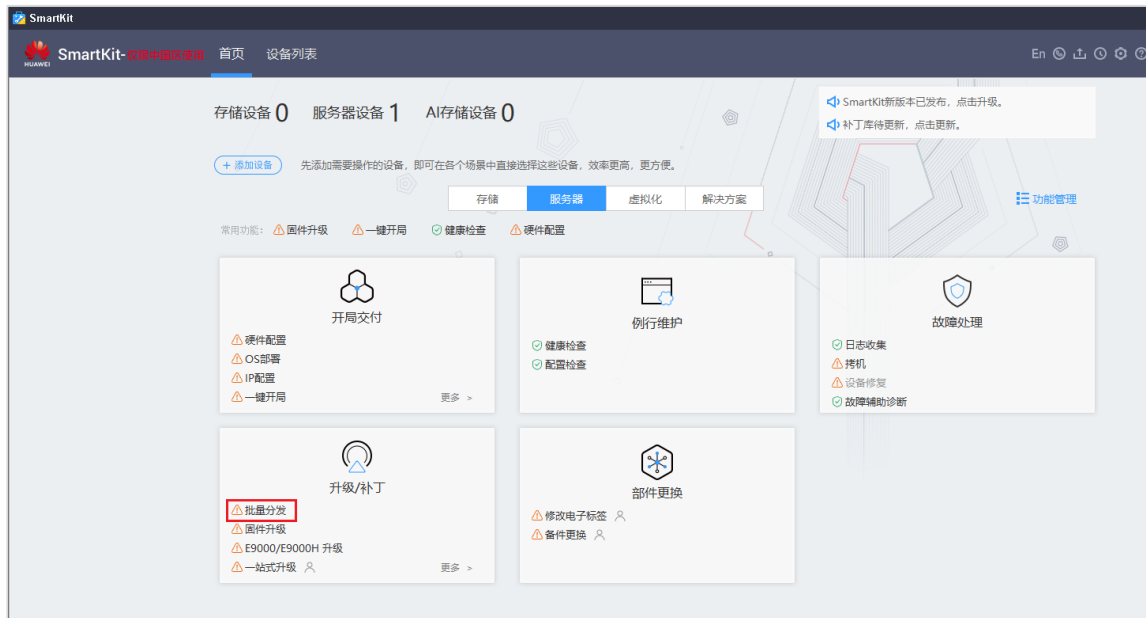


图3-22 进入批量分发

步骤 4 选择设备

单击“选择设备”，在设备列表中选择“OS”页签，然后勾选测试的设备，单击“确定”，返回至“批量分发”页面。

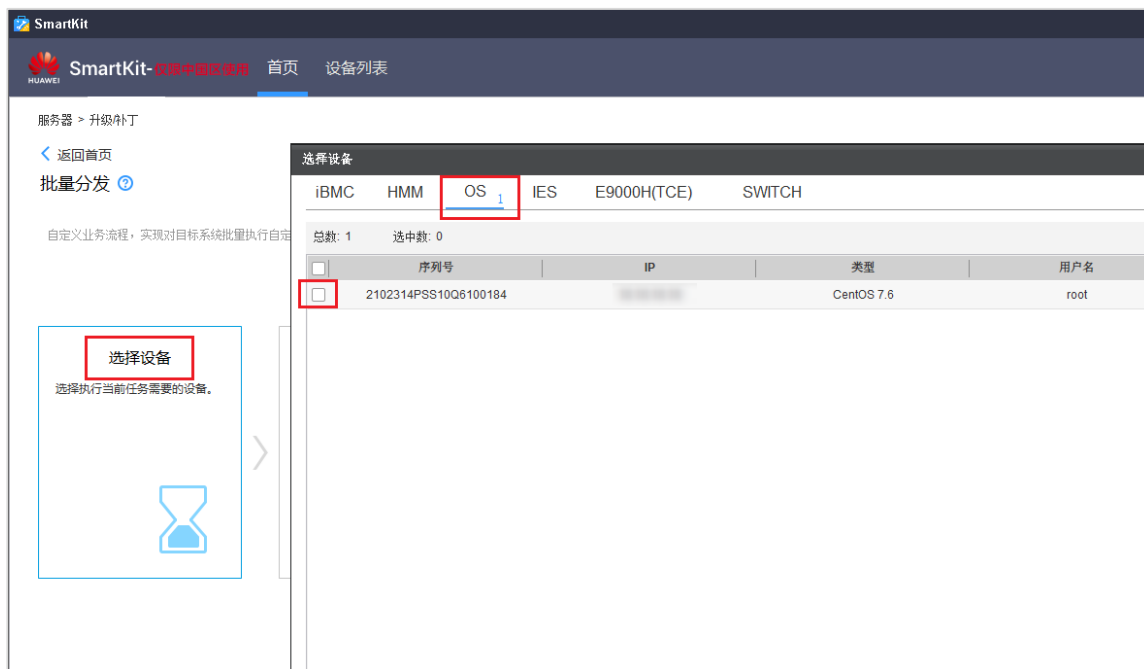


图3-23 选择批量分发设备

步骤 5 业务流配置

在“批量分发”页面，单击“批量分发”，进入批量分发主界面。勾选设备，单击“配置业务流”，进入业务流配置向导界面。



图3-24 配置业务流

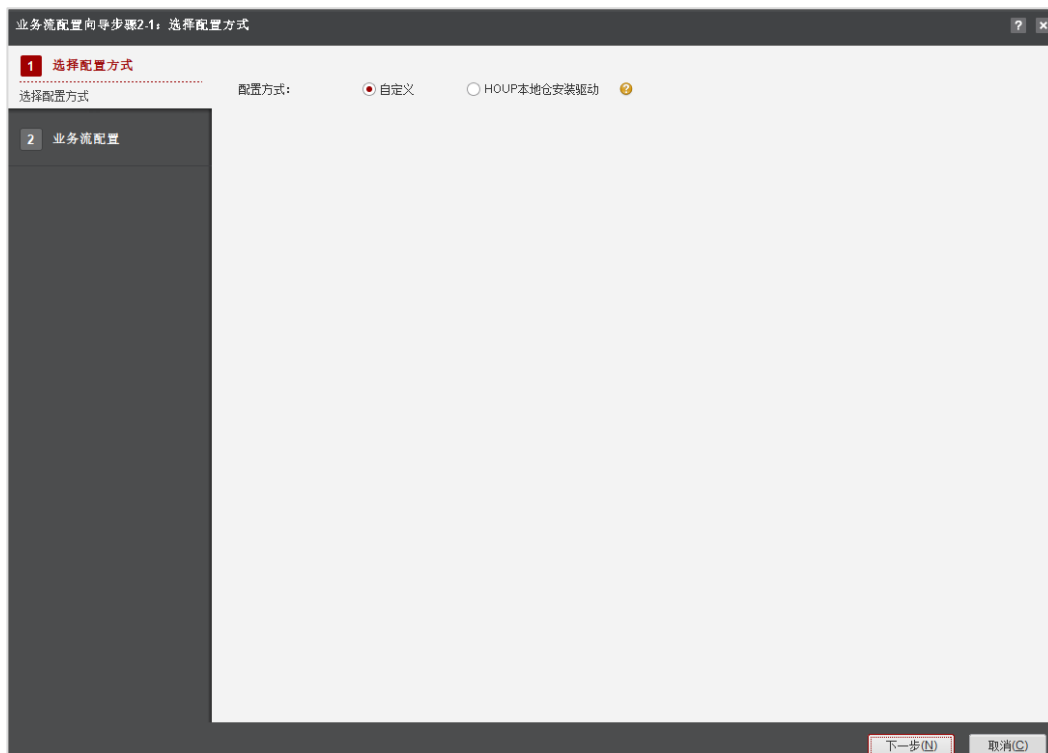


图3-25 选择业务流配置方式

步骤 6 配置业务流

进入“业务流配置”界面。

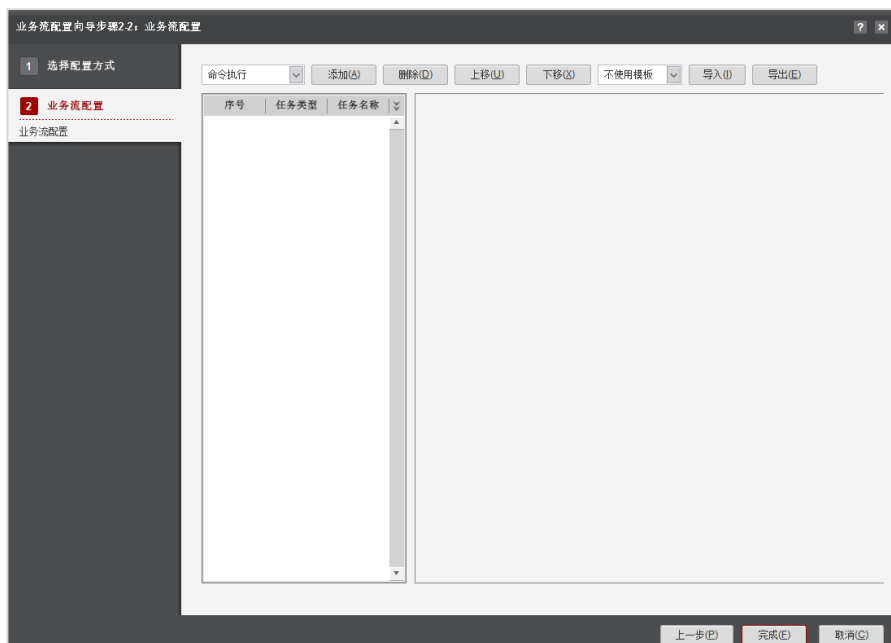


图3-26 业务流配置主界面

步骤 7 导入 workflow 文件

单击“导入”，选择本地的单机检测业务流文件。

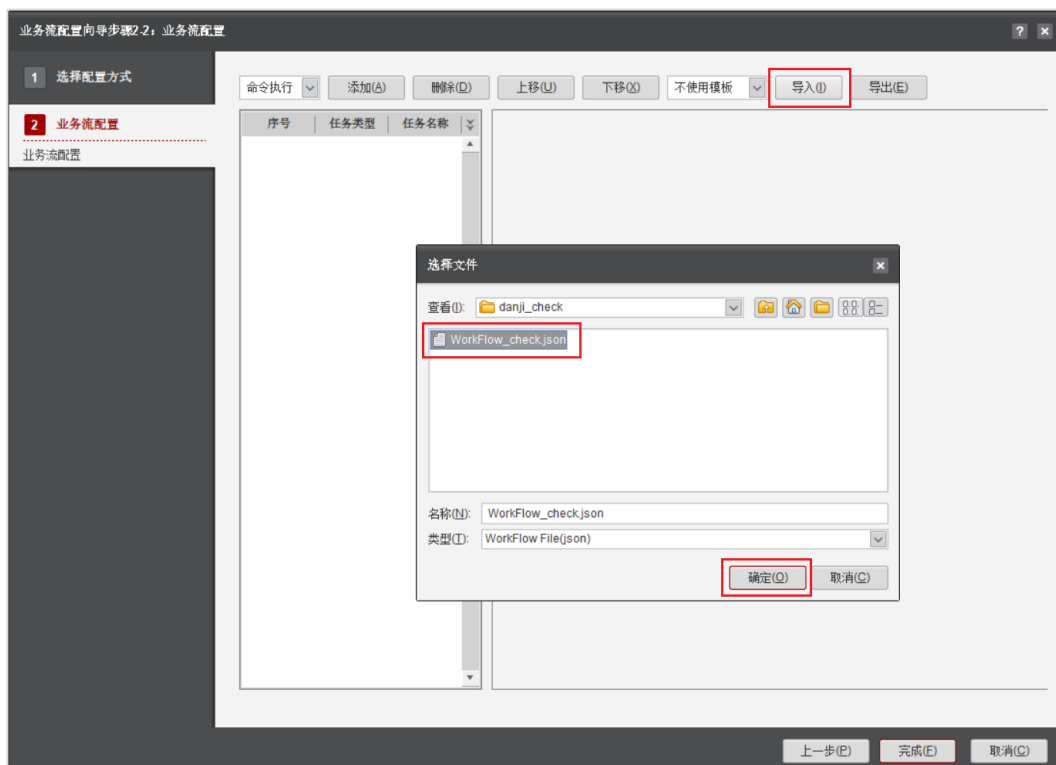


图3-27 导入 workflow 文件

步骤 8 修改任务

根据实际情况修改任务配置，特别是单机综合检测的脚本路径，需要修改为操作机的本地路径。其他任务可根据自己的实际情况进行修改，每完成一个任务的修改，都需要单击“保存”。



图3-28 修改工作流导入检测脚本

步骤 9 执行业务流

业务流配置完成后，单击“完成”按钮，回到“批量分发”页面。然后选择设备，单击“执行业务流”，开始执行单机综合检测。



图3-29 执行业务流

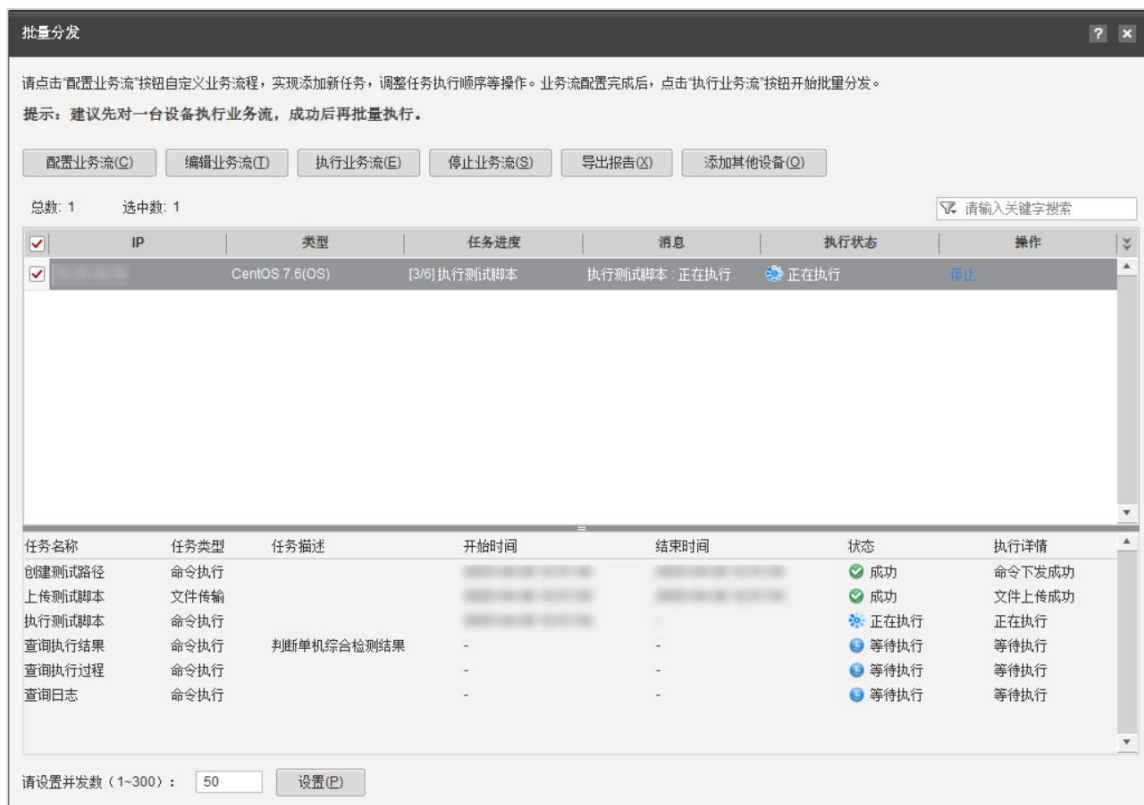


图3-30 业务流执行中

步骤 10 导出报告

执行完成后，设备列表的“执行状态”列会展示单机综合检测结果。单击“导出报告”，将执行结果导出到操作机。

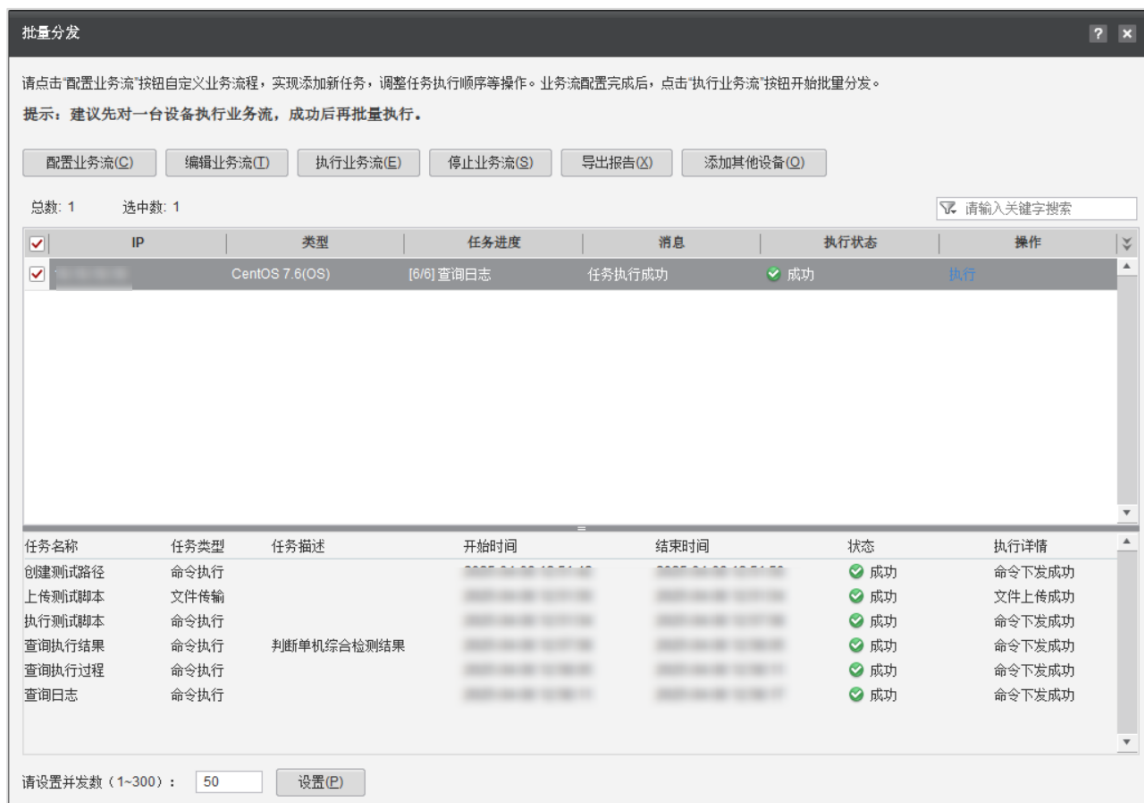


图3-31 业务流执行成功

步骤 11 报告查看

报告导出后得到如下文件，双击打开文件即可查看报告。

名称	修改日期	类型	大小
data		文件夹	
allResult.xls		Microsoft Excel ...	4 KB
index.html		Microsoft Edge ...	1 KB

图3-32 检查报告导出

统计信息					
设备总数	成功	失败	中止	正在执行	未执行
1	1	0	0	0	0

概要信息				
IP地址	设备类型	业务流状态	消息	分发任务数
192.168.1.100	CentOS 7.6	成功	任务执行成功	6

详细信息					
业务流详情					
序号	任务名称	任务描述	状态	消息	详情
1	创建测试路径		成功	命令下发成功	详情
2	上传测试脚本		成功	文件上传成功	详情
3	执行测试脚本		成功	命令下发成功	详情
4	查询执行结果	判断单机综合检测结果	成功	命令下发成功	详情
5	查询执行过程		成功	命令下发成功	详情
6	查询日志		成功	命令下发成功	详情

Copyright © Huawei Technologies Co., Ltd.

图3-33 检查报告查看

4 大模型体验

4.1 ChatGLM3-6B 模型推理体验

ChatGLM3-6B 是一款支持与用户对话的开源大模型，本实验通过验证该开源大模型在使用昇腾 NPU 的情况下，实现模型与用户之间的多轮对话。

步骤 1 新建开发目录并准备实验数据

登录到服务器，创建实验数据目录，比如“/home/data”，从[实验设备及软件介绍](#)获取名为“chatglm.tar”的模型镜像及名为“ChatGLM-6B.tar.gz”的模型权重文件，将这两个文件上传到服务器“/home/data”目录，并将“ChatGLM-6B.tar.gz”文件解压，获得“ChatGLM-6B”目录。

```
pwd  
ls -l
```

```
[root@worker07 data]# ll  
total 76184016  
drwxr-xr-x. 10 root root      4096  
-rw-r--r--.  1 root root 13433279115  
-rw-----.  1 root root 20313025536  
-rw-r--r--.  1 root root 11204129238  
-rw-r--r--.  1 root root      145  
-rw-----.  1 root root          0  
drwxr-xr-x.  3 root root      4096  
-rw-r--r--.  1 root root 33061960714  
[root@worker07 data]#
```

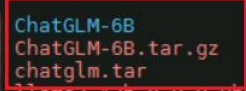


图4-1 准备实验数据

步骤 2 加载容器镜像

执行如下命令加载容器镜像。（由于镜像较大，分层较多，如果发生加载失败的场景，可重复执行加载）

```
docker load -i chatglm.tar
```

```
[root@worker07 data]# docker load -i chatglm.tar
85eb0023ed95: Loading layer 311.1MB/311.1MB
5f70bf18a086: Loading layer [=====>] 1.024kB/1.024kB
dc3e0f73d8af: Loading layer 11.84MB/11.84MB
661e51a1f51c: Loading layer 1.365GB/1.365GB
a4cecbe9c23c: Loading layer [=====>] 182.5MB/182.5MB
3dd79884d940: Loading layer [=====>] 367.8MB/367.8MB
a64f48e250df: Loading layer [=====>] 69MB/69MB
5088a35e3b78: Loading layer [=====>] 11.89MB/11.89MB
87da348fbd10: Loading layer [=====>] 1.614MB/1.614MB
86da566e3251: Loading layer [=====>] 6.144kB/6.144kB
696af1d5db36: Loading layer [=====>] 6.144kB/6.144kB
615348a78cbe: Loading layer [=====>] 4.245GB/4.245GB
91fec05b562f: Loading layer [=====>] 11.96GB/11.96GB
63b3b8d8ac66: Loading layer [=====>] 5.632kB/5.632kB
624d19f69879: Loading layer [=====>] 129.6MB/129.6MB
891df624cf12: Loading layer [=====>] 503.1MB/503.1MB
c03eb81fe8b3: Loading layer [=====>] 42.5kB/42.5kB
1b80aca0704a: Loading layer [=====>] 866.6MB/866.6MB
94ab5726a0a8: Loading layer [=====>] 185.7MB/185.7MB
d17a07ca4036: Loading layer [=====>] 98.51MB/98.51MB
4c95a3bf38f6: Loading layer [=====>] 3.072kB/3.072kB
Loaded image: chatglm:8.1
[root@worker07 data]#
```

图4-2 加载镜像

加载成功后，执行如下命令查看容器镜像。

```
docker images
```

```
[root@worker07 data]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
chatglm       8.1       6b4d34ae83ea   1 month ago   20.2GB
[root@worker07 data]#
```

图4-3 查看镜像

步骤 3 创建并运行容器

切换到/root 目录后，执行如下命令查看当前服务器 NPU 信息：

```
cd
npu-smi info
```

```
[root@worker07 data]# cd [root@worker07 ~]# npu-smi info
```

npu-smi 23.0.7		Version: 23.0.7			
NPU Chip	Name	Health Bus-Id	Power(W) AICore(%)	Temp(C) Memory-Usage(MB)	Hugepages-Usage(page) HBM-Usage(MB)
0	910B3	OK	96.4	49	0 / 0
0		0000:C1:00.0	0	0 / 0	3163 / 65536
1	910B3	OK	95.5	49	0 / 0
0		0000:C2:00.0	0	0 / 0	3163 / 65536
2	910B3	OK	92.0	46	0 / 0
0		0000:81:00.0	0	0 / 0	3163 / 65536
3	910B3	OK	94.6	51	0 / 0
0		0000:82:00.0	0	0 / 0	3163 / 65536
4	910B3	OK	94.0	53	0 / 0
0		0000:01:00.0	0	0 / 0	3163 / 65536
5	910B3	OK	90.9	53	0 / 0
0		0000:02:00.0	0	0 / 0	3163 / 65536
6	910B3	OK	97.6	54	0 / 0
0		0000:41:00.0	0	0 / 0	3310 / 65536
7	910B3	OK	92.4	54	0 / 0
0		0000:42:00.0	0	0 / 0	3313 / 65536
NPU	Chip	Process id	Process name	Process memory(MB)	
No running processes found in NPU 0					

图4-4 查看 NPU 信息

从输出可以看到当前 8 颗 NPU 均无进程使用，在此选择使用第 1 颗为例创建模型容器，执行如下命令：

```
docker run -itd -e ASCEND_VISIBLE_DEVICES=1 --name group001 --pids-limit 409600 --shm-size 128G -v /home/data/ChatGLM-6B:/home/data/ChatGLM-6B chatglm:8.1 /bin/bash
```

```
[root@worker07 ~]# docker run -itd -e ASCEND_VISIBLE_DEVICES=1 --name group001 --pids-limit 409600 --shm-size 128G -v /home/data/ChatGLM-6B:/home/data/ChatGLM-6B chatglm:8.1 /bin/bash
c3e015d4dd944c564d0cf84a03dcfe6126b70cb3108f0e5c44ca3d00f9425be4
[root@worker07 ~]#
```

图4-5 创建容器

创建容器需要修改的参数说明：

- -e ASCEND_VISIBLE_DEVICES=5：表示 NPU 设备号，选择空闲 NPU，比如本例中的 1
- --name group001：容器名字，可自定义
- -v /home/data:/home/data：目录映射，表示将服务器/home/data 这个包含了模型文件的目录映射到容器内部的/home/data 目录
- chatglm:8.1：创建容器使用的镜像（如果镜像 tag 不是 8.1，请自行修改 tag，或将创建容器命令中的 tag 修改为环境的实际 tag）

步骤 4 进入容器

查询容器列表，找到本组的容器 ID。

```
docker ps
```

```
[root@worker07 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
31999deb2f8e   chatglm:8.1  "/bin/bash"             3 minutes ago  Up 3 minutes                group001
[root@worker07 ~]#
```

图4-6 查看容器

根据 ID 进入本组容器环境。

```
docker exec -it -u root af544e312a2b /bin/bash
```

```
[root@worker07 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
31999deb2f8e   chatglm:8.1  "/bin/bash"             3 minutes ago  Up 3 minutes                group001
[root@worker07 ~]# docker exec -it -u root 31999deb2f8e /bin/bash
[root@31999deb2f8e HwHiAiUser]#
```

图4-7 进入容器

步骤 5 查看 NPU 使用情况

在容器内部，执行命令查看该容器对 NPU 的使用情况。可以看到该容器已正常挂载了 NPU 编号为 1 的设备。

```
npu-smi info
```

```
[root@31999deb2f8e HwHiAiUser]# npu-smi info
+-----+
| npu-smi 23.0.7 | Version: 23.0.7 |
+-----+
| NPU   Name      | Health   | Power(W)  | Temp(C)   | Hugepages-Usage(page) |
| Chip  |              | Bus-Id    | AICore(%) | Memory-Usage(MB)      | HBM-Usage(MB) |
+-----+-----+
| 1     910B3     | OK       | 95.7      | 49        | 0 / 0                 | 0 / 0         |
| 0     0000:C2:00.0 | 0        | 0         | 0 / 0     | 3163 / 65536         | 3163 / 65536 |
+-----+-----+
| NPU   Chip      | Process id | Process name | Process memory(MB) |
+-----+-----+
| No running processes found in NPU 1 |
+-----+-----+
[root@31999deb2f8e HwHiAiUser]#
```

图4-8 查看容器内部 NPU 信息

步骤 6 模型推理测试

在容器内部，进入模型脚本路径，加载环境变量。

```
cd /home/data/ChatGLM-6B
```

```
source /usr/local/Ascend/ascend-toolkit/set_env.sh
```

运行推理，加载完成后，就可以和模型进行对话。

```
python3 infer.py
```



```

torch.logspace, torch.randint, torch.hann_window, torch.rand, torch.full_like, torch.ones_like, torch.rand_like, torch.randperm, torch.arange,
torch.frombuffer, torch.normal, torch.empty_per_channel_affine_quantized, torch.empty_strided, torch.empty_like, torch.scalar_tensor, torch.tril,
ndices, torch.bartlett_window, torch.ones, torch.sparse_coo_tensor, torch.randn, torch.kaiser_window, torch.tensor, torch.triu_indices, torch.as_t
nsor, torch.zeros, torch.randint_like, torch.full, torch.eye, torch.sparse_csr_tensor, torch.empty, torch.sparse_coo_tensor_unsafe, torch
blackman_window, torch.zeros_like, torch.range, torch.sparse_csr_tensor, torch.randn_like, torch.from_file, torch.cudnn_init_dropout_state, torch
empty_affine_quantized, torch.linspace, torch.hamming_window, torch.empty_quantized, torch.pin_memory, torch.autocast, torch.Tensor.new_empty, t
rch.Tensor.new_empty_strided, torch.Tensor.new_full, torch.Tensor.new_ones, torch.Tensor.new_tensor, torch.Tensor.new_zeros, torch.Tensor.to, torch
.nn.Module.to, torch.nn.Module.to_empty
*****
warnings.warn(msg, ImportWarning)
Explicitly passing a 'revision' is encouraged when loading a model with custom code to ensure no malicious code has been contributed in a newer re
ision.
Explicitly passing a 'revision' is encouraged when loading a configuration with custom code to ensure no malicious code has been contributed in a
ewer revision.
Explicitly passing a 'revision' is encouraged when loading a model with custom code to ensure no malicious code has been contributed in a newer re
ision.
Loading checkpoint shards: 100%| 8/8 [00:06<00:00, 1.18it/s
请输入对话
>>1+1
The dtype of attention mask (torch.int64) is not bool
/root/.cache/huggingface/modules/transformers_modules/model/modeling_chatglm.py:686: UserWarning: AutoNonVariableTypeMode is deprecated and will b
removed in 1.10 release. For kernel implementations please use AutoDispatchBelowADInplaceOrView instead. If you are looking for a user facing API
to enable running your inference-only workload, please use c10::InferenceMode. Using AutoDispatchBelowADInplaceOrView in user code is under risk o
producing silent wrong result in some edge cases. See Note [AutoDispatchBelowAutograd] for more details. (Triggered internally at /opt/_internal
python-3.7.17/lib/python3.7/site-packages/torch/include/Aten/core/LegacyTypeDispatch.h:79.)
attention_mask[:, :, -context_length] = 1
The sum of 1 and 1 is 2.
>>

```

图4-9 执行推理测试

此时，新打开一个终端连接登录到服务器，在不进入容器的情况下，查看 NPU 信息，可以看到编号为 1 的这个 NPU 正在被使用中。

npu-smi info				
0	0000:41:00.0	0	0 / 0	3310 / 65536
7	910B3	OK	92.8	54
0	0000:42:00.0	0	0 / 0	3313 / 65536

NPU	Chip	Process id	Process name	Process memory(MB)
No running processes found in NPU 0				
1	0	96253	python3	12334
No running processes found in NPU 2				
No running processes found in NPU 3				
No running processes found in NPU 4				
No running processes found in NPU 5				
No running processes found in NPU 6				
No running processes found in NPU 7				

图4-10 查看宿主机 NPU 占用情况

步骤 7 环境清理

在容器内部，按“Ctrl+C”结束正在推理的模型，然后输入 exit 退出容器。


```
[root@31999deb2f8e ChatGLM-6B]# Process ForkServerProcess-10:
Traceback (most recent call last):
  File "/usr/local/python3.7.5/lib/python3.7/multiprocessing/proc
self.run()
  File "/usr/local/python3.7.5/lib/python3.7/multiprocessing/proc
self._target(*self._args, **self._kwargs)
  File "/usr/local/Ascend/ascend-toolkit/latest/python/site-packag
os.kill(mgr_pid, signal.SIGKILL)
ProcessLookupError: [Errno 3] No such process
/usr/local/python3.7.5/lib/python3.7/multiprocessing/semaphore_tra
res to clean up at shutdown
len(cache))

[root@31999deb2f8e ChatGLM-6B]# exit
exit
[root@worker07 ~]#
```

图4-11 退出容器环境

停止并删除刚刚创建的容器：

```
docker ps
docker stop 31999deb2f8e
docker rm 31999deb2f8e
```

```
[root@worker07 ~]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
31999deb2f8e   chatglm:8.1  "/bin/bash"             22 hours ago   Up 20 hours                   group001
[root@worker07 ~]# docker stop 31999deb2f8e
31999deb2f8e
[root@worker07 ~]# docker rm 31999deb2f8e
31999deb2f8e
[root@worker07 ~]#
```

图4-12 清理环境

4.2 RAG 模型推理体验（可选）

RAG 即检索增强生成模型，本实验通过与前端页面结合，实现模型对话。

步骤 1 加载镜像

通过[实验设备及软件介绍](#)小节获取 RAG 模型镜像及文件后（“rag_hyde.zip”），在本地解压得到 “rag_hyde.tar” 容器镜像文件和 “Rag_hyde” 模型文件目录，上传到服务器 “/home/data” 目录。

```
[root@worker07 data]# ll
total 43877376
-rw-----. 1 root root 20313025536 Mar 28 16:37 chatglm.tar
drwxr-xr-x. 3 root root 30 Mar 28 14:18 ModelZoo
drwxr-xr-x. 7 root root 4096 Mar 28 14:18 Rag_hyde
-rw-----. 1 root root 24617400320 Mar 28 14:18 rag_hyde.tar
[root@worker07 data]#
```

图4-13 数据准备

执行镜像加载命令完成容器镜像加载，加载完成后查看镜像。（由于镜像较大，分层较多，如果发生加载失败的场景，可重复执行加载）

```
docker load -i rag_hyde.tar
docker images
```

```
[root@worker07 data]# docker load -i rag_hyde.tar
91db0b785fa0: Loading layer [=====>] 1.482GB/1.482GB
d69ccf89366e: Loading layer [=====>] 1.598GB/1.598GB
54a7ad3efa9c: Loading layer [=====>] 64.01MB/64.01MB
8d436ec4e7c6: Loading layer [=====>] 294MB/294MB
d48232cfd2bd: Loading layer [=====>] 867.9MB/867.9MB
ae5263b25873: Loading layer [=====>] 184.7MB/184.7MB
a2055390a4f7: Loading layer [=====>] 98.51MB/98.51MB
eb77bf51c5ab: Loading layer [=====>] 3.072kB/3.072kB
Loaded image: rag_hyde:8.2
[root@worker07 data]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
rag_hyde 8.2 03ca8e747f42 4 days ago 24.4GB
chatglm 8.1 6b4d34ae83ea 4 days ago 20.2GB
[root@worker07 data]#
```

图4-14 加载镜像

步骤 2 创建并运行容器

查看服务器 NPU 信息，可以看到所有 NPU 均空闲，无程序运行。

```
cd
npu-smi info
```

NPU	Chip	Process id	Process name	Process memory(MB)
No running processes found in NPU 0				
No running processes found in NPU 1				
No running processes found in NPU 2				
No running processes found in NPU 3				
No running processes found in NPU 4				
No running processes found in NPU 5				
No running processes found in NPU 6				
No running processes found in NPU 7				

图4-15 查看 NPU 信息

修改容器名称，比如使用 group05 创建容器：

```
docker run -it -d --net=host --shm-size=8g \
  --privileged \
  --name group05 \
  --device=/dev/davinci_manager \
  --device=/dev/hisi_hdc \
  --device=/dev/devmm_svm \
  -v /home/data/Rag_hyde:/home/data/Rag_hyde \
  rag_hyde:8.2
```

```
[root@worker07 ~]# docker run -itd -e ASCEND_VISIBLE_DEVICES=7 --pids-limit 409600 --shm-size 128G -p 7861:7861 -v /home/data/Rag_hyde:/home/data/Rag_hyde --name group07 rag_hyde:8.2 /bin/bash
8d682ce8129f5b9d873115f522348537e2ec88be500d09605541288342afad99
[root@worker07 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
8d682ce8129f   rag_hyde:8.2   "/bin/bash"             19 seconds ago   Up 16 seconds   22/tcp, 7860/tcp, 7871-7872/tcp, 7877/tcp, 0.0.0.0:7861->7861/tcp, :::7861->7861/tcp
group07
```

图4-16 创建容器

参数说明：

- -p 7861:7861：将容器内部的 7861 端口映射到服务器的 7861 端口
- 其他参数参考 [ChatGLM3-6B 模型推理测试](#) 小节创建模型容器的参数解释

步骤 3 进入容器

```
docker ps
docker exec -it -u root 8d682ce8129f /bin/bash
```

```
[root@worker07 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
8d682ce8129f   rag_hyde:8.2   "/bin/bash"             4 minutes ago   Up 4 minutes
group07
[root@worker07 ~]# docker exec -it -u root 8d682ce8129f /bin/bash
[root@8d682ce8129f HwHiAiUser]#
```

图4-17 进入容器

步骤 4 修改端口号

进入挂载的模型运行代码目录，根据创建时候指定的端口号，修改代码，比如本例中使用的端口号为 7861。

```
cd /home/data/Rag_hyde
vi GLM-6B-langchain.py
```

打开文件后，按“G”跳到文件最后，将默认的端口号修改为 7861（创建多个容器时，端口需要不同）。

```
# 向量库数据输出按钮
#上传文件路径
    upload_folder = "/tmp/gradio/wyf"
#点击加载文件按钮，把新内容加载到向量库
persist_directory_input = gr.Textbox(value=persist_directory)
print_msg = other.click(fn=read_files_by_extension, inputs=
ipplay, inputs=None, outputs=None)
torch.npu.set_device(torch.device("npu:0"))
print_msg = other.click(fn=refresh_vectordb, outputs=[])
db_wo_his_btn.click(handle_query, inputs=[radio,
msg, chatbot], outputs=[chatbot]).then(
clear.click(model_center.clear_template, inputs = [], output
gr.Markdown("""提醒: <br>
1. 使用深度检索耗时可能较长，请耐心等待。
2. 使用中如果出现异常，将会在文本输入框进行展示。 <br>
"""))
gr.close_all()
# 直接启动
demo.launch(server_name='0.0.0.0', server_port=7861, debug=True)
-- INSERT --
```

图4-18 修改服务端口

步骤 5 运行模型推理

加载环境变量，无任何输出表示加载成功。

```
source /usr/local/Ascend/ascend-toolkit/set_env.sh
```

```
[root@8d682ce8129f HwHiAiUser]# cd /home/data/Rag_hyde/
[root@8d682ce8129f Rag_hyde]# vi GLM-6B-langchain.py
[root@8d682ce8129f Rag_hyde]#
[root@8d682ce8129f Rag_hyde]# source /usr/local/Ascend/ascend-toolkit/set_env.sh
[root@8d682ce8129f Rag_hyde]#
```

图4-19 加载环境变量

运行模型。

```
python3.9 GLM-6B-langchain.py
```

当出现如下提示时表示模型运行成功。

```

Loading checkpoint shards: 100%|
完成本地模型的加载
向量库数据条数 889
QA_CHAIN_PROMPT input_variables=['context', 'question'] output_parser=None partial_variables={} template='仅仅使用以下
说你不知道，不要试图编造答\n 案。尽量使答案简明扼要。总是在回答的最后说“谢谢你的提问！”。\n  <system>: {context}\n
plate_format='f-string' validate_template=True
/usr/local/lib/python3.9/site-packages/gradio/routes.py:1019: DeprecationWarning:
    on_event is deprecated, use lifespan event handlers instead.

    Read more about it in the
    [FastAPI docs for Lifespan Events](https://fastapi.tiangolo.com/advanced/events/).

    @app.on_event("startup")
/usr/local/lib/python3.9/site-packages/fastapi/applications.py:4495: DeprecationWarning:
    on_event is deprecated, use lifespan event handlers instead.

    Read more about it in the
    [FastAPI docs for Lifespan Events](https://fastapi.tiangolo.com/advanced/events/).

    return self.router.on_event(event_type)
/usr/local/lib/python3.9/site-packages/gradio/components/base.py:187: UserWarning: 'scale' value should be an integer.
warnings.warn(
/usr/local/lib/python3.9/site-packages/gradio/route_utils.py:800: DeprecationWarning: The loop argument is deprecated
n Python 3.10.
    app.stop_event = asyncio.Event(loop=loop)
Running on local URL: http://0.0.0.0:7861
To create a public link, set `share=True` in `launch()`.

```

图4-20 运行模型推理

步骤 6 登录验证

使用宿主机 IP，也就是物理服务服务器的 IP 和端口 7861 打开浏览器，可体验和模型进行对话。



图4-21 登录推理应用

步骤 7 环境清理

退出模型推理程序，在容器内部，按“Ctrl+C”终止程序运行后，输入“exit”并回车退出容器。（因模型较大，可多按几次“Ctrl+C”加快程序终止速度）

```
KeyboardInterrupt
/usr/local/lib/python3.9/tempfile.py:817: ResourceWarning: Implicitly
warnings.warn(warn_message, ResourceWarning)
^C
[root@8d682ce8129f Rag_hyde]# ^C
[root@8d682ce8129f Rag_hyde]# exit
exit
[root@worker07 ~]#
```

图4-22 停止推理并退出容器

停止并删除容器。

```
docker ps
docker stop 8d682ce8129f
docker rm 8d682ce8129f
docker ps
```

```
[root@worker07 ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
NAMES
8d682ce8129f   rag_hyde:8.2   "/bin/bash"             17 minutes ago Up 17 minutes
group07
[root@worker07 ~]# docker stop 8d682ce8129f
8d682ce8129f
[root@worker07 ~]# docker rm 8d682ce8129f
8d682ce8129f
[root@worker07 ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS   PORTS   NAMES
[root@worker07 ~]#
```

图4-23 清理环境

4.3 DeepSeek 推理性能测试

本实验将通过对一个基于 LLaMA-70B 架构并使用 bfloat16 (bf16)精度训练好的 DeepSeek 模型进行不同卡的推理性能测试，以此观测 NPU 卡对推理性能的影响。

步骤 1 数据准备

根据[实验设备及软件介绍](#)小节，在服务器自定义目录中，比如 “/home/data”，创建 “deepseek-R1-bf16” 目录，并下载模型文件。

```
cd /home/data
mkdir deepseek-R1-bf16
cd deepseek-R1-bf16
pip install modelscope
modelscope download --model deepseek-ai/DeepSeek-R1-Distill-Llama-70B
```

步骤 2 镜像准备

根据[实验设备及软件介绍](#)小节，申请 MindIE 镜像，镜像版本选择 2.0.T3.1-800I-A2-py311-openeuler24.03-lts，参考网页提示完成镜像拉取，拉取完成后出现如下类似输出：

```
2.0.T3.1-800I-A2-py311-openeuler24.03-lts: Pulling from ascendhub/mindie
fb3a8b863e2c: Pull complete
.....
886002f8aa60: Pull complete
Digest: sha256:42e7aa006b1d22f6e38ec7bd8d2e272200b7e556beb9a65379a694880c060a93
Status: Downloaded newer image for swr.cn-south-1.myhuaweicloud.com/ascendhub/mindie:2.0.T3.1-800I-A2-py311-openeuler24.03-lts
swr.cn-south-1.myhuaweicloud.com/ascendhub/mindie:2.0.T3.1-800I-A2-py311-openeuler24.03-lts
```

加载成功后，查看镜像：

```
docker images

[root@worker07 mindie]# docker images
REPOSITORY                                TAG
rag_hyde                                  8.2
chatglm                                    8.1
swr.cn-south-1.myhuaweicloud.com/ascendhub/mindie 2.0.T3.1-800I-A2-py311-openeuler24.03-lts
[root@worker07 mindie]#
```

图4-24 查看镜像

步骤 3 推理容器部署

执行如下命令，使用 mindie 容器镜像部署推理容器：

```
docker run -it -d --net=host --shm-size=16g \
    --privileged \
    --name mindie \
    --device=/dev/davinci_manager \
    --device=/dev/hisi_hdc \
    --device=/dev/devmm_svm \
    -v /usr/local/Ascend/driver:/usr/local/Ascend/driver:ro \
```

```
-v /usr/local/sbin:/usr/local/sbin:ro \
-v /home/data/deepseek-R1-bf16/DeepSeek-R1-Distill-Llama-70B:/DeepSeek-R1-Distill-Llama-70B-bf16:rw \
swr.cn-south-1.myhuaweicloud.com/ascendhub/mindie:2.0.T3.1-800I-A2-py311-openeuler24.03-lts
```

参数解释：

- --net：指定容器运行使用的网络命名空间，host 表示直接使用主机的命名空间
- --shm-size：为容器分配共享内存大小
- --privileged：特权容器，拥有更多的权限
- --name：指定容器名称
- --device：指定映射到容器的设备，将宿主机的相关设备都映射到容器
- -v：将宿主机的目录映射到容器，本例中将 NPU 相关驱动文件及模型文件目录的映射到容器

步骤 4 对话测试

进入容器：

```
docker ps
docker exec -it 18c91 bash
```

```
[root@worker07 ~]# docker ps
CONTAINER ID   IMAGE                                PORTS          NAMES
18c91c0ba12c   swr.cn-south-1.myhuaweicloud.com/ascendhub/mindie:2.0.T3.1-800I-A2-py311-openeuler24.03-lts  mindie
"bash"        11 minutes ago  Up 11 minutes
[root@worker07 ~]# docker exec -it 18c91 bash

Welcome to 4.14.0-115.el7a.0.1.aarch64

System information as of time: Tue Apr  8 09:25:25 CST 2025

System load:    9.06
Memory used:    1.9%
Swap used:      0%
Usage On:       14%
Users online:   0

[root@worker07 /]#
```

图4-25 进入容器

进入模型路径后，发起对话测试：

```
cd $ATB_SPEED_HOME_PATH
torchrun --nproc_per_node 8 \
--master_port 20037 \
-m examples.run_pa \
--model_path /DeepSeek-R1-Distill-Llama-70B-bf16 \
--input_texts '你是什么模型' \
--max_output_length 20 \
```



```

)
[ 3] [4236] [281473001998624] [llm] [INFO] [cache.py-102] : kv cache will allocate 0.0439453125GB memory
[ 1] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : hbm_capacity(GB): 64.0, init_memory(GB): 14.7
[ 1] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : pa_runner: PARunner(model_path=/DeepSeek-R1-D
eddings=None, max_input_length=1024, max_output_length=20, max_prefill_tokens=-1, load_tokenizer=True, enable_atb_torch=False, m
h.bfloat16, block_size=128, model_config=ModelConfig(num_heads=8, num_kv_heads=1, num_kv_heads_origin=8, head_size=128, k_head_s
type=torch.bfloat16, soc_info=NPU SocInfo(soc_name='', soc_version=223, need_nz=False, matmul_nd_nz=False), kv_quant_type=None, f
pp_rank=0, pp_groups=[[0], [1], [2], [3], [4], [5], [6], [7]], micro_batch_size=1), cla_share_factor=1, model_type=llama), max
[2025-04-08 00:20:40.701] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : -----begin warm up-----
[ 1] [4230] [281472987973920] [llm] [INFO] [cache.py-102] : kv cache will allocate 0.0439453125GB memory
[ 5] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : -----total req num: 1, infer start-----
[ 8] [4232] [281473342720288] [llm] [INFO] [cache.py-102] : kv cache will allocate 0.0439453125GB memory
[ 0] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : <<<<<< ori k.caches[0].shape=torch.Size[9,
[ 1] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : >>>>>id of kcache is 281469947578032 id of v
[ 2] [4237] [281473740261664] [llm] [INFO] [cache.py-102] : kv cache will allocate 0.0439453125GB memory
[ 2] [4233] [281473681148192] [llm] [INFO] [cache.py-102] : kv cache will allocate 0.0439453125GB memory
[ 2] [4231] [281473127368992] [llm] [INFO] [cache.py-102] : kv cache will allocate 0.0439453125GB memory
[ 2] [4235] [281473685801248] [llm] [INFO] [cache.py-102] : kv cache will allocate 0.0439453125GB memory
[ 7] [4234] [281473542998304] [llm] [INFO] [cache.py-102] : kv cache will allocate 0.0439453125GB memory
[ 6] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : warmup memory(GB): 16.00
[ 6] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : -----end warm up-----
[ 6] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : -----begin inference-----
[ 4] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : -----total req num: 1, infer start-----
[ 8] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : -----end inference-----
[ 8] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : Answer[0] : ?

</think>

</think>

我是一个AI模型，旨在通过输入和输出的形式
[2 , 3] [4230] [281472987973920] [llm] [INFO] [logging.py-331] : Generate[0] token num: (0, 20)
[root@worker07 atb-models]#

```

图4-26 发起服务对话测试

参数解释：

- --nproc_per_node 8: 每个节点使用 8 颗 NPU 设备
- --master_port 20037: 主节点监听的端口号
- -m examples.run_pa: 把 examples 包下的 run_pa 模块当作脚本来执行
- --model_path /DeepSeek-R1-Distill-Llama-70B-bf16: 指定模型文件路径，该路径在创建容器的时候映射到容器内部
- --input_texts: 向脚本输出的参数
- --max_output_length 20: 限定模型生成文本的最大长度为 20

步骤 5 推理性能测试

进入性能测试路径，使用 2 颗 NPU 执行测试脚本：

```
cd $ATB_SPEED_HOME_PATH/tests/modeltest/
bash run.sh pa_bf16 performance [[256,256]] 1 llama /DeepSeek-R1-Distill-Llama-70B-bf16 2
```

测试完成后，执行如下命令查看并记录测试结果：

```
cat /usr/local/Ascend/atb-
models/tests/modeltest/outputs/results/NPU/performance_test/performance/bf16_w8a8/llama3.1_70b/p
erformance_pa_batch1_tp2_result.csv
```


5 基础工具运用

5.1 HCCN Tool 工具使用

在安装昇腾软件时，会同步安装 HCCN Tool 工具。该工具可以对算力服务器的参数面网络进行配置管理，也就是对 NPU 的网络信息进行配置及查看等操作。

步骤 1 查看 NPU 网络信息

查看第一颗 NPU 的网络信息，服务器共有 8 颗 NPU，分别编号 0-7。

```
hccn_tool -i 0 -ip -g
```

```
[root@worker01 data]# hccn_tool -i 0 -ip -g  
ipaddr:192.168.21.10  
netmask:255.255.255.0  
[root@worker01 data]#
```

图5-1 查看 NPU IP 地址

查看 NPU 的网关信息。

```
hccn_tool -i 0 -gateway -g
```

```
[root@worker01 data]# hccn_tool -i 0 -gateway -g  
default gateway:192.168.21.1, Iface:eth0  
[root@worker01 data]#
```

图5-2 查看 NPU 网关信息

可以添加循环，同时查看 8 颗 NPU 信息。

```
for i in {0..7};do hccn_tool -i $i -gateway -g;done
```

```
[root@worker01 data]# for i in {0..7};do hccn_tool -i $i -gateway -g;done  
default gateway:192.168.21.1, Iface:eth0  
default gateway:192.168.21.1, Iface:eth1  
default gateway:192.168.21.1, Iface:eth2  
default gateway:192.168.21.1, Iface:eth3  
default gateway:192.168.21.1, Iface:eth4  
default gateway:192.168.21.1, Iface:eth5  
default gateway:192.168.21.1, Iface:eth6  
default gateway:192.168.21.1, Iface:eth7  
[root@worker01 data]#
```

图5-3 查看所有 NPU 信息

步骤 2 修改网络信息

如果在运行过程中需要修改 NPU 网络信息，也可以通过 HCCN Tool 工具完成。

先查询第 8 颗 NPU 的 IP 地址。

```
hccn_tool -i 7 -ip -g
```

再修改 IP 地址，并查看是否修改成功。

```
hccn_tool -i 7 -ip -s address 192.168.21.18 netmask 255.255.255.0
```

```
hccn_tool -i 7 -ip -g
```

```
[root@worker01 data]# hccn_tool -i 7 -ip -s address 192.168.21.18 netmask 255.255.255.0
[root@worker01 data]# hccn_tool -i 7 -ip -g
ipaddr:192.168.21.18
netmask:255.255.255.0
[root@worker01 data]#
```

图5-4 NPU 信息修改

步骤 3 其他用法

HCCN Tool 支持对 NPU 进行丰富的网络操作，可通过如下命令查看具体用法。

```
hccn_tool -h
```

具体用法如下：

```
hccn_tool -i devid -ip
[-s]          : set ip cfg
               address %s : set ip address, such as x.x.x.x
               netmask %s : set netmask, such as x.x.x.x
[-g]          : get ip address and netmask

hccn_tool -i devid -gateway
[-s]          : set gateway cfg
               gateway %s : set gateway, such as x.x.x.x
[-g]          : get gateway address

hccn_tool -i devid -netdetect
.....
```

5.2 ascend-dmi 工具使用

ascend-dmi 可随昇腾软件包一起安装，主要用于测试 NPU 的算力、带宽及功耗。

步骤 1 环境变量加载

确保[昇腾软件包部署](#)章节已完成 ToolBox 工具的安装，执行如下命令加载环境变量：

```
source /usr/local/Ascend/ascend-toolkit/set_env.sh
```

```
source /usr/local/Ascend/toolbox/set_env.sh
```

步骤 2 带宽测试

测试 D2H 带宽，表示从某个 NPU 设备到主机的带宽。请确保在无业务的情况下进行测试，否则测试结果不准确且会对业务造成影响，执行测试命令后，根据提示输入“Y”开始测试。

```
ascend-dmi --bw -t d2h
```

```
[root@worker07 ~]# ascend-dmi --bw -t d2h
This test will affect the business on this server. To ensure the correctness and accuracy
to continue?(Y/N)Y
Device to Host Test
Device 0: Ascend 910B3.
-----
Size(Bytes)  Execute Times  Bandwidth(GB/s)  Elapsed Time(us)
-----
      2           5         0.000186         10.73
      4           5         0.000373         10.73
      8           5         0.000729         10.97
     16           5         0.001428         11.21
     32           5         0.002856         11.21
     64           5         0.005836         10.97
    128           5         0.011185         11.44
    256           5         0.022370         11.44
    512           5         0.042108         12.16
   1024           5         0.084215         12.16
   2048           5         0.168430         12.16
   4096           5         0.381775         10.73
   8192           5         0.731058         11.21
  16384           5         1.527099         10.73
  32768           5         3.054199         10.73
  65536           5         6.108398         10.73
 131072           5         9.012390         14.54
 262144           5         7.003259         37.43
 524288           5        11.453246         45.78
1048576           5        17.179869         61.04
2097152           5        22.787806         92.03
4194304           5        25.570038        164.03
8388608           5        26.756176        313.52
16777216           5        27.638941        607.01
33554432           5        28.108146       1193.76
-----
[root@worker07 ~]#
```

图5-5 测试 NPU 到主机的带宽

测试 H2D 带宽，表示从主机到某个 NPU 设备的带宽。


```
[root@worker07 ~]# ascend-dmi --bw -t h2d
This test will affect the business on this server. To ensure the correctness
N)Y
Host to Device Test
Device 0: Ascend 910B3.
```

Size(Bytes)	Execute Times	Bandwidth(GB/s)	Elapsed Time(us)
2	5	0.000186	10.73
4	5	0.000365	10.97
8	5	0.000780	10.25
16	5	0.001637	9.78
32	5	0.003196	10.01
64	5	0.006711	9.54
128	5	0.012783	10.01
256	5	0.026189	9.78
512	5	0.049941	10.25
1024	5	0.095444	10.73
2048	5	0.204522	10.01
4096	5	0.419021	9.78
8192	5	0.858993	9.54
16384	5	1.636178	10.01
32768	5	3.352170	9.78
65536	5	6.704339	9.78
131072	5	8.589935	15.26
262144	5	13.089424	20.03
524288	5	13.658530	38.39
1048576	5	17.113021	61.27
2097152	5	21.144454	99.18
4194304	5	23.425015	179.05
8388608	5	24.518726	342.13
16777216	5	24.786454	676.87
33554432	5	25.199192	1331.57

图5-6 测试主机到 NPU 设备的带宽

测试 D2D 带宽，表示某个 NPU 设备两端的带宽。

```
[root@worker07 ~]# ascend-dmi --bw -t d2d
This test will affect the business on this server. To ensure the correctness
N)Y
Device to Device Test
Device 0: Ascend 910B3.
```

Size(GB)	Execute Times	Bandwidth(GB/s)	Elapsed Time(us)
20.97	1	1519.783203	13799.02
20.97	1	1534.881714	13663.28
20.97	1	1511.471802	13874.90
20.97	1	1533.516968	13675.44
20.97	1	1534.951538	13662.66
20.97	1	1528.309204	13722.04
20.97	1	1533.909546	13671.94
20.97	1	1537.724731	13638.02
20.97	1	1534.428101	13667.32
20.97	1	1537.436157	13640.58
20.97	1	1543.071899	13590.76
20.97	1	1536.420166	13649.60
20.97	1	1543.523926	13586.78
20.97	1	1518.152710	13813.84
20.97	1	1542.722412	13593.84
20.97	1	1520.978027	13788.18
20.97	1	1533.936523	13671.70
20.97	1	1534.533569	13666.38
20.97	1	1541.148682	13607.72
20.97	1	1535.376221	13658.88
20.97	1	1540.157227	13616.48
20.97	1	1534.973877	13662.46
20.97	1	1532.889282	13681.04
20.97	1	1536.960449	13644.80
20.97	1	1535.023315	13662.02

图5-7 测试 NPU 设备自身带宽

测试 P2P 带宽，表示 NPU 设备之间的带宽。

```
[root@worker07 ~]# ascend-dmi --bw -t p2p
This test will affect the business on this server. To ensure the
N)Y
Unidirectional Peer to Peer Test Bandwidth Matrix (GB/s)
  D\D  0    1    2    3    4    5    6    7
  0    ***  26.17 26.16 26.17 26.17 26.17 26.17 26.17
  1    26.16 ***  26.16 26.16 26.16 26.16 26.16 26.17
  2    26.16 26.16 ***  26.16 26.15 26.16 26.16 26.16
  3    26.15 26.16 26.16 ***  26.16 26.16 26.16 26.16
  4    26.16 26.16 26.16 26.16 ***  26.17 26.16 26.17
  5    26.16 26.16 26.16 26.16 26.16 ***  26.16 26.17
  6    26.16 26.16 26.16 26.16 26.16 26.16 ***  26.17
  7    26.16 26.16 26.16 26.16 26.16 26.16 26.16 ***

Bidirectional Peer to Peer Test Bandwidth Matrix (GB/s)
  D\D  0    1    2    3    4    5    6    7
  0    ***  51.02 51.01 50.44 51.00 50.99 51.01 50.98
  1    51.00 ***  51.02 50.99 50.98 51.01 51.02 50.99
  2    51.01 51.01 ***  51.13 50.97 51.02 51.00 51.00
  3    50.98 50.99 51.02 ***  50.98 50.97 51.01 51.02
  4    50.99 51.00 50.99 50.94 ***  51.02 50.98 50.98
  5    50.99 51.01 51.02 50.99 51.01 ***  51.03 50.99
  6    51.00 51.02 50.99 50.99 50.99 51.04 ***  51.00
  7    50.95 50.98 50.99 51.01 51.04 51.01 51.01 ***
```

图5-8 测试 NPU 设备之间单项和双向通信带宽

步骤 3 算力测试

可针对单个 NPU 卡进行算力测试。

```
ascend-dmi -f -d 0
```

```
[root@worker01 ~]# ascend-dmi -f -d 0
-----
Device          Execute Times      Duration(ms)      TFL0PS@FP16      Power(W)
-----
0                360,000,000         1703              313.527          253
-----
```

图5-9 测试单个 NPU 卡算力

默认使用 FP16 的精度，如果想指定精度，可以使用-t 参数。

```
ascend-dmi -f -d 0 -t fp32
```

```
[root@worker01 ~]# ascend-dmi -f -d 0 -t fp32
-----
Device          Execute Times      Duration(ms)      TFL0PS@FP32      Power(W)
-----
0                360,000,000         1703              83.007           195.199997
-----
[root@worker01 ~]#
```


图5-10 指定精度测试 NPU 卡算力

步骤 4 功耗测试

```
ascend-dmi -p
```

```
[root@worker01 ~]# ascend-dmi -p
```

Type	NPU Count		
Device ID Chip Name	Health AI Core Usage	Temperature Power	Voltage
Ascend 910 B3	8		
0 Ascend 910B3	OK 0%	48C 349.6W	0.75V
1 Ascend 910B3	OK 0%	48C 349.6W	0.84V
2 Ascend 910B3	OK 0%	49C 94.6W	0.72V
3 Ascend 910B3	OK 0%	48C 103.4W	0.73V
4 Ascend 910B3	OK 0%	54C 99.8W	0.75V
5 Ascend 910B3	OK 0%	54C 101.6W	0.74V
6 Ascend 910B3	OK 0%	53C 103.3W	0.74V
7 Ascend 910B3	OK 0%	56C 94.3W	0.72V

图5-11 NPU 卡功耗检测

华为认证计算系列教程

HCIE-Computing

计算性能优化实验指导手册

版本：2.0



华为技术有限公司

版权所有 © 华为技术有限公司 2025。保留一切权利。

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

商标声明



和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

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

华为技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<https://e.huawei.com>

华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构，打造的覆盖ICT（Information and Communications Technology，信息通信技术）全技术领域的认证体系，包含ICT基础设施认证、基础软硬件认证、云平台及云服务认证三类认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

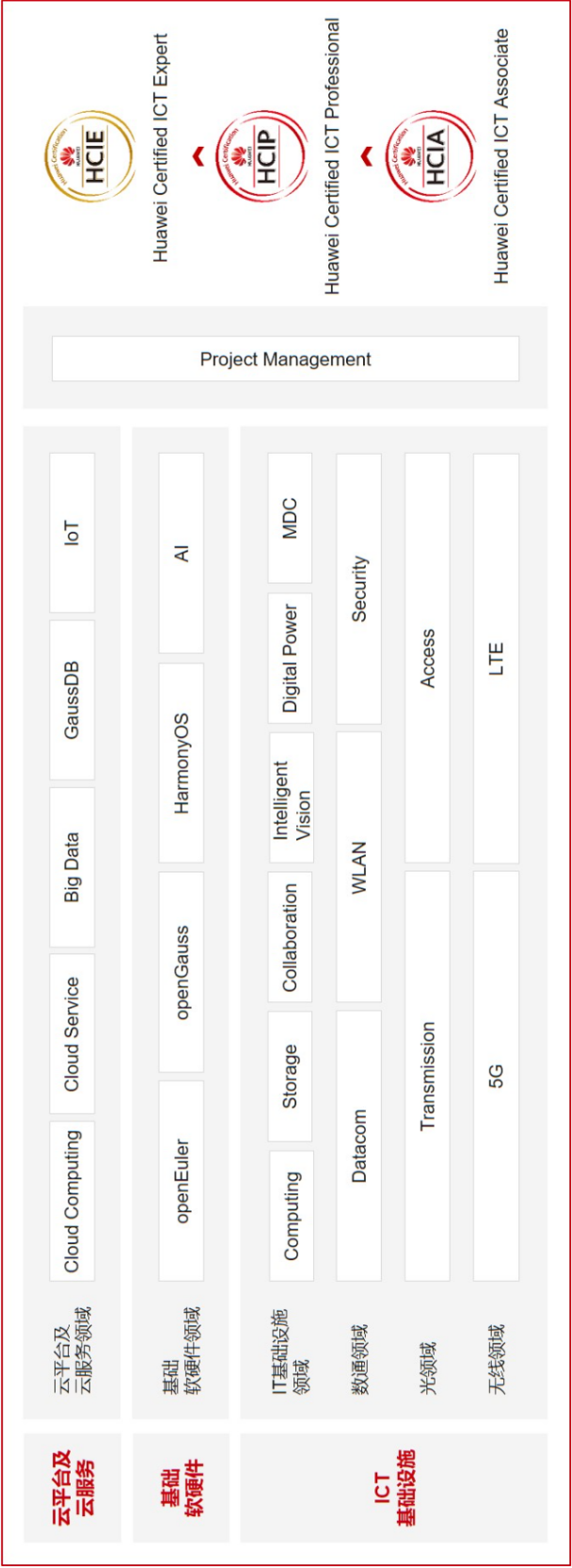
华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

华为认证HCIE-Computing V2.0定位于培养与认证具备人工智能计算中心规划设计、安装部署、性能优化与运维能力的计算专家。

通过HCIE-Computing V2.0认证，您将了解人工智能的发展历程及大模型相关原理，理解人工智能计算中心面临的挑战与解决方案，掌握数据中心的设计方法及人工智能计算中心训练、推理场景的方案，并能进行人工智能数据中心部署，掌握计算设备的性能调优及维护方法。

华为职业认证全景图

华为职业认证覆盖ICT全领域，致力于提供领先的人才培养体系和认证标准，培养数字化时代的新型ICT人才，构建良性的ICT人才生态。



简介

本书为 HCIE-Computing 认证培训教程，适用于准备参加 HCIE-Computing 考试的学员或者希望了解昇腾及鲲鹏解决方案的读者。

内容描述

本实验指导书共包含 3 个实验，逐一介绍了 Linux 系统中性能观测工具 iostat 和 sar 的使用、BoostKit 的性能优化实践及 DevKit 的性能优化实践。

- 实验一为 Linux 常用性能工具相关的实验，本实验中挑选了功能比较强大和全面的 iostat 和 sar 进行练习，掌握如何收集 CPU、内存、网络 and 存储性能数据。
- 实验二为 BoostKit 相关的实验，包括系统库、数学库和 KAE 的使用实践。
- 实验三为 DevKit 相关的实验，包括了 DevKit 性能优化相关模块的安装、如何通过 DevKit 进行对非并行计算和并行计算应用的性能分析。

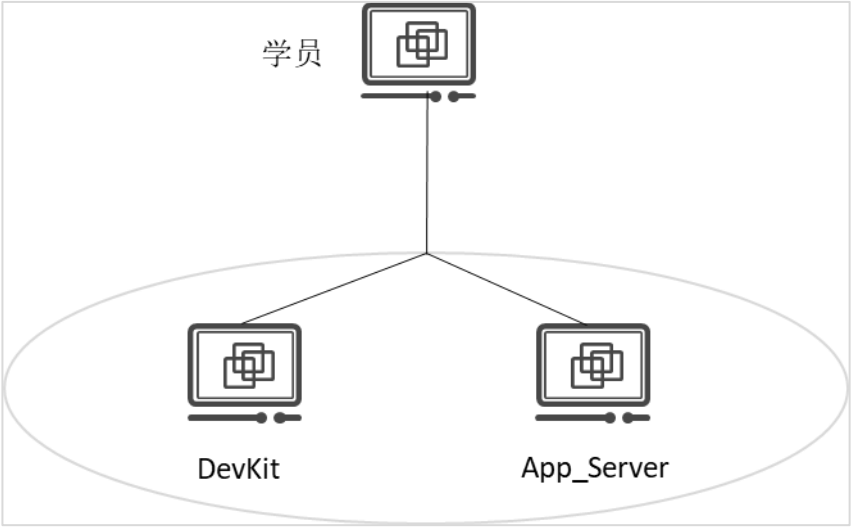
读者知识背景

本课程为华为认证专家级别课程，为了更好地掌握本书内容，阅读本书的读者应首先具备以下基本条件：

- 具有基本的 Linux 知识背景，及相关性能调优的基础知识。

实验环境说明

本实验共使用两台虚拟机，分别是 DevKit 和 App_Server，其中 App_Server 将用于完成实验一的全部内容，并在实验三中作为 DevKit 的客户端，DevKit 用于完成实验二，并在实验三中作为 DevKit 的服务端。具体结构如下图所示：



准备实验环境

检查设备

实验开始之前请每组学员按照下表申请或创建虚拟机：

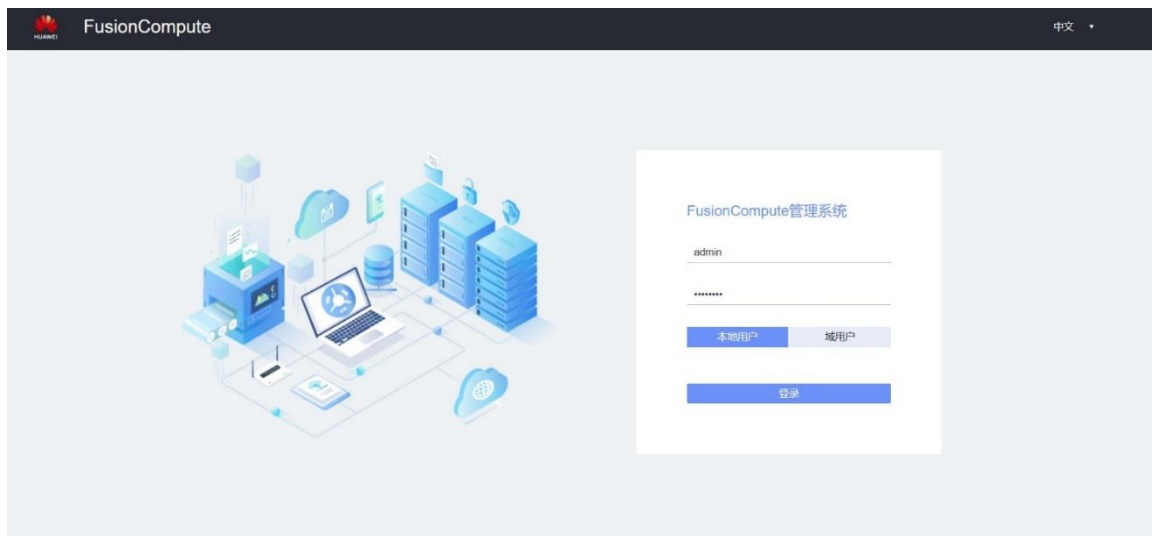
虚拟机名称	规格	备注
DevKit	4vCPUs 12GiB 40G硬盘	\
App_Server	2vCPUs 4GiB 40G硬盘	\

创建虚拟机

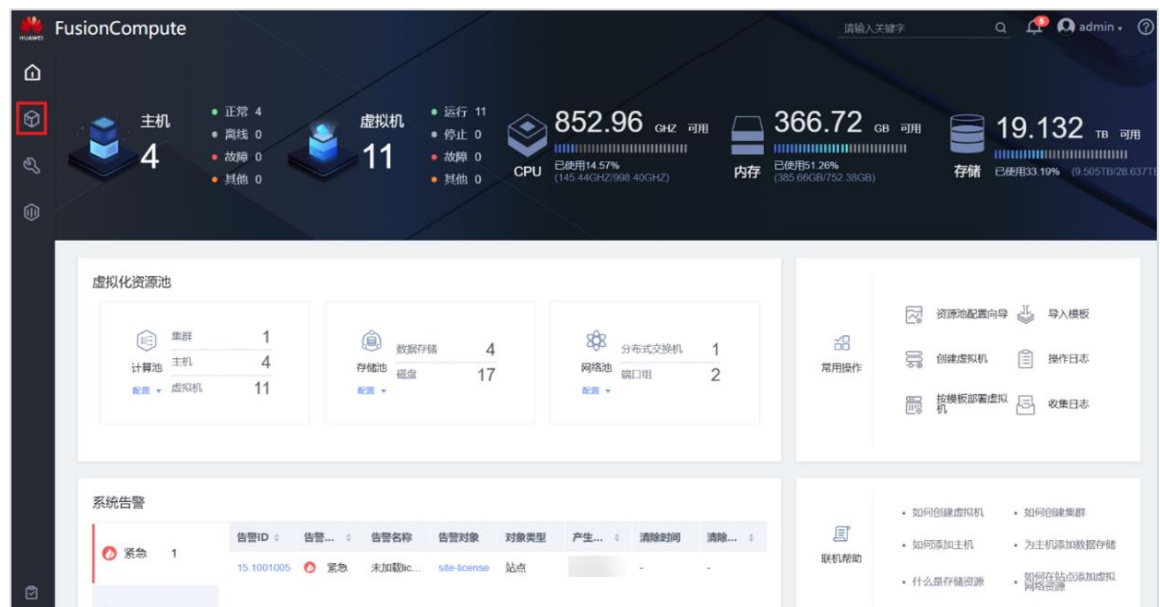
以下将以 App_Server 为例来创建实验过程需要的虚拟机，DevKit 虚拟机可以按照相同的方式创建（DevKit 使用的镜像文件为：ARM-DevKit-vda.qcow2），注意，创建 DevKit 虚拟机时，需要将规格修改为：vCPU：4，内存：12G。

步骤 1 导入虚拟机模板

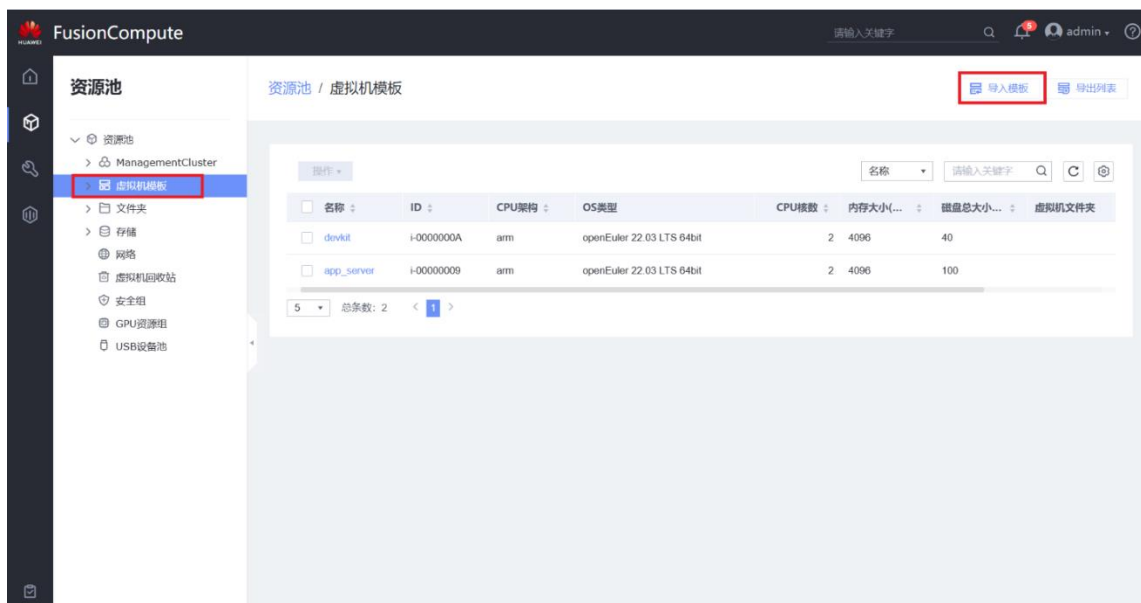
首先登陆到实验环境的 FusionCompute 管理界面，输入用户和密码进行登陆：



登陆后进入如下主界面，点击立方体图标进入资源列表：



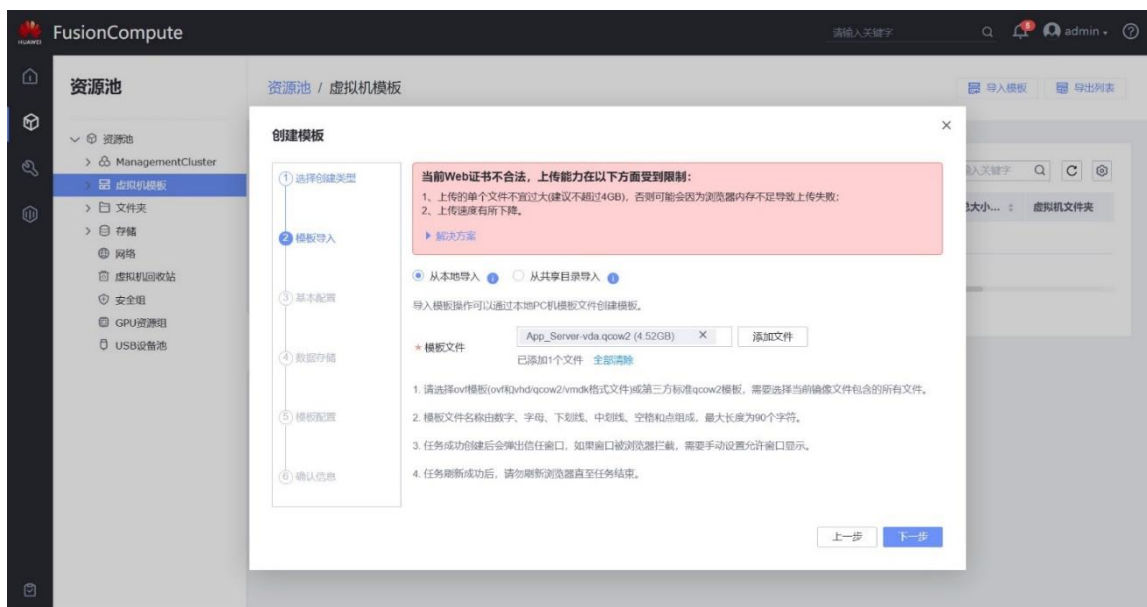
选择虚拟机模板，点击右上角“导入模板”按钮，进入模板导入页面：



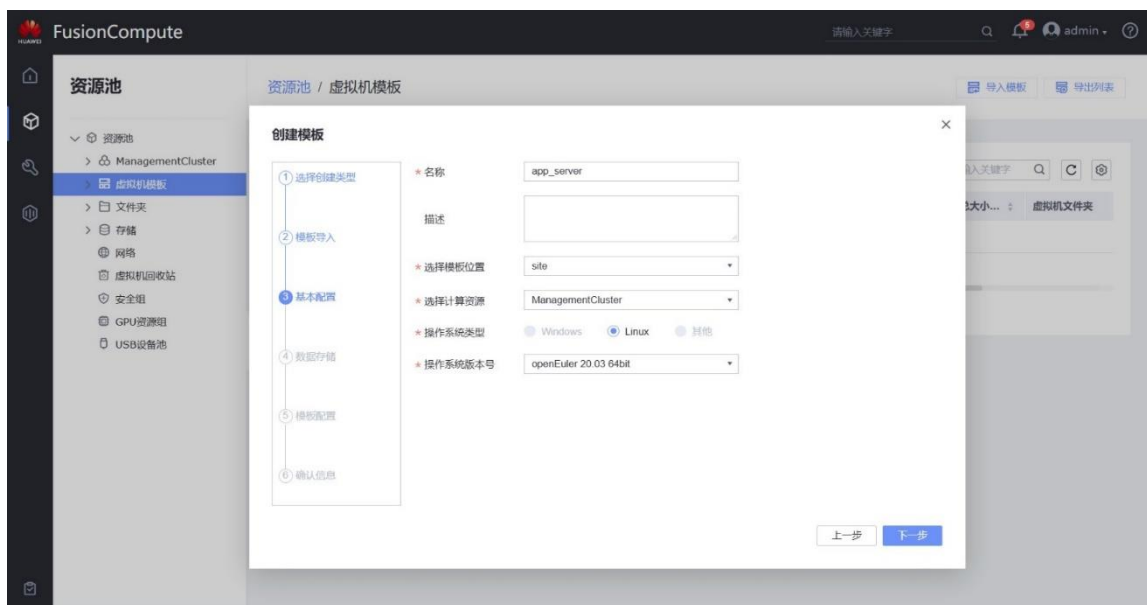
点击“添加文件”，在本地选择该实验预先提供的镜像文件“App-Server-vda.qcow2”进行添加。注意：此处镜像文件的命名必须以*vda.qcow2 的格式进行命名，否则系统会提示规范命名。



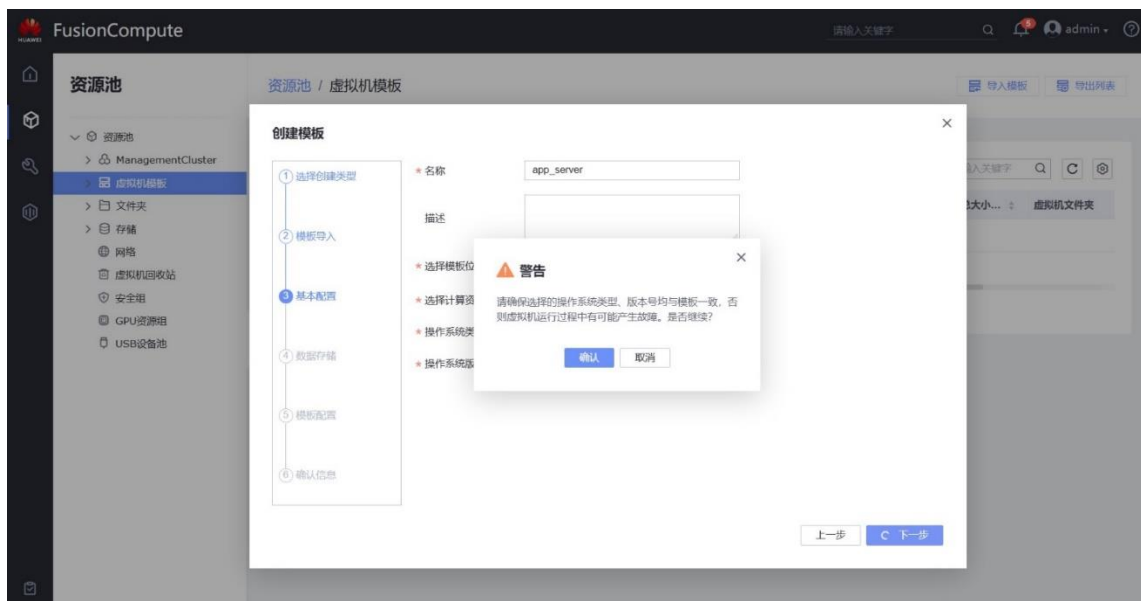
添加后，点击“下一步”：



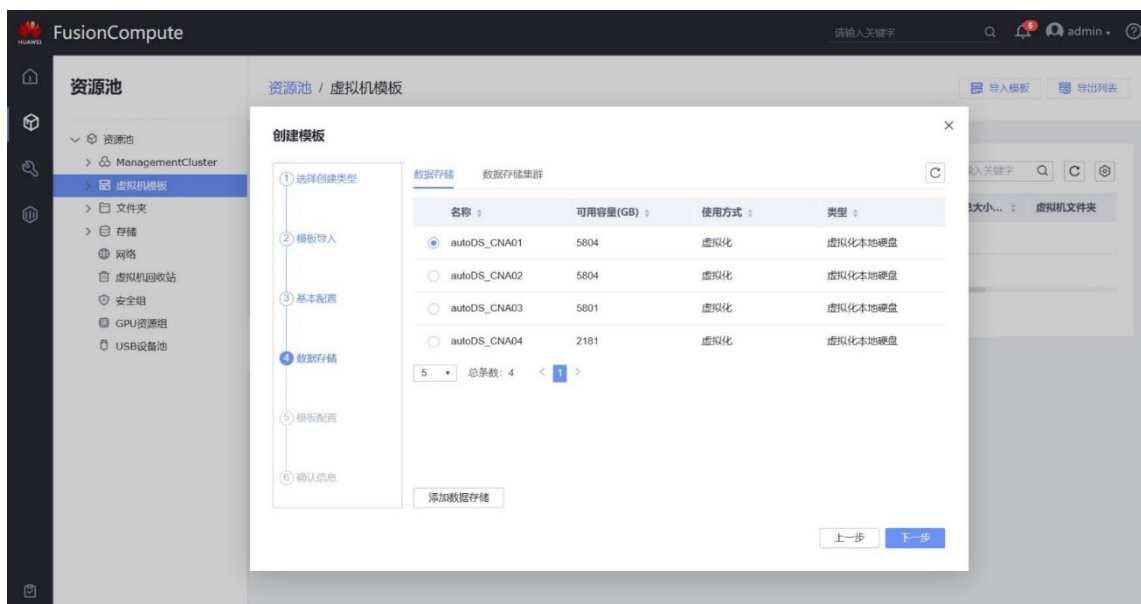
参考下图填写相应的选项, 点击“下一步”:



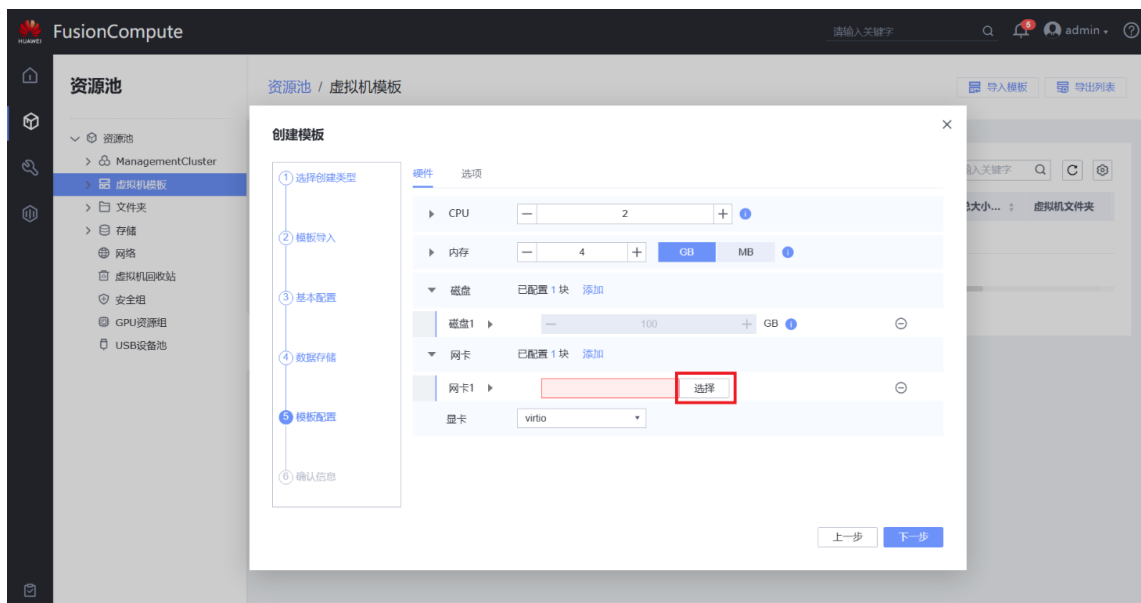
如果弹出如下警告, 请点击“确认”继续:



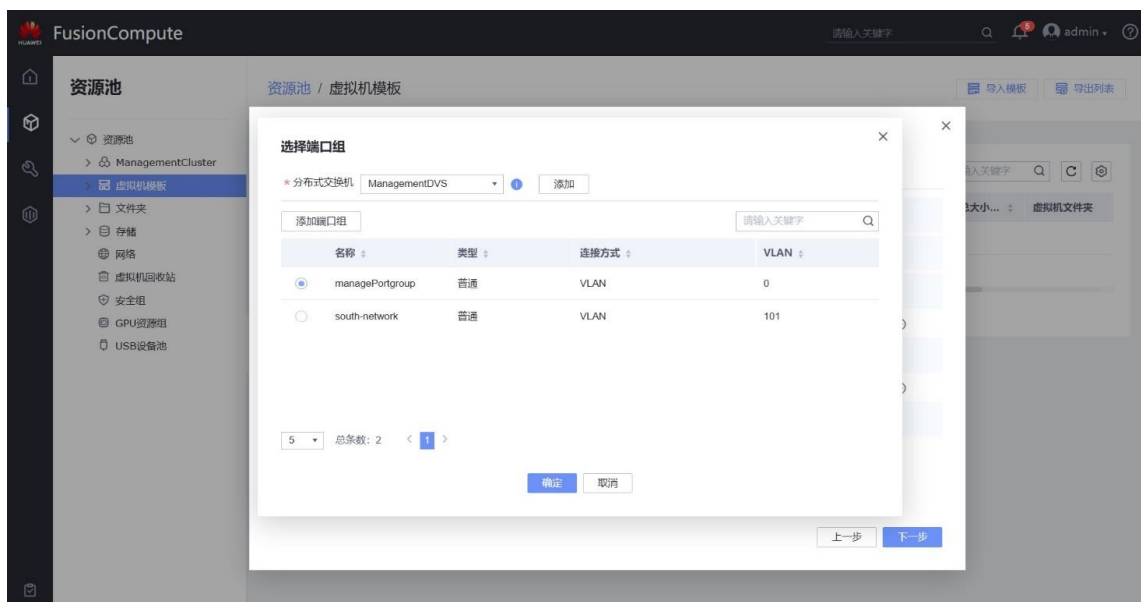
选择一个可用的数据存储，用来给虚拟机分配本地硬盘，然后点击“下一步”：



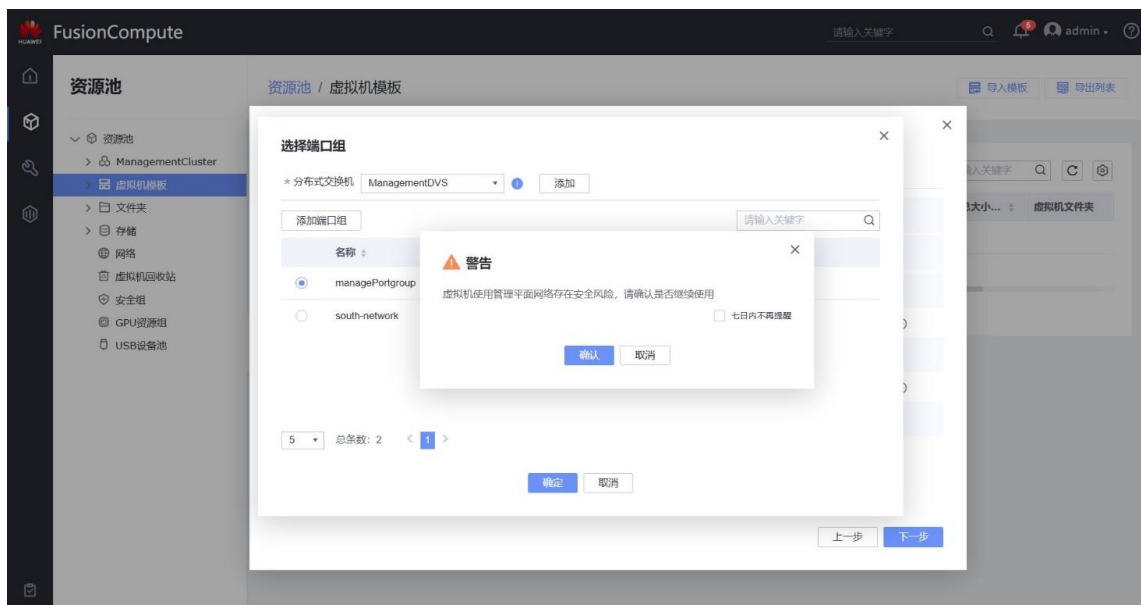
进入模板配置页，请保持其他参数不变，点击“选择”按钮，配置网卡 1 的网络：



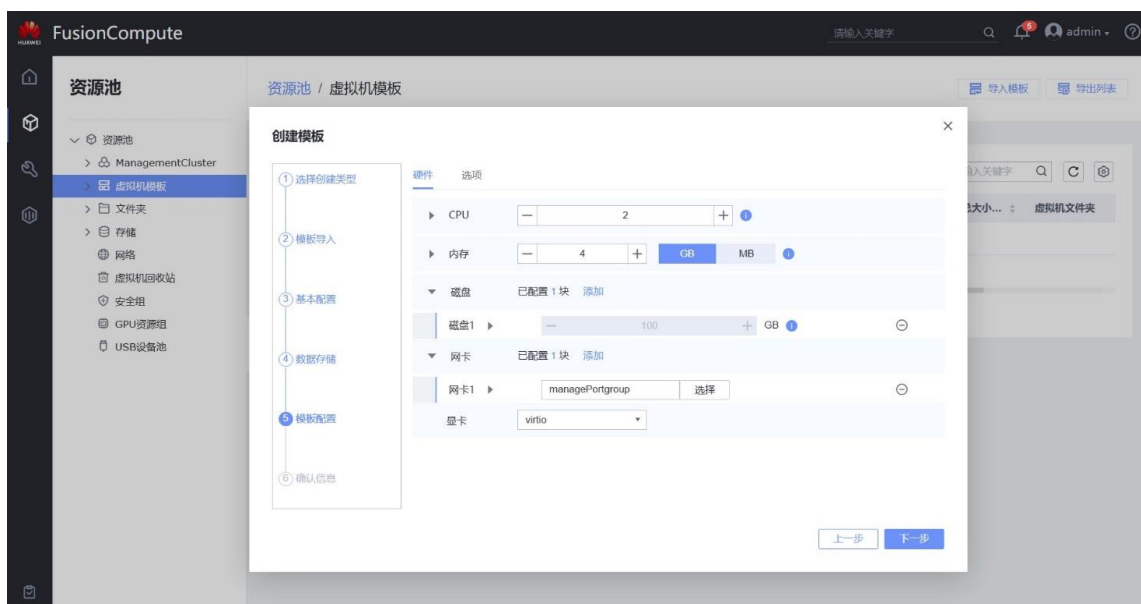
请根据网络规划，选择管理网络，点击“确定”：



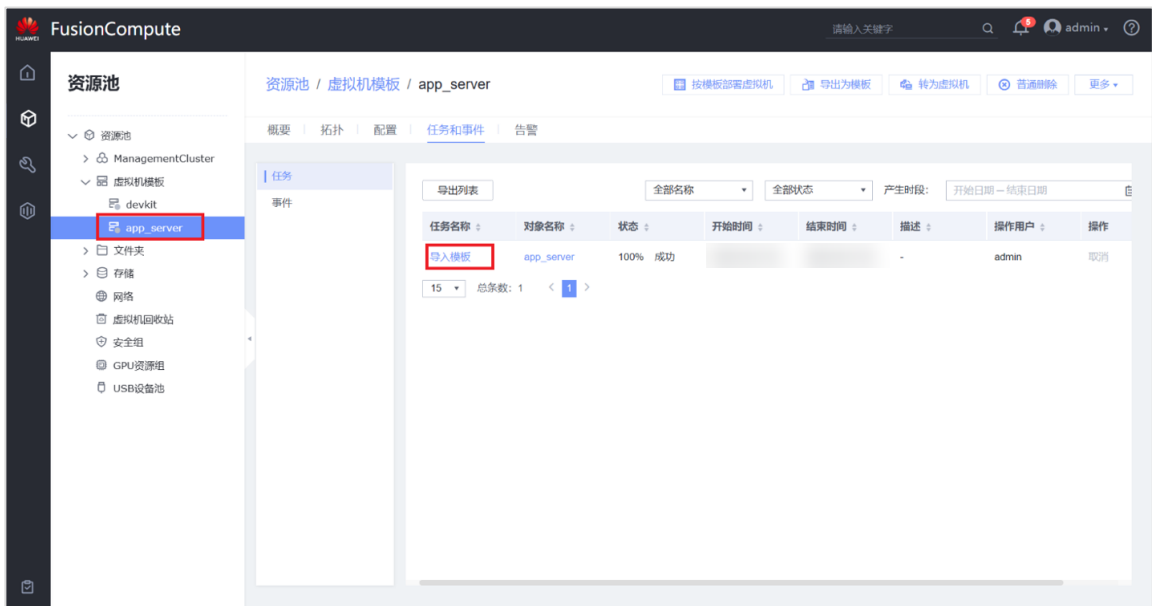
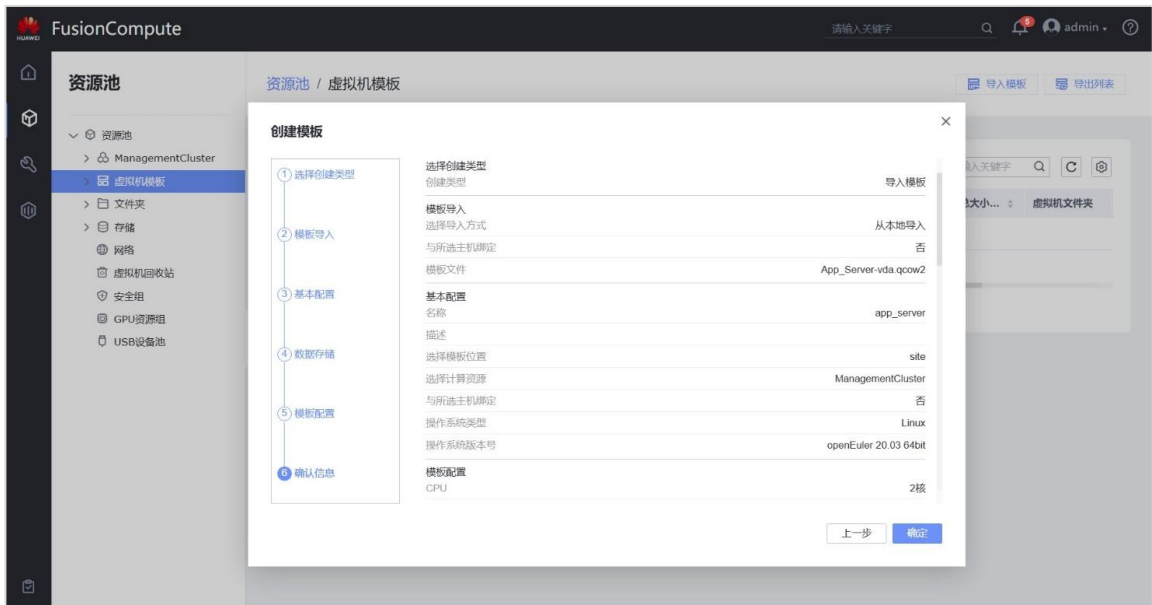
如果弹出以下警告，请点击“确认”继续：

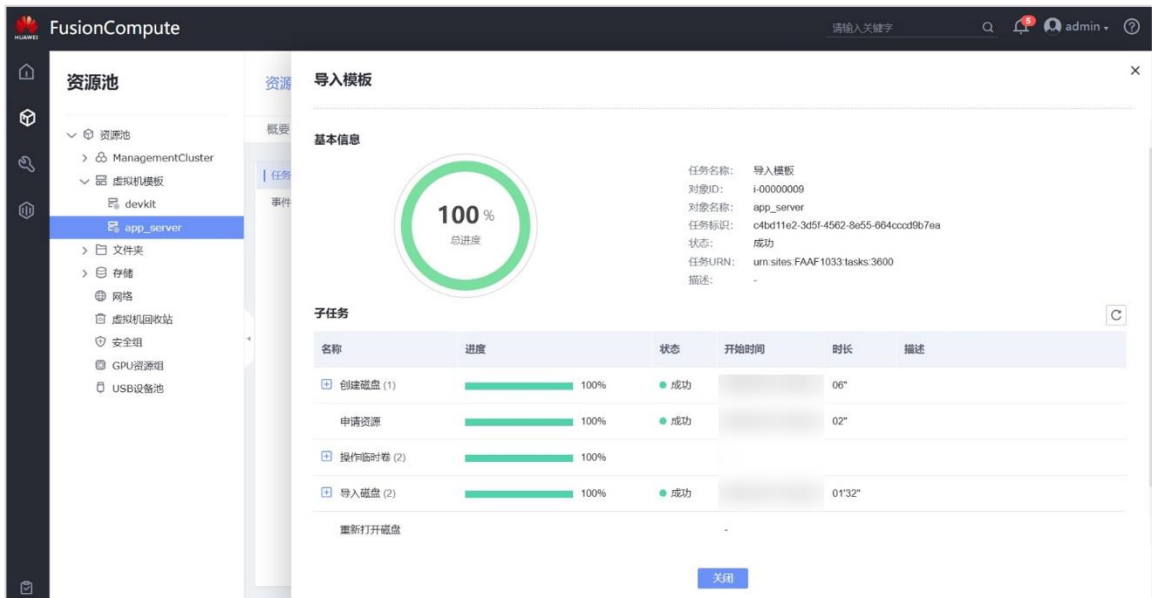


选择好网卡的网络配置，点击“下一步”继续：



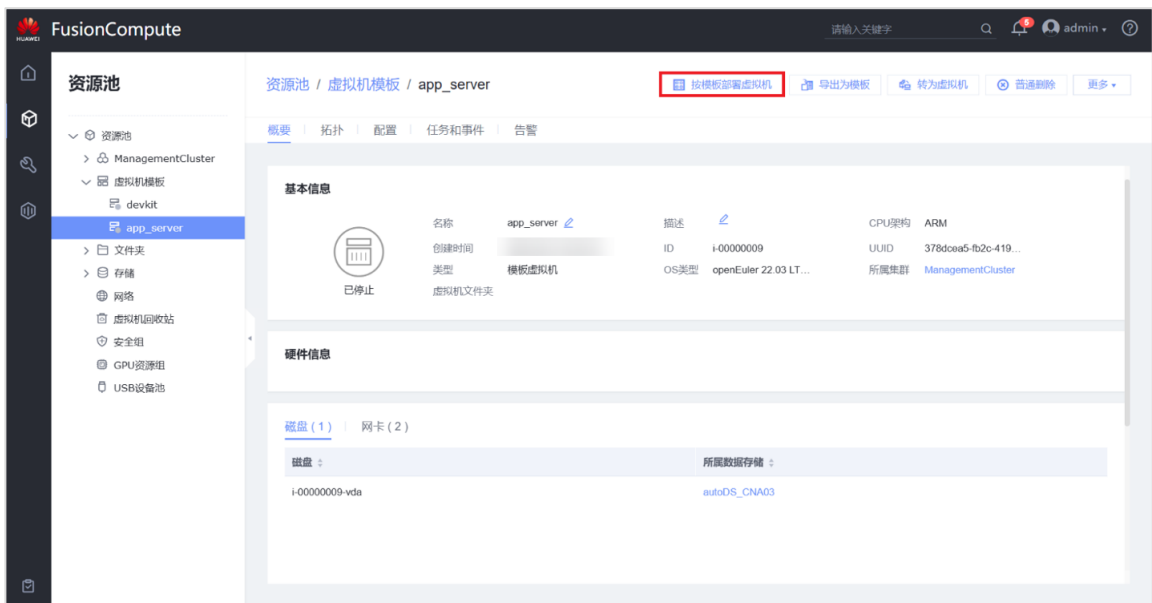
最后，确认信息无误后，点击“确定”按钮：



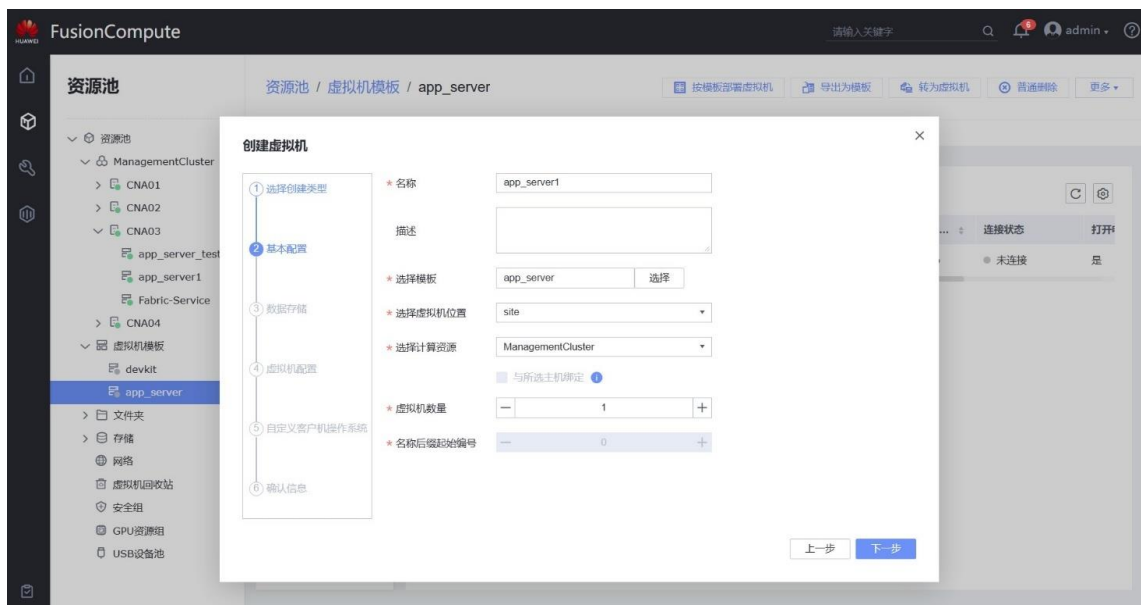


步骤 2 通过虚拟机模板创建虚拟机

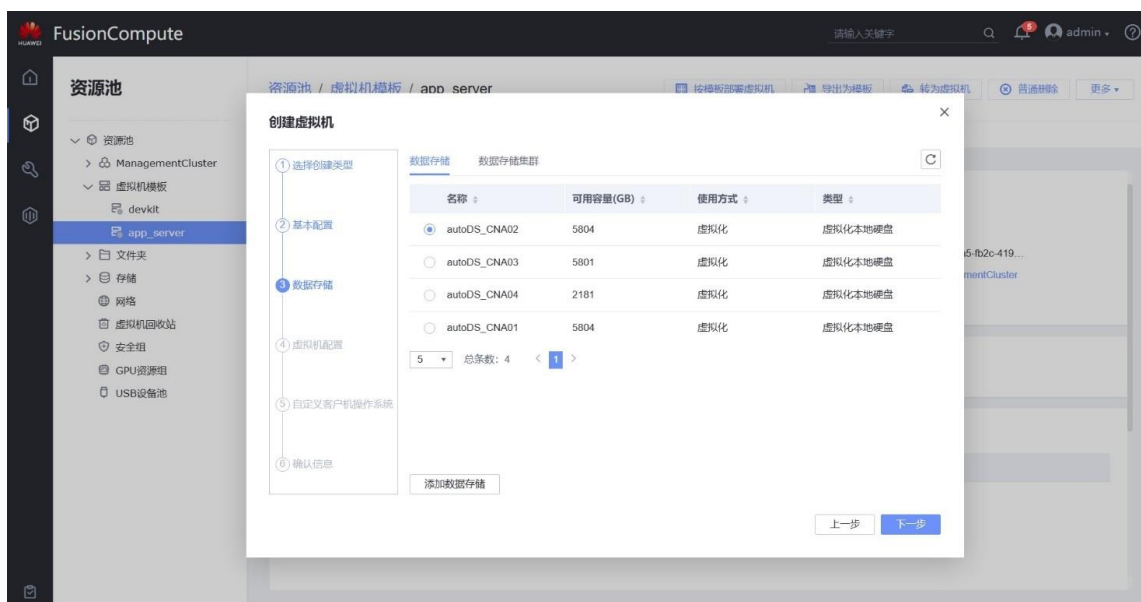
虚拟机模板导入成功以后，就可以通过模板创建虚拟机。在虚拟机模板资源的下拉列表里选择“app_server”，然后在模板信息的页面点击按钮“按模板部署虚拟机”：



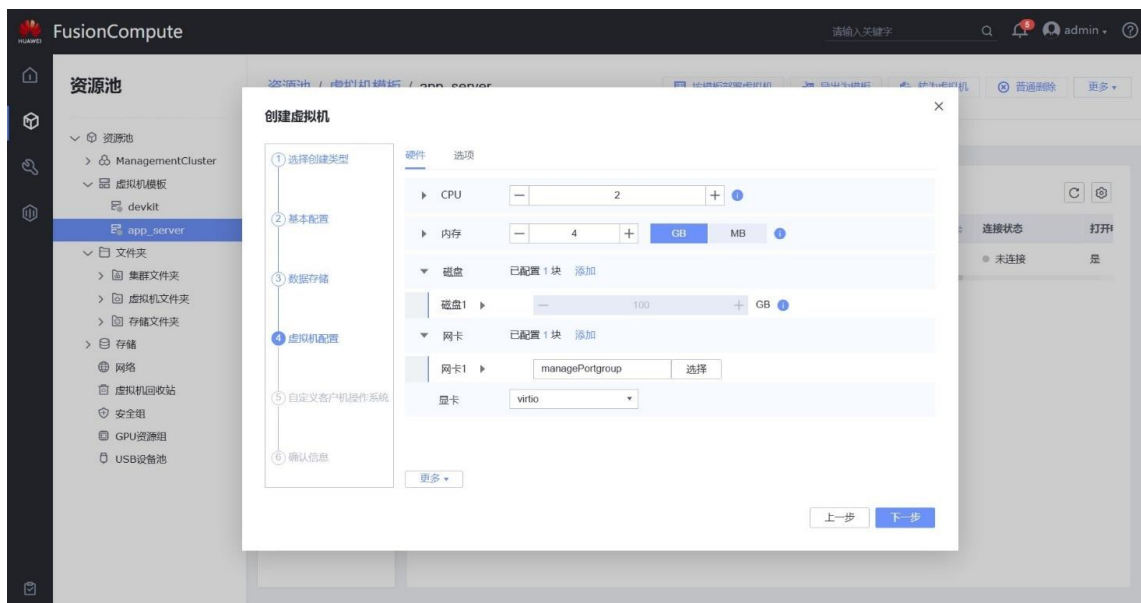
在弹出的创建虚拟机页面中，虚拟机名称根据实际需要进行修改，此处命名虚拟机名称为“app_server1”，其他项可以保持不变，然后点击“下一步”：



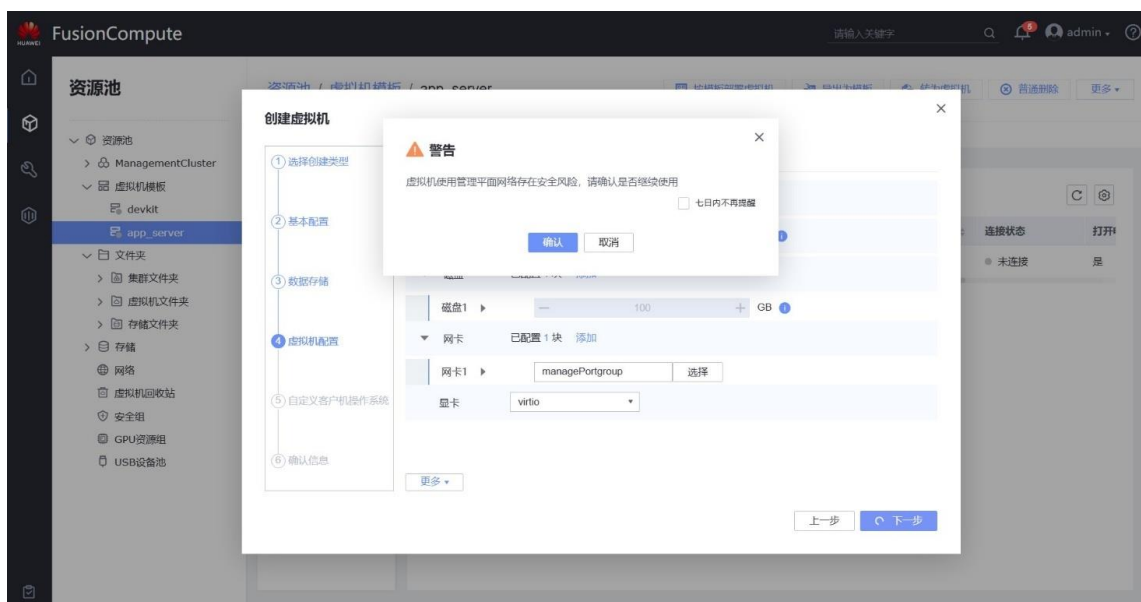
选择可用的数据存储，然后点击“下一步”继续：



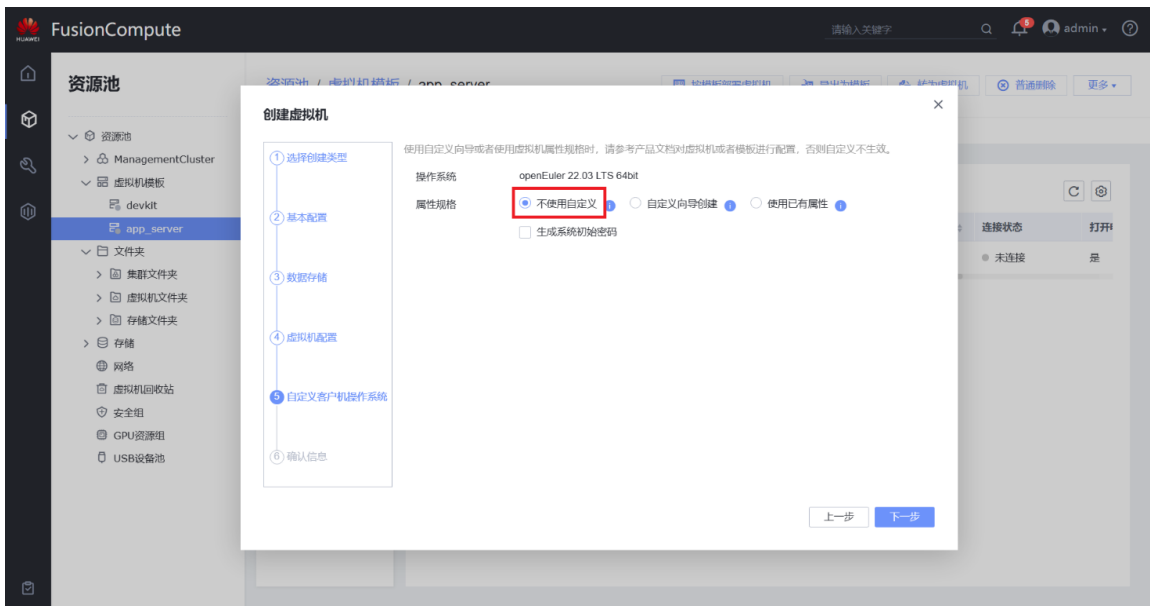
在虚拟机配置页面，由于模板中已经配置好了参数，因此此处可以直接点击“下一步”继续：



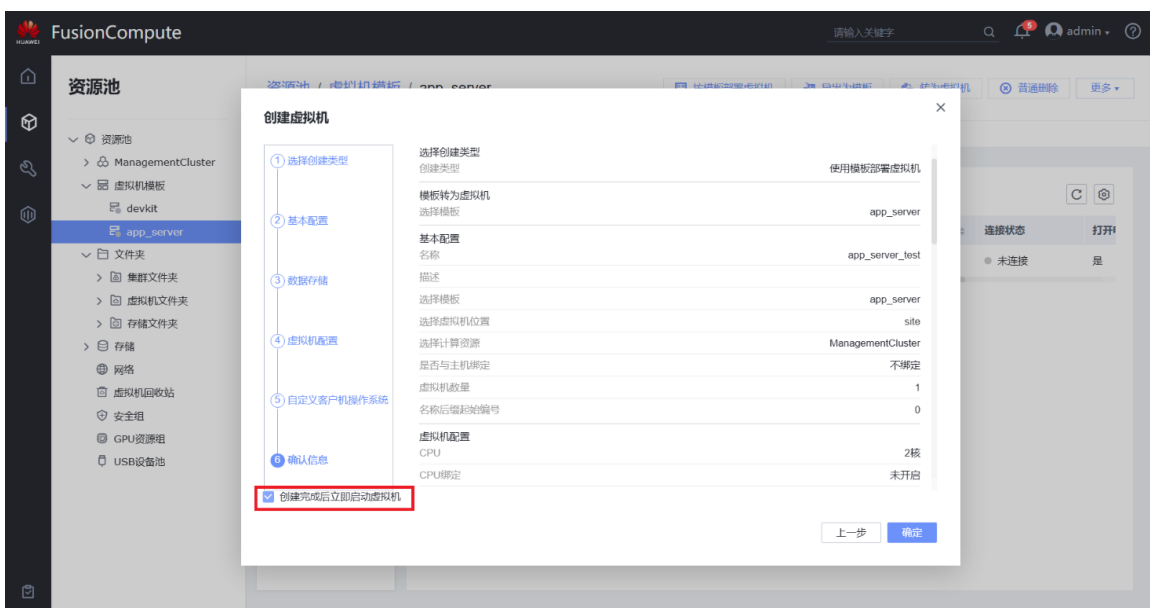
如果弹出以下警告信息，请点击“确认”按钮继续：



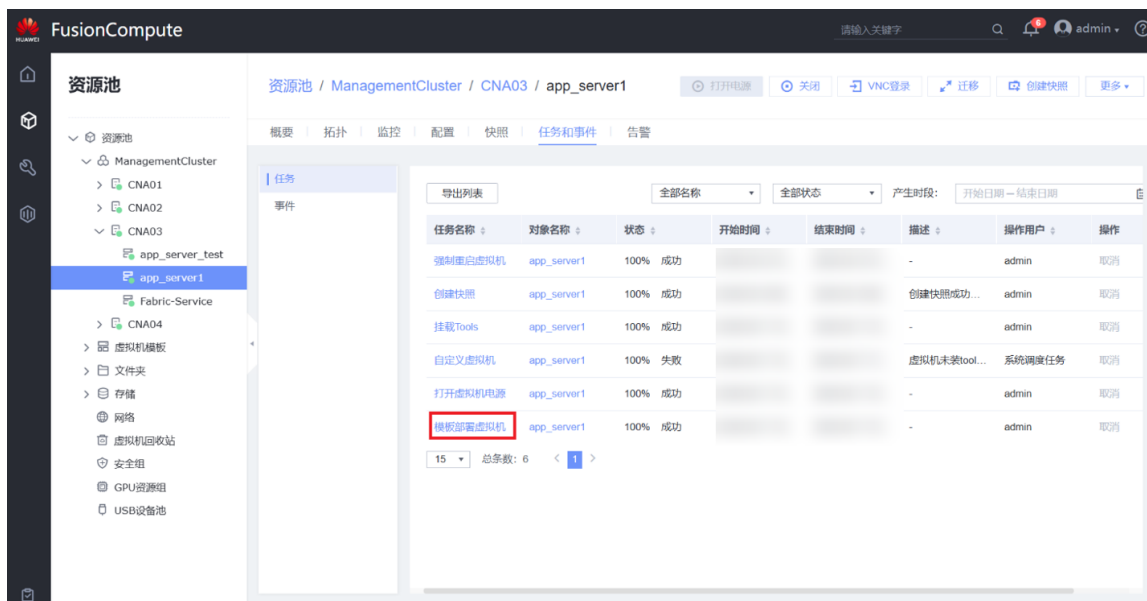
在自定义客户操作系统页面，选择选项“不使用自定义”，然后点击“下一步”继续：



确认创建虚拟机的信息，同时勾选“创建完成后立即启动虚拟机”选项，无误后点击“确认”按钮，此时将成功创建任务：



此时在资源池的下拉列表中可以找到正在创建的虚拟机“app_server1”，选择该虚拟机，在“任务和事件”标签页面可以看到“模板部署虚拟机”任务的状态：

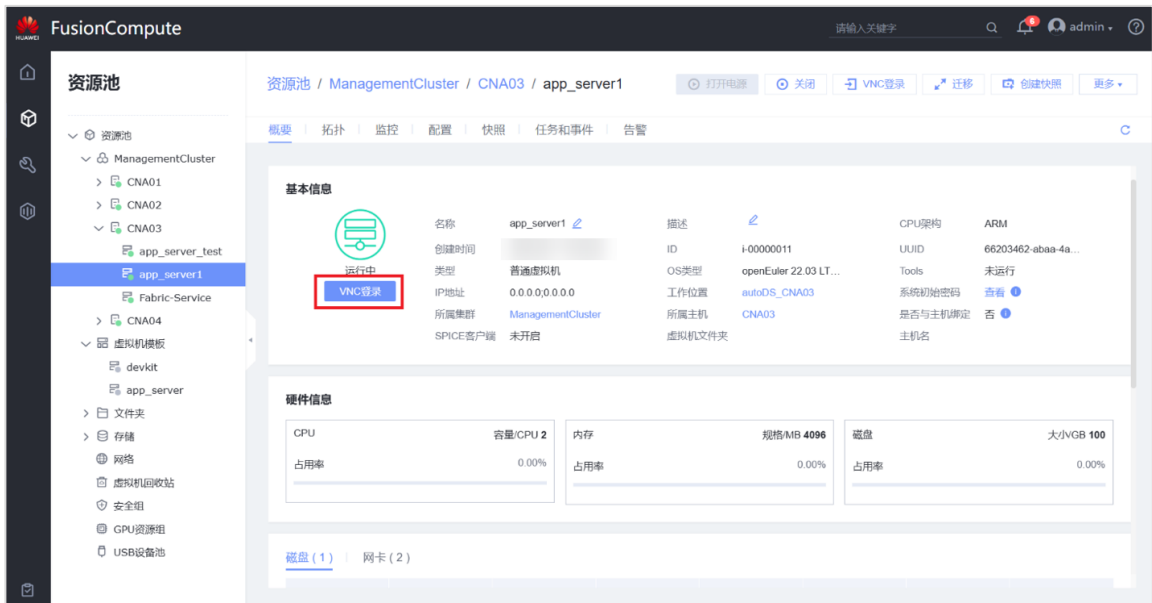


点击“模板部署虚拟机”任务可以查看详细进度和状态，此时等待虚拟机创建成功：

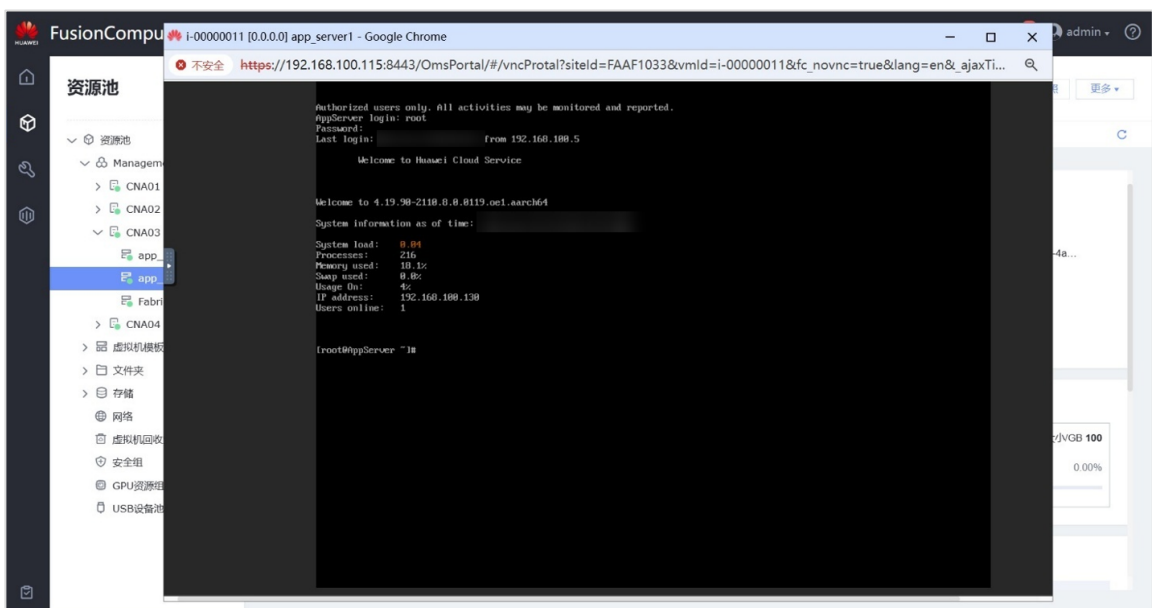


步骤 3 配置静态 IP 地址，通过 SSH 登陆

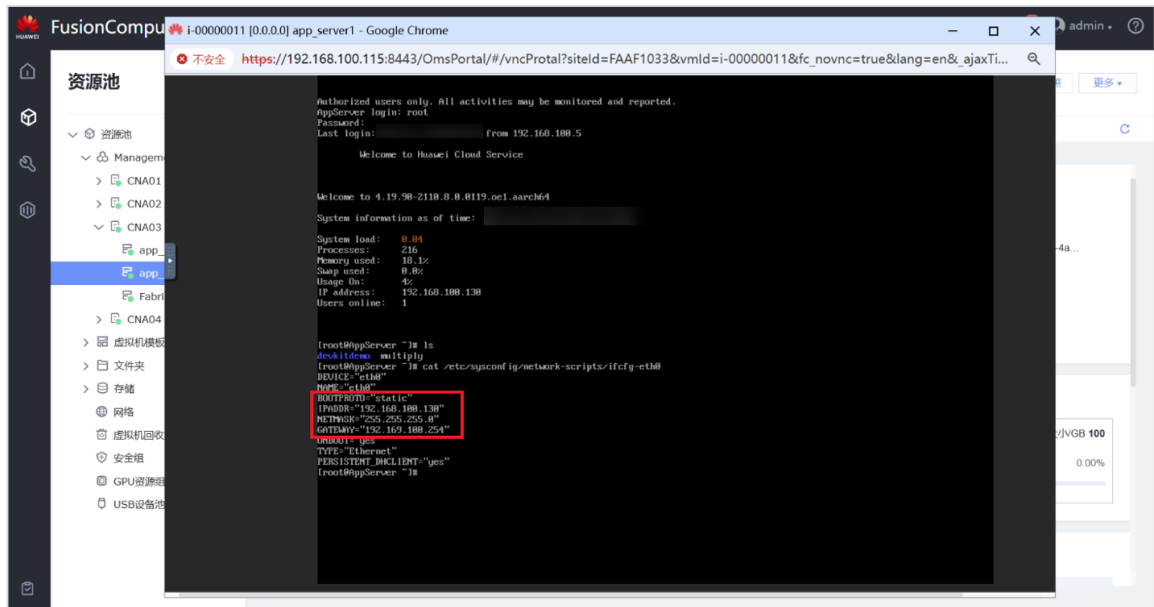
当虚拟机创建成功并启动后，此时虚拟机还没有配置 IP 信息，无法通过 ssh 工具登陆，首先需要通过 VNC 登陆，配置静态 IP 信息。在资源池列表中找到相应的虚拟机，选择后，点击“概要”标签页，然后点击“VNC 登陆”按钮：



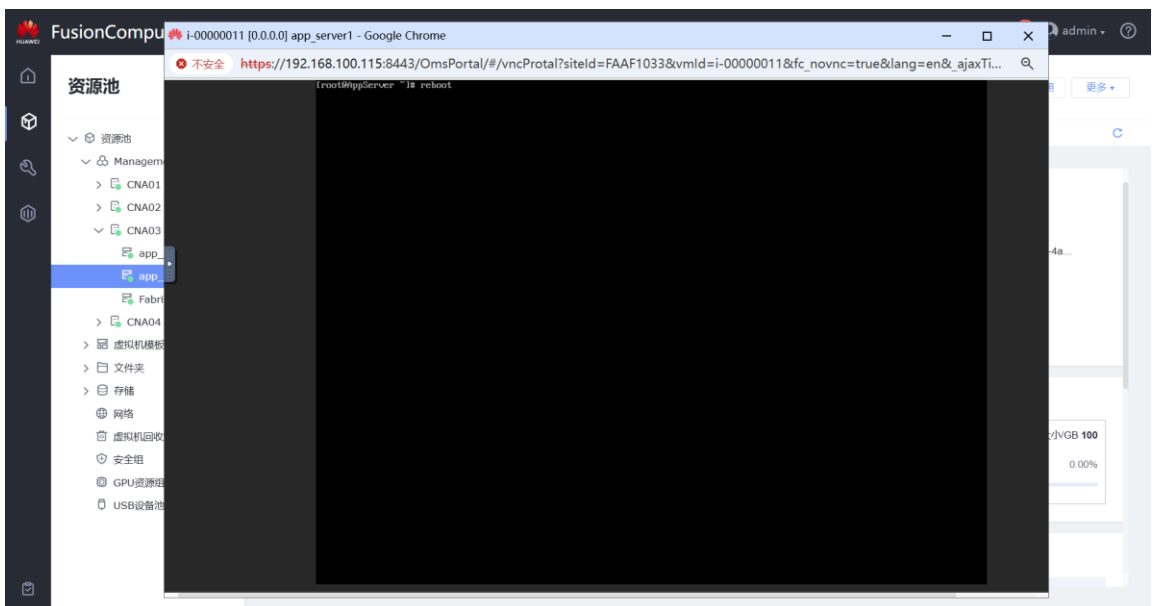
当看到“AppServer login”的提示后，输入用户名：root，默认密码：Huawei@123 登陆到虚拟机：



登陆到虚拟机后，然后参考下图，修改网络配置文件“/etc/sysconfig/network-scripts/ifcfg-eth0”，将参数：BOOTPROTO 改成 static，添加 IPADDR，NETMASK，GATEWAY，这些参数请根据实际实验室的网络规划进行填写，下图只是作为参考。修改完成后保存。



重启虚拟机：



等待重启完成后，在本地电脑使用 ping 命令检查 IP 是否能够 ping 通：

```
命令提示符
Microsoft Windows [版本 10.0.19045.2965]
(c) Microsoft Corporation。保留所有权利。

C:\Users\lyqtest>ping 192.168.100.130

正在 Ping 192.168.100.130 具有 32 字节的数据:
来自 192.168.100.130 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.100.130 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.100.130 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.100.130 的回复: 字节=32 时间<1ms TTL=64

192.168.100.130 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 0ms, 最长 = 0ms, 平均 = 0ms

C:\Users\lyqtest>
```

如果网络配置成功，则可以通过 ssh 工具 putty 进行登陆，用户密码和 VNC 登陆时相同：

```
192.168.100.130 - PuTTY

login as: root
Pre-authentication banner message from server:
|
| Authorized users only. All activities may be monitored and reported.
| End of banner message from server
root@192.168.100.130's password: 
```

登陆成功：

root@AppServer:~

root@192.168.100.130's password:

Welcome to Huawei Cloud Service

Last login: 2025-10-27 10:00:00

Welcome to 4.19.90-2110.8.0.0119.oel.aarch64

System information as of time: 2025-10-27 10:00:00

System load: 0.00
Processes: 211
Memory used: 20.6%
Swap used: 0.0%
Usage On: 4%
IP address: 192.168.100.130
Users online: 2

[root@AppServer ~]# ls

devkitdemo multiply

[root@AppServer ~]#

目录

前 言	3
简介	3
内容描述	3
读者知识背景	3
实验环境说明	3
准备实验环境	4
1 常用性能观测工具实践	22
1.1 iostat 命令实践	22
1.2 Sar 命令实践	23
1.2.1 Sar 命令收集 CPU 信息	23
1.2.2 Sar 命令收集内存信息	24
1.2.3 Sar 命令收集网络信息	25
2 鲲鹏应用使能套件 BoostKit 实践	27
2.1 系统库使用实践	27
2.2 数学库使用实践	30
2.3 KAE 使用实践	34
3 鲲鹏 DevKit 实践	37
3.1 安装鲲鹏 DevKit	37
3.2 鲲鹏 DevKit 系统性能分析实践（1）	42
3.3 鲲鹏 DevKit 系统性能分析实践（2）	52

1

常用性能观测工具实践

1.1 iostat 命令实践

实践描述：本实验在同一主机上实现，因此需要同时打开两个终端，其中一个终端使用 dd 命令模拟数据的写入，另外一个终端使用 iostat 命令观测对应磁盘的 IO 数据。

本实践仅提供观测方式，得出的数据会由于实际硬件性能的差异而与实践中的数据有差异。

步骤 1 准备工作

在终端 1 中使用以下命令模拟数据的写入：

```
dd if=/dev/zero of=/home/test bs=8k count=1024000 oflag=direct
```

```
[root@node01 ~]# dd if=/dev/zero of=/home/test bs=8k count=1024000 oflag=direct
212404+0 records in
212404+0 records out
1740013568 bytes (1.7 GB, 1.6 GiB) copied, 174.37 s, 10.0 MB/s
```

步骤 2 抓取数据

在另外一个终端使用以下命令进行 IO 数据观测：

```
iostat -d 1 /dev/sda2
```

```
[root@node01 ~]# iostat -d 1 /dev/sda2
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      08/08/2024      _x86_64_      (4 CPU)

Device            tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
sda2              102.39      225.03      15500.32      0.00      124277      8560364      0

Device            tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
sda2             1235.64       0.00      9885.15      0.00         0      9984         0

Device            tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
sda2             1135.00       0.00      9080.00      0.00         0      9080         0

Device            tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
sda2             1144.00       0.00      9152.00      0.00         0      9152         0

Device            tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
sda2             1209.00       0.00      9692.00      0.00         0      9692         0

Device            tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
sda2             1095.00       0.00      8760.00      0.00         0      8760         0
```

步骤 3 分析数据

取任意抓取到的数据可使用公式 $(\text{kB_read} + \text{kB_wrtn}) / \text{tps}$ 计算分区 sda 的 IOPS 为：

$(0+8760)/1095=8$ ，单位为 kB，由于当前任务只有 dd，因此 IOPS 刚好等于步骤 1 命令中的 bs 对应的值

1.2 Sar 命令实践

1.2.1 Sar 命令收集 CPU 信息

实践描述：使用 sar 命令收集 CPU 信息，先收集全部 CPU 信息，然后再依次收集每个 CPU 核心信息，最后将收集到的信息保存在文件中。

步骤 1 收集全部 CPU 信息

使用以下命令收集全部 CPU 信息，并间隔 2 秒收集一次，共收集 5 次：

```
sar -u 2 5
```

```
[root@node01 ~]# sar -u 2 5
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)

04:53:43 PM    CPU      %user      %nice    %system    %iowait    %steal     %idle
04:53:45 PM    all        0.00        0.00        0.13        0.00        0.00      99.87
04:53:47 PM    all        0.00        0.00        0.25        0.00        0.00      99.75
04:53:49 PM    all        0.00        0.00        0.00        0.00        0.00     100.00
04:53:51 PM    all        0.00        0.00        0.13        0.00        0.00      99.87
04:53:53 PM    all        0.00        0.00        0.13        0.00        0.00      99.87
Average:        all        0.00        0.00        0.13        0.00        0.00      99.87
```

从收集到的信息可得出，当前主机有 4 个 CPU，且暂时无任何负载，CPU 空闲率都近 100%

步骤 2 收集每个 CPU 核心信息

使用以下命令收集 0 号 CPU 信息，间隔时间为 2 秒，共收集 5 次：

```
sar -P 0 1 5
```

```
[root@node01 ~]# sar -P 0 1 5
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)

05:16:13 PM    CPU      %user      %nice    %system    %iowait    %steal     %idle
05:16:14 PM     0        0.00        0.00        0.00        0.00        0.00     100.00
05:16:15 PM     0        0.00        0.00        0.00        0.00        0.00     100.00
05:16:16 PM     0        0.00        0.00        0.00        0.00        0.00     100.00
05:16:17 PM     0        0.00        0.00        0.00        0.00        0.00     100.00
05:16:18 PM     0        0.00        0.00        0.00        0.00        0.00     100.00
Average:        0        0.00        0.00        0.00        0.00        0.00     100.00
```

使用相同命令收集其他 CPU 信息，如下图所示：

```
[root@node01 ~]# sar -P 1 1 5
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)

05:17:52 PM    CPU    %user    %nice    %system    %iowait    %steal    %idle
05:17:53 PM      1      0.00      0.00      0.00      0.00      0.00    100.00
05:17:54 PM      1      0.00      0.00      0.00      0.00      0.00    100.00
05:17:55 PM      1      0.00      0.00      0.00      0.00      0.00    100.00
05:17:56 PM      1      0.00      0.00      0.00      0.00      0.00    100.00
05:17:57 PM      1      0.00      0.00      0.00      0.00      0.00    100.00
Average:        1      0.00      0.00      0.00      0.00      0.00    100.00

[root@node01 ~]# sar -P 2 1 5
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)

05:17:59 PM    CPU    %user    %nice    %system    %iowait    %steal    %idle
05:18:00 PM      2      0.00      0.00      1.00      0.00      0.00     99.00
05:18:01 PM      2      1.09      7.61      2.17      0.00      0.00     89.13
05:18:02 PM      2      0.00      0.00      0.00      0.00      0.00    100.00
05:18:03 PM      2      0.00      0.00      0.00      0.00      0.00    100.00
05:18:04 PM      2      0.00      0.00      0.00      0.00      0.00    100.00
Average:        2      0.20     1.42      0.61      0.00      0.00     97.77

[root@node01 ~]# sar -P 3 1 5
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)

05:18:08 PM    CPU    %user    %nice    %system    %iowait    %steal    %idle
05:18:09 PM      3      0.00      0.00      0.00      0.00      0.00    100.00
05:18:10 PM      3      0.00      0.00      0.00      0.00      0.00    100.00
05:18:11 PM      3      0.00      0.00      2.04      0.00      0.00     97.96
05:18:12 PM      3      0.00      0.00      0.00      0.00      0.00    100.00
05:18:13 PM      3      0.00      0.00      0.00      0.00      0.00    100.00
Average:        3      0.00      0.00      0.40      0.00      0.00     99.60
```

步骤 3 将收集到的信息保存入文件中

使用以下命令将步骤 1 中收集到的信息保存到名称为 cpu 的文件中：

```
sar -u -o cpu 2 5
```

```
[root@node01 ~]# sar -u -o cpu 2 5
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)

05:21:14 PM    CPU    %user    %nice    %system    %iowait    %steal    %idle
05:21:16 PM    all      0.00      0.00      0.38      0.00      0.00     99.62
05:21:18 PM    all      0.00      0.00      0.13      0.00      0.00     99.87
05:21:20 PM    all      0.00      0.00      0.00      0.00      0.00    100.00
05:21:22 PM    all      0.00      0.00      0.38      0.25      0.00     99.37
05:21:24 PM    all      0.00      0.00      0.00      0.00      0.00    100.00
Average:      all      0.00      0.00      0.18      0.05      0.00     99.77
```

使用以下命令读取该文件信息：

```
sar -u -f cpu
```

```
[root@node01 ~]# sar -u -f cpu
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)

05:21:14 PM    CPU    %user    %nice    %system    %iowait    %steal    %idle
05:21:16 PM    all      0.00      0.00      0.38      0.00      0.00     99.62
05:21:18 PM    all      0.00      0.00      0.13      0.00      0.00     99.87
05:21:20 PM    all      0.00      0.00      0.00      0.00      0.00    100.00
05:21:22 PM    all      0.00      0.00      0.38      0.25      0.00     99.37
05:21:24 PM    all      0.00      0.00      0.00      0.00      0.00    100.00
Average:      all      0.00      0.00      0.18      0.05      0.00     99.77
```

1.2.2 Sar 命令收集内存信息

实践描述：使用 sar 命令收集当前主机内存利用率、swap 分区利用率。

步骤 1 收集内存利用率信息

使用以下命令收集内存利用率信息，间隔时间为 2，收集 3 次：

```
sar -r 2 3
```

```
[root@node01 ~]# sar -r 1 3
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)
05:37:51 PM kbmemfree  kbavail  kbmempused  %memused  kbbuffers  kbcached  kbcommit  %commit  kbactive  kbinact  kbdirty
05:37:52 PM 7160808 7259532 95304 1.25 17164 297888 260580 2.68 112072 245332 24
05:37:53 PM 7160808 7259532 95304 1.25 17164 297888 260580 2.68 112072 245332 24
05:37:54 PM 7160808 7259532 95304 1.25 17164 297888 260580 2.68 112072 245332 24
Average: 7160808 7259532 95304 1.25 17164 297888 260580 2.68 112072 245332 24
```

步骤 2 收集 swap 分区利用率信息

使用以下命令收集 swap 分区利用率信息，间隔时间为 2，收集 3 次：

```
sar -S 2 3
```

```
[root@node01 ~]# sar -S 2 3
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)
09:05:12 AM kbswpfree kbswpused  %swpused  kbswpcad  %swpcad
09:05:14 AM 2097148 0 0.00 0 0.00
09:05:16 AM 2097148 0 0.00 0 0.00
09:05:18 AM 2097148 0 0.00 0 0.00
Average: 2097148 0 0.00 0 0.00
```

步骤 3 收集内存页状态信息

使用以下命令收集内存页信息，间隔时间为 2，收集 3 次：

```
sar -B 2 3
```

```
[root@node01 ~]# sar -B 2 3
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)
01:42:58 PM pgpgin/s pgpgout/s  fault/s  majflt/s  pgfree/s  pgscank/s  pgscand/s  pgsteal/s  %vmeff
01:43:00 PM 0.00 18.00 8.50 0.00 5.00 0.00 0.00 0.00 0.00
01:43:02 PM 0.00 0.00 0.50 0.00 5.00 0.00 0.00 0.00 0.00
01:43:04 PM 0.00 0.00 0.00 0.00 4.50 0.00 0.00 0.00 0.00
Average: 0.00 6.00 3.00 0.00 4.83 0.00 0.00 0.00 0.00
```

1.2.3 Sar 命令收集网络信息

实践描述：检查当前主机哪个网卡有业务流量，并判断该流量使用哪种网络协议。

步骤 1 抓取网络信息

使用以下命令抓取当前主机所有网络信息：

```
sar -n DEV 2 3
```

```
[root@node01 ~]# sar -n DEV 2 3
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)
02:12:54 PM IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s %ifutil
02:12:56 PM lo 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
02:12:56 PM enp0s3 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
02:12:56 PM enp0s8 1.50 1.50 0.10 0.12 0.00 0.00 0.00 0.00
02:12:56 PM enp0s9 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
02:12:56 PM IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s %ifutil
02:12:58 PM lo 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
02:12:58 PM enp0s3 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
02:12:58 PM enp0s8 1.50 1.50 0.10 0.14 0.00 0.00 0.00 0.00
02:12:58 PM enp0s9 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
02:12:58 PM IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s %ifutil
02:13:00 PM lo 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
02:13:00 PM enp0s3 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
02:13:00 PM enp0s8 1.50 1.50 0.10 0.13 0.00 0.00 0.00 0.00
02:13:00 PM enp0s9 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Average: IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s %ifutil
Average: lo 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Average: enp0s3 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Average: enp0s8 1.50 1.50 0.10 0.13 0.00 0.00 0.00 0.00
Average: enp0s9 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

从抓取到的信息得出，仅有 enp0s3 中有流量存在。

步骤 2 抓取网络协议信息

使用以下命令分别抓取 TCP 和 UDP 网络协议流量数量：

```
sar -n TCP 2 3
sar -n UDP 2 3
```

```

Average:
[root@node01 ~]# sar -n TCP 2 3
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)

02:25:32 PM active/s passive/s      iseg/s      oseg/s
02:25:34 PM      0.00      0.00      0.50      0.50
02:25:36 PM      0.00      0.00      0.50      1.50
02:25:38 PM      0.00      0.00      0.50      0.50
Average:      0.00      0.00      0.50      0.83
[root@node01 ~]# sar -n UDP 2 3
Linux 5.10.0-182.0.0.95.oe2203sp3.x86_64 (node01)      _x86_64_      (4 CPU)

02:25:53 PM idgm/s      odgm/s      noport/s idgmerr/s
02:25:55 PM      0.00      0.00      0.00      0.00
02:25:57 PM      0.00      0.00      0.00      0.00
02:25:59 PM      0.00      0.00      0.00      0.00
Average:      0.00      0.00      0.00      0.00

```

从抓取到的信息可得出，过去的时间中，仅存在 TCP 的流量经过该网口。

2 鲲鹏应用使能套件 BoostKit 实践

2.1 系统库使用实践

实践说明：本实践将在鲲鹏平台的 openEuler 中安装 AVX2KI 库，并在 C 语言开发的应用程序编译时调用该库。本实践请使用“ARM-DevKit-vda.qcow2”该镜像创建的虚拟机，由于该镜像中除了 AVX2KI 库，其他相关的库已经下载到了目录/root，因此可以跳过下载环节，直接进行操作：

```
[root@DevKit ~]# ls
devkit
kae-1.3.12-1.openEuler20.03LTS_SP1.aarch64
kae-1.3.12-1.openEuler20.03LTS_SP1.aarch64.rar
kml-2.4.0-1.aarch64.rpm
kml-2.4.0.aarch64.deb
KML_2.4.0.zip
'Kunpeng BoostKit User License Agreement 2.0.txt'
lapack_adapt
lapack-v3.12.0-master
master.zip
test.sh
tmp
'鲲鹏应用使能套件BoostKit用户许可协议 2.0.txt'
```

步骤 1 下载 AVX2KI 库

在鲲鹏社区的“资源下载”页面（<https://www.hikunpeng.com/zh/developer/software>）搜索 AVX2KI，如下图所示：



点击搜索到的结果，进入对应页面，并找到“立即下载”，如下图所示：



点击，并按照提示下载到该软件包，并将下载到的软件包上传到鲲鹏架构主机中。

步骤 2 安装 AVX2KI 库

使用以下命令在鲲鹏架构主机解压下载到的软件包：

```
unzip BoostKit-ksl_2.3.1.zip
```

```
[root@ecs-a5c9 ~]# unzip BoostKit-ksl_2.3.1.zip
Archive: BoostKit-ksl_2.3.1.zip
  inflating: boostkit-ksl-2.3.1-1.aarch64.rpm
  inflating: Kunpeng BoostKit User License Agreement 2.0.txt
  inflating: 鲲鹏应用使能套件BoostKit用户许可协议 2.0.txt
```

解压完成后使用以下命令安装对应加速库：

```
rpm -ivh boostkit-ksl-2.3.1-1.aarch64.rpm
```

```
[root@ecs-a5c9 ~]# rpm -ivh boostkit-ksl-2.3.1-1.aarch64.rpm
Verifying... ##### [100%]
Preparing... ##### [100%]
Updating / installing...
 1:boostkit-ksl-2.3.1-1 ##### [100%]
```

安装完成后，使用以下命令检查是否安装成功，下图为安装成功后的输出：


```
ll /usr/local/ksl
```

```
[root@ecs-a5c9 ~]# ll /usr/local/ksl
total 12K
drwxr-xr-x 2 root root 4.0K Aug 12 15:54 bin
drwxr-xr-x 3 root root 4.0K Aug 12 15:54 include
drwxr-xr-x 2 root root 4.0K Aug 12 15:54 lib
```

步骤 3 调用 AVX2KI 库

使用命令“source /etc/profile”重新加载环境变量，然后使用以下命令检查环境变量是否生效，下图为生效后正确的输出：

```
env | grep LD_LIBRARY_PATH
```

```
[root@ecs-a5c9 ~]# env | grep LD_LIBRARY_PATH
LD_LIBRARY_PATH=/usr/local/ksl/lib:
```

编写以下 C 代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "avx2ki.h"

void Avx()
{
    int32_t a[4] = {-5, 13, 4, -20};
    int32_t b[4] = {12, 3, 0, 7};
    int32_t c[4] = {0};
    __m128i t1 = _mm_load_epi32(a);
    __m128i t2 = _mm_load_epi32(b);
    __m128i dst = _mm_add_epi32(t1, t2);
    _mm_store_epi32(c, dst);
    printf("dst: %d %d %d %d\n", c[0], c[1], c[2], c[3]);
}

int main(void) {
    Avx();
    return 0;
}
```

使用以下命令编译代码，并在编译时指定 AVX2KI 库的路径：

```
gcc test.c -L /usr/local/ksl/lib -lavx2ki -I /usr/local/ksl/include -lm -o code
```

编译完成后，运行编译后的代码，如下图所示：

```
[root@ecs-a5c9 ~]# ./code
dst: 7 16 4 -13
```


2.2 数学库使用实践

实践说明：本实践将在鲲鹏平台的 openEuler 中安装 KML_LAPACK 库，并在 C 语言开发的应用程序编译时调用该库。

步骤 1 安装 KML_LAPACK 库

在鲲鹏社区的“资源下载”页面下载 GCC 版本数学库，下载地址可参考实验 2.1 的步骤 1 查找，下载页面如下：



鲲鹏应用使能套件BoostKit > 基础加速软件包 > 数学库

系统库 压缩 加解密 媒体 **数学库**

数学库

鲲鹏数学库（Kunpeng Math Library，以下简称KML）提供了基于鲲鹏平台优化的高性能数学函数，所有接口实现，部分接口兼容Fortran语言调用，部分提供Java语言封装的接口。

开发语言	C
BoostKit版本号	24.0.RC2
版本类型	候选版本
组件版本号	2.4.0

立即下载

[GCC版本](#) [GCC版本数字签名](#) [毕昇版本](#) [毕昇版本数字签名](#)

[数学库文档](#)

[数学库文档](#)

步骤 2 安装数学库

将下载到的软件包上传至鲲鹏架构的主机中，并参考实验 2.1 中步骤 2 进行安装，具体安装命令如下：

```
unzip KML_2.4.0.zip
rpm -ivh kml-2.4.0-1.aarch64.rpm
```

```
[root@ecs-a5c9 ~]# rpm -ivh kml-2.4.0-1.aarch64.rpm
Verifying... ##### [100%]
Preparing... ##### [100%]
Updating / installing...
 1:kml-2.4.0-1 ##### [100%]
```

安装完成后，使用命令“source /etc/profile”使环境变量生效，并检查环境变量 LD_LIBRARY_PATH 是否包含 KML 的安装路径“/usr/local/kml/lib”，具体操作命令如下：

```
source /etc/profile
env | grep LD_LIBRARY_PATH
```

```
[root@ecs-a5c9 ~]# source /etc/profile

Welcome to 4.19.90-2110.8.0.0119.oel.aarch64

System information as of time:
System load: 0.53
Processes: 141
Memory used: 37.4%
Swap used: 0.0%
Usage on /: 10%
IP address: 10.0.0.54
Users online: 1

[root@ecs-a5c9 ~]# env | grep LD_LIBRARY_PATH
LD_LIBRARY_PATH=/usr/local/knl/lib/kspblas/multi:/usr/local/knl/lib/kspblas/single:/usr/local/knl/lib/kvml/multi:/usr/local/knl/lib/kvml/single:/usr/local/knl/lib/kblas/pthread:/usr/local/knl/lib/kblas/locking:/usr/local/knl/lib/kblas/omp:/usr/local/knl/lib/kblas/nolocking:/usr/local/knl/lib:/usr/local/ksl/lib:/usr/local/ksl/lib:
```

步骤 3 下载 Netlib LAPACK v3.12.0 的源代码包

KML_LAPACK 依赖于开源的 Netlib LAPACK 以提供完整的 LAPACK 接口功能。因此安装 KML_LAPACK 后需自行获取开源的 Netlib LAPACK v3.12.0 的源代码包。源码包可从以下链接拉取：<https://gitee.com/yftyxa/lapack-v3.12.0.git>，下载完成后使用命令“unzip master.zip”解压该源码包，解压完成后，在目录中生成下图中对应的文件：

```
[root@ecs-a5c9 ~]# ls
lapack-v3.12.0-master master.zip
[root@ecs-a5c9 ~]# cd lapack-v3.12.0-master/
[root@ecs-a5c9 lapack-v3.12.0-master]# ls
lapack-3.12.0.tar.gz README.md
[root@ecs-a5c9 lapack-v3.12.0-master]#
```

请记录 lapack-3.12.0.tar.gz 所在的路径，如本实践中所在的路径为：/root/lapack-v3.12.0-master。

```
[root@ecs-a5c9 lapack-v3.12.0-master]# pwd
/root/lapack-v3.12.0-master
```

步骤 4 生成动态库

编辑以下 shell 脚本，注意脚本中的加粗部分（第三行）为上步中获取到的路径：

```
set -eE

echo "LAPACK_SRC_DIR      ${LAPACK_SRC_DIR:-<undefined>}"
echo "LAPACK_TGZ           /${LAPACK_TGZ:=/root/lapack-v3.12.0/lapack-3.12.0.tar.gz}"
echo "LIBKLAPACK_A          ${LIBKLAPACK_A:=/usr/local/knl/lib/libklapack.a}"
echo "LIBKSERVICE_A        ${LIBKSERVICE_A:=${LIBKLAPACK_A}/klapack/kservice}"
echo "ADAPT_DIR              ${ADAPT_DIR:=./lapack_adapt}"
echo "CMAKE_BUILD_TYPE       ${CMAKE_BUILD_TYPE:=Release}"
echo "LIBLAPACK_ADAPT_A       ${LIBLAPACK_ADAPT_A:=liblapack_adapt.a}"
echo "LIBKLAPACK_FULL_SO     ${LIBKLAPACK_FULL_SO:=libklapack_full.so}"
echo "CC                     ${CC:=gcc}"
echo "FC                     ${FC:=gfortran}"

mkdir -p ${ADAPT_DIR}
cd ${ADAPT_DIR}

# build netlib lapack
if [ ! -r "${LAPACK_SRC_DIR}/CMakeLists.txt" ]; then
    mkdir -p netlib
    ( cd netlib ; tar xzpf ${LAPACK_TGZ} )
    LAPACK_SRC_DIR=$(cd netlib/* ; pwd)
```

```

fi

mkdir -p build
cmake_flags=(
    -DCMAKE_BUILD_TYPE=${CMAKE_BUILD_TYPE}
    -DCMAKE_POSITION_INDEPENDENT_CODE=ON
    -DCMAKE_C_COMPILER=${CC}
    -DCMAKE_Fortran_COMPILER=${FC}
    -DCMAKE_RULE_MESSAGES=off
    -DBUILD_DEPRECATED=on
    -DBUILD_TESTING=off
)
( cd build ; cmake ${cmake_flags[*]} ${LAPACK_SRC_DIR} )
( cd build ; make -j )

cp build/lib/liblapack.a ${LIBLAPACK_ADAPT_A}

# get symbols defined both in klapack and netlib lapack
nm -g ${LIBLAPACK_ADAPT_A} | grep 'T ' | grep -oP '\Kw+(?=_$)' | sort | uniq > netlib.sym
nm -g ${LIBKLAPACK_A} | grep 'T ' | grep -oP '\Kw+(?=_$)' | sort | uniq > klapack.sym
comm -12 klapack.sym netlib.sym > comm.sym

# update symbols name of ${LIBLAPACK_ADAPT_A}
while read sym; do
    (
        if ! nm ${LIBLAPACK_ADAPT_A} | grep -qe " T ${sym}_\$"; then
            continue
        fi
        ar x ${LIBLAPACK_ADAPT_A} ${sym}.f.o
        mv ${sym}.f.o ${sym}_netlib.f.o

        objcopy --redefine-sym ${sym}_=${sym}_netlib_ ${sym}_netlib.f.o
    ) &
done < comm.sym
wait
ar d ${LIBLAPACK_ADAPT_A} $(sed -ne 's/\$/./p' comm.sym)
ar d ${LIBLAPACK_ADAPT_A} xerbla.f.o
ar ru ${LIBLAPACK_ADAPT_A} *_netlib.f.o
rm *_netlib.f.o
${FC} -o ${LIBKLAPACK_FULL_SO} -shared -fPIC \
    -Wl,--whole-archive ${LIBKLAPACK_A} ${LIBLAPACK_ADAPT_A} -Wl,--no-whole-archive \
    ${LIBKSERVICE_A} -fopenmp -lpthread -lm

```

为确保动态库编译成功，请使用以下命令安装所需的编译器：

```
yum install -y gcc-gfortran gcc-c++ gcc gdb cmake
```

安装完成后执行这个脚本进行动态库编译，编译完成后，在当前目录生成 lapack_adapt 子目录，并在目录中生成适配的 liblapack_adapt.a，如下图所示：

```
[root@ecs-a5c9 ~]# ll
total 344M
-rw-r--r-- 1 root root 93M Jun 21 15:56 kml-2.4.0-1.aarch64.rpm
-rw-r--r-- 1 root root 76M Jun 21 15:56 kml-2.4.0.aarch64.deb
-rw-r--r-- 1 root root 168M Aug 13 15:05 KML_2.4.0.zip
-rw-r--r-- 1 root root 15K Jun 21 15:56 'Kunpeng BoostKit User License Agreement 2.0.txt'
drwxr-xr-x 4 root root 12K Aug 13 15:07 lapack_adapt
drwxr-xr-x 2 root root 4.0K Aug 13 14:28 lapack-v3.12.0-master

[root@ecs-a5c9 ~]# ll lapack_adapt/
total 46M
drwxr-xr-x 9 root root 4.0K Jun 23 10:19 build
-rw-r--r-- 1 root root 1.6K Jun 23 10:21 comm.sym
-rw-r--r-- 1 root root 2.4K Jun 23 10:21 klapack.sym
-rwxr-xr-x 1 root root 36M Jun 23 10:21 libklapack_full.so
-rw-r--r-- 1 root root 9.9M Jun 23 10:21 liblapack_adapt.a
drwxr-xr-x 3 root root 4.0K Jun 23 10:19 netlib
-rw-r--r-- 1 root root 15K Jun 23 10:21 netlib.sym
```

步骤 5 测试安装效果

首先编写测试代码 app.c, 代码内容具体如下:

```
#include <stdio.h>
#include "klapack.h" // 使用 KML 提供的 LAPACK 接口声明

int main() {
    int m = 4;
    int n = 4;
    int lda = 4;
    int ipiv[4];
    int info = 0;

    double a[] = {
        1.80, 5.25, 1.58, -1.11,
        2.88, -2.95, -2.69, -0.66,
        2.05, -0.95, -2.90, -0.59,
        -0.89, -3.80, -1.04, 0.80
    };

    dgetrf_(&m, &n, a, &lda, ipiv, &info);

    if(info == 0) {
        printf("LU 分解成功, 结果如下: \n");
        for(int i = 0; i < m; i++) {
            for(int j = 0; j < n; j++) {
                printf("%8.4f ", a[j * lda + i]); // 注意列主序输出
            }
            printf("\n");
        }
    } else {
        printf("dgetrf_ 返回 info = %d\n", info);
    }
}
```

```
return 0;
}
```

使用以下命令编译

```
gcc app.c -o app -fopenmp -I /usr/local/kml/include/ -L /usr/local/kml/lib/ -lklapack -L
/root/lapack_adapt -l:liblapack_adapt.a -L /usr/local/kml/lib/kblas/omp/ -lkblas -lgfortran -lm -lkservice
```

编译完成后将生成 app 程序，具体如下：

```
[root@ecs-c1bc test]# ll
total 1.1M
-rwxr-xr-x 1 root root 1.1M Jun 23 10:24 app
-rw-r--r-- 1 root root 758 Jun 23 09:25 app.c
```

使用命令 “./app” 运行该脚本，会返回以下结果：

```
[root@ecs-c1bc test]# ./app
LU 分解成功，结果如下：
 5.2500 -2.9500 -0.9500 -3.8000
 0.3429  3.8914  2.3757  0.4129
 0.3010 -0.4631 -1.5139  0.2948
-0.2114 -0.3299  0.0047  0.1314
```

如果编译成功且能运行成功，这表示该数学库被成功调用。

2.3 KAE 使用实践

步骤 1 安装 OpenSSL（可选）

使用以下命令检查 OpenSSL 是否被安装：

```
openssl version
```

如果没安装，系统会返回 OpenSSL 的版本信息，如下图所示：

```
[root@ecs-a5c9 ~]# openssl version
OpenSSL 1.1.1f 31 Mar 2020
```

如果未安装，请使用 yum 命令安装，且版本不得低于 1.1.1a。

步骤 2 安装 KAE1.0

安装 KAE 时，请选择正确版本，如果 Linux 的内核为 4.19，请安装 KAE1.0，如果为 5.1.0，可选择安装 KAE2.0，本实践中的 Linux 内核版本为 4.19，因此选择安装 KAE1.0，使用以下链接下载对应版本 rpm 包：

https://gitee.com/kunpengcompute/KAE/releases/download/v1.3.12/kae-1.3.12-1.openEuler20.03LTS_SP1.aarch64.rpm

下载完成后，解压下载到的安装包，并将其上传至鲲鹏架构的主机中。

进入 KAE rpm 包所在目录，然后使用以下命令安装全部 rpm 包：

```
rpm -ivh *.rpm
```

```
[root@ecs-a5c9 kae-1.3.12-1.openEuler20.03LTS.SP1.aarch64]# ls
hisi_hpre-1.3.12-1.openEuler20.03LTS.SP1.aarch64.rpm  hisi_zip-1.3.12-1.openEuler20.03LTS.SP1.aarch64.rpm  libwd-1.3.12-1.openEuler20.03LTS.SP1.aarch64.rpm
hisi_rde-1.3.12-1.openEuler20.03LTS.SP1.aarch64.rpm  libkae-1.3.12-1.openEuler20.03LTS.SP1.aarch64.rpm  uacce-1.3.12-1.openEuler20.03LTS.SP1.aarch64.rpm
hisi_sec2-1.3.12-1.openEuler20.03LTS.SP1.aarch64.rpm  libkaezip-1.3.12-1.openEuler20.03LTS.SP1.aarch64.rpm
[root@ecs-a5c9 kae-1.3.12-1.openEuler20.03LTS.SP1.aarch64]# rpm -ivh *.rpm
Verifying...##### [100%]
Preparing...##### [100%]
checking installed modules
uacce modules start to install
Updating / installing...
 1:uacce-1.3.12-1.openEuler20.03LTS_##### [ 13%]
uacce modules installed
 2:libwd-1.3.12-1.openEuler20.03LTS_##### [ 25%]
 3:libkae-1.3.12-1.openEuler20.03LTS_##### [ 38%]
 4:libkaezip-1.3.12-1.openEuler20.03LTS_##### [ 50%]
checking installed modules
hisi_hpre modules start to install
 5:hisi_hpre-1.3.12-1.openEuler20.03LTS_##### [ 63%]
hisi_hpre modules installed
checking installed modules
hisi_rde modules start to install
 6:hisi_rde-1.3.12-1.openEuler20.03LTS_##### [ 75%]
hisi_rde modules installed
checking installed modules
hisi_sec2 modules start to install
 7:hisi_sec2-1.3.12-1.openEuler20.03LTS_##### [ 88%]
hisi_sec2 modules installed
checking installed modules
hisi_zip modules start to install
 8:hisi_zip-1.3.12-1.openEuler20.03LTS_##### [100%]
hisi_zip modules installed
```

安装完成后，使用以下命令检查是否安装成功：

```
rpm -ql uacce
rpm -ql hisi_sec2 hisi_hpre hisi_rde
```

```
[root@ecs-a5c9 ~]# rpm -ql uacce
/lib/modules/4.19.90-2109.1.0.0108.oe1.aarch64/extra/hisi_qm.ko
/lib/modules/4.19.90-2109.1.0.0108.oe1.aarch64/extra/uacce.ko
[root@ecs-a5c9 ~]# rpm -ql hisi_sec2 hisi_hpre hisi_rde
/etc/modprobe.d/hisi_sec2.conf
/lib/modules/4.19.90-2109.1.0.0108.oe1.aarch64/extra/hisi_sec2.ko
/etc/modprobe.d/hisi_hpre.conf
/lib/modules/4.19.90-2109.1.0.0108.oe1.aarch64/extra/hisi_hpre.ko
/etc/modprobe.d/hisi_rde.conf
/lib/modules/4.19.90-2109.1.0.0108.oe1.aarch64/extra/hisi_rde.ko
```

使用以下命令查看在内核模块配置目录中是否生成对应的配置文件：

```
ls -al /etc/modprobe.d/
```

```
[root@ecs-a5c9 ~]# ls -al /etc/modprobe.d/
total 32
drwxr-xr-x. 2 root root 4096 Aug 13 15:44 .
drwxr-xr-x. 85 root root 4096 Aug 13 15:44 ..
-rw-r--r--. 1 root root 166 Mar 24 2020 firewalld-sysctls.conf
-rw-r--r--. 1 root root 45 Jan 12 2024 hisi_hpre.conf
-rw-r--r--. 1 root root 44 Jan 12 2024 hisi_rde.conf
-rw-r--r--. 1 root root 61 Jan 12 2024 hisi_sec2.conf
-rw-r--r--. 1 root root 44 Jan 12 2024 hisi_zip.conf
-rw-r--r--. 1 root root 674 Jul 5 2018 tuned.conf
```

步骤 3 加载加速器驱动到内核中

使用以下命令将加速器驱动加载到内核中：

```
modprobe uacce
modprobe hisi_sec2
modprobe hisi_hpre
modprobe hisi_rde
```

加载完成后，使用以下命令检查是否加载成功：

```
lsmod | grep uacce
```



```
[root@ecs-a5c9 ~]# lsmod | grep uacce
[root@ecs-a5c9 ~]# modprobe uacce
[root@ecs-a5c9 ~]# modprobe hisi_sec2
[root@ecs-a5c9 ~]# modprobe hisi_hpre
[root@ecs-a5c9 ~]# modprobe hisi_rde
[root@ecs-a5c9 ~]# lsmod | grep uacce
uacce                262144  1 hisi_qm
```

确认加载完成后，设置 KAE 的环境变量，具体命令为：

```
export OPENSSL_ENGINES=/usr/local/lib/engines-1.1
```

步骤 4 安装后测试

使用 OpenSSL 的软件算法测试 RSA 性能，具体命令为：

```
openssl speed -elapsed rsa2048
```

```
[root@ecs-a5c9 ~]# openssl speed -elapsed rsa2048
You have chosen to measure elapsed time instead of user CPU time.
Doing 2048 bits private rsa's for 10s: 7680 2048 bits private RSA's in 10.00s
Doing 2048 bits public rsa's for 10s: 311259 2048 bits public RSA's in 10.00s
OpenSSL 1.1.1f  31 Mar 2020
built on: Fri Apr  2 10:53:13 2021 UTC
options:bn(64,64) md2(char) rc4(char) des(int) aes(partial) idea(int) blowfish(ptr)
compiler: gcc -fPIC -pthread -Wa,--noexecstack -Wall -O3 -O2 -g -pipe -Wall -Werror=
s=/usr/lib/rpm/openEuler/openEuler-hardened-cc1 -fasynchronous-unwind-tables -fstack
SL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_BN_ASM_MONT -DSHA1_ASM -DSHA256_ASM -DSHA512_A
""
              sign    verify    sign/s verify/s
rsa 2048 bits 0.001302s 0.000032s    768.0  31125.9
```

在当前性能的主机中，RSA 的性能如下：

	sign	verify	sign/s	verify/s
rsa 2048 bits	0.001302s	0.000032s	768.0	31125.9

使用 KAE 加速引擎再次测试 RSA 性能，具体命令为：

```
export OPENSSL_ENGINES=/usr/local/lib/engines-1.1
openssl speed -elapsed -engine kae rsa2048
```

```
[root@ecs-a5c9 ~]# openssl speed -elapsed -engine kae rsa2048
engine "kae" set.
You have chosen to measure elapsed time instead of user CPU time.
Doing 2048 bits private rsa's for 10s: 7304 2048 bits private RSA's in 10.01s
Doing 2048 bits public rsa's for 10s: 220721 2048 bits public RSA's in 10.00s
OpenSSL 1.1.1f  31 Mar 2020
built on: Fri Apr  2 10:53:13 2021 UTC
options:bn(64,64) md2(char) rc4(char) des(int) aes(partial) idea(int) blowfish(ptr)
compiler: gcc -fPIC -pthread -Wa,--noexecstack -Wall -O3 -O2 -g -pipe -Wall -Werror=
s=/usr/lib/rpm/openEuler/openEuler-hardened-cc1 -fasynchronous-unwind-tables -fstack
SL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_BN_ASM_MONT -DSHA1_ASM -DSHA256_ASM -DSHA512_A
""
              sign    verify    sign/s verify/s
rsa 2048 bits 0.001370s 0.000045s    729.7  22072.1
```

使用 KAE 加速引擎后，RSA 的性能如下：

	sign	verify	sign/s	verify/s
rsa 2048 bits	0.001370s	0.000045s	729.7	22072.1

3 鲲鹏 DevKit 实践

3.1 安装鲲鹏 DevKit

实践说明：本实践仅针对鲲鹏 DevKit 的性能优化和系统诊断功能，因此只安装对应工具，即系统性能分析工具、Java 性能分析工具及系统诊断工具。本实践请使用“ARM-DevKit-vda.qcow2”该镜像创建的虚拟机，DevKit 已经下载到了目录/root/devkit，同时已经安装了相关的依赖包，因此可以跳过下载和依赖安装环节，直接进行安装操作：

```
[root@DevKit devkit]# ls
DevKit-All-24.0.RC2-Linux-Kunpeng          DevKit-Sys-Diagnosis-24.0.RC2-Linux-Kunpeng.tar.gz
DevKit-All-24.0.RC2-Linux-Kunpeng.tar.gz   DevKit-Sys-Perf-24.0.RC2-Linux-Kunpeng.tar.gz
DevKit-Java-Perf-24.0.RC2-Linux-Kunpeng.tar.gz  qe
[root@DevKit devkit]#
```

步骤 1 部署运行环境

使用以下命令安装实践工具依赖的软件包：

```
yum install -y unzip make expect perf gcc-c++ gcc glibc openssl sudo util-linux binutils dmidecode
sysstat numactl sqlite perl logrotate curl zip libffi-devel pcre pcre-devel zlib zlib-devel libunwind openssl-
devel graphviz psmisc strace pciutils lsscsi procs initscripts policycoreutils ethtool smartmontools kmod
net-tools rsyslog gzip iputils traceroute tcpdump fio ipmitool man bc crontabs libaio-devel numactl-
devel
```

执行完该命令后，等待所有依赖包安装完成

步骤 2 安装鲲鹏 DevKit 全量包

使用以下命令下载获取本实践所需的软件包，其中全量包获取命令为：

```
wget https://mirrors.huaweicloud.com/kunpeng/archive/DevKit/Packages/Kunpeng_DevKit/DevKit-All-
24.0.RC2-Linux-Kunpeng.tar.gz
```


下载完成后的安装如下图所示：

```
[root@ecs-a5c9 devkit]# ll
total 1.5G
-rw-r--r-- 1 root root 982M Jun 29 18:34 DevKit-All-24.0.RC2-Linux-Kunpeng.tar.gz
```

使用以下命令分别解压软件包：

```
tar --no-same-owner -xzf DevKit-All-24.0.RC2-Linux-Kunpeng.tar.gz
```

解压完成后生成对应目录，如下图所示：

```
[root@ecs-a5c9 devkit]# ll
total 1.5G
drwxr-xr-x 2 root root 4.0K Feb 28 2024 DevKit-All-24.0.RC2-Linux-Kunpeng
-rw-r--r-- 1 root root 982M Jun 29 18:34 DevKit-All-24.0.RC2-Linux-Kunpeng.tar.gz
```

进入对应目录安装并执行“./install.sh”安装命令，具体操作，如下图所示：

```
[root@ecs-a5c9 devkit]# cd DevKit-All-24.0.RC2-Linux-Kunpeng
[root@ecs-a5c9 DevKit-All-24.0.RC2-Linux-Kunpeng]# ./install.sh
Installing... Please wait.
```

在安装的过程中，按要求输入所需信息，首先设置认证方式，本实践采取用户名密码方式认证，则设置为 1，如下所示：

```
ot@ecs-a5c9 DevKit-All-24.0.RC2-Linux-Kunpeng]# ./install.sh
talling... Please wait.
he system uses yum to manage packages.

ect a tool access mode for your installation:
1]: User name and password
2]: OAuth 2.0-based SSO (You will be automatically logged in to the DevKit once you logged in
you select [2], to ensure proper use of the Kunpeng DevKit, ensure that you have installed an
er the sequence number of the tool access mode for your installation (default: 1): 1
```

然后设置安装模式，本实践选择模式 2，如下图所示：

```
Select an installation mode:
[1] : Install the DevKit framework and all plugins
[2] : Install the DevKit framework and selected plugins
[3] : Install the DevKit framework
Enter the sequence number of the tool to be installed (default: 1): 2
```

然后按照规划选择安装的插件，本实践将安装系统性能分析工具、Java 性能分析工具及系统诊断工具，在对话框中输入“5,6,7”，如下图所示：

```
Select the plugins to be installed::
SequenceNumber  Plug-in  Support
[1]             porting  y
[2]             affinity y
[3]             devtools y
[4]             debugger y
[5]             sys_perf y
[6]             java_perf y
[7]             sys_diagnosis y

Enter the serial number of the plug-in that can be installed. Format: a single digit or 'digit+comma', for example, (1 or 1,2 or 1,2,3). To exit the plug-in installation, enter no: 5,6,7
Selected install_tool: DevKit framework with plugins including sys_perf,java_perf,sys_diagnosis
```

安装过程中如果遇到如下依赖检查失败，请直接输入“y”继续，这些依赖不会影响后续实验：

```

The ipmitool(ipmitool -V) tool cmd check: OK
The man(man -V) tool cmd check: OK
The bc(bc -v) tool cmd check: OK
The route(route -n) tool cmd check: OK
The arp(type arp) tool cmd check: OK
The ibv(ibv_devinfo) tool cmd check: Failed
The crontabs cmd check: OK
The libaio-devel tool package check: OK
The numactl-devel tool package check: OK
Kernel-debuginfo may not be installed, which may cause some kernel symbol information not found
The following optional components have not been installed:ibv kernel-debuginfo

Installation environment check result:


| SequenceNumber | CheckItem                  | CheckResult | Suggestion                             |
|----------------|----------------------------|-------------|----------------------------------------|
| [1]            | installation user          | ok          | N/A                                    |
| [2]            | sudo permission            | ok          | N/A                                    |
| [3]            | su permission              | ok          | N/A                                    |
| [4]            | pam_rootok                 | ok          | N/A                                    |
| [5]            | sudoers configuration file | ok          | N/A                                    |
| [6]            | devkit user                | ok          | N/A                                    |
| [7]            | service configuration file | ok          | N/A                                    |
| [8]            | tool installation path     | ok          | N/A                                    |
| [9]            | user home directory        | ok          | N/A                                    |
| [10]           | framework dependency       | ok          | N/A                                    |
| [11]           | devkitworkers              | ok          | The tool will create 10 users (default |


t devkitworker1-10) to execute background tasks. All the users belong to the devkit user group and have the same use
r permissions.
[12] sys_perf optional dependencies fail The following optional components hav
e not been installed:modulecmd lshw ibdev2netdev cma_roce_tos ofed_info mlxconfig mlnx_qos show_gids ucx_info nvidia
-smi java kernel-debuginfo
[13] sys_diagnosis optional dependencies fail The following optional components hav
e not been installed:ibv kernel-debuginfo

Do you want to authorize the tool to handle the items failed in the installation environment check?[y/N]:

```

后续安装需设置安装路径及访问端口，保持默认即可，如下图所示：

```

Obtain the IP address, port number, and installation path requi
Enter the installation path. (The default path is /opt):
Selected install_path: /opt

The permission check of the installation path is successful.
The space check of the installation path is successful.
Selected IP address of the web server: 10.0.0.54

Please enter the installation port (default: 8086):
Selected Nginx port: 8086
Selected HTTP server port: 8002
Selected RPC cluster server port: 50051
Selected the internal port of the tool: 7996

```

安装时间大概需要五分钟，请等待安装完成。

```
Applying nfs_dpc_diagnostic_analysis.0001_initial... OK
Applying storageio_diagnostic_task.0001_initial... OK
Run sys_diagnosis install script
Check disk space
Check install path disk space Success
Creating an Application Running User
User:devkitworker1 is exist!
Locking password for user devkitworker1.
passwd: Success
Creating an Application Running User
Copy hyper_tuner_memleak to devkit_tools success
Copy bcc_wrapper.sh to devkit_tools success
Copy check_user_password_diagnose to devkit_tools success
Copy leak_check.sh to devkit_tools success
Copy clean_memdiagnostic.sh to devkit_tools success
Copy tcpdump.sh to devkit_tools success
Copy iperf3.sh to devkit_tools success
Copy fio.sh to devkit_tools success
Copy netflow.sh to devkit_tools success
Copy risky command success
Add sudoers
Install iperf3 Success
Install trace_net_flow Success
Load database success
Run sys_diagnosis install script success
install sys_diagnosis success
Generate agent package success
The sys_perf plugin is successfully installed in the /opt/DevKit/devkitplugins/sys_perf directory.
The java_perf plugin is successfully installed in the /opt/DevKit/devkitplugins/java_perf directory.
The sys_diagnosis plugin is successfully installed in the /opt/DevKit/devkitplugins/sys_diagnosis directory.
The Kunpeng Devkit Advisor is successfully installed in the /opt/DevKit directory.
The Devkit Web console is running. Visit https://192.168.100.131:8086
If 192.168.100.131:8086 has mapping IP, please use the mapping IP.
```

步骤 3 登录 DevKit

通过“<https://主机IP:8086>”登录鲲鹏 DevKit，首次登录时浏览器会提示证书风险，可按照提示点击“高级”、“继续前往 xx.xx.xx.xx（不安全）”继续访问，如下图所示：





登录鲲鹏 DevKit 后，系统会提示设置管理员密码，如下图所示：

欢迎使用

鲲鹏DevKit

首次登录请创建管理员密码。

devadmin

密码

确认密码

登录

设置密码完成后，即可进入主页，点击“调优”即可进入对应页面，如下所示：



根据上图中功能显示，表示已安装规划的功能模块。

3.2 鲲鹏 DevKit 系统性能分析实践（1）

实践说明：在创建运行测试脚本的虚拟机中，开启 NUMA 绑定功能，使虚拟机有至少两个 NUMA 节点，如下图所示：



步骤 1 增加 Agent

在鲲鹏 DevKit 主页，点击右上角“设置”图片，进入对应页面，然后点击“Agent 管理”，再点击“添加节点”，如下图所示：



在弹出的页面中输入待监测节点的信息，包括节点 IP、端口、认证方式、用户名及密码，如下图所示：

添加节点

节点名称

QE

安装路径

/opt

SSH登录信息

* SSH传输须知

节点管理使用服务器默认的SSH传输通道，为了保护数据的安全完整，建议使用安全的SSH服务，如使用安全的SSH版本，配置使用安全的加密算法等。

☒ 我已阅读以上内容。

* 节点IP

10 · 0 · 0 · 72

* 端口

22

* 用户名

root

* 认证方式

密码认证

采用密码认证时，将生成新的公钥用于配置免密认证

确定

取消

输入完成后点击确定，系统会有安全提示，如下图所示：

提示

 您正在使用root用户登录节点进行操作，请保证SSH通道安全。您也可以选择使用普通用户登录。

1.92.149.254下的指纹

加密类型	协议类型	指纹
SHA256	RSA	tylpJN4kK1JcstgJg8rcwUc7PYgksWgdxPfbqNztsao
SHA256	ECDSA	vGGVAfiTElr/YBU2FHMBQ/90oan+/DCC4JVik3h4elc
SHA256	ED25519	MwCQ7e/5cxilZin4C6UMXgh5Sz64uXI53PSRWWCV9Oc
MD5	RSA	b8:5e:74:61:9d:57:49:c6:2e:5b:f3:bd:e2:d0:ad:b2
MD5	ECDSA	11:fa:74:eb:f1:c3:7c:4b:cc:98:50:39:6d:e9:da:8c
MD5	ED25519	2e:05:b3:76:fb:91:cd:4b:e1:1b:56:ca:cd:7f:20:b8

继续添加

取消

点击“继续添加”可完成添加，等待节点状态转为在线即表示添加完成，如下所示：

添加节点	批量导入	批量删除
节点名称 	节点IP 	节点状态 
Node 1	10.0.0.54	 在线
QE	10.0.0.72	 在线

步骤 2 环境准备

说明：如果实验环境可以联网使用以下命令进行环境准备，如果实验环境无法联网，请向讲师获取代码并上传到业务主机中。

使用以下命令下载全部代码：

```
git clone https://gitee.com/kunpengcompute/devkitdemo.git
```

下载完成后，使用以下命令进入对应目录，并修改所涉代码的权限：

```
cd devkitdemo/Hyper_tuner/testdemo/multiply/ && chmod 777 multiply.c multiply_simd.c
multiply_start.sh multipliesimd_start.sh
```

然后使用以下命令编译对应代码并修改编译后应用程序的权限：

```
gcc -g multiply.c -o multiply && chmod 777 multiply
```

编译完成后，当前目录的文件如下图所示：

```
[root@ecs-9175 multiply]# ll
total 36K
-rwxrwxrwx 1 root root 74K Aug 15 16:53 multiply
-rwxrwxrwx 1 root root 2.2K Aug 15 16:46 multiply.c
-rwxrwxrwx 1 root root 2.6K Aug 15 16:46 multiply_simd.c
-rwxrwxrwx 1 root root 658 Aug 15 16:46 multiplysimd_start.sh
-rwxrwxrwx 1 root root 653 Aug 15 16:46 multiply_start.sh
```

使用以下命令循环运行程序：

```
nohup bash multiply_start.sh >>multiply.out 2>&1 &
```

步骤 3 系统性能分析

在鲲鹏 DevKit 页面“系统性能分析”菜单中点击新增图标，并选择“通用分析”，如下图所示：



在弹出的页面中选择“全景分析”并选择节点为新添加的业务主机，如下图所示：

★ 任务名称

test1

分析类型

通用分析

全景分析

进程

通过采集系统软硬件配置信息，以及系统CPU、内

系统部件分析

微架构分析

访存

专项分析

资源调度分析

锁与

★ 选择节点

已选： 1/10

全部： 2

最多支持

☐ 节点IP

☐ 10.0.0.54

☒ 10.0.0.72

采样的参数按照下图所示配置：

基本设置

分析对象

☒ 系统

系统资源上通常运行多个应用，采集整个系统的数据，可以忽略应用的个数或

* 选择场景

通用场景

* 采样类型

☒ 系统设置 ☒ PCIe拓扑 ☒ 系统性能 ☒ 任务调度

* 采样时长 (s)

20 (20~900)

高级设置

系统性能扩展分析 ☐

* 采样间隔 (s)

1 (1~5)

* 任务时间

☒ 立即执行 ☐ 预约定时启动

然后等待数据采集并分析完成，如下图所示：



当前节点：10.0.0.72 执行时间：00:00:11

i 系统负载高以及分析预处理均会使实际采集时长大于用户设置时长。

☒ 71%

☒ 数据采集中...

☐ 数据分析

☐ 任务完成

步骤 4 结果分析

在分析结果中，点击“系统性能”进入对应页面，然后展开 CPU 占用率页签，并将类型设置为“Core”，如下图所示：



将鼠标移到折线图的多个时间点，可发现总有的核占用率很高，而其他核则比较空闲，如下图所示：

	%system	%user
cpu0	60.61	38.38
cpu1	3.96	95.05
cpu2	0	99
cpu3	58	41

步骤 5 抓取进程/线程分析数据

按照步骤 3 的方法再新创一个任务，将分析类型设置为“进程/线程分析”，如下图所示：



The screenshot shows the 'New Task' form with the 'Process/Thread Analysis' option selected.

采集参数配置如下图所示：

分析对象

☒ 系统
 ☐ 应用

系统资源上通常运行多个应用，采集整个系统的数据，可以忽略应用的个数

* 采样时长 (s)

—

10

+

(2~300)

* 采样类型

☒ CPU
 ☒ 内存
 ☒ 存储IO
 ☒ 上下文切换

高级设置 ▾

* 采样间隔 (s)

—

1

+

采样间隔应当小于或等于采样时长的1/2且

采集线程信息

☒

* 任务时间

☒ 立即执行
 ☐ 预约定时启动

等待数据抓取并分析完成，DevKit 会根据当前情况给出优化建议，如下图所示：

总览

CPU

内存

存储IO

任务信息

🚀 优化建议 ▾

▼ 检测到CPU利用率不均衡，线程Tid_28971:multiply的CPU占用率过高。

优化建议：建议优化线程Tid_28971:multiply的实现。
 修改方法：将线程的计算任务分解到其它线程或增加此类线程的线程数。

步骤 6 热点函数分析

在热点函数分析时，建议仍然先启动程序，然后通过指定进程的方式进行采集。

首先使用以下命令查看任务的进程名称，具体如下所示：

```
[root@ecs-36f4 multiply]# ps -aux | grep multiply
root      34311  0.0  0.0 214400  3840 pts/1    S   15:32   0:00 bash multiply_start.sh
root      34318 92.7 49.9 11721408 7666112 pts/1    R   15:32   0:08 ./multiply
root      34321  0.0  0.0 213952  1536 pts/1     S+  15:32   0:00 grep --color=auto multiply
```

如果未查询到，请再次使用以下命令进行任务启动：

```
nohup bash multiply_start.sh >>multiply.out 2>&1 &
```

使用程序“multiply”对应的进程号进行热点函数分析，具体设置如下图所示：

任务名称

test3

导入模板

分析类型

通用分析

全景分析

进程/线程分析

热点函数分析

支持分析C/C++源代码识别性能瓶颈，给出对应的Top30热点函数和调用栈详情。通过火焰图展示函数的调用关系，发现优化路径。详情见[联机帮助](#)。

系统部件分析

微架构分析

访存分析

I/O分析

专项分析

资源调度分析

锁与等待分析

HPC应用分析

HPC集群检查

Roofline分析

选择节点

已选: 1/10

全部: 2

最多支持10个节点。管理节点

节点IP	节点名称	状态
☐ 10.154.175.102	Node 1	● 在线
☒ 10.154.175.103	QE	● 在线

在“基本设置”和“高级设置”中，按照下图进行配置：

基本设置

分析对象

系统

应用

应用运行在系统上，采集应用使用的系统资源数据，分析应用的性能。

模式

Launch application

Attach to process

启动采集任务时应用正在运行，采样时长由配置参数控制，适用于应用运行时间较长的场景。

进程名称

./multiply

PID

支持输入至多128个PID，请用“,”分隔

采样时长 (s)

—

20

+

(1~1,800)

随着采样时长增加，采集处理可能会因超过设定的采集数据大小而终止。

采集调用栈

?

高级设置

采样频率 (次/秒)

—

100

+

(1~500)

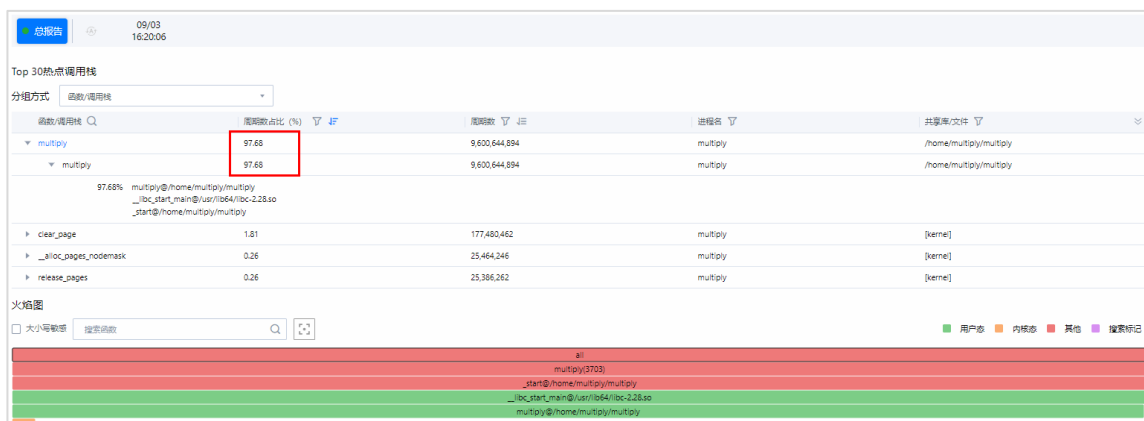
采样范围

所有

C/C++ 源文件目录

输入C/C++ 源文件工程目录

从获取到的结果可看出 multiply 占用了近 100%的时钟周期数，如下图所示：



点击该函数即可找到它在应用中对应的代码，如下图所示：

```

51     free(g_a);
52     free(g_b);
53     free(g_c);
54 }
55
56 void multiply(void)
57 {
58     unsigned i;
59     for (i = 0; i < N; i++) {
60         g_c[i] = g_a[i] * g_b[i];
61     }
62 }
63
64 void print_data(void)
65 {
66     // print first 2 and last 2 element.

```

可以看到这是一个 for 循环，且循环了 N 次，向上找到变量 N 的定义，如下图所示：

```

/*
一维矩阵计算
gcc -g multiply.c -o multiply
*/

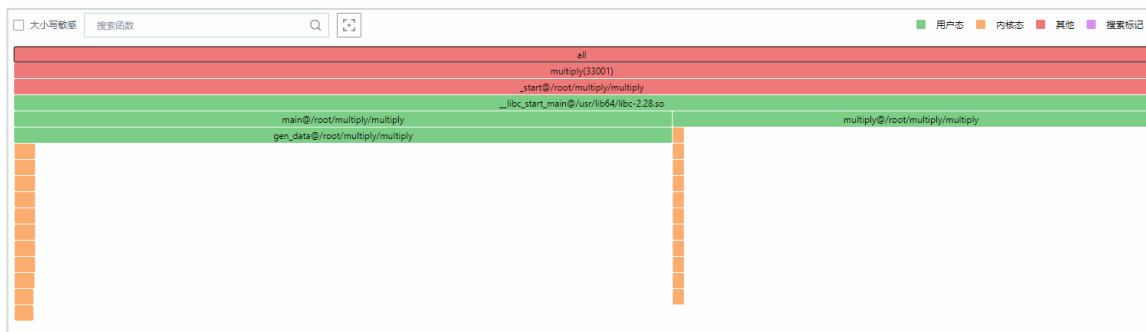
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
#include <sys/time.h>

#define N 1000000000
#define SEED 0x1234

```

假设在实际场景中不需要这么多次的循环，可在源代码 multiply.c 中将这个值修改为 100000，以减少时钟周期的占用。

同时还可以观察到对应的火焰图，如下图所示：



3.3 鲲鹏 DevKit 系统性能分析实践（2）

实验说明：本实践将复用 3.2 中的虚拟机，因此不需要再次安装 agent。所使用的代码包括在 3.2 步骤二中下载的项目中。

步骤 1 安装 MPI 应用的相关命令

使用命令 “<https://gitee.com/yftyxa/openmpi-4.1.4.git>” 下载应用编译所需要的工具源码
下载完成后使用命令解压下载后的文件，具体命令为：

```
tar -zxvf openmpi-4.1.4.tar.gz
```

解压完成后进入该目录，然后使用以下命令进行相关命令的安装：

```
./configure --prefix=/install/path CC=gcc CXX=g++ FC=gfortran && make -j4 && make install
```

其中 “/install/path” 为安装完成后命令所在的目录，可根据实际情况修改。

安装过程大概需要五分钟，安装完成后使用以下命令设置环境变量：

```
export PATH=/install/path/bin:$PATH
export LD_LIBRARY_PATH=/install/path/lib:$LD_LIBRARY_PATH
```

设置完成后，使用以下命令测试是否安装及配置正确：

```
mpicc --version
```

如果配置正确，返回的信息如下图所示：

```
[root@ecs-36f4 ~]# mpicc --version
gcc (GCC) 7.3.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

步骤 2 应用编译

应用所需代码为 ring.c，已在实验 3.2 的步骤二中下载，请使用 find 命令查找其所在位置，具体命令及结果可参考下图：

```
[root@ecs-36f4 ~]# find / -name ring.c
/root/devkitdemo/Hyper_tuner/testdemo/omp-mpi/ring.c
```

将代码所在目录全部复制到/opt 下，如下图所示：

```
[root@ecs-36f4 ~]# cp -r /root/devkitdemo/Hyper_tuner/testdemo/omp-mpi/ /opt
[root@ecs-36f4 ~]# ls /opt
DevKit omp-mpi
```

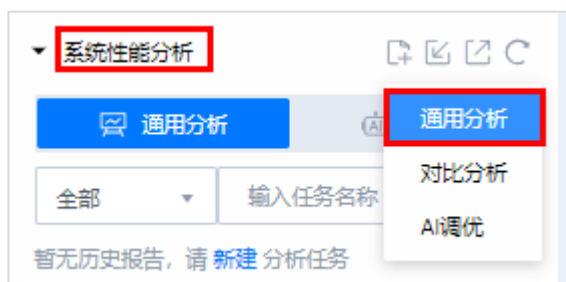
进入/opt/omp-mpi 目录，使用以下命令编译应用，并为编译后的程序添加可执行权限，具体命令为：

```
mpicc ring.c -O3 -o ring -fopenmp -lm && chmod 777 ring
```

```
[root@ecs-36f4 ~]# cd /opt/omp-mpi/
[root@ecs-36f4 omp-mpi]# mpicc ring.c -O3 -o ring -fopenmp -lm && chmod 777 ring
[root@ecs-36f4 omp-mpi]# ll
total 32K
-rw-r--r-- 1 root root 938 Sep  5 14:26 memory_bound.c
-rw-r--r-- 1 root root 1.1K Sep  5 14:26 memory_bound_mod.c
-rwxrwxrwx 1 root root 70K Sep  5 14:28 ring
-rw-r--r-- 1 root root 1.4K Sep  5 14:26 ring.c
-rw-r--r-- 1 root root 2.2K Sep  5 14:26 utils.h
```

步骤 3 应用分析

登录 DevKit，创建“系统性能分析任务”，如下图所示：



在创建任务具体参数时，将“分析类型”设置为“HPC 应用分析”，并选中对应节点，如下图所示：

新建任务

任务名称

HPC1

导入模板

分析类型

通用分析

全景分析

进程/线程分析

热点函数分析

系统部件分析

微架构分析

访存分析

I/O分析

专项分析

资源调度分析

锁与等待分析

HPC应用分析

通过采集系统的PMU事件，获取面向OpenMP和MPI应用的关键指标，以及内存带宽、指令分布、微架构指标等信

选择节点

已选: 1/10

全部: 2

最多支持10个节点。管理节点

节点IP

节点名称

10.154.175.102

Node 1

10.154.175.103

QE

然后按照下图所示配置应用所在的绝对路径、采集模式、分析模式及共享目录：

应用路径

/opt/omp-mpi/ring

注意：请确保应用无安全风险。
当前仅支持应用在/opt/或/home/或/usr/lib64/mpich/或/install/路径下执行，如有

应用参数

请输入应用参数

采集模式

OpenMP模式

MPI / MPI+OpenMP混合模式

分析模式

统计分析

精细化分析

共享目录

/home

最后按照下图所示设置 mpirun 相关配置：

mpirun配置

* 运行mpirun节点

10.154.175.103

* mpirun运行用户

root

* 密码

.....

* mpirun所在路径

/install/path/bin/mpirun

其中“mpirun 节点”及对应的运行用户和密码根据实际情况修改。Mpirun 所在的路径为本实验步骤一中编译安装所指定的路径。

最后按照下图所示设置采集相关的配置：

采样配置

* 采样时长 (s)

☒ 限制应用运行时长

☐ 不限制应用运行时长

因总采样时长包含采集准备时间，设置的采样时长需大于实际采样时长。

应用分析时可能会造成应用实际运行时间延长，请耐心等待。

—

20

+

(1~600)

随着采样时长增加，采集处理可能会因超过设定的采集数据大小而终止。

基础信息采集

☐ RDMA

☐ 共享存储

高级设置

配置完成后点击“确定”启动系统分析任务。

任务完成后，可查看到具体的分析结果，如下图所示：

HPC1

10.154.175.103_应用_ Launch_ MPI / MPI+OpenMP混合模式

通信热力图

Top N 低效通信

MPI timeline

任务信息

优化建议

绑核配置优化建议，详见 [rankfile.txt](#)，点击即可下载。

统计对象

Rank To Rank

统计指标

Latency

通信类型

Point To Point

(I,J)对应的指标为I发往J的指标加接收自J的指标；索引经过通信域和模糊化后重编，实际代表信息hover后显示。

华为认证计算系列教程

HClE-Computing

计算维护实验指导手册

版本：2.0



华为技术有限公司

版权所有 © 华为技术有限公司 2025。保留一切权利。

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

商标声明



和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

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

华为技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<https://e.huawei.com>

华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构，打造的覆盖ICT（Information and Communications Technology，信息技术）全技术领域的认证体系，包含ICT基础设施认证、基础软硬件认证、云平台及云服务认证三类认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

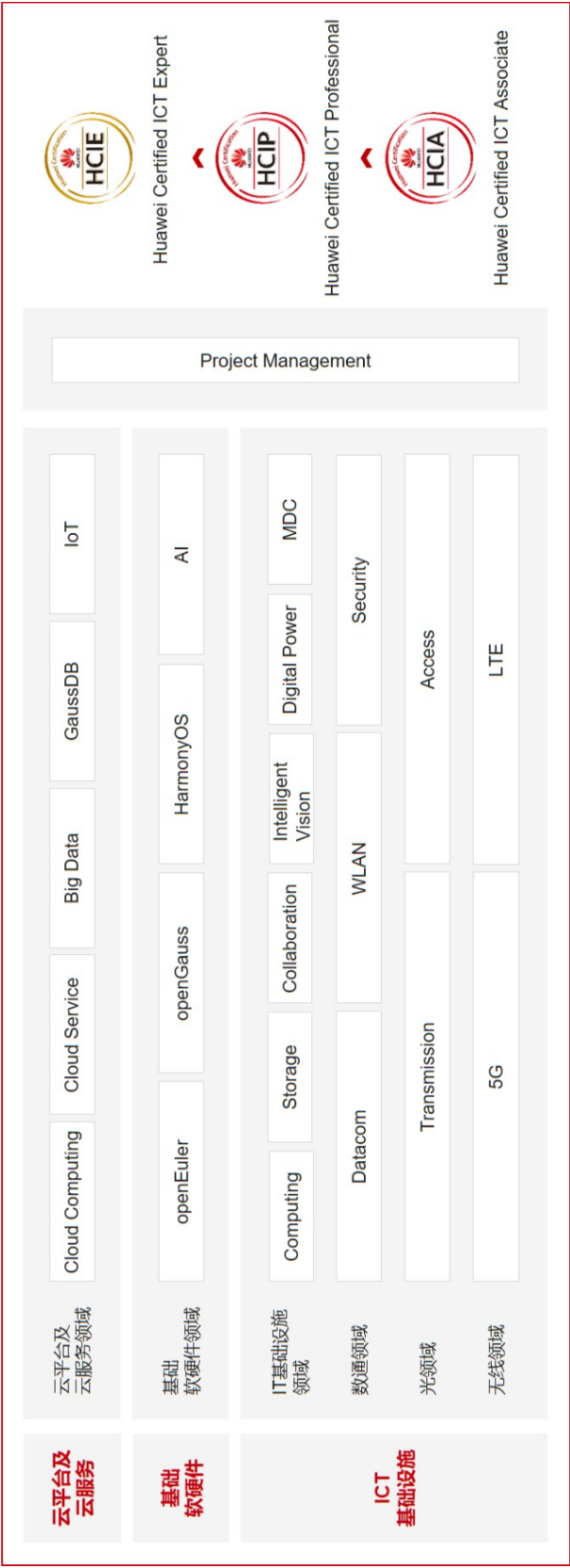
华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

华为认证HCIE-Computing V2.0定位于培养与认证具备人工智能计算中心规划设计、安装部署、性能优化与运维能力的计算专家。

通过HCIE-Computing V2.0认证，您将了解人工智能的发展历程及大模型相关原理，理解人工智能计算中心面临的挑战与解决方案，掌握数据中心的设计方法及人工智能计算中心训练、推理场景的方案，并能进行人工智能数据中心部署，掌握计算设备的性能调优及维护方法。

华为职业认证全景图

华为职业认证覆盖ICT全领域，致力于提供领先的人才培养体系和认证标准，培养数字化时代的新型ICT人才，构建良性的ICT人才生态。



简介

本书为 HCIE-Computing 认证培训教程，适用于准备参加 HCIE-Computing 考试的学员或者希望了解昇腾解决方案的读者。

在大模型快速发展的社会背景下，算力成为新的增长点，华为提供了以鲲鹏、昇腾为核心的第二算力平面。在算力使用过程中，CCAE 为计算中心提供端到端运维管理解决方案，提供计算设备、网络设备和存储设备管理能力，帮助用户提升集群场景运维效率，降低运维成本。

本次实验内容为 CCAE(Cluster Computing Autonomous Engine，简称 CCAE) 集群自智引擎的安装部署和对接配置，CCAE 是华为面向集群运维和管理的软件，致力于提供计算、网络、存储设备的一站式跨域管理，提供自动故障诊断等高阶能力。



图1-1 CCAE 介绍

内容描述

本实验指导书共包含 2 个实验，从 CCAE 的安装部署到 CCAE 跟各个平台的对接，实现告警的监测等功能。具体如下：

- 实验一为 CCAE 安装部署，包括使用 EasySuite 安装工具进行组网配置，并在服务器上安装 OS、CCAE 集群等内容。
- 实验二为 CCAE 资源对接，将服务器、存储、网络 and AI 集群对接到 CCAE 环境中，通过 CCAE 实现对各类资源的纳管，并查看资源对接情况。

读者知识背景

本课程为华为认证专家级别课程，为了更好地掌握本次实验内容，实验操作前建议首先具备以下基本条件：

- 具备一定的服务器、存储、网络相关基础知识

- 对 Linux 操作系统有一定的了解
- 已经具备同方向 HCIA 及 HCIP 同等知识技能

实验环境说明

本实验环境面向准备 HCIE-Computing 学习的工程师。前期相关环境预配置，可参考《HCIE-Computing 实验环境搭建指南》，进行前期的环境部署和配置。

实验环境组总体网示意图：

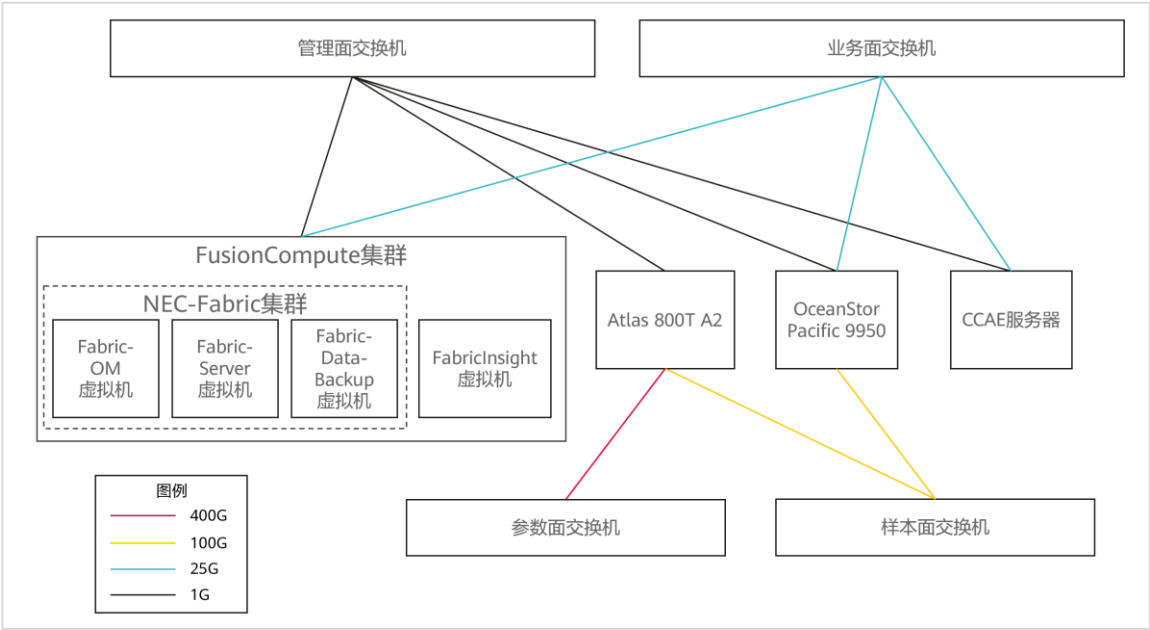


图1-2 组网图

本次 CCAE 实验所用到的设备如下，为了满足实验的需要，建议实验环境采用以下配置：

设备名称	设备型号	数量	备注
交换机	CE6865E-48S8CQ	1	业务交换机（25GE）
交换机	CE5855F-48T4S2Q	1	管理交换机（GE）
服务器	TaiShan 200 (Model 2280)	3	安装CCA集群
服务器	Atlas 800T A2	1	训练服务器

准备实验环境

软件及工具

实验开始之前请每组学员检查自己的软件及工具是否齐全，实验清单如下：

设备名称	版本	备注
CCAE相关软件包	V100R024C10SPC101	CCAE软件及证书
EasySuite安装工具	V100R024C10SPC101	EasySuite安装工具
笔记本或台式机	Windows10/11	运行EasySuite安装工具
PuTTY工具	N/A	SSH方式远程登录服务器

安装所需要的软件包如下，仅供安装时进行参考。

说明：以下为软件包名称，对应的同名签名文件（如 cms 和 crt）未列出，但仍需下载。

设备名称	版本	备注
EasySuite 安装软件	CCAE_V100R024C10SPC101_EasySuiteContainer_windows.zip	部署工具
	CCAE_V100R024C10SPC101_CCAEESolution.zip	解决方案包
CCAE安装 软件	CCAE_V100R024C10SPC101_CCAEService_image_linux-aarch64.7z	业务服务包
	CCAE_V100R024C10SPC101_CaaS_linux-aarch64.zip	平台CaaS包
	CCAE_V100R024C10SPC101_NetGraph_image_linux-aarch64.7z	平台服务包
	CCAE_V100R024C10SPC101_SolutionChart_linux-noarch.7z	业务部署配置包
	CCAE_V100R024C10SPC101_WeaveletSolution_linux-aarch64.zip	部署服务包
	CCAE_V100R024C10SPC101_BootstrapDeployTool_linux-aarch64.zip	平台部署工具包
	CCAE_V100R024C10SPC101_Euleros-iso_linux-aarch64.7z	操作系统软件包
	CCAE_V100R024C10SPC101_EasySuiteSrvtool_linux-aarch64.zip	EasySuite服务器 工具软件包
	CCAE_V100R024C10SPC101_LingQuService_image_linux-aarch64.7z	灵衢业务服务包

目录

前 言	3
简介	3
内容描述	3
读者知识背景	3
实验环境说明	4
准备实验环境	4
1 安装 CCAE	7
1.1 关于本实验	7
1.2 实验目的	7
1.3 组网拓扑	7
1.4 实验规划	8
1.5 实验任务配置	9
1.5.1 运行 EasySuite	9
1.5.2 安装 CCAE	12
1.6 结果验证	24
1.6.1 登录 CCAE 网页	24
1.7 思考题	27
2 CCAE 资源对接	28
2.1 关于本实验	28
2.2 实验目的	28
2.3 实验组网介绍	28
2.4 实验任务配置	29
2.4.1 对接前准备	29
2.4.2 接入服务器资源	34
2.4.3 接入存储系统	39
2.4.4 接入网络资源	46
2.4.5 接入 AI 集群资源	53
2.5 结果验证	59
2.5.1 告警查看	59
2.6 思考题	63

1 安装 CCAE

1.1 关于本实验

本实验通过 PC 电脑或其他 Windows 电脑上运行 EasySuite 安装工具，通过安装工具在服务器上安装 CCAE 集群环境，此外通过 EasySuite 安装工具还可以在服务器配置 RAID、安装系统、部署 CCAE 等。

1.2 实验目的

- 掌握 CCAE 的安装部署流程
- 掌握 CCAE 的组网规划及相关配置
- 掌握 CCAE 的网段及各网段的功能及作用
- 熟悉安装部署的相关注意事项

1.3 组网拓扑

CCAIE 网络有多个网络平面，各网络平面信息如下：

网络平面	功能	备注
容器内部通信网络	用于CCAIE各个物理节点内部通信使用	CCAIE各个物理节点内部相互通信
北向网络	用于CCAIE和客户OSS对接使用	与客户的OSS对接，如告警中心、日志中心等
南向带外网络	用于CCAIE和计算节点的iBMC及其他网管系统对接使用	连接计算节点iBMC、灵衢总线网络设备、NCE-Fabric、NCE-FabricInsight和DME等
南向带内网络	用于CCAIE和客户作业系统及计算节点OS对接使用	对接客户作业系统和计算节点OS，即连接CCAgent、LDAP、NTP、DNS以及作业平台

CCAE 支持北向网络和南向带外网络不隔离，南向带外网络和南向带内网络隔离，该配置为默认推荐配置，也是典型组网配置，物理网口组合如下：

- 容器内部通信网络
- 北向网络 + 南向带外网络
- 南向带内网络

因此本次实验部署，采用三平面的部署方式，即：容器内部通信网络、北向网络 + 南向带外网络、南向带内网络。

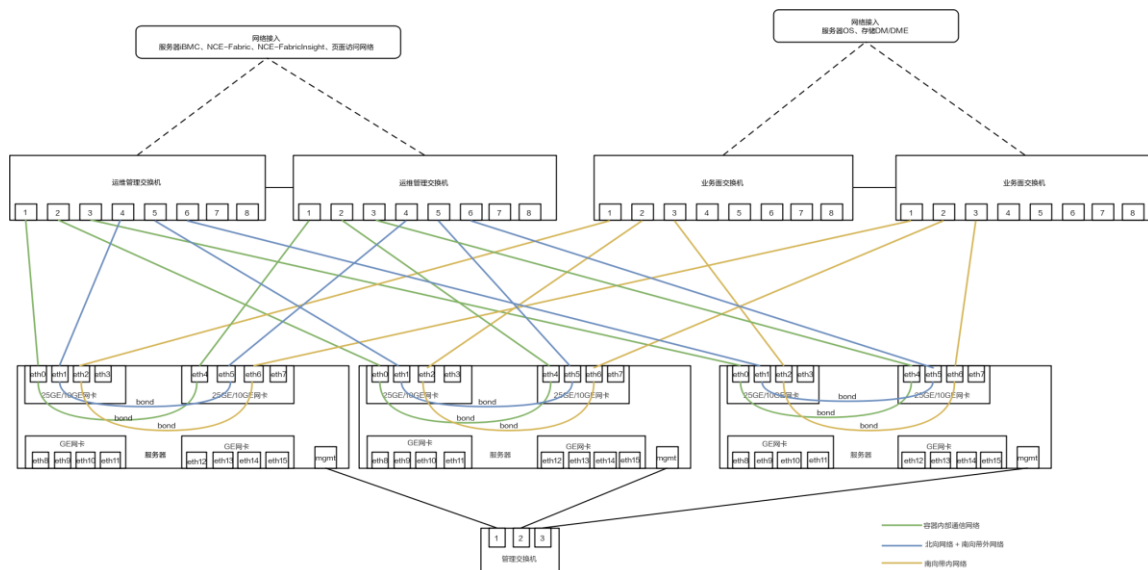
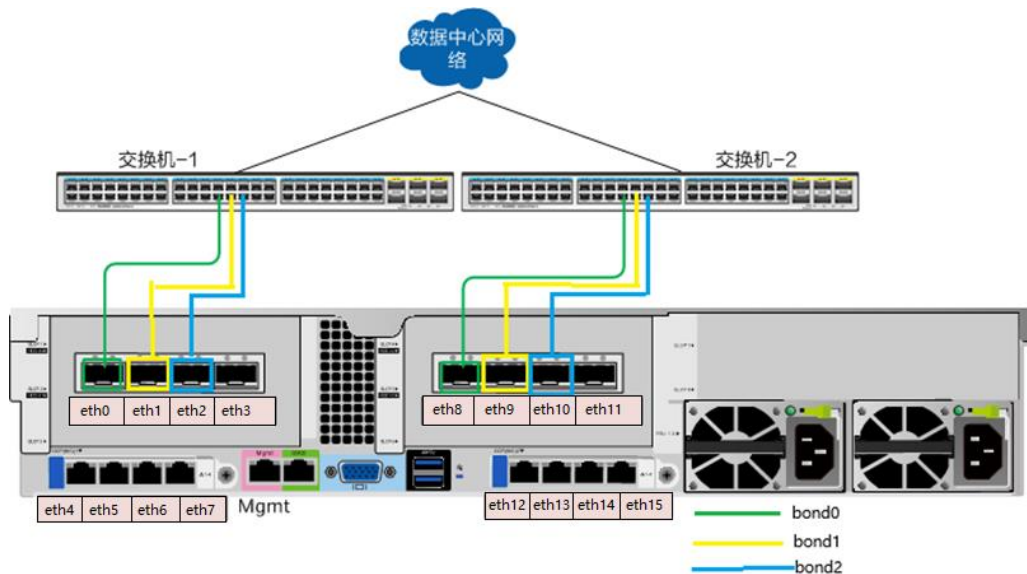


图1-1

1.4 实验规划

本次 CCAE 部署规模为 3 节点集群组网，服务器的网口连线规划示意图，如下：



同色标示的2个网口配置bond

图1-2

网络平面	功能	备注
容器内部通信网络	bond0	用于CCAE各个物理节点内部通信使用
北向网络+南向带外网络	Bond1	用于CCAE系统和客户OSS、计算节点的iBMC及其他网管系统对接使用
南向带内网络	Bond2	用于CCAE系统和客户南向带内平面对接使用

IP 地址规划示例

类型		网口	IP地址示例	备注
节点IP	容器内部通信网络	bond0	172.28.1.120	用于产品安装过程中安装节点之间同步软件包和镜像仓库
	北向网络+南向带外网络	Bond1	192.168.101.160	用于产品安装过程中EasySuite连接安装节点进行配置部署下发和服务安装
	南向带内网络	Bond2	192.168.102.120	用于产品的南向带内网络通信
浮动IP	容器内部通信网络浮动IP	bond0	172.28.1.123-124	填写未被使用的容器内部通信网络IP地址，需规划2个IP地址
	北向网络+南向带外网络浮动IP	Bond1	192.168.101.163	填写未被使用的对外IP地址，建议目的网段、目的掩码填写0.0.0.0
	南向带内网络浮动IP	Bond2	192.168.102.123	填写未被使用的对外IP地址

说明：如果有多个节点，相应的节点 IP 也需要规划多个。

1.5 实验任务配置

1.5.1 运行 EasySuite

步骤 1 在 <https://support.huawei.com> 官网下载 CCAE 相关软件，或联系管理员获取相关软件。

步骤 2 将安装用的笔记本或台式机，接入到待安装的网络环境中。将 EasySuite 安装包下载或拷贝到安装用的笔记本或台式机，并解压 EasySuite 安装软件包“CCA_E_V100R024C10SPC101_EasySuiteContainer_windows.zip”。

步骤 3 进入解压的软件路径 CCAE_V100R024C10SPC101_EasySuiteContainer_windows，并运行脚本“start.bat”，启动安装脚本。

名称	类型	大小
EasySuiteDesktopTool	文件夹	
es_svc	文件夹	
start.bat	Windows 批处理文件	7 KB

图1-3

步骤 4 等待加载完成。



图1-4

步骤 5 加载完成后，点击右上角的“新建 > 安装项目”按钮。

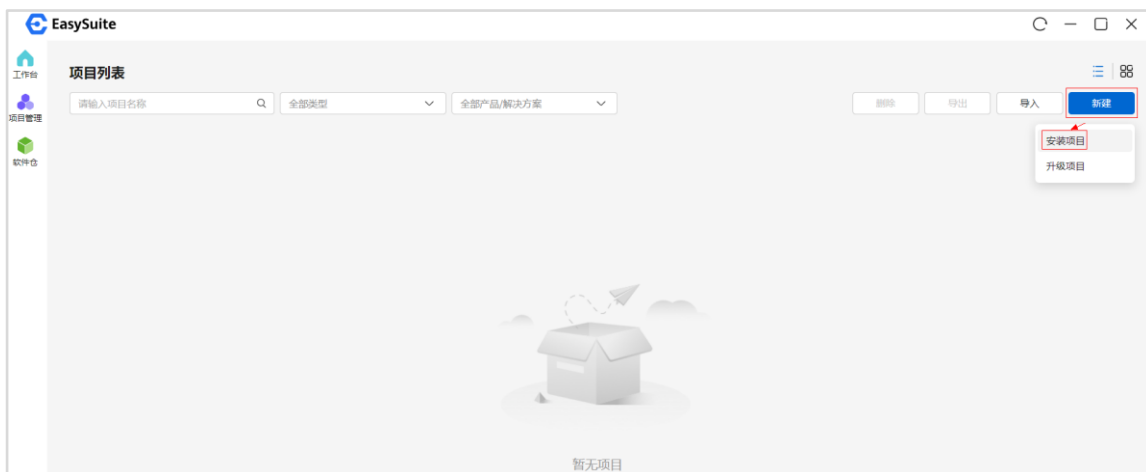


图1-5

步骤 6 在部署方案中，选择“其他解决方案”。

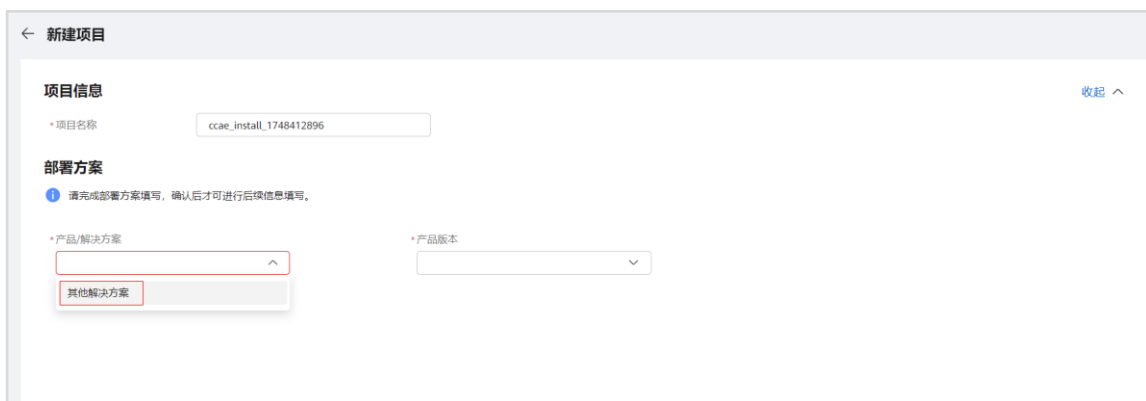


图1-6

步骤 7 在弹出的提示页面，点击“是”。

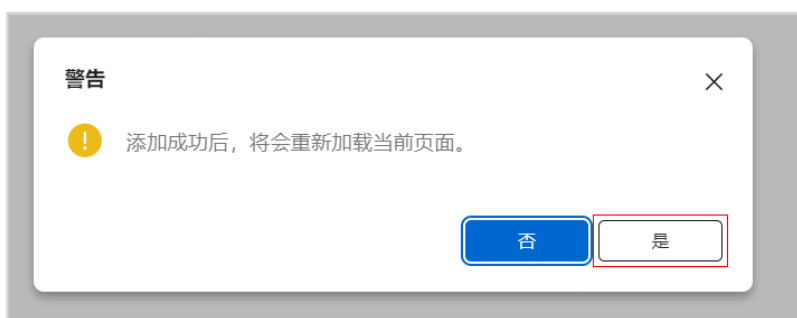


图1-7

步骤 8 根据提示，上传解决方案包，例如本次使用“CCAE_V100R024C10SPC101_CCAEESolution.zip”及其对应的签名文件，并点击“加载”按钮，等待加载完成。



图1-8

1.5.2 安装 CCAE

步骤 1 解决方案包加载完成后，会返回“项目信息”页面，并显示产品版本信息，在业务场景选项中，选择“集群”。



图1-9

步骤 2 本次 CCAE 为 3 节点的 AI 集群安装场景，采用“容器内部通信、北向+南向带外、南向带内”组网方式，根据此组网方式，进行如下配置。

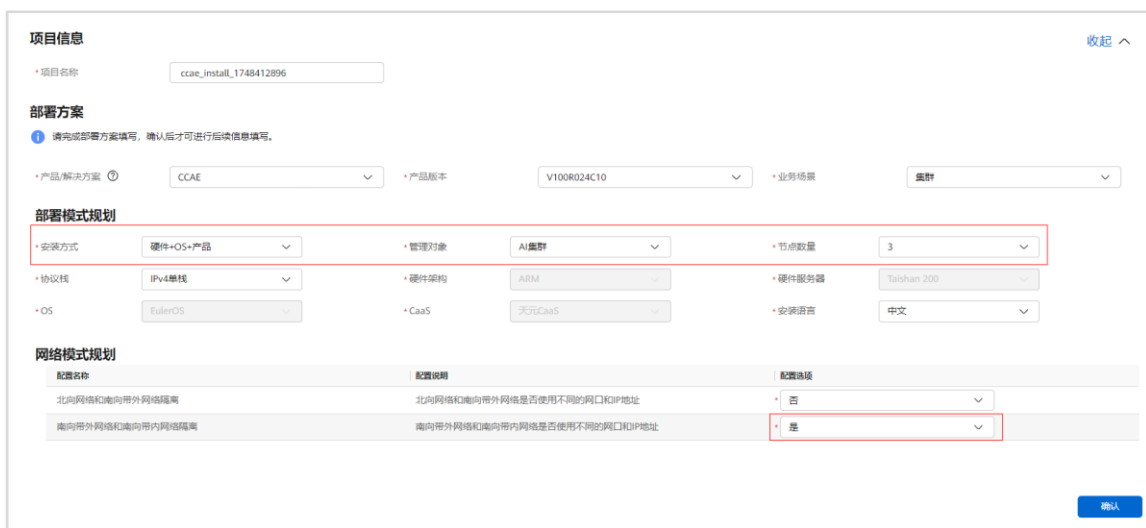
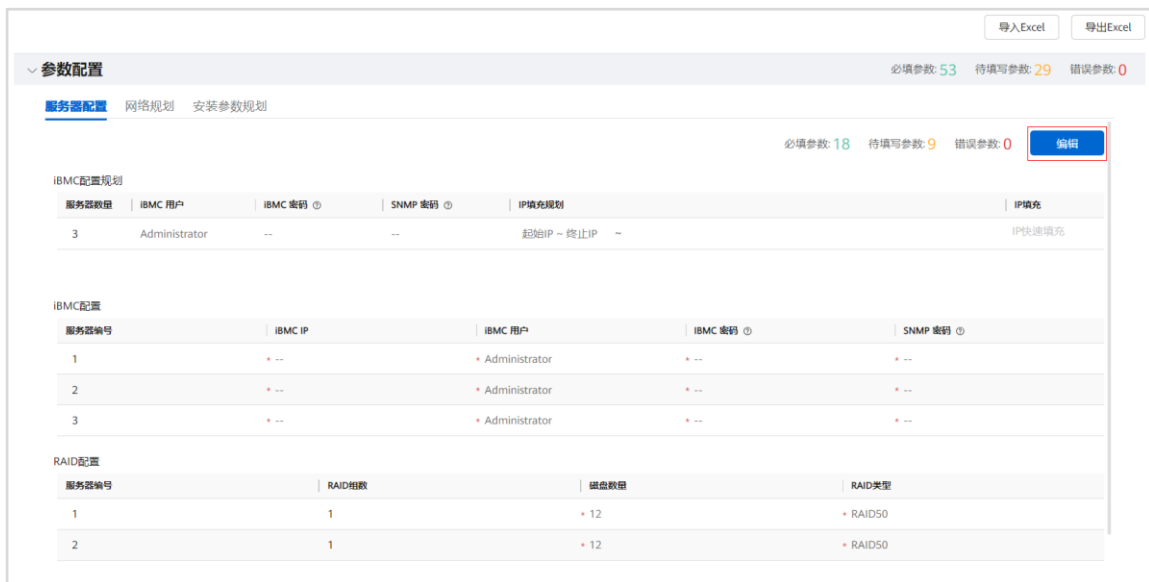


图1-10

步骤 3 在参数配置页面，需完成“服务器配置”、“网络规划”、“安装参数规划”的配置。点击“服务器配置”页签，选择右侧的“编辑”按钮，进行服务器参数配置。

说明：也可以选择导出 excel，填写配置后，再导入，本次实验以编辑配置方式进行讲解。



参数配置

网络规划 安装参数规划

必填参数: 53 待填写参数: 29 错误参数: 0

服务器配置

必填参数: 18 待填写参数: 9 错误参数: 0

编辑

iBMC配置规划

服务器数量	iBMC 用户	iBMC 密码	SNMP 密码	IP 填充规划	IP 填充
3	Administrator	--	--	起始IP ~ 终止IP	IP快速填充

iBMC配置

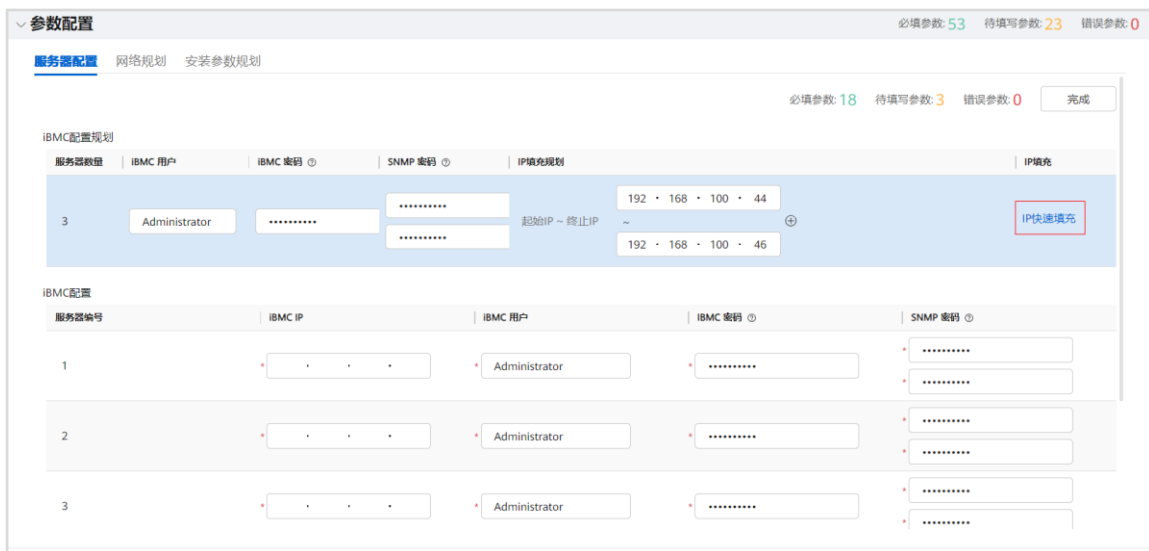
服务器编号	iBMC IP	iBMC 用户	iBMC 密码	SNMP 密码
1	--	Administrator	--	--
2	--	Administrator	--	--
3	--	Administrator	--	--

RAID配置

服务器编号	RAID组数	磁盘数量	RAID类型
1	1	12	RAID50
2	1	12	RAID50

图1-11

步骤 4 若服务器的配置一致，填写完成首行的配置信息，点击右侧的“IP 快速填充”，即可完成所有的节点的填写。



参数配置

网络规划 安装参数规划

必填参数: 53 待填写参数: 23 错误参数: 0

服务器配置

必填参数: 18 待填写参数: 3 错误参数: 0

完成

iBMC配置规划

服务器数量	iBMC 用户	iBMC 密码	SNMP 密码	IP 填充规划	IP 填充
3	Administrator	*****	*****	起始IP ~ 终止IP 192.168.100.44 ~ 192.168.100.46	IP快速填充

iBMC配置

服务器编号	iBMC IP	iBMC 用户	iBMC 密码	SNMP 密码
1	.*.*.*	Administrator	*****	*****
2	.*.*.*	Administrator	*****	*****
3	.*.*.*	Administrator	*****	*****

图1-12

步骤 5 完成服务器配置后，选择“网络规划”标签页，选择“编辑”按钮，进行网络配置。

参数配置

必填参数: 53 待填写参数: 20 错误参数: 0

服务器配置

网络规划

安装参数规划

必填参数: 32 待填写参数: 17 错误参数: 0

编辑

网段规划

网络平面	网口名称	IP数量	IP填充规划	IP填充
容器内部通信网络 (IPv4)	bond0	5	起始IP ~ 终止IP ~ 子网掩码 255.255.255.0	IP快速填充
北向网络+南向带外网络 (IPv4)	bond1	4	起始IP ~ 终止IP ~ 子网掩码 255.255.255.0 目的网段 0.0.0.0 目的掩码 0.0.0.0 平面网关 --	IP快速填充
南向带内网络 (IPv4)	bond2	4	起始IP ~ 终止IP ~ 子网掩码 255.255.255.0 目的网段 -- 目的掩码 --	IP快速填充

图1-13

步骤 6 跟服务器配置类似，填写完成规划的 IP 地址后，点击右侧的“IP 快速填充”，可以快速填充 IP 地址，以下为参考配置：

参数配置

必填参数: 53 待填写参数: 0 错误参数: 0

服务器配置

网络规划

安装参数规划

必填参数: 32 待填写参数: 0 错误参数: 0

完成

网段规划

网络平面	网口名称	IP数量	IP值规划	IP填充
容器内部通信网络 (IPv4)	bond0	5	起始IP ~ 终止IP ~ 子网掩码 255 · 255 · 0 · 0	IP快速填充
北向网络+南向带外网络 (IPv4)	bond1	4	起始IP ~ 终止IP ~ 子网掩码 255 · 255 · 255 · 0 目的网段 0 · 0 · 0 · 0 目的掩码 0 · 0 · 0 · 0 平面网关 192 · 168 · 101 · 254	IP快速填充

图1-14

服务器配置

网络规划

安装参数规划

平面网关 192 · 168 · 101 · 254

起始IP ~ 终止IP ~
192 · 168 · 102 · 120 ~ 192 · 168 · 102 · 125

子网掩码 255 · 255 · 255 · 0

目的网段 0 · 0 · 0 · 0

目的掩码 0 · 0 · 0 · 0

平面网关 192 · 168 · 102 · 254

IP快速填充

南向带内网络 (IPv4) bond2 4

节点准入信息

主机名称	容器内部通信网络 (IPv4)	北向网络+南向带外网络 (IPv4)	南向带内网络 (IPv4)
caasnode1	172 · 28 · 1 · 120	192 · 168 · 101 · 160	192 · 168 · 102 · 120
caasnode2	172 · 28 · 1 · 121	192 · 168 · 101 · 161	192 · 168 · 102 · 121
caasnode3	172 · 28 · 1 · 122	192 · 168 · 101 · 162	192 · 168 · 102 · 122

图1-15

参数配置

必填参数: 53 待填写参数: 0 错误参数: 0

服务器配置

网络规划

安装参数规划

网口名称	网络用途	IP地址	掩码	目的网段	目的掩码	网关
* bond0	CaaS内部浮动IP (DIP) (IPv4)	* 172.28.1.123	* 255.255.0.0	--	--	--
* bond0	CaaS外部浮动IP (DIP) (IPv4)	* 172.28.1.124	* 255.255.0.0	--	--	--
* bond1	北向网络+南向带内网络浮动IP (IPv4)	* 192.168.101.163	* 255.255.255.0	* 0.0.0.0	* 0.0.0.0	* 192.168.101.254
* bond2	南向带内网络浮动IP (IPv4)	* 192.168.102.123	* 255.255.255.0	* 192.168.101.0	* 255.255.255.0	* 192.168.102.254

Bond配置

Bond名称	Bond模式
bond0	* mode1 (active-backup)
bond1	* mode1 (active-backup)
bond2	* mode1 (active-backup)

CaaS准入信息

配置项	规划值	配置项说明
容器网络网段	* 172.20.0.0/16	CAAE集群中所有容器服务实例的IP地址范围, 用于服务实例之间的通信。CIDR格式, 掩码16位, 必须为网段内起始地址, 即x.x.0.0/16, 例如172.20.0.0/16, 不能与主机节点已有网段冲突。
服务网络网段	* 10.41.0.0/24	CAAE集群中所有容器服务的浮动IP地址范围, 用于服务之间的负载均衡和通信。CIDR格式, 掩码24位, 必须为网段内起始地址, 即x.x.x.0/24, 例如10.41.0.0/24, 不能与主机节点已有网段冲突。

图1-16

步骤 7 配置完成后, 点击“安装参数规划”, 点击右侧“编辑”按钮, 编辑相关配置。

参数配置

必填参数: 53 待填写参数: 3 错误参数: 0

服务器配置

网络规划

安装参数规划

密码规划

用户分类	用户名	密码	用户说明
操作系统用户	root	* --	操作系统管理员用户, 可以执行所有命令。产品安装完成后, root用户不允许远程登录。
操作系统用户	sopuser	* --	安装天元CaaS时创建, 用于SSH登录的用户。
操作系统用户	soplocal	* --	安装天元CaaS时创建, 不允许远程登录, 可使用登录用户切换至该用户。

必填参数: 3 待填写参数: 3 错误参数: 0

编辑

图1-17

步骤 8 根据账号规格, 设置操作系统用户 root、sopuser、soplocal 的密码, 填写完成后点击“完成”。

参数配置

必填参数: 53 待填写参数: 0 错误参数: 0

服务器配置

网络规划

安装参数规划

密码规划

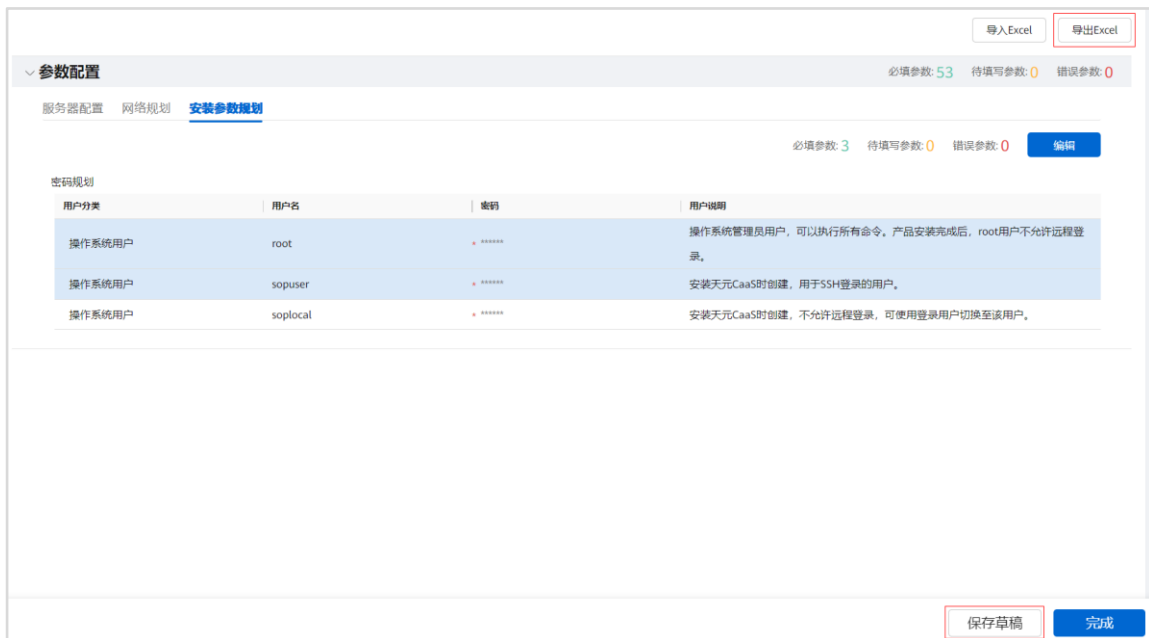
用户分类	用户名	密码	用户说明
操作系统用户	root	* *	操作系统管理员用户, 可以执行所有命令。产品安装完成后, root用户不允许远程登录。
操作系统用户	sopuser	* *	安装天元CaaS时创建, 用于SSH登录的用户。
操作系统用户	soplocal	* *	安装天元CaaS时创建, 不允许远程登录, 可使用登录用户切换至该用户。

必填参数: 3 待填写参数: 0 错误参数: 0

完成

图1-18

步骤 9（可选）点击右下角的“保存草稿”等待保存完成，点击右上角的“导出 excel”，可以导出规格填写的配置信息。



参数配置

服务器配置 网络规划 **安装参数规划**

必填参数: 53 待填写参数: 0 错误参数: 0

密码规划

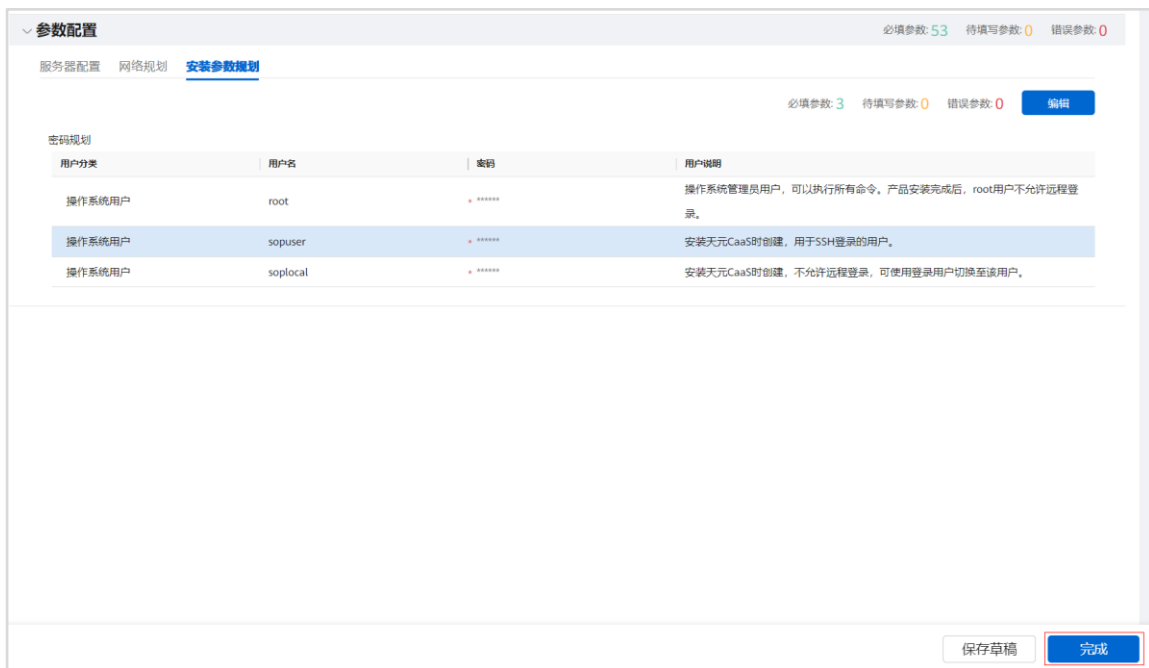
用户分类	用户名	密码	用户说明
操作系统用户	root	*****	操作系统管理员用户，可以执行所有命令。产品安装完成后，root用户不允许远程登录。
操作系统用户	sopuser	*****	安装天元CaaS时创建，用于SSH登录的用户。
操作系统用户	soplocal	*****	安装天元CaaS时创建，不允许远程登录，可使用登录用户切换至该用户。

必填参数: 3 待填写参数: 0 错误参数: 0 编辑

保存草稿 完成

图1-19

步骤 10 检查配置无误后，点击右下角的“完成”。



参数配置

服务器配置 网络规划 **安装参数规划**

必填参数: 53 待填写参数: 0 错误参数: 0

密码规划

用户分类	用户名	密码	用户说明
操作系统用户	root	*****	操作系统管理员用户，可以执行所有命令。产品安装完成后，root用户不允许远程登录。
操作系统用户	sopuser	*****	安装天元CaaS时创建，用于SSH登录的用户。
操作系统用户	soplocal	*****	安装天元CaaS时创建，不允许远程登录，可使用登录用户切换至该用户。

必填参数: 3 待填写参数: 0 错误参数: 0 编辑

保存草稿 完成

图1-20

步骤 11 在弹出的提示页面，点击“确定”。

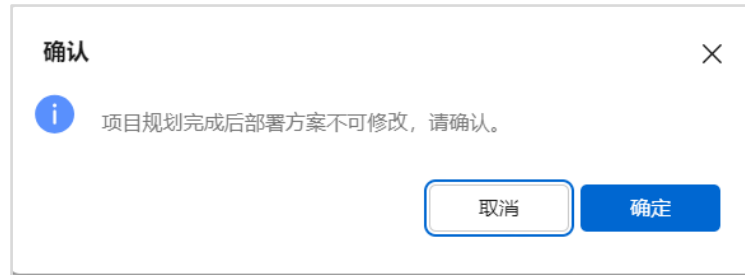


图1-21

步骤 12 接下来进入软件包上传的相关页面。

软件包详情					
序号	类型	软件包名称	软件包描述	结果	可能原因
1	配置RAID软件包	{product}_{version}_EasySuiteServertool_linux-aarch...	CCAE服务器工具软件包	● 待获取	
2	操作系统软件包	{product}_{version}_Euleros-iso_linux-aarch64.7z	CCAE操作系统软件包	● 待获取	
3	业务包	{product}_{version}_CCAEService_image_linux-aarch...	CCAE业务服务包	● 待获取	
4	产品软件包	{product}_{version}_SolutionChart_linux-noarch.7z	CCAE业务部署配置包	● 待获取	
5	业务包	{product}_{version}_NetGraph_image_linux-aarch64....	CCAE平台服务包	● 待获取	
6	CaaS软件包	{product}_{version}_CaaS_linux-aarch64.zip	CCAE平台CaaS包	● 待获取	
7	业务包	{product}_{version}_WeaveletSolution_linux-aarch64....	CCAE部署服务包	● 待获取	
8	业务包	{product}_{version}_BootstrapDeployTool_linux-aarc...	CCAE平台部署工具包	● 待获取	

图1-22

步骤 13 点击“导入软件包”。



图1-23

步骤 14 选择需要上传的文件或文件夹，选择完成后，点击“开始导入”。



图1-24

步骤 15 等待导入完成。



图1-25

步骤 16 软件包导入后，会自动进行效验，等待效验完成。

软件包详情					
序号	类型	软件包名称	软件包描述	结果	可能原因
1	配置RAID软件包	CCAЕ_V100R024C10SPC101_EasySuiteServertool_linu...	CCAЕ服务器工具软件包	就绪	
2	操作系统软件包	CCAЕ_V100R024C10SPC101_Euleros-iso_linux-aarch6...	CCAЕ操作系统软件包	就绪	
3	业务包	CCAЕ_V100R024C10SPC101_CCAEService_image_linu...	CCAЕ业务服务包	就绪	
4	产品软件包	CCAЕ_V100R024C10SPC101_SolutionChart_linux-noa...	CCAЕ业务部署配置包	就绪	
5	业务包	CCAЕ_V100R024C10SPC101_NetGraph_image_linux-...	CCAЕ平台服务包	就绪	
6	CaaS软件包	CCAЕ_V100R024C10SPC101_CaaS_linux-aarch64.zip	CCAЕ平台CaaS包	就绪	
7	业务包	CCAЕ_V100R024C10SPC101_WeaveletSolution_linux-...	CCAЕ部署服务包	就绪	
8	业务包	CCAЕ_V100R024C10SPC101_BootstrapDeployTool_lin...	CCAЕ平台部署工具包	就绪	

图1-26

步骤 17 校验无误后，点击“下一步”，进行环境监测。

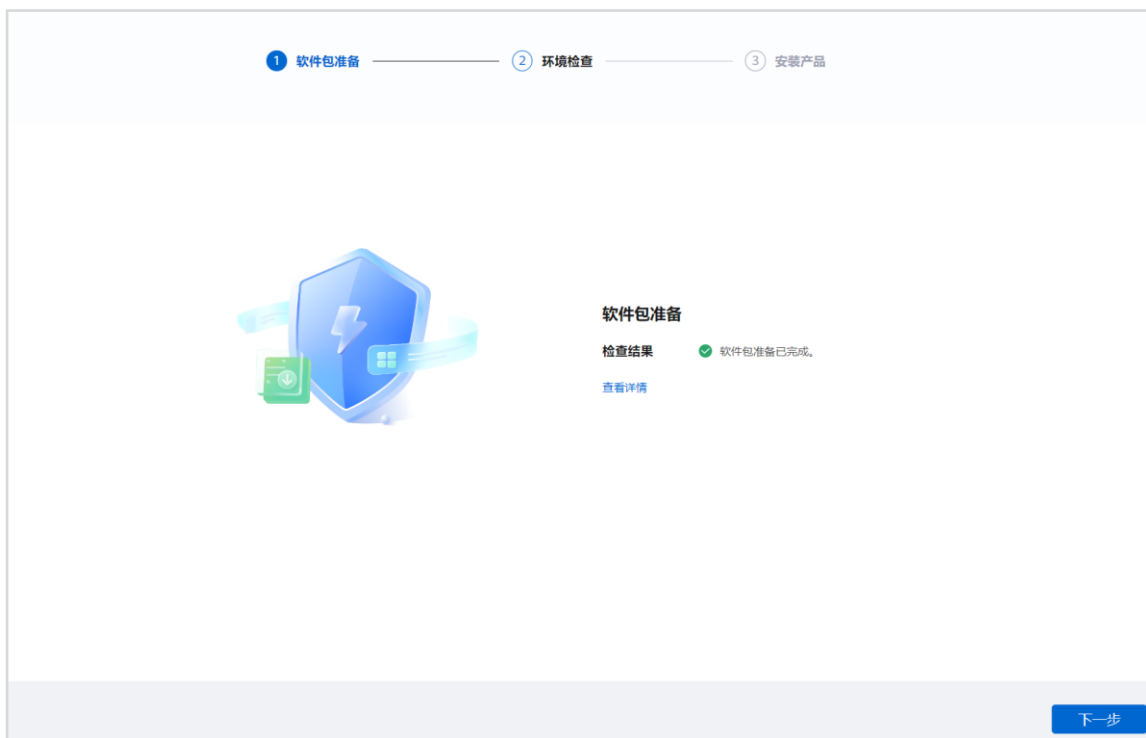


图1-27

步骤 18 点击“检查”按钮，进行安装环境监测。



图1-28

步骤 19 如果有检测不通过的选项，请根据提示，进行配置或环境的调整。

说明：若环境检查失败后需要调整规划参数，在“环境检查”界面单击“跳转至规划页面”。

环境检查详情

检查项	检查规则	类型	结果	处理建议
检查本地硬盘大小	检查硬盘剩余空间是否大于软件包总大小的2倍。	必过项	通过	--
检查本地CPU和内存	检查本地CPU及内存资源是否满足要求。	必过项	通过	--
检查服务器管理模块连通性	检查BMC能否SSH登录。	必过项	通过	--
检查服务器硬件信息	1、检查服务器类型。 2、查询RAID类型是否为规格内RAID卡...	必过项	通过	--
检查挂载镜像	检查服务器是否支持挂载镜像安装系统。	必过项	通过	--
检查BIOS安全启动项	检查BIOS安全启动项是否关闭。	必过项	通过	--
网络配置页面IP校验	1、检查是否有重复ip 2、检查同一网络平面IP是否在同一网段 ...	必过项	通过	--
多平面配置检查	1、业务容器网络平面请使用不同的网卡 2、南向多平面请使用...	必过项	失败	✖

2025-05-28 15:46:11 +0800 **ERROR** Please input different des_ipaddress in multi-south-plane !!!
2025-05-28 15:46:12 +0800 **ERROR** if floatingip is not in same ip range,please use different des_ip !!!

| 生成磁盘规划 | 生成磁盘规划 | 必过项 | | -- |

软件包准备

2 环境检查

3 安装产品



环境检查

检查结果

通过 7
失败 1
未开始 1

重试

跳转至规划页面

查看详情

图1-29

步骤 20 环境检查通过后，点击“下一步”。

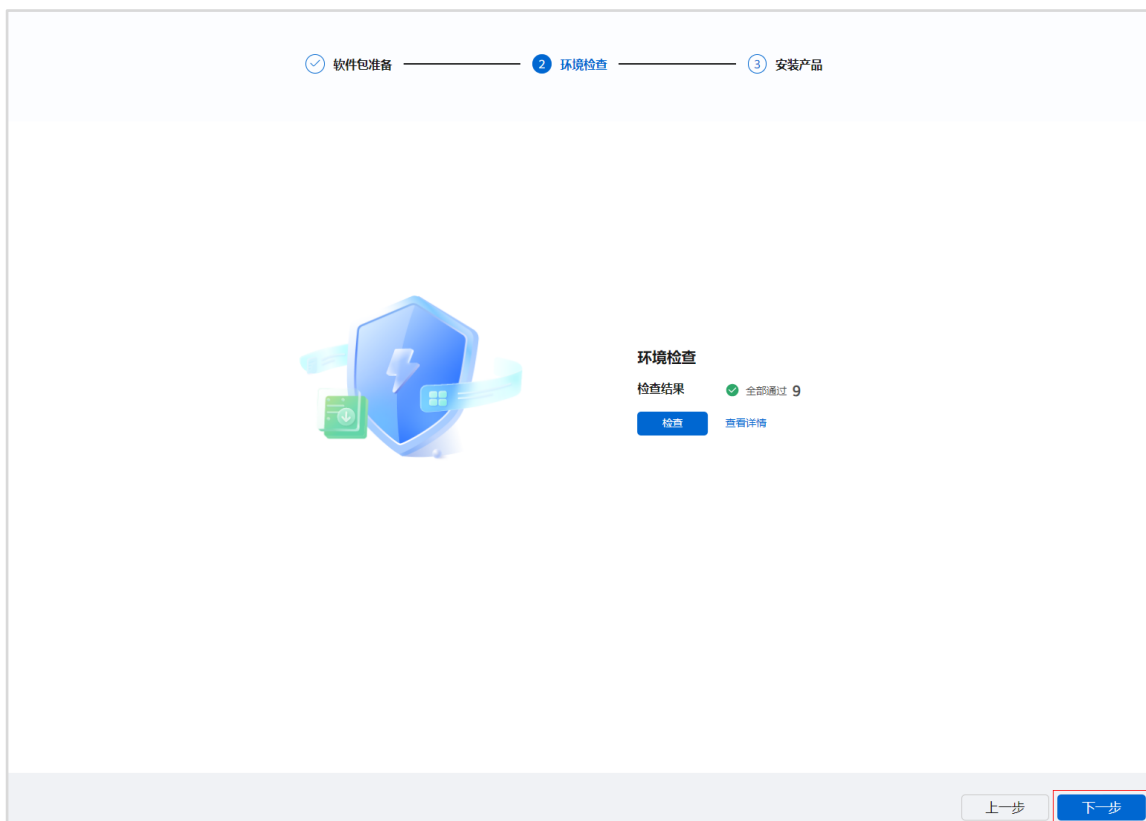


图1-30

步骤 21 在安装产品页面，点击“执行安装”。



图1-31

步骤 22 在安装提示页面，点击“是”，并等待安装完成。

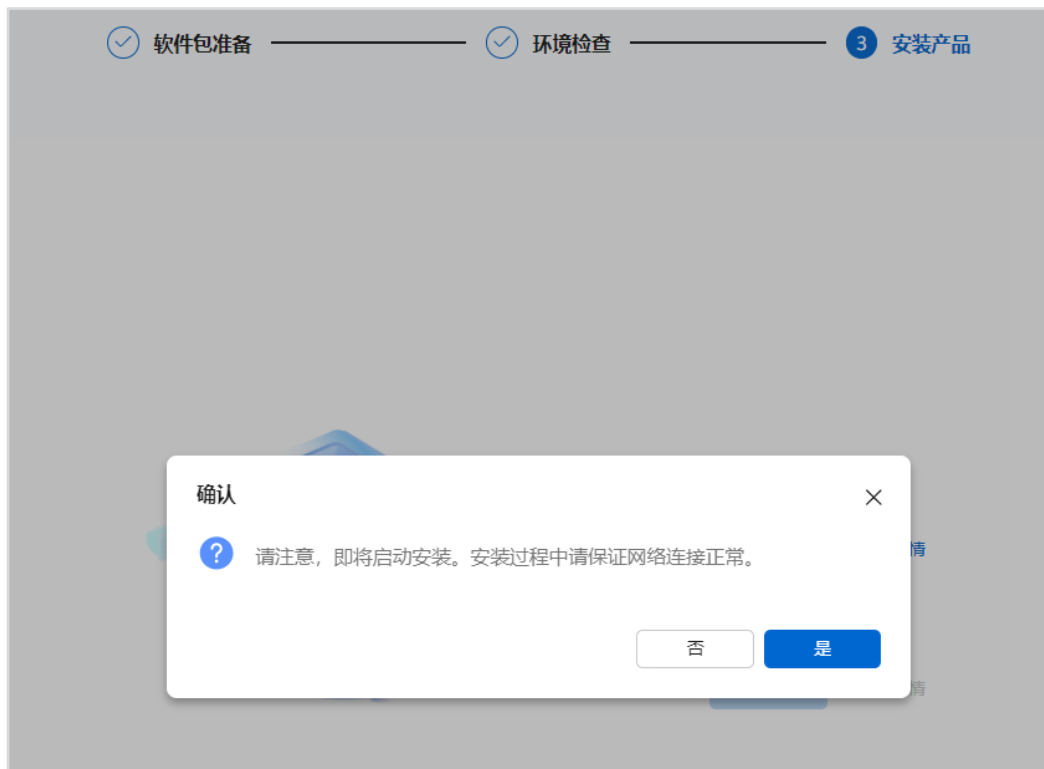


图1-32

步骤 23 在安装产品详情页面，可以查看安装的进度信息。

安装产品详情				
任务名称	任务进度	结果	开始时间	完成时间
✓ 安装参数配置	0%			
✓ 配置BMC	0%			
✓ 配置RAID	0%			
✓ 安装操作系统	0%			
✓ 配置操作系统	0%			
✓ 上传软件包	0%			
✓ 导入镜像	0%			
✓ 安装产品	0%			
✓ 安装后处理	0%			

安装产品详情					
任务名称	任务进度	结果	开始时间	完成时间	
安装参数配置	100%	已完成			
配置BMC	100%	已完成			
配置RAID	12%	运行中			
安装操作系统	0%				
配置操作系统	0%				
上传软件包	0%				
导入镜像	0%				
安装产品	0%				
安装后处理	0%				

安装产品详情					
任务名称	任务进度	结果	开始时间	完成时间	
安装参数配置	100%	已完成			
配置BMC	100%	已完成			
配置RAID	100%	已完成			
安装操作系统	100%	已完成			
配置操作系统	100%	已完成			
上传软件包	100%	已完成			
导入镜像	100%	已完成			
安装产品	96%	运行中			
安装后处理	0%				

图1-33

步骤 24 安装耗时预计 1-2.5 小时，安装期间请确保 PC 机电量充足，安装过程中 PC 机不能休眠，且网络不中断。

安装产品详情					
任务名称	任务进度	结果	开始时间	完成时间	
安装参数配置	100%	已完成			
配置BMC	100%	已完成			
配置RAID	100%	已完成			
安装操作系统	100%	已完成			
配置操作系统	100%	已完成			
上传软件包	100%	已完成			
导入镜像	100%	已完成			
安装产品	100%	已完成			
安装后处理	100%	已完成			

图1-34

步骤 25 安装完成后，页面会提示“产品安装已完成”，并显示安装后的健康检查等信息。



图1-35

步骤 26 页面显示为“安装产品已完成。可以登录管理面、业务面进行后续工作”，表示 CCAE 安装成功，单击“管理面”或“业务面”可跳转至对应 CCAE 登录界面。

步骤 27 CCAE 安装完成并验证后，单击界面右上角关闭按钮，关闭 EasySuite，并清理 EasySuite 解压目录。

1.6 结果验证

1.6.1 登录 CCAE 网页

页面显示为“安装产品已完成。可以登录管理面、业务面进行后续工作”时，表示 CCAE 安装成功，此时单击“管理面”或“业务面”可跳转至对应 CCAE 登录界面。也可以根据规划的 IP 地址+端口号，在浏览器直接登录。

步骤 1 分别登录管理面和业务面，登录用户为“admin”，首次登录初始密码为“Changeme_123”，登录后会提示修改登录密码，根据提示修改默认的初始密码。例如本次安装环境的登录地址如下：

- 管理面：https://192.168.101.163:31945/
- 业务面：https://192.168.101.163:31943/

步骤 2 登录 CCAE 平台的管理面页面，并修改默认密码。

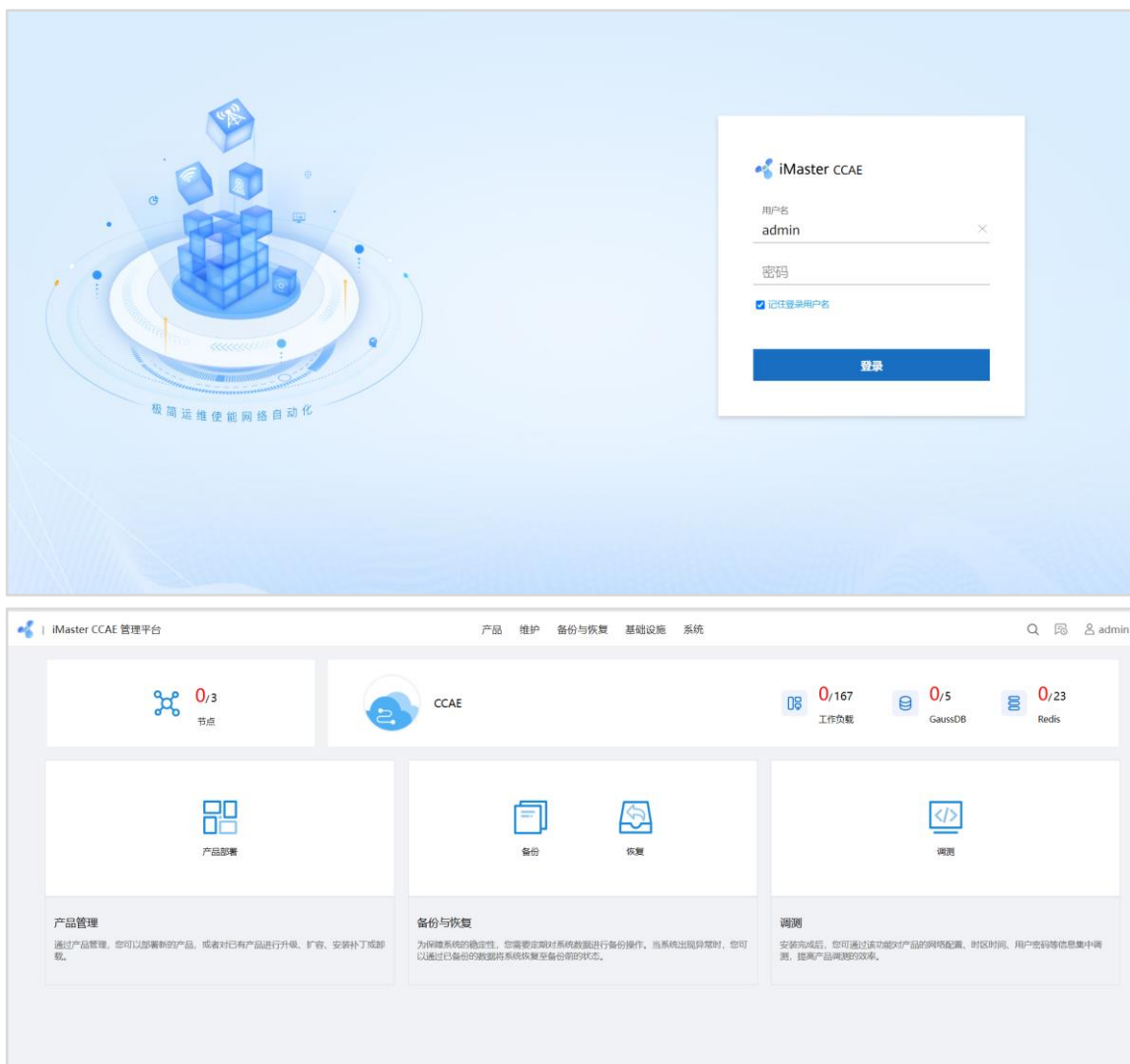


图1-36

步骤 3 登录 CCAE 平台业务面网页，，并修改默认密码。

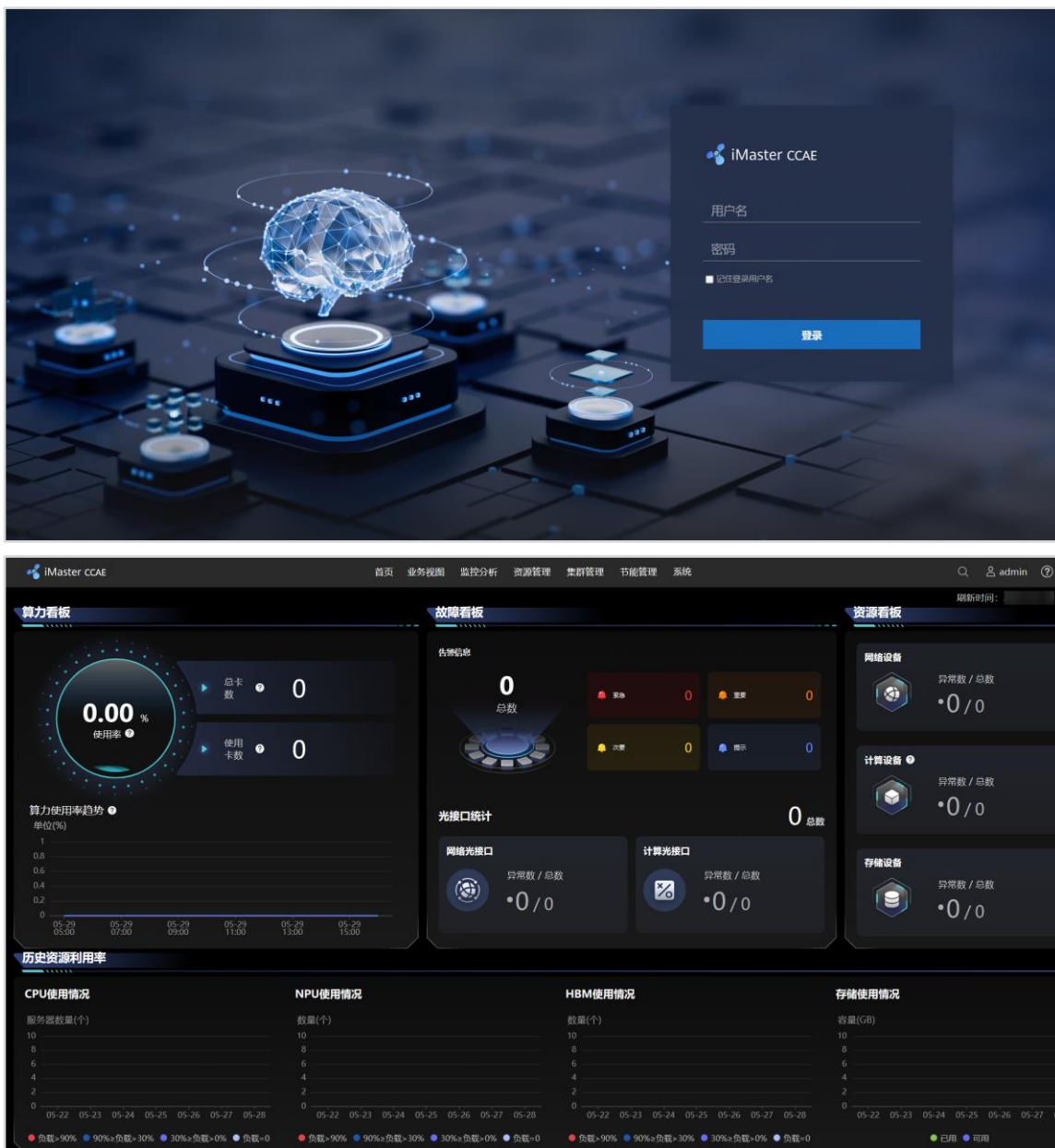


图1-37

步骤 4 登录新安装的系统，确认内核版本等信息。

使用 PuTTY（或其他 SSH 工具）以 sopuser 用户，通过 SSH 方式登录任一 CCAE 管理节点，执行以下命令，查看当前操作系统内核版本。

```
login as: sopuser
Pre-authentication banner message from server:
| Authorized users only. All activities may be monitored and reported.
End of banner message from server
sopuser@192.168.101.160's password:
Authorized users only. All activities may be monitored and reported.
Last login: Fri Jun 20 17:00  from 192.168.100.6
[sopuser@caasnode1 ~]$ uname -r
```

```
5.10.0-136.12.0.86.h2441.eulerosv2r12.aarch64  
[sopuser@caasnode1 ~]$
```

1.7 思考题

CCAIE 安装完成后，会自动完成网口的 bond 配置，在服务器有多个网口的情况下，如何快速确认 bond 的是哪两个网口？

答：在 CCAIE 集群中，查看网卡 bond 信息可使用如下命令（需要以 SSH 方式登录节点）。以查看 bond0 绑定的网卡为例，通过命令可以看到 bond0 绑定的 eth0 和 eth6，两个网口都是 25000 Mbps 速率。

```
[root@caasnode1 network-scripts]# cat /proc/net/bonding/bond0  
Ethernet Channel Bonding Driver: v5.10.0-136.12.0.86.h2441.eulerosv2r12.aarch64  
  
Bonding Mode: fault-tolerance (active-backup)  
Primary Slave: None  
Currently Active Slave: eth0  
MII Status: up  
MII Polling Interval (ms): 100  
Up Delay (ms): 0  
Down Delay (ms): 0  
Peer Notification Delay (ms): 0  
  
Slave Interface: eth0  
MII Status: up  
Speed: 25000 Mbps  
Duplex: full  
Link Failure Count: 0  
Permanent HW addr: c0:50:64:e8:81:44  
Slave queue ID: 0  
  
Slave Interface: eth6  
MII Status: up  
Speed: 25000 Mbps  
Duplex: full  
Link Failure Count: 0  
Permanent HW addr: c0:50:64:e8:80:a8  
Slave queue ID: 0  
[root@caasnode1 network-scripts]#
```

2 CCAE 资源对接

2.1 关于本实验

本实验将在刚刚完成安装的 CCAE 基础上，进行资源对接配置，包括服务器资源、存储资源、网络资源、AI 集群资源。并在对接完成后，进行相关资源的查看操作。

2.2 实验目的

- 熟悉 CCAE 的页面及操作
- 掌握 CCAE 和服务器的对接配置操作
- 掌握 CCAE 和存储设备的对接配置操作
- 掌握 CCAE 和网络平台的对接配置操作
- 掌握 CCAE 和 AI 集群的对接配置操作
- 熟悉 CCAE 的告警查看及资源的查看操作

2.3 实验组网介绍

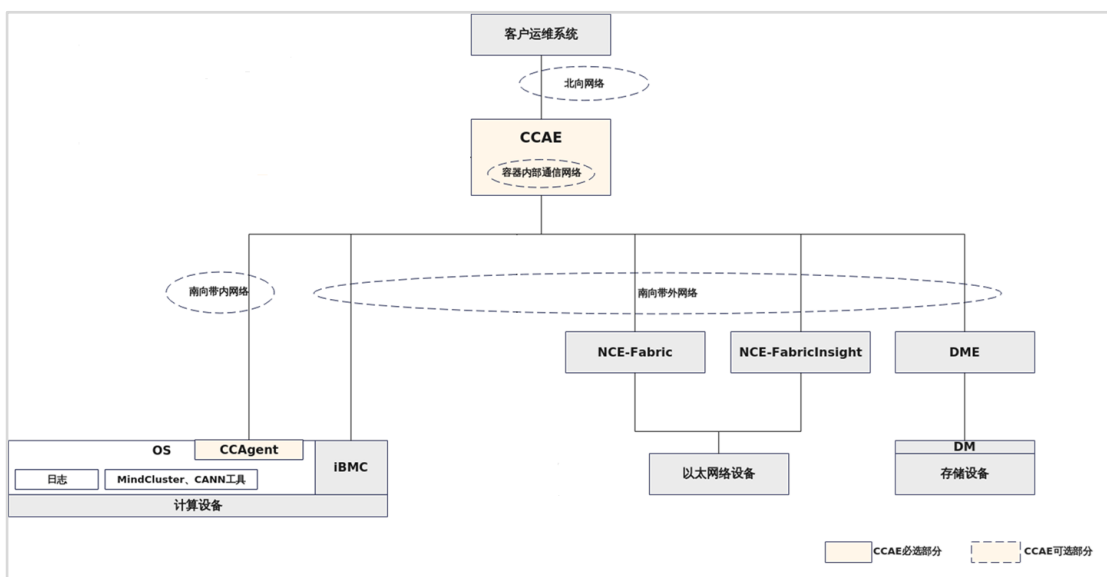


图2-1 CCAE 组网示意图

2.4 实验任务配置

2.4.1 对接前准备

步骤 1 登录 CCAE 管理面，在主菜单中选择“维护 > 网络配置 > 配置 VIP”。



图2-2

步骤 2 在“配置 VIP”页面的“配置 VIP”区域中，查看并记录所有产品名称为“CCAEC(cae)”、VIP 为 IPv4 类型，网络用途包含“南向网元接入”信息所对应的 VIP 地址。

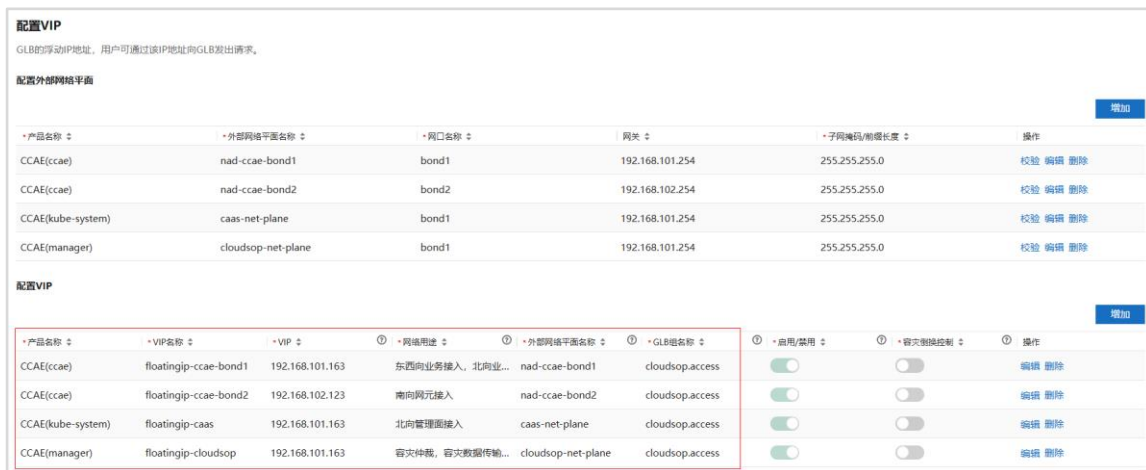


图2-3

步骤 3 配置 NTP 时钟。在 CCAE 管理面主菜单中选择“维护 > 时间管理 > 配置 NTP”。输入 NTP 服务器地址，并点击“确定”。



图2-4

步骤 4 在主菜单中选择“维护 > 时间管理 > 配置时区时间”。选择“UTC+8:00”的时区，并对时间进行确认。



图2-5

步骤 5 登录 CCAE 业务面网页，在主菜单中选择“系统 > 安全管理 > CA 服务”。在左侧导航树中选择“PKI 管理 > CA 管理 > 下载 CA 证书”。



图2-6

步骤 6 在弹出的对话框“文件格式”中选择“PEM”，并提交。pem 格式的证书会自动下载至本地。

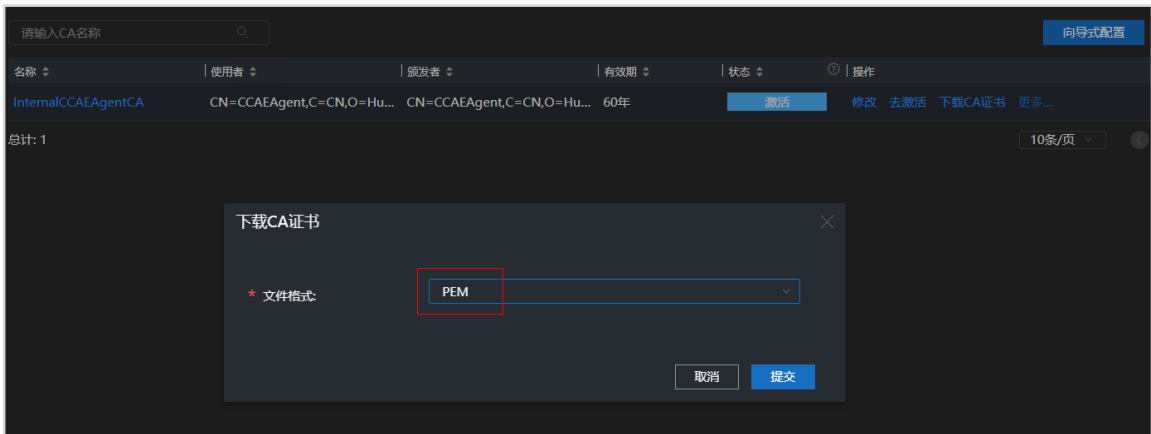


图2-7

步骤 7 配置项管理。在 CCAE 业务面主菜单中选择“系统 > 系统设置 > 配置项管理”，在左侧导航树中选择“SSH 配置”，开启“AutoInitialConfiguration”。在弹出的对话框中，仔细阅读提示信息后，勾选“我已明确该操作的风险”，单击“确定”。



图2-8

步骤 8 在左侧导航树中选择“证书认证配置”，选择“HTTPS 协议”页签。

开启“CCAEEAgentMgmtServiceActionServerCertAuthentication”、
“CCAEEAgentMgmtServiceHeartServerCertAuthenticatio”、
“CCAEEAgentMgmtServiceRegServerCertAuthentication”和
“CCAEEAgentMgmtServiceRenewServerCertAuthentication”。

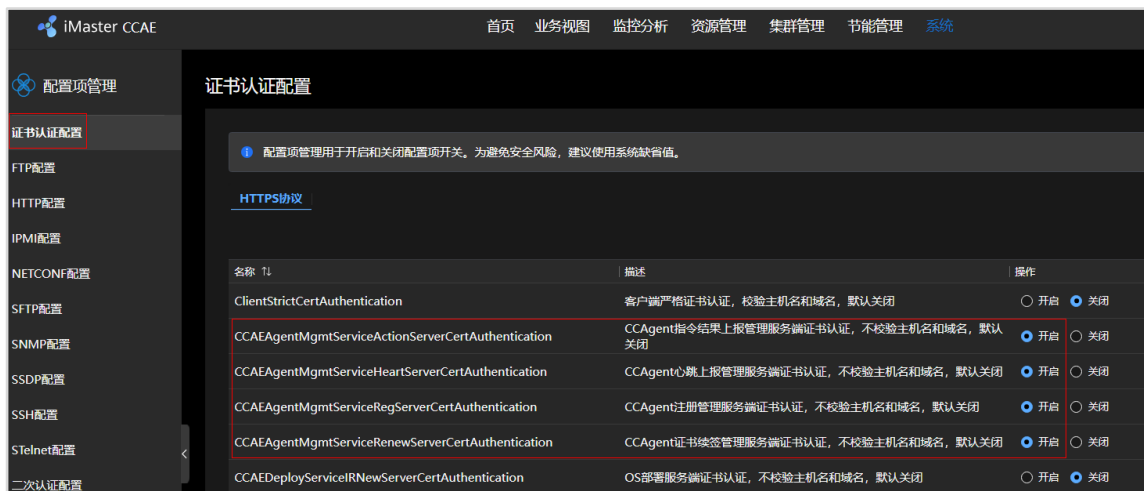


图2-9

步骤 9 若使用 Redfish 协议接入时, 不导入设备证书, 则需要在“系统 > 系统设置 > 配置项管理”中, 将“证书认证配置 > HTTPS 协议”的“ClientCertAuthentication”设置为“关闭”。

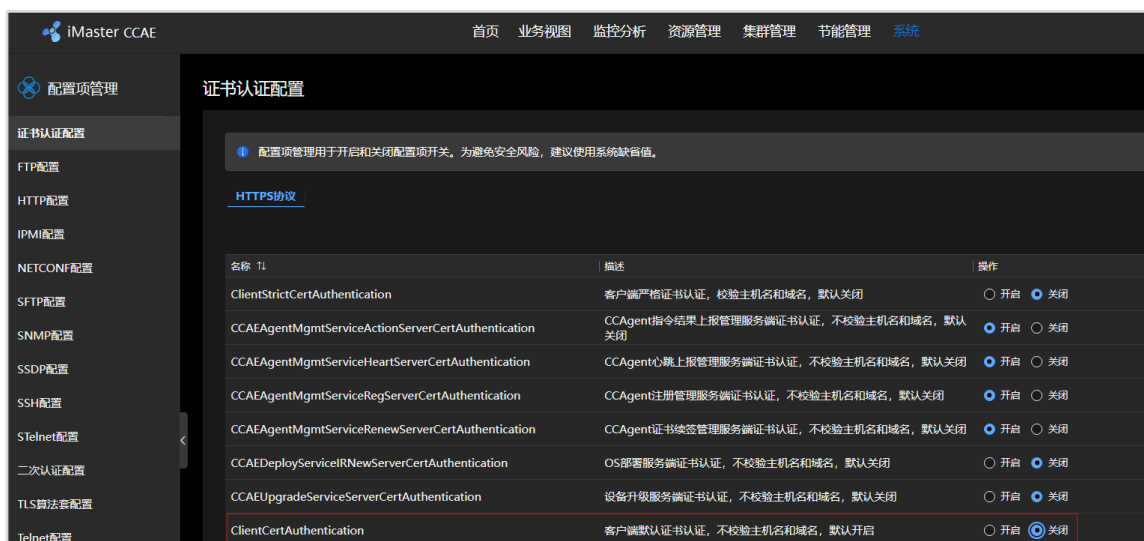


图2-10

步骤 10 在 CCAE 业务面主菜单中选择“系统 > 安全管理 > CA 服务”，在左侧导航树中选择“PKI 管理 > CA 管理”，在“CA 管理”界面, 单击名称为“InternalCCAEEAgentCA”的 CA 证书, 手动复制该 CA 的“默认模板”名称。



图2-11

步骤 11 CCAgent 证书配置。在“系统 > 证书管理 > 服务证书管理 > SouthBoundNodeService”中，选择“导入”。



图2-12

步骤 12 选择之前下载的 PEM 证书，填写完成后，点击“提交”。

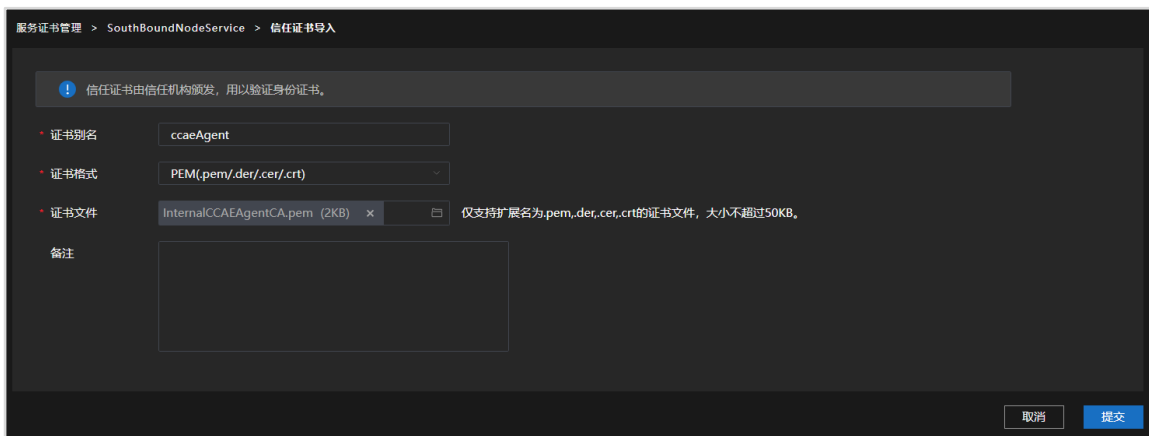


图2-13

步骤 13 查看导入的证书。



图2-14

步骤 14 创建 CCAgent 用户。通过 Putty 或其他工具，以 SSH 方式登录计算节点，参照以下命令，添加 CCAgent 用户并设置密码。

说明：在计算节点安装 CCAgent，或在线安装方式添加计算节点时，需提前在计算节点创建 CCAgent 运行用户。

```
[root@hcworker ~]# groupadd -g 5000 ccaeuser
[root@hcworker ~]# useradd -g ccaeuser -d /home/ccauser -u 5000 -m ccaeuser -s /bin/bash
[root@hcworker ~]# passwd ccaeuser
Changing password for user ccaeuser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@hcworker ~]#
```

步骤 15 通过 date 命令检查服务器系统时间和时区是否正确。

```
[root@hcworker ~]# date
Sat Jun 28 11:20:42 CST 20xx
[root@hcworker ~]# date -R
Sat, 28 Jun 20xx 11:21:31 +0800
```

或使用 timedatectl 进行检查。

```
[root@hcworker ~]# timedatectl
Local time: Sat 20xx-06-28 11:23:07 CST
Universal time: Sat 20xx-06-28 03:23:07 UTC
RTC time: Sat 20xx-06-28 11:23:11
Time zone: Asia/Shanghai (CST, +0800)
NTP enabled: no
NTP synchronized: no
RTC in local TZ: yes
DST active: n/a
```

步骤 16 如果时间或时区不正确，请使用 date 或 timedatectl 命令对其进行修改，或配置 NTP 服务器。

修改时间/时区的参考命令如下，本次以 CentOS7 为例。

```
[root@atlas800_01 ~]# timedatectl set-time "20xx-06-28 11:32:00"
[root@atlas800_01 ~]# timedatectl set-timezone Asia/Shanghai
[root@atlas800_01 ~]# timedatectl
Local time: Sat 20xx-06-28 11:32:43 CST
Universal time: Sat 20xx-06-28 03:32:43 UTC
RTC time: Sat 20xx-06-28 03:32:42
Time zone: Asia/Shanghai (CST, +0800)
NTP enabled: no
NTP synchronized: no
RTC in local TZ: no
DST active: n/a
```

2.4.2 接入服务器资源

本次接入的这台服务器，即在前面模型相关实验中使用的训练服务器，本次通过 iBMC 接入。

设备类型	型号	iBMC地址	备注
------	----	--------	----

服务器	Atlas 800T A2	192.168.100.50	训练
-----	---------------	----------------	----

步骤 1 登录需要进行对接服务器的 iBMC，选择 “用户&安全 > 本地用户 > 添加用户” 为其创建对接账号。



图2-15

步骤 2 根据提示，完成密码设置，角色权限需要设置为 “管理员”，设置完成后，点击 “保存”。

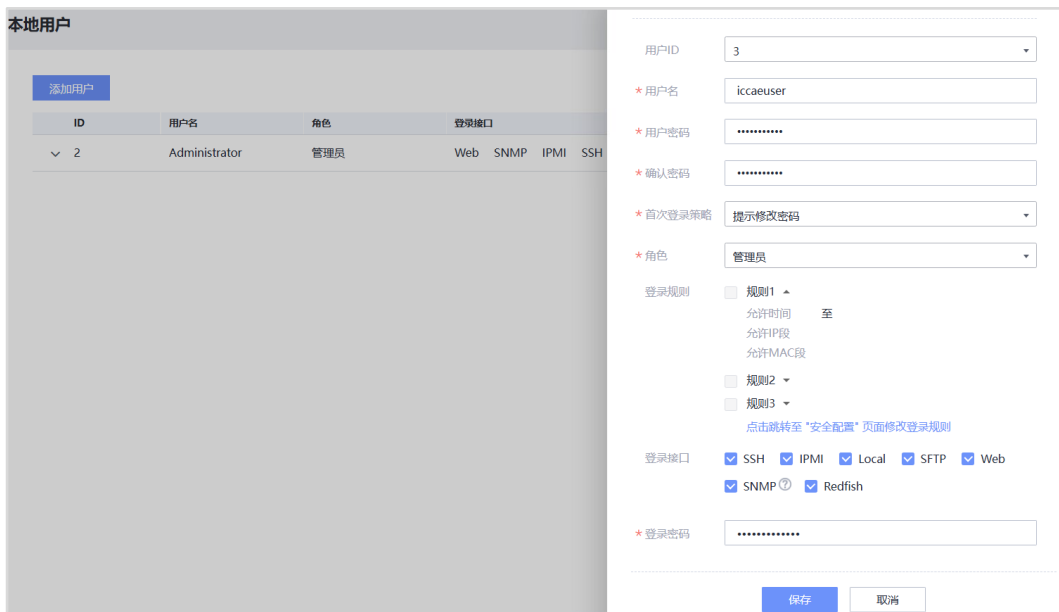


图2-16

步骤 3 使用新创建的用户，登录 iBMC，并根据提示修改密码。

密码修改

.....

.....

.....

✓ 密码修改成功。

确认

返回

图2-17

步骤 4 在 CCAE 业务面主菜单中选择“资源管理 > 计算资源 > 服务器”。



图2-18

步骤 5 单击右上角“单个添加”。

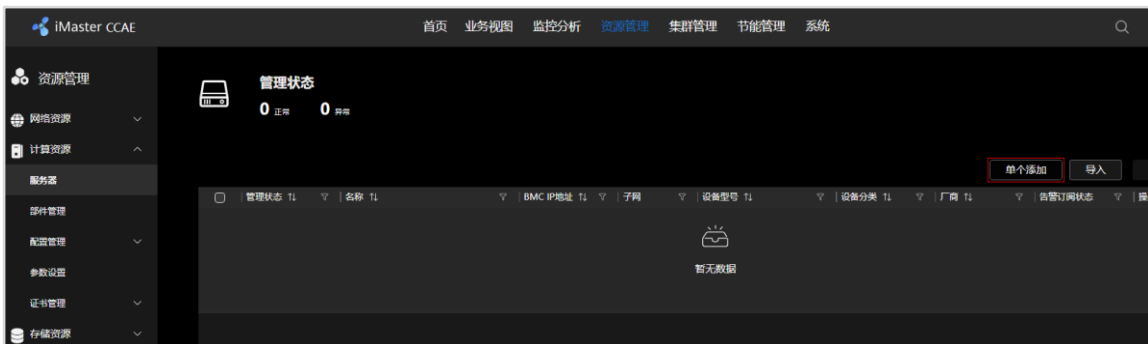
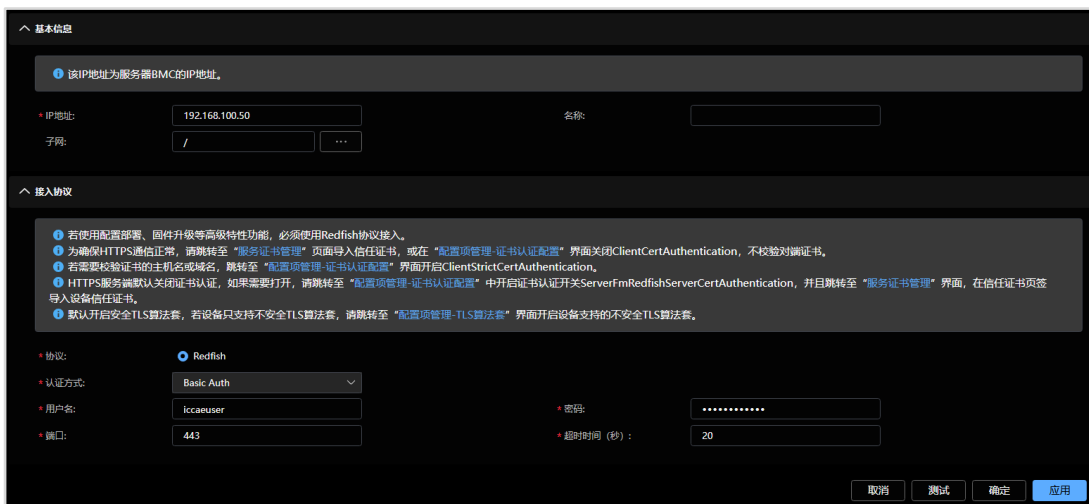


图2-19

步骤 6 根据界面提示及现网服务器信息，填写接入信息。



基本信息

该IP地址为服务器BMC的IP地址。

IP地址: 192.168.100.50 名称:

子网: / ...

接入协议

若使用配置部署、固件升级等高级特性功能，必须使用Redfish协议接入。
 为确保HTTPS通信正常，请跳转至“服务证书管理”页面导入信任证书，或在“配置项管理-证书认证配置”界面关闭ClientCertAuthentication，不校验对端证书。
 若需要校验证书的主机名或域名，跳转至“配置项管理-证书认证配置”界面开启ClientStrictCertAuthentication。
 HTTPS服务端默认关闭证书认证，如果需要打开，请跳转至“配置项管理-证书认证配置”中开启证书认证开关ServerFmRedfishServerCertAuthentication，并且跳转至“服务证书管理”界面，在信任证书页签导入设备信任证书。
 默认开启安全TLS算法套，若设备只支持不安全TLS算法套，请跳转至“配置项管理-TLS算法套”界面开启设备支持的不安全TLS算法套。

协议: ☒ Redfish

认证方式: Basic Auth

用户名: iccauser 密码:

端口: 443 超时时间 (秒): 20

取消 测试 确定 应用

图2-20

步骤 7 填写完成后，点击“测试”。测试成功后，点击右下角的“应用”。

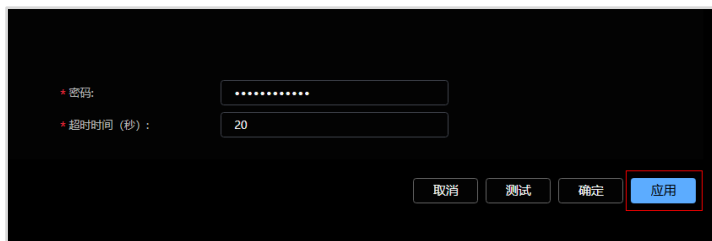


信息

协议	测试结果	测试详情
Redfish	连通性测试成功	连通性测试成功

确定

图2-21



密码:

超时时间 (秒): 20

取消 测试 确定 应用

图2-22

步骤 8 添加完成后，会有提示。



信息

✓ 添加设备成功

确定

图2-23

步骤 9 如果还有其他服务器需要添加，可以参照以上步骤，完成剩余服务器的添加。

添加完成后，点击“计算 > 服务器”，查看已经添加的服务器。



图2-24

步骤 10 添加完成后，点击“计算 > 部件管理”，查看服务器的 NPU 等信息。

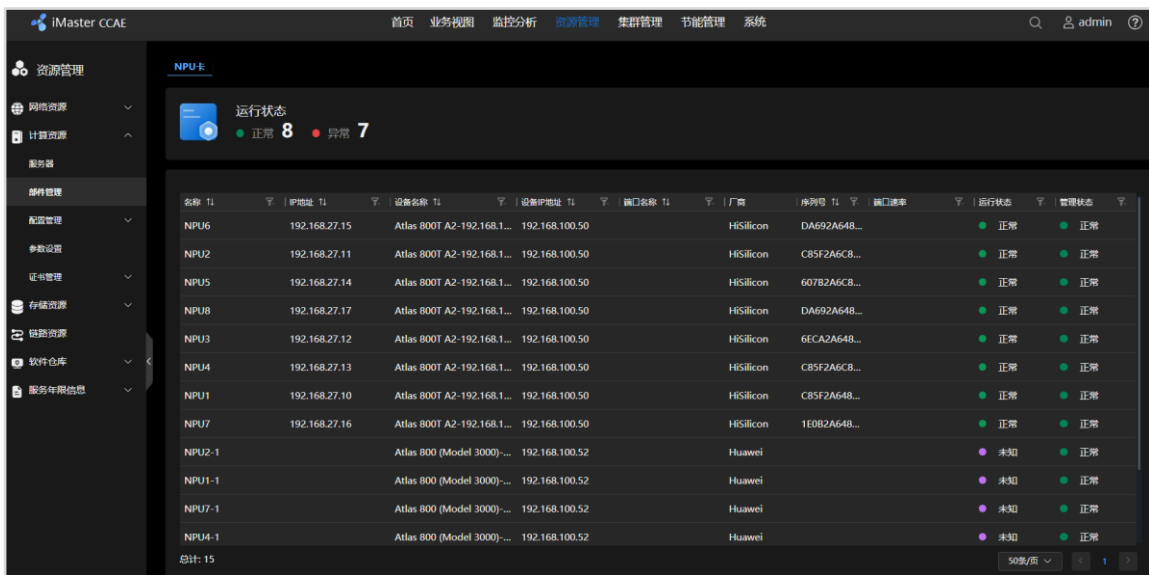


图2-25

步骤 11 还可以查看服务器状态，并支持上下电等操作。



图2-26

2.4.3 接入存储系统

当需要在 CCAE 中添加少量存储设备时，可以通过单个添加的方式将存储设备添加到 CCAE 中。当存储设备有多台且部署有 DME 管理系统时，推荐使用 DME 进行对接，本次实验以添加 DM 的方式进行演示。

步骤 1 通过浏览器访问存储的管理地址，例如 <https://xx.xx.xx.xx:8088>，登录存储设备的 DeviceManager（简称 DM）页面。

步骤 2 进入到添加用户的页面。例如本次分布式存储的路径为“设置 > 用户和安全 > 用户和角色管理 > 用户管理”。



图2-27

步骤 3 点击“增加”，新增对接用户，根据规划，填写对接用户，角色权限需要设置为“管理员”。填写完成后，点击“确定”。



图2-28

步骤 4 退出当前用户，使用新设置的用户进行登录并修改密码。



图2-29

步骤 5 使用新密码登录存储系统。



图2-30

步骤 6 在存储 DM 页面中，点击“设置 > SNMP 管理 > SNMP 协议”，在 USM 用户管理中，点击“增加”新建对接用户。



图2-31

步骤 7 设置 USM 用户信息。

增加USM用户

* 用户名: ?

ccaeuser

用户权限:

可读写

用户鉴权:

☒ 启用

鉴权算法:

SHA512

* 鉴权密码:

.....

* 确认鉴权密码:

.....

数据加密:

☒ 启用

加密算法:

AES256

* 数据加密密码:

.....

* 确认数据加密密码:

.....

图2-32

步骤 8 查看设置的 USM 用户信息。

SNMP协议

配置SNMP

通过配置SNMP，用户可以查询系统信息和接收系统的告警信息。SNMP支持SNMPv1、SNMPv2c和SNMPv3三种协议。

* 端口:

161

* 上下文名称: ?

Array

SNMPv1&SNMPv2c协议开关:

☐ 开启

USM用户管理

通过设置USM用户，可以为SNMPv3进行鉴权和数据加密。最多支持4个USM用户。

增加

修改

删除

☒

用户名

☒

用户权限

☒

鉴权算法

☒

加密算法

☒

操作

ccaeuser

可读写

SHA512

AES256

更多

总数: 1, 选中: 1

<

1

>

图2-33

步骤 9 CCAE 对接存储资源默认为 DME，如果采用 DM 进行对接，需要先切换场景，将 CCAE 存储资源管理对接系统选择为 DM。在 CCAE 业务面主菜单中选择“资源管理 > 存储资源 > 参数设置”，然后在“场景切换”区域的“对接系统”中，选择“DM”。设置完成后，单击“应用”。

说明：切换对接系统后，将清除当前已对接的存储数据，且无法恢复，请谨慎选择操作。



图2-34

步骤 10（可选）如果存储设备比较老，或某些原因，无法支持 SNMPv3，则需要在 SNMP 配置中，将此选项设置为“开启”。



图2-35

步骤 11 在 CCAE 业务面主菜单中选择“资源管理 > 存储资源 > 存储设备”。单击“单个添加”，进行 DM 设备的相关信息录入。



图2-36

步骤 12 根据实验环境，设置“基本信息”和“SNMP 参数”。设置完成后，点击“确定”。

图2-37

图2-38

The screenshot displays the 'Resource Management' (资源管理) interface of iMaster CCAE. The top navigation bar includes links for 'Home' (首页), 'Task View' (业务视图), 'Monitoring Analysis' (监控分析), 'Resource Management' (资源管理), 'Group Management' (集群管理), 'Energy Management' (节能管理), and 'System' (系统). The left sidebar lists various resource categories: 'Network Resources' (网络资源), 'Computing Resources' (计算资源), 'Storage Resources' (存储资源), 'Storage Devices' (存储设备), 'Snapshots' (快照管理), 'Naming Spaces' (命名空间), and 'Storage Pools' (存储池). The main content area shows a table of resources. A single resource is listed: 'OceanStor-Distributed-Storage' with IP address '192.168.101.205' and subnet '/'. The resource is marked as 'Online' (在线). The table also shows the device type 'OceanStor Pacific', total capacity '88.415 TB', used capacity '0.000 MB', and capacity utilization '0.00%'. A summary row at the bottom indicates 'Total: 1' (总计: 1) resource.

状态	名称	IP地址	子网	设备型号	总容量	已用容量	容量利用率
在线	OceanStor-Distributed-Storage	192.168.101.205	/	OceanStor Pacific	88.415 TB	0.000 MB	0.00%

总计: 1

图2-39

步骤 15 点击“存储设备 > 部件管理 > 硬盘/电源/节点/存储池/等”，可以查看对应的硬件设备信息。

The figure consists of four screenshots of the iMaster CCAE interface, demonstrating the '部件管理' (Component Management) section under '存储设备' (Storage Devices).

Screenshot 1: Storage Devices View

健康状态	名称	SN	固件版本	型号	容量	转速(RPM)	生产厂家	所属设备
健康	1volume10009	2102314BQGFSP70...	--	--	3.493 TB	--	--	OceanStor-Distrib...
健康	3volume10008	2102314BQGFSP70...	--	--	3.493 TB	--	--	OceanStor-Distrib...
健康	2volume10009	2102314BQGFSP70...	--	--	3.493 TB	--	--	OceanStor-Distrib...
健康	2volume10004	2102314BQGFSP70...	--	--	3.493 TB	--	--	OceanStor-Distrib...
健康	2volume10005	2102314BQGFSP70...	--	--	3.493 TB	--	--	OceanStor-Distrib...
健康	4volume10002	2102314BQGFSP70...	--	--	3.493 TB	--	--	OceanStor-Distrib...
健康	2volume10000	2102314BQGFSP70...	--	--	3.493 TB	--	--	OceanStor-Distrib...
健康	1volume10000	2102314BQGFSP70...	--	--	3.493 TB	--	--	OceanStor-Distrib...
健康	3volume10007	2102314BQGFSP70...	--	--	3.493 TB	--	--	OceanStor-Distrib...
健康	3volume10001	2102314BQGFSP70...	--	--	3.493 TB	--	--	OceanStor-Distrib...

总计: 40

Screenshot 2: Components View

健康状态	名称	所属设备	机框序列号	节点管理IP
正常	PSU1	OceanStor Distributed-Storage	--	--
正常	PSU0	OceanStor Distributed-Storage	--	--
不在位	PSU3	OceanStor Distributed-Storage	--	--
不在位	PSU2	OceanStor Distributed-Storage	--	--
正常	PSU5	OceanStor Distributed-Storage	--	--
正常	PSU4	OceanStor Distributed-Storage	--	--

总计: 6

Screenshot 3: Nodes View

健康状态	名称	所属设备	管理IP地址	存储IP地址	存储软件	序列号	机框	角色	备注
健康	Storage03	OceanStor-Distrib...	192.168.101.203	192.168.104.31,19...	已安装	2103050HER10QA...	C6-14	存储节点	--
健康	Storage04	OceanStor-Distrib...	192.168.101.204	192.168.104.41,19...	已安装	2103050HER10QA...	C6-14	存储节点	--
健康	Storage02	OceanStor-Distrib...	192.168.101.202	192.168.104.21,19...	已安装	2103050HER10QA...	C6-14	管理,存储节点	--
健康	Storage01	OceanStor-Distrib...	192.168.101.201	192.168.104.11,19...	已安装	2103050HER10QA...	C6-14	管理,存储节点	--

总计: 4

Screenshot 4: Storage Pools View

状态	名称	总容量	所属设备	已用容量	容量利用率	缓存类型
正常	StoragePool	88.415 TB	OceanStor-Distributed-Stor...	117.000 MB	0.00%	无缓存

总计: 1

图2-40

2.4.4 接入网络资源

CCAIE 支持通过南向接口，对接 NCE-Fabric 和 NCE-FabricInsight 网络管理平台，本次对接以 NCE-Fabric 为例，通过 CCAIE 跟 NCE-Fabric 的告警对接，实现 NCE-Fabric 告警报至 CCAIE。

步骤 1 通过浏览器访问“https://NCE-Fabric 的业务面 IP 地址:31943”，登录 NCE-Fabric。

步骤 2 单击界面左上方的“≡”图标，在菜单中选择“系统 > 系统设置 > 北向接口”。选择“SNMP 接口 > 基本配置”，并记录和设置基本配置信息。



图2-41

步骤 3 在“高级设置”区域中，记录“SNMPv3 协议参数配置”下的“鉴权协议”、“加密协议”和“Engine ID”的值。将“支持 RFC 规范”设置为“否”。点击“保存”保存参数配置。



图2-42

步骤 4 在左侧导航树中选择“SNMP 接口 > 第三方系统配置”，单击“创建”。



图2-43

步骤 5 根据实际环境配置，完成以下相关内容的填写。

IP 地址类型：选择为 IPv4

IPv4 地址：CCAIE 接入 NCE-Fabric 的南向网络平面的浮动 IP 地址

端口：10160

协议版本：SNMPv3

安全用户名：自定义，确保唯一性

授权认证密码：自定义

数据加密密码：自定义



图2-44

步骤 6 单击“展开过滤条件”，在“过滤模式”勾选为“上报”。完成后点击“创建”。

隐藏过滤条件

过滤模式 ?

☐ 屏蔽
☒ 上报

过滤条件生效... ?

☒ 满足任意条件
☐ 满足所有条件

告警级别 ?

☒ 紧急
☒ 重要
☒ 次要
☒ 提示

告警分类 ?

☒ 事件
☒ 故障告警
☒ 清除告警
☒ 确认告警
☒ 反确认告警
☒ 变更告警

工程告警 ?

☒ 工程态告警

告警制式 ?

告警源类型

图2-45

步骤 7 创建完成提示页面，点击“确定”。

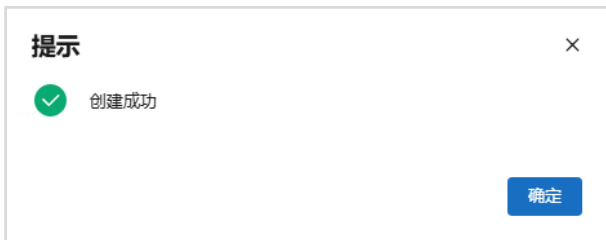


图2-46

步骤 8 创建三方系统接入用户，并将该用户赋予角色权限。单击界面左上方的“≡”，在菜单中选择“系统 > 管理员 > 用户管理”，在“用户”页签中单击“创建”。

默认工作组

概述

用户

用户组

角色

在线用户

请输入用户名、用户全名或者用户编...
 全部用户类型
 全部使用状态

刷新

删除

创建

☐	用户名 T1	用户全名 T1	用户类型 T1	描述 T1	登录状态	使用状态	操作
>	☐ admin		本地用户	管理员	已登录	正常	重置密码 删除 停用 复制

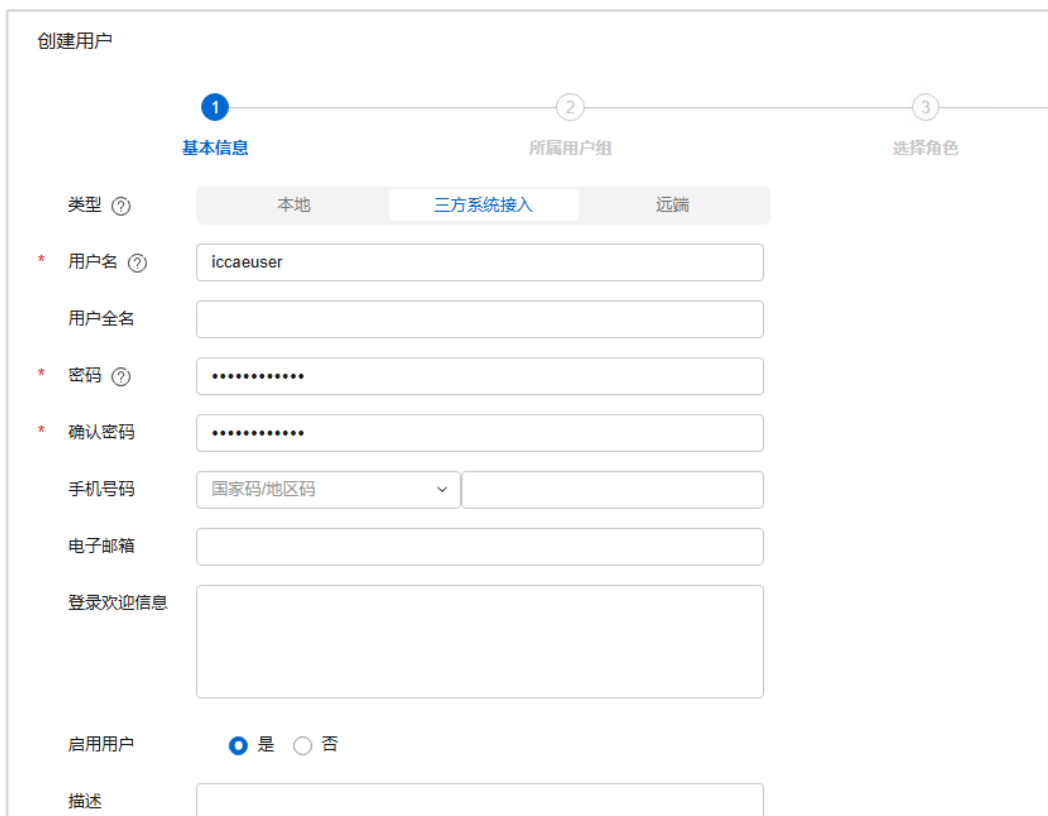
总计: 1

10条/页

< 1 >

图2-47

步骤 9 在创建新用户界面，“类型”选择“三方系统接入”。定义用户名和密码后，单击“下一步”。



创建用户

1 基本信息 2 所属用户组 3 选择角色

类型 ? 本地 三方系统接入 远端

* 用户名 ? iccaeuser

用户全名

* 密码 ?

* 确认密码

手机号码 国家码/地区码

电子邮箱

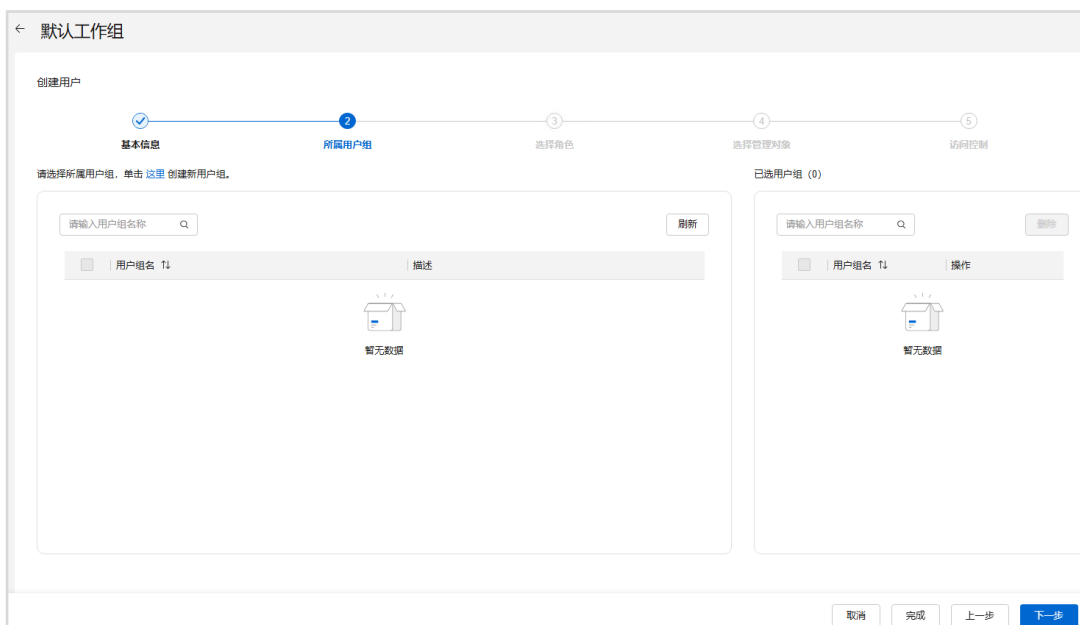
登录欢迎信息

启用用户 ☒ 是 ☐ 否

描述

图2-48

步骤 10 用户根据实际情况选择用户组，或根据界面提示创建用户组后再选择；也可以不设置，直接点“下一步”。



默认工作组

创建用户

1 基本信息 2 所属用户组 3 选择角色 4 选择管理对象 5 访问控制

请选择所属用户组，单击 [这里](#) 创建新用户组。

已选用户组 (0)

请输入用户组名称 刷新

用户组名 T1 描述

暂无数据

请输入用户组名称 删除

用户组名 T1 操作

暂无数据

取消 完成 上一步 下一步

图2-49

步骤 11 选择角色“NBI User Group”和“Northbound Interface Operator”，单击“完成”。

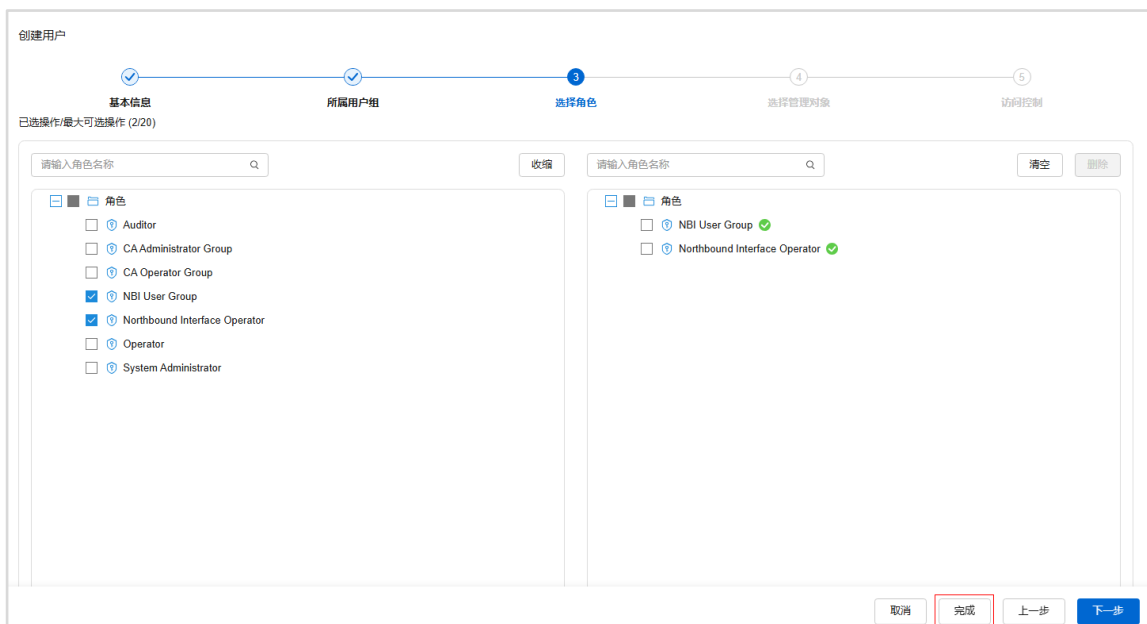


图2-50

步骤 12 登录 CCAE 业务面，在 CCAE 业务面主菜单中选择“系统 > 南向接入 > 接入系统设置”，单击“创建”。

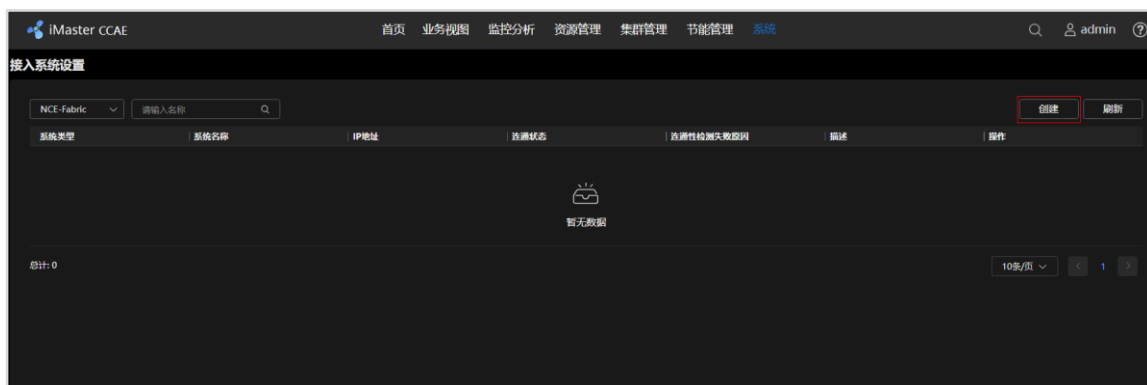


图2-51

步骤 13 在“创建系统”面板页面，根据实际环境配置创建 NCE-Fabric 连接的参数。分别完成 HTTP 和 SNMP 设置。相关配置请务必与 NCE-Fabric 中的参数相互匹配。

基本信息

* 系统类型

NCE-Fabric

* 系统名称

NCE-Fabric

SSO登录地址

请输入

描述

请输入

HTTP协议配置

* 协议类型

HTTPS

* IP地址

192.168.101.15

* 端口

18002

* 用户名

iccaeuser

* 用户密码

.....

图2-52

SNMP配置

* SNMP IP

192.168.101.15

* SNMP协议版本

SNMP v3

* 网元端口

9812

* 安全用户名

iccae

上下文

请输入

* 引擎ID

192.168.101.15

* 授权认证协议

HMAC-SHA2-512

* 授权认证密码

.....

* 数据加密协议

AES-256

* 数据加密密码

.....

* 超时时间 (秒)

5

* 重试次数

3

作为客户端访问CCA

!

客户端IP与服务IP不同时填写，多个IP使用分号间隔。

图2-53

步骤 14 在“接入系统设置”界面，单击该配置所在操列的“”，可以进行连通性测试，验证 CCAE 对接 NCE-Fabric 的 HTTPS 协议的连通性。



图2-54

步骤 15（可选）如果遇到报错，可根据提示进行问题分析，并进行调整。

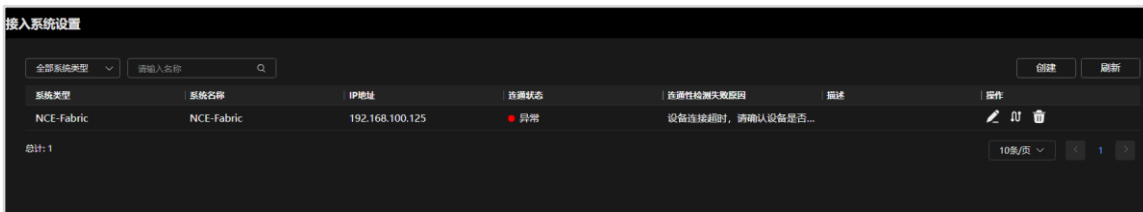


图2-55

步骤 16 进行 SNMP 告警发送测试。在 NCE-Fabric 菜单中选择“系统 > 系统设置 > 北向接口”。在左侧导航树中选择“SNMP 接口 > 第三方系统配置”，在对接 CCAE 的 IP 地址所在行，单击“”发送测试告警。

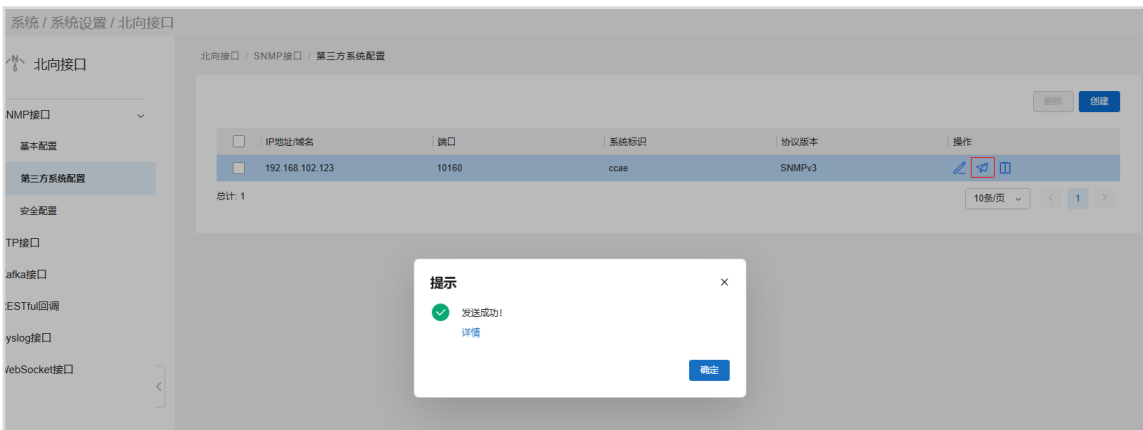


图2-56

步骤 17 在 CCAE 业务面上查看是否存在 NCE-Fabric 发送的测试告警。在 CCAE 业务面主菜单中选择“监控分析 > 告警管理 > 当前告警”，查看是否存在“北向接口测试告警”。如果配置都正确，可以看到有个告警提示，名称为“北向接口测试告警”。



图2-57

2.4.5 接入 AI 集群资源

CCAE 系统接入 AI 集群资源有两种方式，分别为：在线安装方式，适用于 CCAE 与计算节点网络双向连通的场景；离线安装方式，适用于 CCAE 与计算节点网络单向连通的场景。本次实验环境以在线安装方式为例进行讲解。

操作流程如下，供参考：

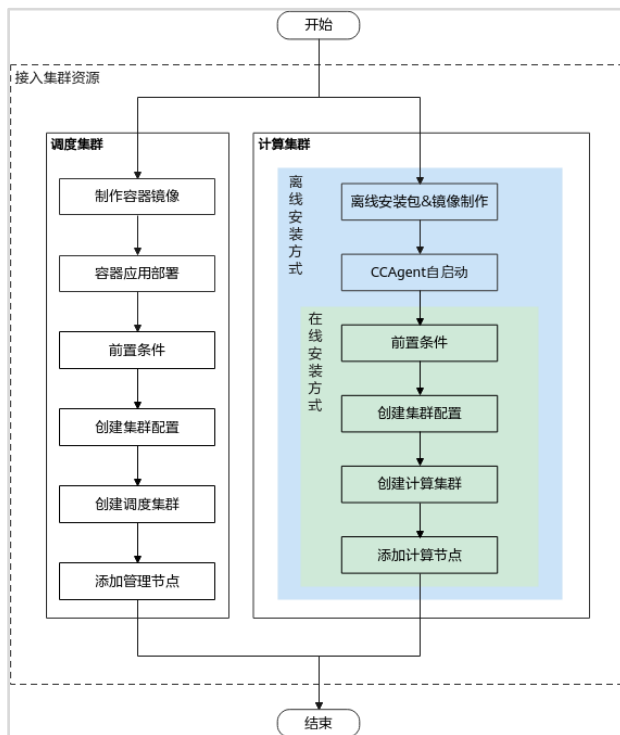


图2-58 操作流程图

步骤 2 创建 AI 集群配置。在 CCAE 业务面主菜单中选择“集群管理 > AI 集群 > 参数配置”。并在“参数配置”界面，单击“创建”。



图2-59

步骤 3 在“创建”界面进行集群参数配置。名称自定义，其他参数可以默认也可以根据需要进行调整。配置完成后，单击“确定”。

基本信息

名称

AI_计算

集群类型

AI计算集群

描述

参数配置

CCAgent运行配置

心跳周期 (秒)

20

?

SFTP协议上传限流

0KB/s

?

AI日志回收

启用

Device侧日志路径

操作系统日志路径

CANN应用日志路径

用户训练日志路径

磁盘IO采集

启用

HBM信息采集

启用

图2-60

步骤 4 在 CCAE 业务面主菜单中选择“集群管理 > AI 集群 > 集群资源”。在左侧导航中选择“集群资源 > AI 计算集群”。在“AI 计算集群”界面，单击“创建”。



图2-61

步骤 5 在“创建集群”面板，填写集群名称并选择集群配置信息。在所属 AI 调度集群项为可选项，可以根据需要进行选择。配置完成后，单击“确定”。



图2-62

步骤 6 在 CCAE 业务面主菜单中选择“资源管理 > 软件仓库 > 软件包”。在“软件包”页面选择“升级包”页签。

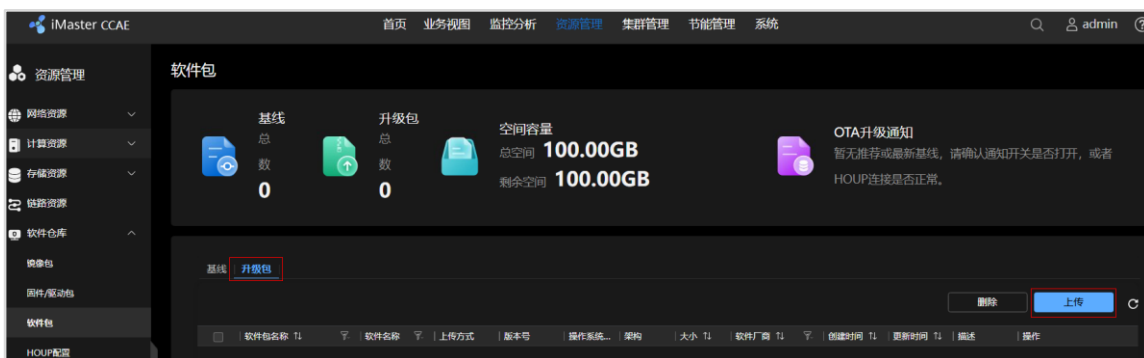


图2-63

步骤 7 根据计算节点的操作系统架构类型，上传对应的 CCAgent 软件包和数字签名文件。以下软件包供参考。

架构	软件包名称	数字签名文件
ARM架构 物理机	CCAIE_V100R024C10SPC101_Agent_aarch64.zip	CCAIE_V100R024C10SPC101_Agent_aarch64.zip.asc

X86架构 虚拟机	CCAE_V100R024C10SPC101_Agent_x86_64.zip	CCAE_V100R024C10SPC101_Agent_x86_64.zip.asc
--------------	---	---

完成软件上传后，点击“确定”。

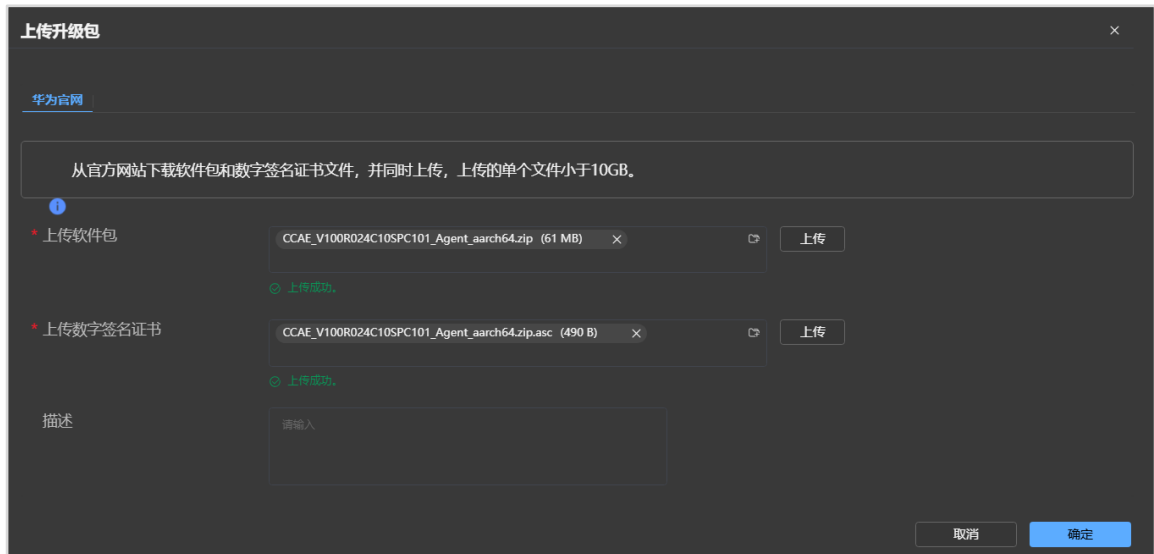


图2-64

步骤 8 上传完成后，刷新页面，可以看到上传的文件。

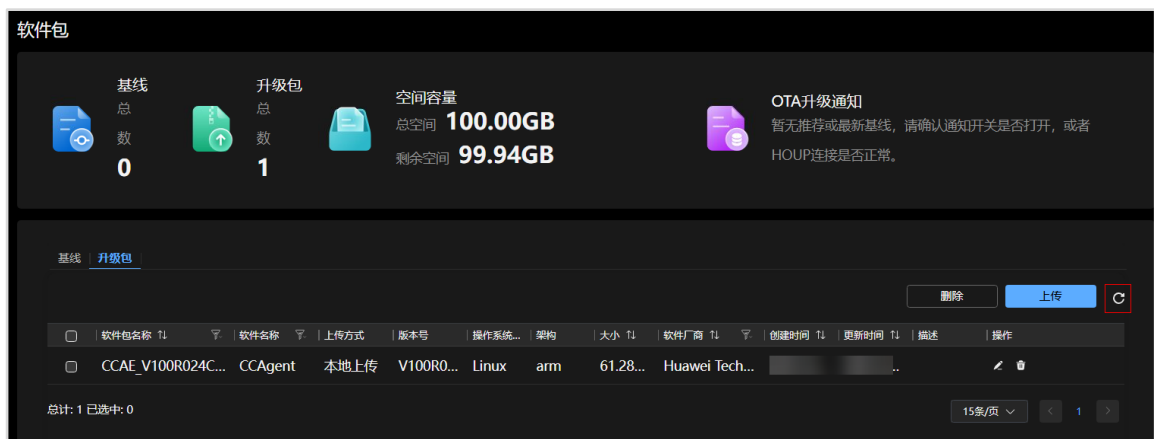


图2-65

步骤 9 在 CCAE 业务面主菜单中选择“集群管理 > AI 集群 > 集群资源”。在左侧导航中选择“集群资源 > AI 计算集群”。



图2-66

步骤 10 在“AI 计算集群”界面，在需要添加节点的集群所在行，单击集群名称前面的“”展开该集群，然后单击“添加节点”。

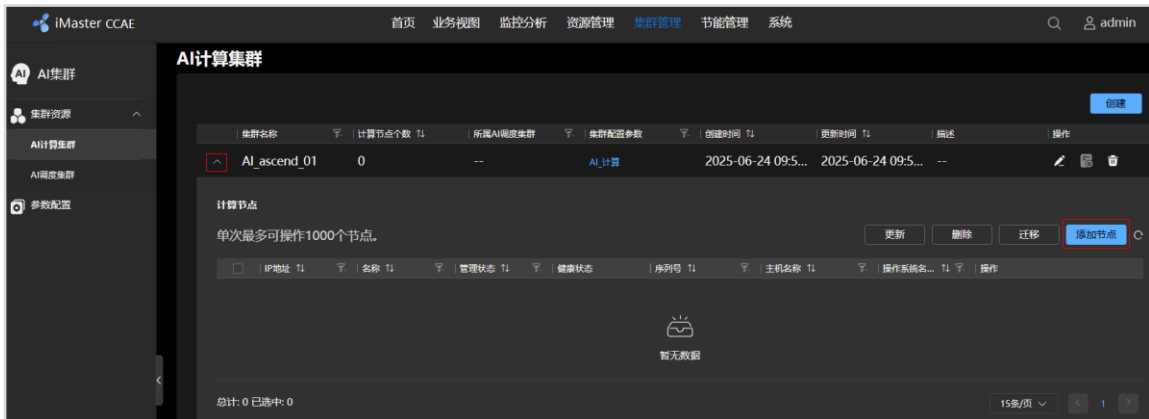


图2-67

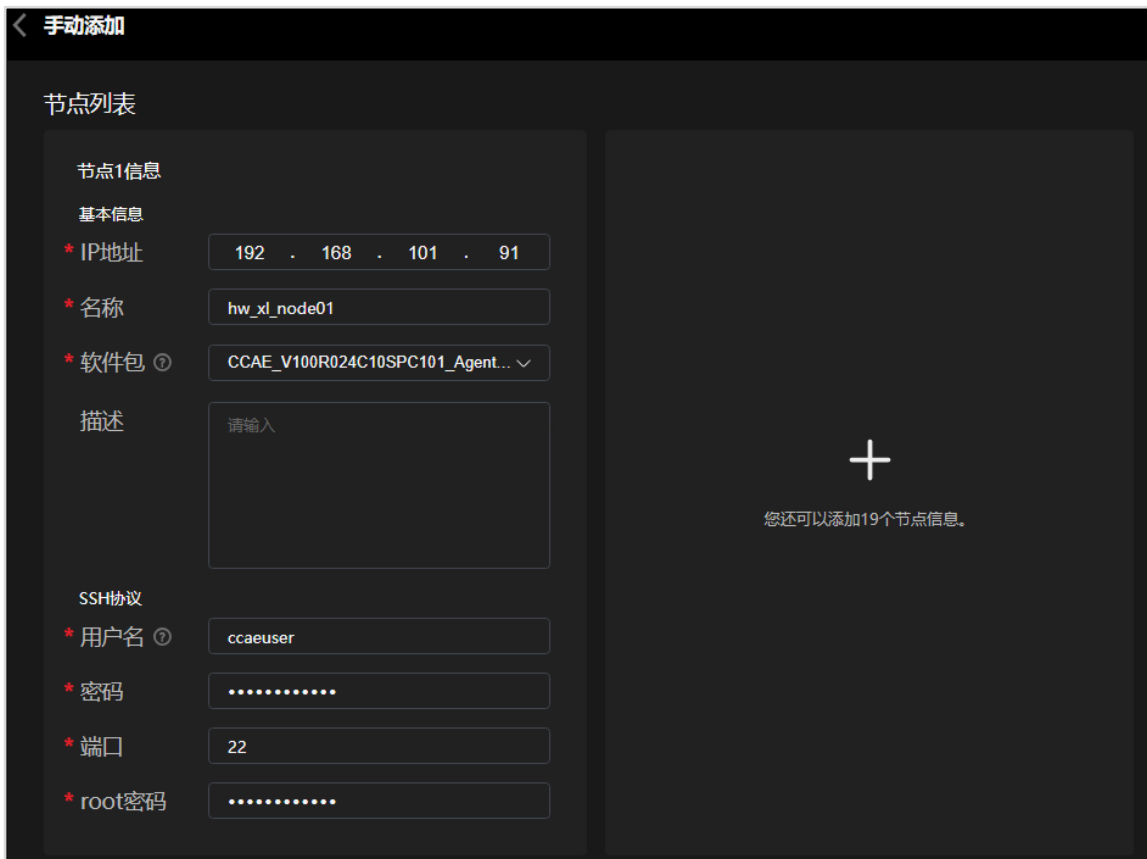
步骤 11 在“添加节点”弹窗中选择添加方式，本次以“手动添加”为例。



图2-68

步骤 12 在配置页面，根据提示，完成相关参数填写。此处的用户名，填写前边为 CCAgent 创建的户即可。确认无误后，点击确定。

说明：添加前，请确保系统时间跟 CCAE 平台时间保持一致。



手动添加

节点列表

节点1信息

基本信息

* IP地址: 192 . 168 . 101 . 91

* 名称: hw_xl_node01

* 软件包 ⑦: CCAE_V100R024C10SPC101_Agent... ▾

描述: 请输入

SSH协议

* 用户名 ⑦: ccaeuser

* 密码:

* 端口: 22

* root密码:

您还可以添加19个节点信息。

图2-69

步骤 13 添加计算节点任务会出现在“当前任务”。添加成功后，在集群详情界面显示添加后的计算节点的详细信息。

在“系统 > 任务管理 > 当前任务”页面可以看到添加的进度。



iMaster CCAE 首页 业务视图 监控分析 资源管理 集群管理 节能管理 系统

任务管理

当前任务

任务名称	操作对象	子任务	任务类型	创建人	创建时间	TL	进度	操作
添加节点	计算节点	2	添加节点	admin			100%	
子任务列表: 进行中 1 ● 等待中 0 ● 成功 1 ● 失败 0 ●								
任务名称	操作对象	任务类型	进度	错误原因	处理建议	耗时	操作	
添加节点	192.168.101.91	添加节点	进行中 15%			1min2s	📄	
同步资源	资源	添加节点	成功			8s	📄	
总计: 2								

15条/页 1

图2-70

步骤 14 节点添加完成后，“AI 计算集群”界面在已添加节点的集群所在行，单击集群名称前面的展开该集群，可查看已添加的计算节点，“管理状态”为“就绪”表示节点添加成功。



图2-71

2.5 结果验证

2.5.1 告警查看

对接完成后，可以在 CCAE 平台进行告警或资源等信息的查看，还可以进行健康检查等操作。

步骤 1 登录 CCAE 的业务面，选择“监控分析 > 告警管理 > 当前告警”，可以查看和处理告警等信息。



图2-72

步骤 2 单击告警名称，可以查看详细。

iMaster CCAE									
首页 业务视图 监控分析 资源管理 集群管理 节能管理 系统									
模板管理 过滤									
自动刷新 快速过滤 组合排序									
	操作	级别	告警流水号	告警ID	名称	告警源	来源系统类型	来源系统	
☒		重要	14694	19660818	网卡光模块的功率异常严重告警	OceanStor-Distrib...	NMS_Storage	存储	
☒		重要	17324	687865873	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	
☒		重要	17352	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	
☒		重要	17351	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	
☒		重要	17350	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	
☒		重要	17323	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	
☒		重要	17322	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	
☒		重要	17321	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	
☒		重要	17349	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	
☒		重要	9110	687865871	网卡光模块电压异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	

图2-73

步骤 3 在告警详情页面，除了告警记录的查看外，还可以查看历史维护经验和最近处理记录，并可以对告警进行“备注、清除或确认”等操作。

告警详情

告警详情和处理建议

维护经验

最近2个月处理记录

基本信息

名称

网卡光模块的功率异常严重告警

告警源

Atlas 800T A2-192.168.100.50

来源系统

服务器

领域

首次发生时间

持续时间

--

级别

重要

附加信息

修复建议

-

产生原因和修复建议

告警源类型

Atlas 800T A2

来源系统类型

NMS_Server

厂商

Huawei

发生时间

次数

1

类型

设备告警

定位信息

事件主体=Port2--FLEX IO B4.Port 2(Channel 3 Tx Power)

产生原因

光模块故障。

处理状态

清除状态

清除状态: 未清除

确认状态

确认状态: 未确认

备注

单击编辑备注信息。

清除

确认

关闭

图2-74

步骤 4 选择“监控分析 > 告警管理 > 告警日志”，可以查看相关告警日志。

iMaster CCAE									
首页 业务视图 监控分析 资源管理 集群管理 节能管理 系统									
模板管理 过滤									
组合排序 导出 备注 发送通知									
	操作	级别	告警流水号	告警ID	名称	告警源	来源系统类型	来源系统	定位信息
☒		紧急	9325	800301	接入系统断连	OSS	OSS	OSS	系统IP=192.168.100.125...
☒		重要	14694	19660818	SNMP 连接失败	OceanStor-Distrib...	NMS_Storage	存储	网元=OceanStor-Distrib...
☒		重要	17324	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17352	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17351	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17350	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17323	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17322	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17321	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17349	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	9110	687865871	网卡光模块电压异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17376	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port2--FLEX L...
☒		重要	17348	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port2--FLEX L...
☒		重要	17347	687865873	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port2--FLEX L...
☒		重要	17346	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port2--FLEX L...
☒		重要	17345	687865873	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port2--FLEX L...
☒		重要	17375	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port2--FLEX L...
☒		重要	17374	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port2--FLEX L...
☒		重要	17373	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port2--FLEX L...
☒		重要	17332	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17331	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17330	687865873	网卡光模块电流异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...
☒		重要	17320	687865871	网卡光模块电压异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port2--FLEX L...
☒		重要	17360	687865879	网卡光模块的功率异常严重告警	Atlas 800T A2-192...	NMS_Server	服务器	事件主体=Port1--FLEX L...

图2-75

步骤 5 选择“监控分析 > 计算资产管理 > 资产概览”，可以查看服务器等资产的概览信息及变化数据。



图2-76

步骤 6 选择“监控分析 > 计算资产管理 > 资产明细”。可以查看接入的服务器等资产信息。可以对 CPU、内存、硬盘等硬件信息进行统计，并支持导出为资产表格。

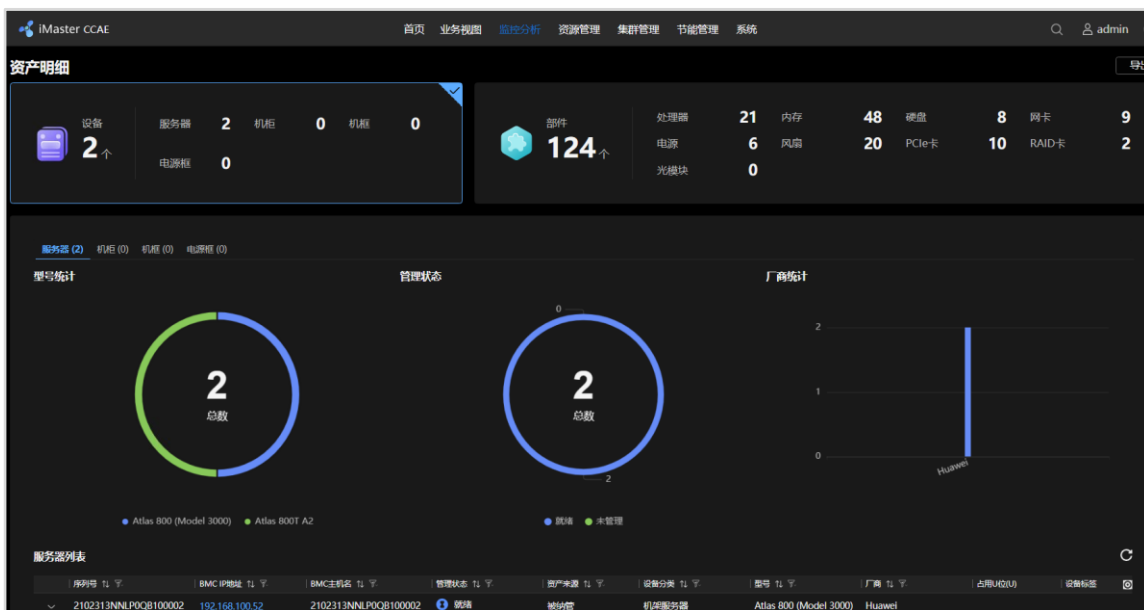


图2-77

步骤 7 选择“资源管理 > 软件仓库 > 软件包”，可以对基线/升级包进行管理，并支持进行 OTA 相关的升级管理。



图2-78

步骤 8 选择“业务视图 > 健康检查”，点击“创建”按钮，可以直接创建监控检查。



图2-79

步骤 9 通过执行健康检查，可以对纳管的设备和固件等信息进行健康检测。



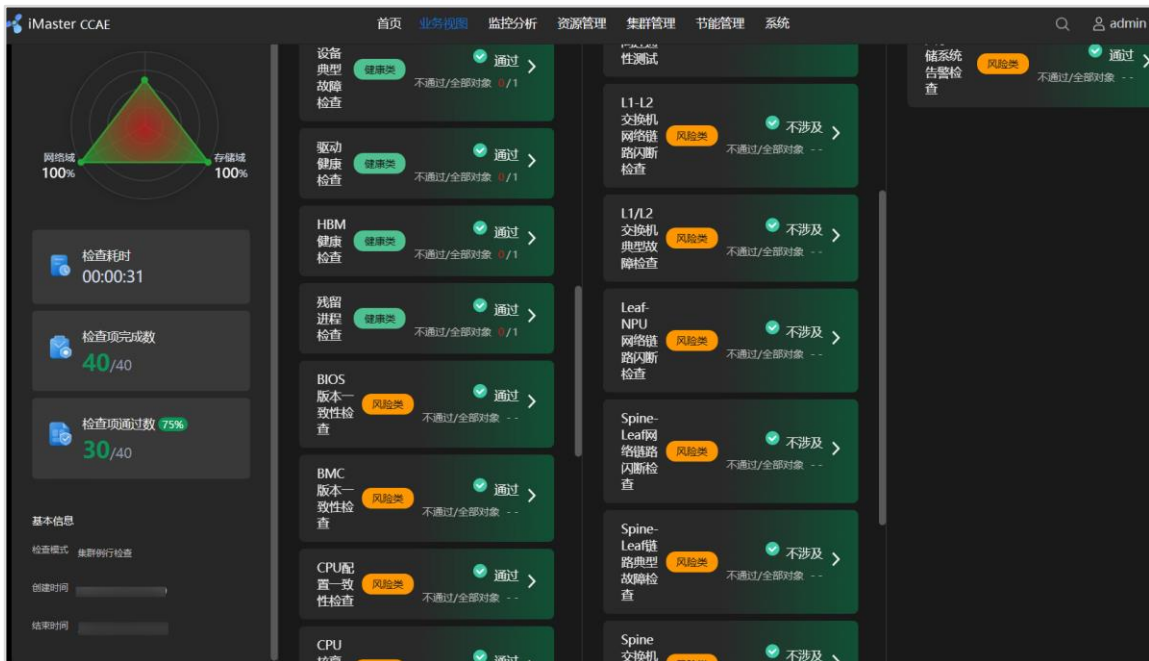


图2-80

步骤 10 节能管理。选择“节能管理 > 总体趋势/用能特征”，可以进行设备能耗相关的查看或管理。

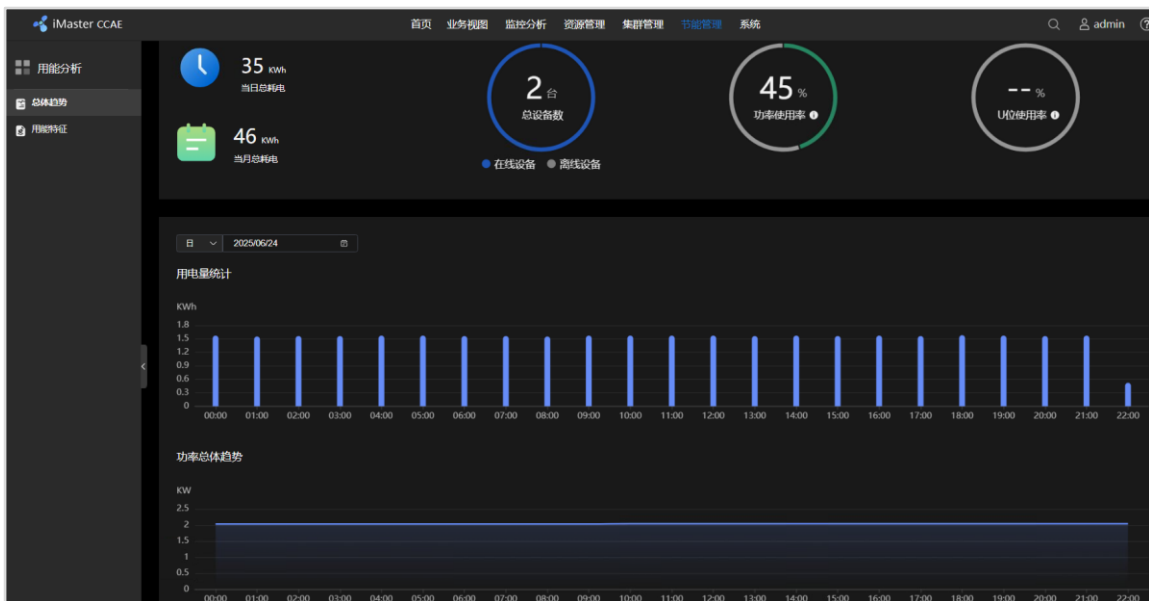


图2-81

2.6 思考题

在接入 AI 集群的时候 Agent 可以安装在哪些节点，安装 Agent 有什么作用？

CCAgent 支持安装在两种集群中的节点上，根据安装位置不同，完成不同的业务功能，如下：

管理节点：主要用于与 MindCluster 中的 ClusterD 组件交互，获取 AI 作业信息，包括作业 ID、作业起止时间、训练框架类型、作业调度的计算节点及卡信息，用于支撑 CCAE 基于作业的故障诊断、作业拓扑、作业日志、劣化诊断等高级特性。

计算节点：主要用于采集计算节点操作系统及 NPU 指标数据，支撑故障诊断、健康检查、日志分析、劣化诊断等高级特性。

