

全爱科技后羿智盒 HOUYI-PI-B

用户指南

文档版本 01

发布日期 2025-10-20



全爱科技（上海）有限公司

版权所有 全爱科技（上海）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



和其他全爱商标均为全爱科技（上海）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、服务或特性等应受全爱科技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，全爱公司对本文档内容不做任何明示或默示的声明或保证。

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

全爱科技（上海）有限公司

地址：上海市闵行区剑川路920号2栋3层 邮编：200240

网址：www.quanaichina.com

前言

概述

本文档详细介绍了 HOUYI-PI-B 的产品特点，系统架构，产品规格，接口测试等。使得客户对 HOUYI-PI-B 有更加深入的了解。

读者对象

本文档主要适用于以下人员

- 用户企业开发人员，售前人员
- AI 开发者与研究人员

文档更新记录

版本	日期	更新记录
01	2025. 10. 20	初版发布

支持版本如下表：

操作系统版本	Ubuntu 22.04
全爱科技 硬件产品：	全爱科技后羿智盒 HOUYI-PI-B
产品账号密码 (root 账号不支持 远程和界面登陆)	账号：root 密码：12345678 账号：dev 密码：12345678

目 录

- 1 安全5
 - 1.1 电气安全5
 - 1.2 操作安全5
 - 1.3 软件安全6
 - 1.4 维护与处置6
 - 1.5 其他注意事项6
- 2 产品简介7
 - 2.1 产品概述7
 - 2.2 产品特点7
- 3 接口测试9
 - 3.1 NVME 接口测试10
 - 3.2 USB 接口测试10
 - 3.3 DP 测试10
 - 3.4 emmc 测试11
 - 3.5 网口测试11
 - 3.6 IIC 测试12
 - 3.6.1 编译操作12
 - 3.6.2 调试操作12
- 4 环境安装15
 - 4.1 环境要求15
 - 4.2 软件清单15
 - 4.3 依赖关系16
 - 4.4 安装与部署16
 - 4.4.1 准备工作16
 - 4.4.2 系统基础环境18
 - 4.4.3 AI 基础环境19
 - 4.4.4 基础环境验证20

- 4.5 常用命令20
 - 4.5.1 命令清单 20
 - 4.5.2 命令操作 21
- 4.6 常见问题23
 - 4.6.1 问题清单 23
 - 4.6.2 解决方法 24
- 5 开发资料导航 28
 - 5.1 资料网站28
 - 5.2 使用 WinSCP 传输文件28

1 安全

1.1 电气安全

电源要求

- 必须使用官方提供的 12V/5A 电源适配器，严禁使用不符合规格的电源
- 连接电源前，确保电源电压与当地电网电压一致
- 电源适配器应放置在通风良好的位置，避免覆盖任何物品
- 插拔电源时应握住插头部分，而非电源线

防触电措施

- 操作开发板前确保手部干燥，避免在潮湿环境中使用
- 开发板运行时，请勿触摸电路板上的元器件和接口引脚
- 严禁在开发板通电状态下进行插拔扩展模块等操作
- 如遇液体泼溅到设备上，应立即断开电源并联系技术支持

1.2 操作安全

机械安装

- 安装或拆卸扩展模块时，应先断开电源
- 连接 M 摄像头等接口时，注意对齐接口方向，避免用力过猛损坏针脚
- 开发板应固定在不平穩表面，避免跌落或碰撞
- 请勿在开发板上放置任何重物

环境要求

- 工作环境温度应保持在 0 °C 至 40 °C 之间
- 避免在多尘、潮湿或有腐蚀性气体的环境中使用
- 远离强电磁干扰源，如大功率电机、变压器等
- 确保设备周围有足够空间散热，至少预留 10cm 通风空间

1.3 软件安全

系统安全

- 开发板接入网络时，应配置适当的防火墙规则
- 重要数据应定期备份，防止意外丢失

权限管理

- 避免长期使用 root 权限进行开发工作
- 为不同开发任务创建专用用户账户，并分配适当权限
- 对外提供网络服务时，限制不必要的端口开放

1.4 维护与处置

日常维护

- 清洁设备表面时，应先断开电源，使用干燥柔软的布料擦拭
- 如需清洁接口，可使用专用电子清洁剂和软毛刷
- 长期不使用时，应断开电源并妥善存放

故障处理

- 如发现设备异常发热、有异味或异响，应立即断开电源
- 遇到硬件故障时，请勿自行拆解设备，应联系官方技术支持

废弃处置

- 本设备包含电子元器件，应按照当地环保规定进行废弃处理
- 处置前请删除设备中存储的所有敏感数据

1.5 其他注意事项

- 开发板不具备防水功能，请勿在可能接触液体的环境中使用
- 禁止私自改装设备或更换非官方配件
- 设备运行时，避免堵塞散热孔
- 运输过程中应使用原包装或具备防震保护的包装材料
- 遵循以上安全注意事项可确保设备正常运行，减少故障风险，并保障使用者的人身安全。

2 产品简介

2.1 产品概述

全爱科技后羿智盒 HOUYI-PI-B 用于帮助开发者完成全功能、多形态的 AI 应用开发与设计评估，最大可提供 20TOPS INT8 的计算能力。

全爱科技后羿智盒 HOUYI-PI-B 可以实现语音、图像与视频等多种数据分析与推理计算，可广泛用于智能监控、机器人、无人机、视频服务器等场景。

2.2 产品特点

- CPU 12 核/8 核 2.65 GHz
- 最大可提供 50 TOPS INT8 算力，适用于 AI 推理及高性能计算场景。
- 支持多路 H.264/H.265 硬件编解码：
解码：2*4K 60fps。
编码： 2*8K 30fps。

多元接口配置

- USB 接口：有 2 组 USB3.1 接口。这些接口可以连接多种外部设备，如鼠标、键盘、移动硬盘、U 盘等，方便数据传输和设备扩展，比如开发者可以通过 USB 接口将训练好的模型数据存储到移动硬盘中。
- 显示接口：配备 DP 接口，可提供 4K 高分辨率显示，能够满足对显示画质有较高要求的场景，如连接高清显示器进行模型训练结果展示、数据可视化等工作。
- 网络通信 2 个千兆网络接口，为设备网络通信提供坚实保障。无论是在智能工厂中，与生产线上的其他设备进行数据交互，协同完成生产流程；还是在远程办公场景下，实现高效的远程调试与控制，都能确保数据传输的高速与稳定，减少延迟与卡顿。满足多样化的联网需求。

丰富软件生态

- 主流模型支持 HOUYI-PI-B 对主流大语言模型和机器视觉模型展现出良好兼容性。DeepSeek、Qwen、Llama 等大语言模型，以及 YOLO、RESNET 等机器视觉模型，均可在该设备上稳定运行。开发者无需在模型适配环节耗费大量精力。能将更多时间与精力投入大模型优化与应用开发中，加速项目进程。
- 高参数量模型处理尤为值得一提的是，它能够流畅运行高达 32B 参数量的语言模型。在自然语言处理任务里，如智能写作、语义理解、机器翻译等场景，该能力确保模型对复杂语言结构与语义关系进行精准解析与生成，极大提升应用的智能化水平与用户体验。

3 接口测试

接口测试说明：进行接口测试需要进入板卡系统，进入系统的操作步骤请参考快速开始文档

注：接口描述详情可参考产品白皮书

个接口连接如图所示：



3.1 NVME 接口测试

进入板卡输入 `fdisk -l`，可查看到一个 nvme 存储盘。

```
(base) root@davinci-mini:~# fdisk -l
Disk /dev/mmcblk0: 29.12 GiB, 31268536320 bytes, 61071360 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mmcblk1: 57.64 GiB, 61891149824 bytes, 120881152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: FA2F9662-4F6F-47EC-988D-451ACDE66435

Device            Start      End  Sectors  Size Type
/dev/mmcblk1p1      559104    561151     2048    1M Linux filesystem
/dev/mmcblk1p2     561152 120776703 120215552 57.3G Linux filesystem
/dev/mmcblk1p3 120776704 120879103    102400    50M Microsoft basic data

Disk /dev/nvme0n1: 465.76 GiB, 500107862016 bytes, 976773168 sectors
Disk model: ZHITAI Ti600 500GB
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

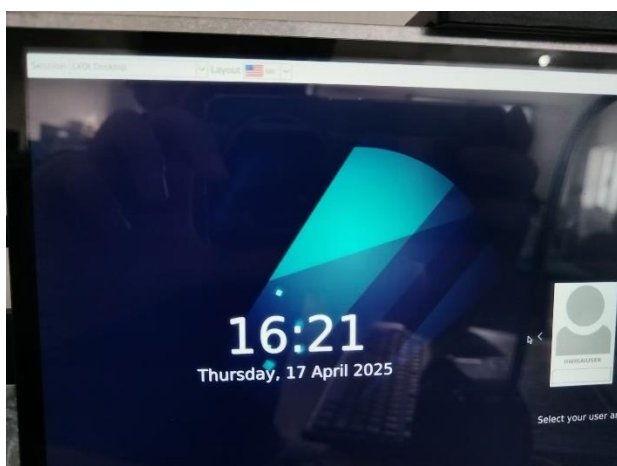
3.2 USB 接口测试

将 U 盘插入开发者套件中，输入 `lsusb` 可以识别 U 盘

```
(base) root@davinci-mini:/opt# lsusb
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 002: ID 17ef:38ac Lenovo
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
(base) root@davinci-mini:/opt#
```

3.3 DP 测试

DP 接口接入显示器，启动后会有登录界面显示



3.4 emmc 测试

使用 `fdisk -l` 查看 emmc 设备

```
(base) root@davinci-mini:~# fdisk -l
Disk /dev/mmcblk0: 29.12 GiB, 31268536320 bytes, 61071360 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mmcblk1: 57.64 GiB, 61891149824 bytes, 120881152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: FA2F9662-4F6F-47EC-988D-451ACDE66435

Device            Start      End  Sectors  Size Type
/dev/mmcblk1p1     559104    561151    2048    1M Linux filesystem
/dev/mmcblk1p2     561152 120776703 120215552 57.3G Linux filesystem
/dev/mmcblk1p3 120776704 120879103   102400    50M Microsoft basic data

Disk /dev/nvme0n1: 465.76 GiB, 500107862016 bytes, 976773168 sectors
Disk model: ZHITAI Ti600 500GB
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

3.5 网口测试

- 使用网线连接 PC 机网口板卡 eth0, 配置 PC 网络链接 IP: 192.168.2.10
- 使用 `ifconfig eth0 192.168.2.100` 配置网络 IP
- PC 机 Ping 192.168.2.100 是可以 ping 通的
- 同理板卡 ping 192.168.2.10 也可以 ping 通

- 使用网线连接 PC 机网口板卡 eth1, 配置 PC 网络链接 IP: 192.168.137.10
- PC 机 Ping 192.168.137.10 是可以 ping 通的

- 同理板卡 ping 192.168.137.100 也可以 ping 通

3.6 IIC 测试

IIC 源码在 i2c-tools-4.3 文件夹下

3.6.1 编译操作

访问如下链接，下载 i2c-tools-4.3.tar.gz。

<https://mirrors.edge.kernel.org/pub/software/utils/i2c-tools/>

登录 Arm 架构的 Linux 服务器。

执行如下命令，切换至 root 用户。

```
su - root
```

使用“WinSCP”，将软件包“i2c-tools-4.3.tar.gz”上传至 Linux 系统 root 用户属组目录下，例如/opt。详细操作请参见 11.2 使用 WinSCP 传输文件。

执行如下命令，进入源码包所在目录，例如/opt。

```
cd /opt
```

执行以下命令，解压 i2c-tools-4.3.tar.gz。

```
tar -zxvf i2c-tools-4.3.tar.gz
```

执行以下命令，进入 i2c-tools-4.3 目录。

```
cd i2c-tools-4.3
```

执行以下命令进行编译。

```
make USE_STATIC_LIB=1
```

编译成功后，在当前目录的子目录“tools”下会生成对应的可执行文件“i2cdetect”

与“i2ctransfer”。

3.6.2 调试操作

1、登录待调试 I 2C 的环境。

2、执行如下命令，切换至 root 用户。

```
su - root
```

- 3、使用“WinSCP”，将 i2cdetect 与 i2ctransfer 上传至 root 用户属组目录下，例如/opt。
- 4、执行以下命令，添加可执行权限。

```
chmod +x i2ctransfer
```

```
chmod +x i2cdetect
```

- 5、执行以下命令，查看系统的 I 2C 总线信息。

```
./i2cdetect -l
```

结果如下所示

```
i2c-0 i2c HiSilicon I2C Controller 300170000.i2c0 I2C adapter
i2c-1 i2c HiSilicon I2C Controller 300180000.i2c1 I2C adapter
i2c-2 i2c HiSilicon I2C Controller 300190000.i2c2 I2C adapter
i2c-3 i2c HiSilicon I2C Controller 3001a0000.i2c3 I2C adapter
i2c-5 i2c HiSilicon I2C Controller c4170000.i2c5 I2C adapter
i2c-6 i2c HiSilicon I2C Controller c4180000.i2c6 I2C adapter
i2c-7 i2c HiSilicon I2C Controller 82060000.i2c7 I2C adapter
i2c-8 i2c HiSilicon I2C Controller 82070000.i2c8 I2C adapter
i2c-9 i2c HiSilicon I2C Controller c4190000.i2c9 I2C adapter
i2c-13 i2c HiSilicon I2C Controller 401040000.i2c13 I2C
adapter
```

- 6、执行以下命令，使用 i2cdetect -y 查询 i2c 上挂载的设备。

```
./i2cdetect -y 5
```

结果如下所示

```
0 1 2 3 4 5 6 7 8 9 a b c d e f
00: -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```

30: -- -- UU -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --

```

说明： 当前表明只有 0x32 挂载有设备。

7、执行以下命令，写 I 2C。

```
./i2ctransfer -y -f 8 w3@0x50 0x00 0x80 0xdf
```

字段说明

8: I2C 设备号。

w3 : 表示写 3 个值，前两位为对应 16 位寄存器地址，之后则为要写的值。

0x50: I2C 从设备地址。

0x00 0x80 : 表示往 0x0080 地址写值。

0xdf : 表示给对应寄存器写 0xdf。

8、执行以下命令，读 I 2C。

```
./i2ctransfer -y -f 8 w2@0x50 0x00 0x80 r1
```

字段说明

8 : I 2C 的总线号。

w2: 表示写 2 个值，即要读的对应 16 位寄存器地址。

0x50: I 2C 从设备地址。

0x00 0x80: 表示读取 0x0080 地址的值。

r1: 表示从寄存器开始读取 1 个字节。

4 环境安装

4.1 环境要求

(出厂已安装，如无必要不用重复安装)

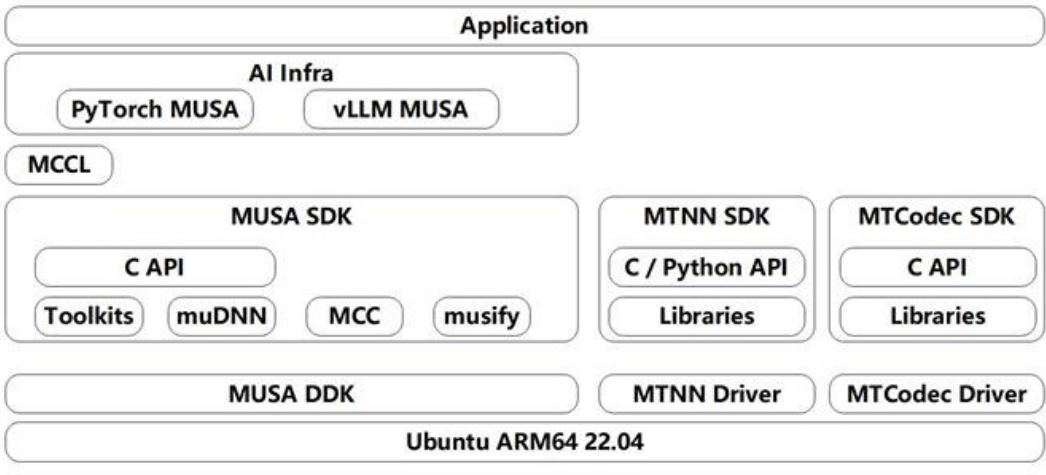
- OS 版本：UbuntuARM6422.04
- MUSADDK 版本：3.1.0
- MUSADDK 版：4.1.2
- Python 版本：3.10
- PyTorch 系列：pytorch-2.5.0|torch_musa-2.1.0
- vLLM 系列：vllm-0.7.4|vllm_musa-1.0|flash_attn-2.6.3|triton-3.1.0

4.2 软件清单

序号	名字	类别	必装	描述	详见
1	Third-PartyOSS	系统	☒	第三方开源依赖： git mpi blas numa stdc++12 等	点击链接
2	MUSADDK	系统	☒	MUSA 驱动 运行时 c	点击链接
3	MUSASDK	系统	☒	MUSA 开发包： mcc toolkits mudnn musify	点击链接
4	MCCL	系统	☒	MUSA 通信开发包	点击链接
5	MTNNSDK	系统	☒	NPU 驱动 运行时 开发包	点击链接
6	MTCCodecSDK	系统	☒	VPU 驱动 FFMPEG 开发包	点击链接
7	mthreads-smi	系统	☒	查询 BIOS CPU GPU VPU Disk MEM 的性能 监测工具	点击链接
8	ac_tool	系统	☐	开发者调试工具（推荐）	点击链接
9	APTSource	系统	☒	APT 源设置	点击链接
10	PIPSource	系统	☐	PIP 源设置（推荐）	点击链接

11	PyTorchMUSA	AI	☒	PyTorchGPU 版本	点击链接
12	vLLMMUSA	AI	☒	vLLMGPU 版本	点击链接

4.3 依赖关系



4.4 安装与部署

4.4.1 准备工作

账户：请使用常规账户安装，切勿使用 root 账户

网络：请保持互联网在线

下载：请参考以下表格，务必在安装之前，确保安装包已下载完整

使用授权账号信息，登录 TOS 网盘下载.

序号	名字	依赖项	路径
1	Third-Party OSS	互联网	N/A
2	MUSADDK	<code>musa_3.1.0-ABI00_arm64.deb</code>	<code>tos//ab100-sw/AI MODULE/AI MODULEUBOOT(平台化版本)/MUSA</code>
3	MUSASDK	<code>musa-sdk_4.1.2_arm64.deb</code>	<code>tos//ab100-sw/AI MODULE/AI MODULEUBOOT(平台化版本)/MUSA</code>
4	MCCL	<code>mccl_M1000.tar.gz</code>	<code>tos//ab100-sw/AI MODULE/AI MODULEUBOOT(平台化版本)/Torch_MUSA_V2.0.1/Dependency Library/</code>
5	MTNNSDK	<code>kernel-module-m1000-npu_1.3-</code>	<code>tos//ab100-sw/AI MODULE/AI MODULEUBOOT(平台化版</code>

		r20250619105729+33cf6e8_all.deb m1000-mtnnrt_1.3- r20250619111908+c0d070b_all.deb m1000-mtnn-unify-lib_1.3- r20250619111734+791de2d_all.deb	本)/Debs/all_deb_20250630.tar
6	MITCodecSDK	mtcodec_1.0.0- r20241227174000+f27a2f1a1_arm64.deb	tos//ab100-sw/AI MODULE/AI MODULEUBOOT(平台化版本)/Debs/all_deb_20250630.tar
7	mthreads-smi	d19665d72_mthreads-smi_1.1.0_arm64.deb	tos//ab100-sw/AI MODULE/AI MODULEUBOOT(平台化版本)/Debs/all_deb_20250630.tar
8	ac_tod	kernel-module-m1000- actod_1.4+gitAUTONC+0db57f7ce5_aarch64.deb m1000-ac-tod_1.0- r20250606163821+459c1a1_arm64.deb	tos//ab100-sw/AI MODULE/AI MODULEUBOOT(平台化版本)/Debs/all_deb_20250630.tar
9	PyTorchMUSA	torch-2.2.0-cp310-cp310-linux_aarch64.whl torch_musa-2.0.1-cp310-cp310-linux_aarch64.whl torchaudio-2.2.2+cefdb36-cp310-cp310- linux_aarch64.whl torchvision-0.17.2+clfd70fe-cp310-cp310- linux_aarch64.whl	https://mt-ai-data.tos-cn-shanghai.volces.com/vllm_musa/v1.0.1/vllm_musa%2Bm1000_v1.0.1.zip
10	vLLMMUSA	requirements.txt triton-3.1.0-cp310-cp310-linux_aarch64.whl flash_attn-2.6.3-py3-none-any.whl vllm-0.7.4.dev1+gb0c51fc69.d20250704-cp310- cp310-linux_aarch64.whl vllm_musa-1.0.1-cp310-cp310-linux_aarch64.whl	https://mt-ai-data.tos-cn-shanghai.volces.com/vllm_musa/v1.0.1/vllm_musa%2Bm1000_v1.0.1.zip

4.4.2 系统基础环境

1 Third-Party OSS

```
sudo apt install -y python3-pip git git-lfs cmake wget vim build-essential g++ net-tools vainfo
sudo apt install -y libstdc++-12-dev libnuma-dev libopenmpi-dev libopenblas-dev
git lfs install
```

2 MUSA DDK

```
# 卸载
sudo dpkg -P musa

# 安装
sudo dpkg -i musa_3.1.0-AB100_arm64.deb
```

3 MUSA SDK

```
# 卸载
sudo dpkg -P musa-sdk
sudo rm -rf /usr/local/musa*

# 安装
sudo dpkg -i musa-sdk_4.1.2_arm64.deb
```

4 MCCL

```
# 确保安装顺序: MUSA SDK -> MCCL
tar zxvf mccl_1.9.1-AB100.tar.gz
sudo sh -c "cd mccl && ./install.sh"
```

5 MTNN SDK

```
sudo apt install -y ./kernel-module-m1000-npu_1.3-r20250619105729+33cf6e8_all.deb
sudo apt install -y ./m1000-mtnn-unify-lib_1.3-r20250619111734+791de2d_all.deb
sudo apt install -y ./m1000-mtnnrt_1.3-r20250619111908+c0d070b_all.deb
```

6 MTCodec SDK

```
sudo dpkg -i mtcodec_1.0.0-r20241227174000+f27a2f1a1_arm64.deb
```

7 mthreads-smi

```
sudo dpkg -i d19665d72_mthreads-smi_1.1.0_arm64.deb
```

8 ac_tool

```
sudo apt install -y ./kernel-module-m1000-actool_1.4+gitAUTOINC+0db57f7ce5_aarch64.deb
sudo apt install -y ./m1000-ac-tool_1.0-r20250606163821+459c1a1_arm64.deb
```

9 APT Source

```
sudo sh -c 'apt_src="https://mirrors.mthreads.com/ubuntu-ports" && \
mv /etc/apt/sources.list /etc/apt/sources.list.bak && \
cat <<EOL > /etc/apt/sources.list
deb ${apt_src} mthreads-aimodule-pro multiverse

deb ${apt_src} jammy main restricted universe multiverse
deb ${apt_src} jammy-updates main restricted universe multiverse
deb ${apt_src} jammy-backports main restricted universe multiverse
deb ${apt_src} jammy-security main restricted universe multiverse
EOL'
```

10 PIP Source

```
pip_src="https://pypi.tuna.tsinghua.edu.cn/simple"
mkdir -p $HOME/.pip && tee $HOME/.pip/pip.conf <<EOL
[global]
index-url = ${pip_src}
EOL
```

4.4.3 AI 基础环境

1 PyTorch MUSA

```
pip install torch-2.2.0-cp310-cp310-linux_aarch64.whl
pip install torch_musa-2.0.1-cp310-cp310-linux_aarch64.whl
pip install torchaudio-2.2.2+cefdb36-cp310-cp310-linux_aarch64.whl
pip install torchvision-0.17.2+c1d70fe-cp310-cp310-linux_aarch64.whl
pip install numpy==1.26.4

# 确认环境及安装
python -c "import torch; import torch_musa; print(torch.musa.is_available())"
```

2 vLLM MUSA

```
pip install -r requirements.txt
pip install triton-3.1.0-cp310-cp310-linux_aarch64.whl
pip install flash_attn-2.6.3-py3-none-any.whl
pip install vllm-0.7.4.dev1+gb0c51fc69.d20250704-cp310-cp310-linux_aarch64.whl
pip install vllm_musa-1.0.1-cp310-cp310-linux_aarch64.whl
pip install numpy==1.26.4

# 确认环境及安装
python -c "from vllm_musa import _musa_custom_ops; _musa_custom_ops.decode_mla"
```

4.4.4 基础环境验证

```
# 方法 1: 一键在线符合性测试
bash <(curl -fsSL https://apollo-appstore-pre.tos-cn-beijing.volces.com/appstore/release/tools/E300_Compliance_Test_Uboot130.sh)

# 方法 2: 下载, 本地符合性测试
wget https://apollo-appstore-pre.tos-cn-beijing.volces.com/appstore/release/tools/E300_Compliance_Test_Uboot130.sh
bash E300_Compliance_Test_Uboot130.sh

Item | Status | Results
#####
musa (pkg) | 3.1.0-AB100 | PASS
musa-sdk (pkg) | 4.1.2 | PASS
python3 (pkg) | 3.10.6-1~22.04.1 | PASS
libopenmpi3 (pkg) | 4.1.2-2ubuntu1 | PASS
libopenblas0 (pkg) | 0.3.20+ds-1 | PASS
libstdc++-12-dev (pkg) | 12.3.0-1ubuntu1~22.04 | PASS
libnuma1 (pkg) | 2.0.14-3ubuntu2 | PASS
kernel-module-m1000-npu (pkg) | 1.3-r20250625172539+e1603c1 | PASS
m1000-mtnn-unify-lib (pkg) | 1.3-r20250619111734+791de2d | PASS
m1000-mtnnrt (pkg) | 1.3-r20250709191916+156fc1f | PASS
mtcodec (pkg) | 1.0.0-r20241227174000+f27a2f1a1 | PASS
mthreads-smi (pkg) | 1.1.0 | PASS
kernel-module-m1000-actool (pkg) | 1.4+gitAUTOINC+0db57f7ce5 | PASS
m1000-ac-tool (pkg) | 1.0-r20250606163821+459c1a1 | PASS
numpy (pip) | 1.26.2 | PASS
torch (pip) | 2.2.0 | PASS
torch_musa (pip) | 2.0.1 | PASS
torchaudio (pip) | 2.2.2+cefdb36 | PASS
torchvision (pip) | 0.17.2+c1d70fe | PASS
triton (pip) | 3.1.0 | PASS
flash_attn (pip) | 2.6.3 | PASS
vllm (pip) | 0.7.4.dev1+gb0c51fc69.d20250704 | PASS
vllm_musa (pip) | 1.0.1 | PASS
git --version (cmd) | OK | PASS
git-lfs --version (cmd) | OK | PASS
cmake --version (cmd) | OK | PASS
wget --version (cmd) | OK | PASS
vim --version (cmd) | OK | PASS
python --version (cmd) | OK | PASS
g++ --version (cmd) | OK | PASS
gcc --version (cmd) | OK | PASS
ifconfig --version (cmd) | OK | PASS
pip --version (cmd) | OK | PASS
vainfo --version (cmd) | OK | PASS
mcc --version (cmd) | OK | PASS
ac_tool --version (cmd) | OK | PASS
mcccl_version (cmd) | OK | PASS
musaInfo (cmd) | OK | PASS
mthreads-smi (cmd) | OK | PASS
musa_version_query (cmd) | OK | PASS
apt_src_check (cmd) | OK | PASS
torch_musa_check (cmd) | OK | PASS
vllm_musa_check (cmd) | OK | PASS

=====
数据: 总计= 43, 通过= 43, 失败= 0
结论: 系统符合性检测通过.
日志: E300_Compliance_Report_20250815_150623.log
```

4.5 常用命令

4.5.1 命令清单

序号	描述	详见
1	mthreads-smi 查询 BIOS CPU GPU VPU Disk MEM 的性能监测工具	点击链接

2	查询模组系统 BIOS 版本	点击链接
3	查询模组芯片 CPU 负载	点击链接
4	查询模组芯片 GPU 负载	点击链接
5	查询模组芯片 VPU 负载	点击链接
6	查询模组芯片 NPU 负载	点击链接
7	查询模组芯片温度	点击链接
8	查询模组芯片功耗	点击链接
9	查询已安装 Python 核心包：torch/vllm/mtnn/triton	点击链接
10	musaInfo 查询 MUSAGPU 配置， MEM Threads Grid Warp	点击链接

4.5.2 命令操作

1 mthreads-smi： 查询 BIOS | CPU | GPU | VPU | Disk |
MEM 的性能监测工具

```
mthreads-smi
mthreads-smi version is 1.1.0
cpu information:
---cpu name: M1000
---cpu utilization: 1.00%
---cpu frequency: 2650000000 HZ
---cpu temperature: 52.93 C
gpu information:
---gpu utilization: 0.00%
---gpu frequency: 730000000 HZ
---gpu temperature: 31.97 C
system memory information:
---used: 6468032 KB
---capacity: 16333300 KB
battery information:
---current power: file system error mWh
---power capacity: file system error mWh
display screen information:
---screen size: file system error inch
fan information:
---index 0: 2187 rpm
---index 1: 2187 rpm
bios information:
---bios version: 1.3.0
disk information:
---size: 1000204632 KB
vpu information:
---encoder: 0.00%
---decoder: 0.00%
```

2 询模组芯片 NPU 负载

```
sudo sh -c 'watch -n1 cat /sys/kernel/debug/gc/load'
Every 1.0s: cat /sys/kernel/debug/gc/load
device      : 0
....core    : 0
....load     : 0%
....core    : 1
....load     : 0%
```

3 查询模组芯片温度

```
ac_tool sensor -r tsensor -i 3
```

4 查询模组芯片功耗

```
# CPU Power (Unit: μW)
sudo sh -c 'cat /sys/devices/LNXSYSTM:00/LNXXSYBUS:00/MI2C0001:12/ACPI000D:01/power1_average'

# GPU Power (Unit: μW)
sudo sh -c 'cat /sys/devices/LNXSYSTM:00/LNXXSYBUS:00/MI2C0001:12/ACPI000D:02/power1_average'

# Others (Unit: μW)
sudo sh -c 'cat /sys/devices/LNXSYSTM:00/LNXXSYBUS:00/MI2C0001:12/ACPI000D:00/power1_average'
```

5 查询已安装 Python 核心包: torch | vllm | mtnn | triton

```
pip list | grep -E 'torch|numpy|vllm|mtt|mtnn|triton'
libmtnn-api-4py      1.1
mttransformer        20240402.dev65+g273eb81
numpy                1.26.0
torch                2.5.0
torch_compat         1.0.0
torch_musa           2.1.0
torchaudio           2.5.0a0+56bc006
torchvision          0.20.0a0+afc54f7
triton               3.1.0
```

6 musaInfo: 查询 MUSA GPU 配置, MEM|Threads |Grid | Warp

```
musaInfo
-----
compiler: mcc
device# 0
Name: M1000
pciBusID: 0x0
pciDeviceID: 0x0
pciDomainID: 0x0
multiProcessorCount: 8
maxThreadsPerMultiProcessor: 6144
isMultiGpuBoard: 1
clockRate: 0 Mhz
memoryClockRate: 0 Mhz
memoryBusWidth: 256
totalGlobalMem: 15.58 GB
sharedMemPerMultiProcessor: 72.00 KB
totalConstMem: 8192
sharedMemPerBlock: 72.00 KB
canMapHostMemory: 1
regsPerBlock: 262144
warpSize: 128
l2CacheSize: 0
computeMode: 0
maxThreadsPerBlock: 1024
maxThreadsDim.x: 1024
maxThreadsDim.y: 1024
maxThreadsDim.z: 1024
maxGridSize.x: 2147483647
maxGridSize.y: 2147483647
maxGridSize.z: 2147483647
major: 2
minor: 2
concurrentKernels: 1
cooperativeLaunch: 0
cooperativeMultiDeviceLaunch: 0
isIntegrated: 1
maxTexture1D: 32768
maxTexture2D.width: 32768
maxTexture2D.height: 32768
maxTexture3D.width: 16384
maxTexture3D.height: 16384
maxTexture3D.depth: 2048
peers:
non-peers: device#0

memInfo.total: 15.58 GB
memInfo.free: 6.00 GB (39%)
```

4.6 常见问题

4.6.1 问题清单

序号	描述	详见
1	执行安装命令时，发现没有安装包	点击链接
2	musaInfo: 未找到命令	点击链接
3	musa_version_query: 未找到命令	点击链接
4	mccl_version: 未找到命令	点击链接
5	sudo dmidecode-t0: NoSMBIOS norDMIentry	点击链接

	point found, sorry.	
6	torch 使用: libmusa_python.so.2: open shared object file: No such file or directory	点击链接
7	torch 使用: libmudnn.so.2: cannot open shared object file: No such file or directory	点击链接
8	torch 使用: libmpi_cxx.so.40: cannot open shared object file: No such file or directory	点击链接
9	torch 使用: libmcl.so.2: cannot open shared object file: No such file or directory	点击链接
10	torch 使用: libmusart.so.1.0: cannot open shared object file: No such file or directory	点击链接
11	torch 使用: CreatePlatformfailed	点击链接
12	torch 使用: AssertionError: Torchnot compiled with CUDAenabled	点击链接
13	torch 使用: A module that was compiled using NumPy1.x cannot be run in NumPy2.2.0 as it may crash.	点击链接
14	torch 使用: Error in cpuinfo:prctl(PR_SVE_GET_VL) failed	点击链接

4.6.2 解决方法

1 执行安装命令时，发现没有安装包

=> 请参考章节《安装与部署》-准备工作部分，[参考链接](#)

2 musaInfo: 未找到命令

=> 与两个因素相关: #1 使用 root 账户; #2MUSASDK 未安装成功

=> 先退出 root 账户, 试试看, 若无此问题, 则#1 导致, 否则, 请参考章节《安装与部署》-MUSASDK 安装部分, [参考链接](#)

3 musa_version_query: 未找到命令

=> 与两个因素相关: #1 使用 root 账户; #2MUSASDK 未安装成功

=> 先退出 root 账户, 试试看, 若无此问题, 则#1 导致, 否则, 请参考章节《安装与部署》-MUSASDK 安装部分, [参考链接](#)

4 mccl_version: 未找到命令

=> 与两个因素相关: #1 使用 root 账户; #2MCCL 未安装成功

=> 先退出 root 账户, 试试看, 若无此问题, 则#1 导致, 否则, 请参考章节《安装与部署》-MCCL 安装部分, [参考链接](#)

5 sudo dmidecode-t0: No SMBIOS nor DMI entry point found,
sorry.

=> 官方固件设备管理默认采用 ACPI, 而 SMBIOS 信息放置于 ACPI 中, 出现报错, 很可能与采用 DTS 有关

6 torch 使用:libmusa_python.so.2: open shared object file:
No such file or directory

=> 常出现在 conda 虚拟环境下, 主要因素: 系统全局安装 TorchMUSA 后, 与虚拟环境未隔离完全, 导致交错调用混乱

=> 解决方法有两种:

#1 只在 conda 虚拟环境中安装 whl, 禁止全局安装;

#2 设置全局环境隔离变量 exportPYTHONNOUSERSITE=1

为使其永久生效, 执行

```
echo"exportPYTHONNOUSERSITE=1">>$HOME/.bashrc&&source$HOME/.bashrc
```

7 torch 使用: libmudnn.so.2: cannot open shared object
file: No such file or directory

=> 与两个因素相关: #1 使用 root 账户; #2MUSASDK 未安装成功

=> 先退出 root 账户, 试试看, 若无此问题, 则#1 导致, 否则, 请参考章节《安装与部署》-MUSASDK 安装部分, [参考链接](#)

8 torch 使用: libmpi_cxx.so.40: cannot open shared object
file: No such file or directory

=> MPI 库未安装成功, 请参考章节《安装与部署》-Third-PartyOSS 安装部分, [参考链接](#)

9 torch 使用: libmccl.so.2: cannot open shared object
file: No such file or directory

=> MCCL 库未安装成功, 请参考章节《安装与部署》-MCCL 安装部分, [参考链接](#)

10torch 使用: libmusart.so.1.0: cannot open shared object
file: No such file or directory

=> 主要因素: MUSASDK 与 DDK 不匹配导致

=> 请参考章节《安装与部署》-MUSASDK 安装部分, [参考链接](#)

11 torch 使用: CreatePlatform failed

=> 该问题与当前用户的登录授权相关, 主要因素: 用户尚未登录进入桌面系统, 导致登录初始化脚本未执行, 无访问授权;

=> 两种方法: #1 登录进入桌面; #2 在串口或 SSH 终端, 执行 `sudo sh-c "usermod-aGrender,video$USER&&systemctlrestartgdm3";`

12 torch 使用: AssertionError: Torch not compiled with
CUDA enabled

=> 该问题与当前所用账户相关, 主要因素: 进入 root 账户, MUSA 相关的
PATH 与 LD_LIBRARY_PATH 环境变量面向普通账户

=> 退出 root 账户, 问题自然消失

13 torch 使用: A module that was compiled using NumPy 1.x
cannot be run in NumPy 2.2.0 as it may crash.

=> 提示 NumPy 版本 2.2.0 可能导致程序崩溃, 降级到 NumPy 1.26.4 版本,
执行 `pip install numpy==1.26.4`

14 torch 使用: Error in cpuinfo: prctl(PR_SVE_GET_VL)
failed

=> 该信息划归为: 警告, 在构建 PyTorch 时, 查询 ARM CPU 是否支持 SVE
指令集, 若支持, 将启用 ATen/native/sve 路径优化;

=> 尚未发现该问题导致软件功能缺陷案例, 可忽略。

5 开发资料导航

5.1 资料网站

- [全爱科技下载中心](#)
- [测试例程下载地址](#)
- 系统镜像：

<https://pan.baidu.com/s/1RnEZEFnWfwSCLGrp6yxQug?pwd=r8j8>

提取码: r8j8

5.2 使用 WinSCP 传输文件

操作场景

在 PC 机上使用 WinSCP 工具进行文件传输。

必备事项

前提条件

目的设备已开启 SFTP 服务。

数据

需准备如下数据：

- 待连接服务器的 IP 地址
- 登录待连接服务器的用户名和密码

软件

WinSCP.exe：此工具为第三方软件，请自行准备。

操作步骤

步骤 1 打开“WinSCP”文件夹，双击“WinSCP.exe”。

弹出“WinSCP 登录”对话框，如图 5-1 所示。

📖 说明 若系统非中文操作系统，可以单击“Languages”进行界面语言的选择。

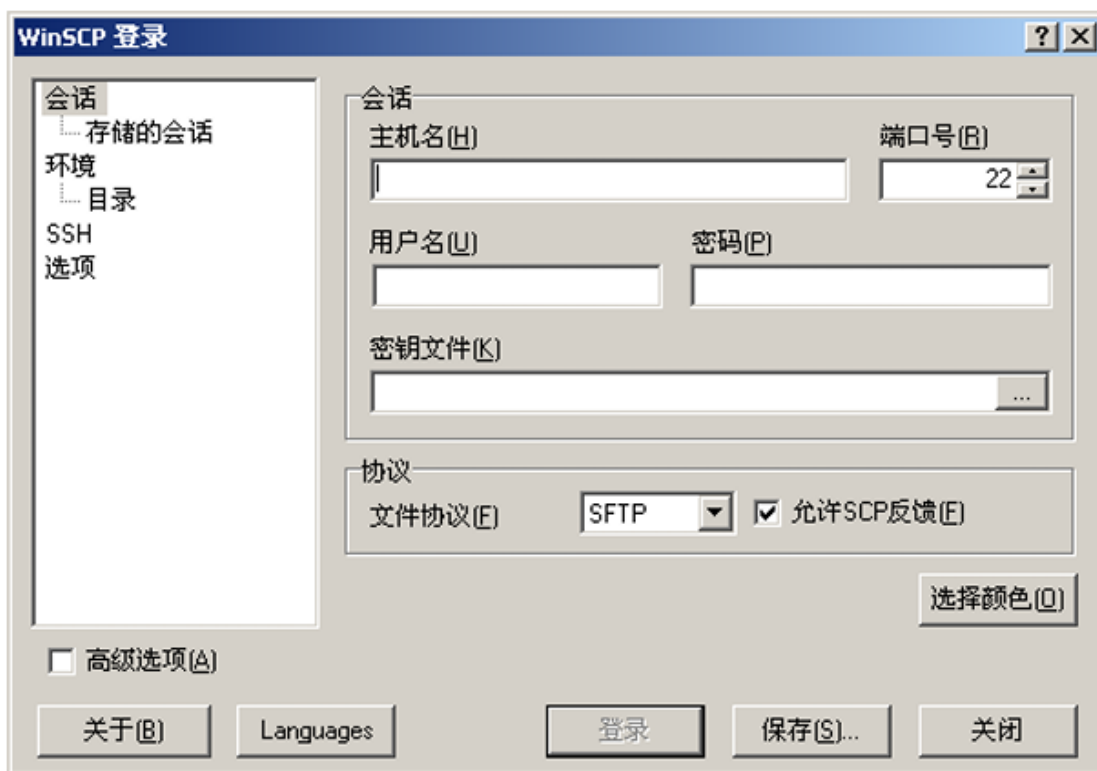


图 5-1

步骤 2 单击“会话”。

步骤 3 设置登录参数。

参数说明如下。

- 主机名 (H)：待连接设备的 IP 地址，例如“192.168.2.10”。
- 端口号 (R)：默认为“22”。
- 用户名 (U)：待连接设备的操作系统用户名，例如“admin123”。
- 密码 (P)：待连接设备的操作系统用户的密码，例如“admin123”。
- 密钥文件 (K)：默认为空，保留默认值。
- 协议：选择默认文件协议“SFTP”，并勾选“允许 SCP 反馈 (F)”。

步骤 4 单击“登录”。

进入“WinSCP”文件传输界面。

说明如果首次登录时没有选择密钥文件，此时会弹出一个警告提示框，询问“是否连接并添加密钥到缓存？”，单击“是（Y）”，进入“WinSCP”文件传输界面。

步骤 5 根据实际需求，在界面左右区的指定目录中进行文件夹的创建、删除和复制等操作

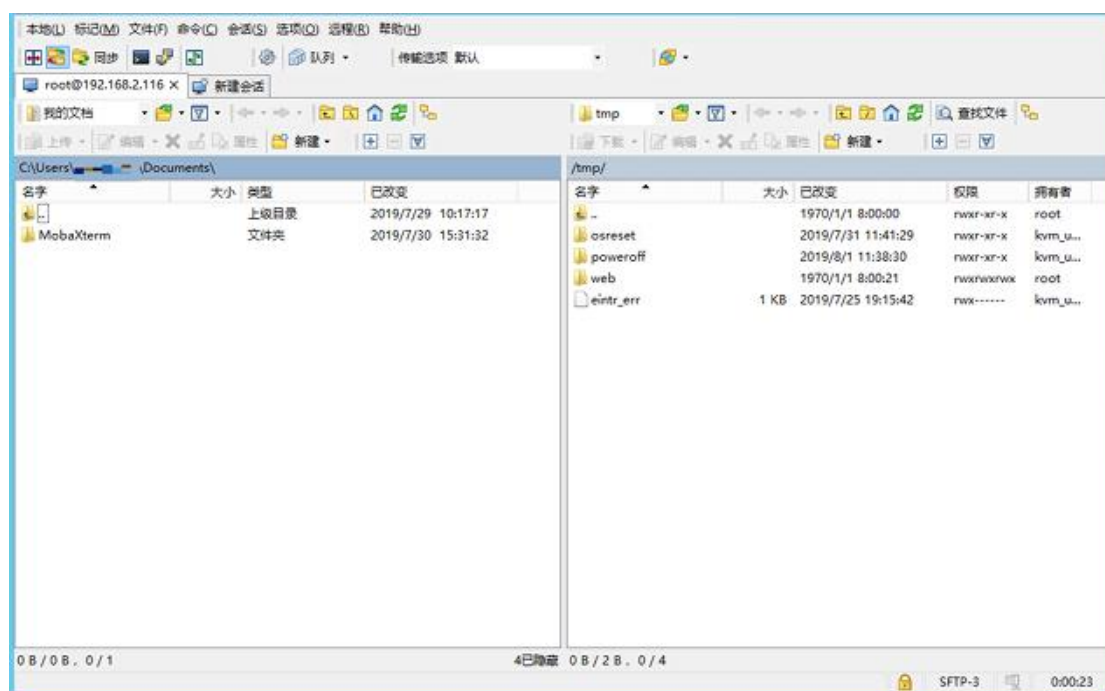


图 5-2WinSCP 界面

说明界面左侧区域为本地 PC 的目录，右侧区域为已连接设备的目录。