DeepSeek-V3/R1部署指南

目录

[DeepSeek-V3/R1部署指南 1](#_Toc192174075)

[1、 硬件要求及组网 4](#_Toc192174076)

[2、 模型权重 5](#_Toc192174077)

[2.1 量化W8A8权重下载 6](#_Toc192174078)

[2.2.1 权重放置 6](#_Toc192174079)

[3、 一键式部署脚本介绍 7](#_Toc192174080)

[3.1 推荐版本 7](#_Toc192174081)

[3.1.1hdk下载方式 7](#_Toc192174082)

[3.2 部署脚本使用 8](#_Toc192174083)

[4、 半自动化按需部署 10](#_Toc192174084)

[4.1 填写配置文件 10](#_Toc192174085)

[4.2 加载推理镜像 10](#_Toc192174086)

[4.3 网络检查 10](#_Toc192174087)

[4.3.1 检查机器网络情况【宿主机】 11](#_Toc192174088)

[4.3.2 检测机器间互联情况【宿主机】 14](#_Toc192174089)

[4.4 环境配置 17](#_Toc192174090)

[4.5 ray集群启动 17](#_Toc192174091)

[4.6 服务启动 18](#_Toc192174092)

[5、服务化测试 19](#_Toc192174093)

[5.4 使用Benchmark测试【容器内】 19](#_Toc192174094)

[5.5 使用客户端测试 20](#_Toc192174095)

[6. FAQ 22](#_Toc192174096)

[6.1 运行oedp run install报错找不到库 22](#_Toc192174097)

[6.2 运行部署时显示空间不足 22](#_Toc192174098)

[7、环境变量说明： 22](#_Toc192174099)

# 硬件要求及组网

部署DeepSeek-V3/R1量化模型至少需要2台Atlas 800I A2（8\*64G）服务器。本文以DeepSeek-V3为主进行介绍，DeepSeek-R1与V3的模型结构和参数量一致，部署方式与V3相同。

组网结构推荐使用直连模式，即服务器通过交换机直连，每张卡都可以ping通其他卡

# 模型权重

|  |  |
| --- | --- |
| **Check List** | |
| 2.A | 下载W8A8权重时，请确保机器内/挂载盘中有大于700G的存储空间。 |
| 2.B | 如需要多机推理，请确保每台机器上都有相同的权重，并且能正常访问。 |
| 2.C | 请确保CPU侧内存能够放下对应权重。  例：w8a8权重需要大约500G左右的内存，通过free -h指令查看空闲cpu内存。  **free\_mem >= (权重大小 / 机器数) \* 1.3 （该计算方式待验证，但需要确保内存足够）** |
| 2.D | 请确保使用的推理镜像是最新版本。 |
| 2.E | 请根据权重大小选择对应的卡数推理：  - W8A8至少需要2台800I/T A2 64G。 |
| 2.F | 请确保权重的正确性，对比权重/tokenizer等文件与源文件的MD5或SHA256值。 |

## 2.1 量化W8A8权重下载

|  |  |
| --- | --- |
| Modelers | 待补充 |
| 待补充 |

### 2.2.1 权重放置

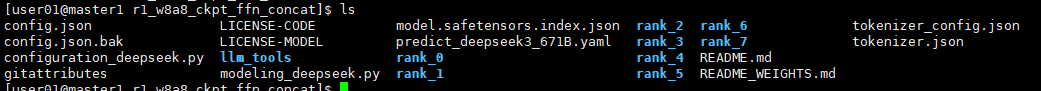
权重下载好后，需要在两台节点各放置一份，且放置到相同的路径。

在主节点上，保留rank\_0~rank\_7

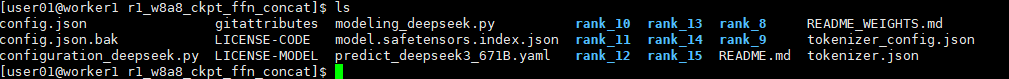
在从节点上，保留rank\_8~rank\_15

两台节点上其余权重相关文件，需保持一致

主节点权重目录示意如下:



从节点权重目录示意如下：



# 一键式部署脚本介绍

|  |  |
| --- | --- |
| **CheckList** | |
| 3.A | 请确保完成第2章节，权重获取 |
| 3.B | 请确保已经在所有宿主机上安装Ascend HDK driver & firemare |
| 3.C | 请确保组网联通性 |

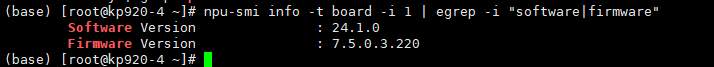
使用一键是部署脚本，可根据配置，一键部署双节点集群，并且拉起deepseek服务。

## 3.1 推荐版本

|  |  |
| --- | --- |
| **部件** | **版本号** |
| Ascend HDK Driver | 24.1.0 |
| Ascend HDK Firmware | 7.5.0.3.220 |

该版本要求内核版本为5.10，安装前校验内核版本

|  |
| --- |
| # 可以使用如下命令获取环境上的驱动&固件版本信息  npu-smi info -t board -i 1 | egrep -i "software|firmware" |



### 3.1.1hdk下载方式

所需的驱动和固件可从昇腾官方社区下载，详细见

<https://www.hiascend.com/hardware/firmware-drivers/commercial?product=1&model=30&cann=8.0.0.beta1&driver=1.0.28.alpha>

## 3.2 部署脚本使用

一键式部署脚本推荐在单独控制节点执行，控制节点需要可以使用ssh访问各个推理节点

**Step1：下载oedeploy工具**

|  |
| --- |
| # 下载插件包并解压  wget https://repo.oepkgs.net/openEuler/rpm/openEuler-24.03-LTS/contrib/oedp/plugins/mindspore-deepseek.tar.gz  tar zxvf mindspore-deepseek.tar.gz  # 安装oedp工具，例如:  yum localinstall oedp-1.0.0-20250208.x86\_64.rpm  rpm包可从如下地址获取  <https://repo.oepkgs.net/openEuler/rpm/openEuler-24.03-LTS/contrib/oedp/aarch64/Packages/oedp-1.0.0-2.oe2503.aarch64.rpm> |

**Step2：调整oedeploy配置文件**

|  |
| --- |
| # 调整插件目录下config.yaml  (base) [root@910b-3 mindspore-deepseek]# cat config.yaml  all:  children:  masters:  hosts:  master1:  ansible\_host: 1.2.3.4  ansible\_port: 22  ansible\_user: root  ansible\_password: "密码"  workers:  hosts:  worker1:  ansible\_host: 1.2.3.5  ansible\_port: 22  ansible\_user: root  ansible\_password: "密码"  vars:  # 容器镜像  image\_name: hub.oepkgs.net/oedeploy/openeuler/aarch64/mindspore  image\_tag: latest  # 推理服务所在容器的名称  container\_name: openeuler\_ds  # 模型路径  model\_path: /workspace/deepseekv3  # ray 开放的端口  ray\_port: 6379  # 节点个数  node\_num: 2  # 拉起服务前是否停止其他容器  is\_stop\_other\_container: 0  # 推理服务端口  llm\_port: 8000  # ray集群使用的网卡  ray\_device: enp67s0f0np0 |

启动master为主节点，worker为从节点，进行部署前，需要调整对应节点的ip地址，用户名，以及密码。

请确保node\_num的值与配置的ip个数一致。

请确保各个节点上使用的网卡设备一致，配置到ray\_device变量中

**Step2：运行一键部署脚本**

|  |
| --- |
| #插件目录下运行  oedp run install |

# 半自动化按需部署

除了使用一键式部署脚本，也可以手动执行分步脚本。

需要将插件目录下workspace/roles/prepare/files/lib目录传输至所有推理节点。

传输脚本前，请务必调整config.cfg配置文件。

## 4.1 填写配置文件

|  |
| --- |
| # 复制模板config，该example在lib文件夹内  cp example\_config config.cfg  # 调整config.cfg |

注意，复制生成的文件名字必须为config.cfg

## 4.2 加载推理镜像

|  |  |
| --- | --- |
| **CheckList** | |
| 3.A | 请确保服务器OS满足：宿主机的内核=5.10 |
| 3.B | 请确保所有机器的OS、HDK、内核保持一样。 |
| 3.C | 推理镜像版本请以社区最新版为准 |

**Step1：拉取推理镜像，启动容器**

|  |
| --- |
| # 需在所有节点执行容器拉取脚本，改脚本包含容器镜像拉取，容器启动两个步骤  ./lib/start\_docker.sh |

在拉取容器前会检测环境，若环境上有同名镜像，则不会进行拉取

在启动容器前会检测环境，若环境上有同名容器(无论是运行态还是停止态)，均会使用已存在的容器

## 4.3 网络检查

可在各个节点调用网络检测脚本

|  |
| --- |
| ./lib/net\_check.sh |

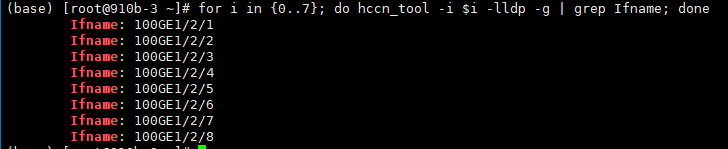
该脚本内，会检测各个节点的npu卡link状态，以及网络健康状态。

也可按照如下步骤手动检查

### 4.3.1 检查机器网络情况【宿主机】

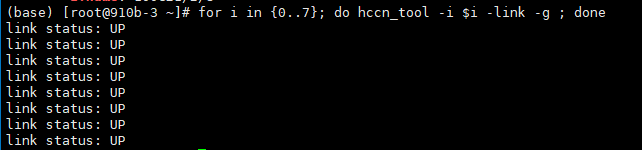
**step1:** 检查物理链接

|  |
| --- |
| for i in {0..7}; do hccn\_tool -i $i -lldp -g | grep Ifname; done |



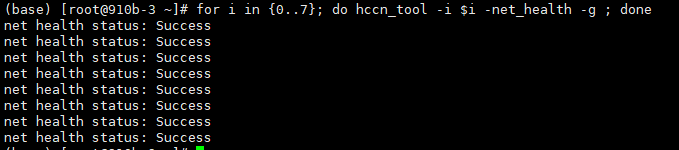
**step2:** 检查链接情况

|  |
| --- |
| for i in {0..7}; do hccn\_tool -i $i -link -g ; done |



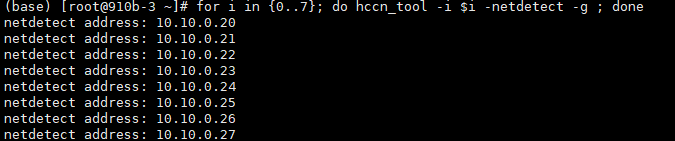
**step3:** 检查网络健康情况

|  |
| --- |
| for i in {0..7}; do hccn\_tool -i $i -net\_health -g ; done |



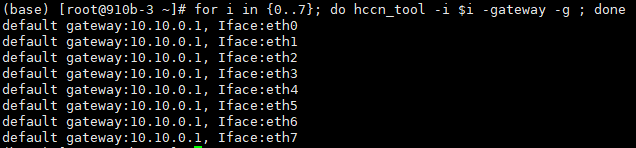
**step4:** 查看侦测ip的配置是否正确

|  |
| --- |
| for i in {0..7}; do hccn\_tool -i $i -netdetect -g ; done |



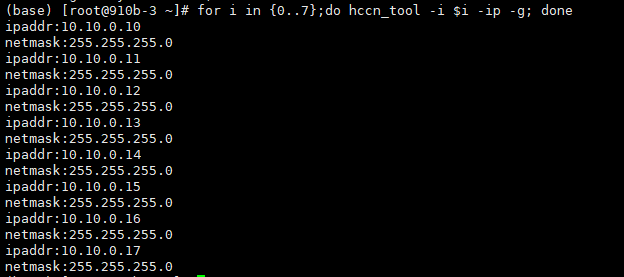
**step5:** 查看网关是否配置正确

|  |
| --- |
| for i in {0..7}; do hccn\_tool -i $i -gateway -g ; done |



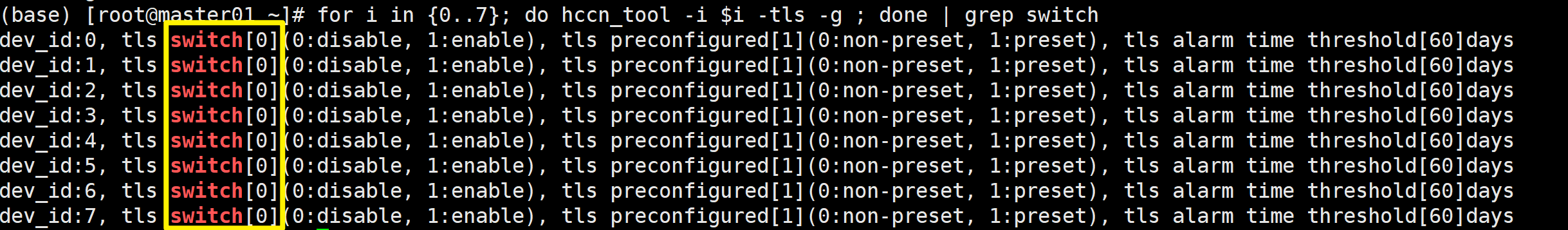
**step6:** 获取每张卡的ip地址

|  |
| --- |
| for i in {0..7};do hccn\_tool -i $i -ip -g; done |



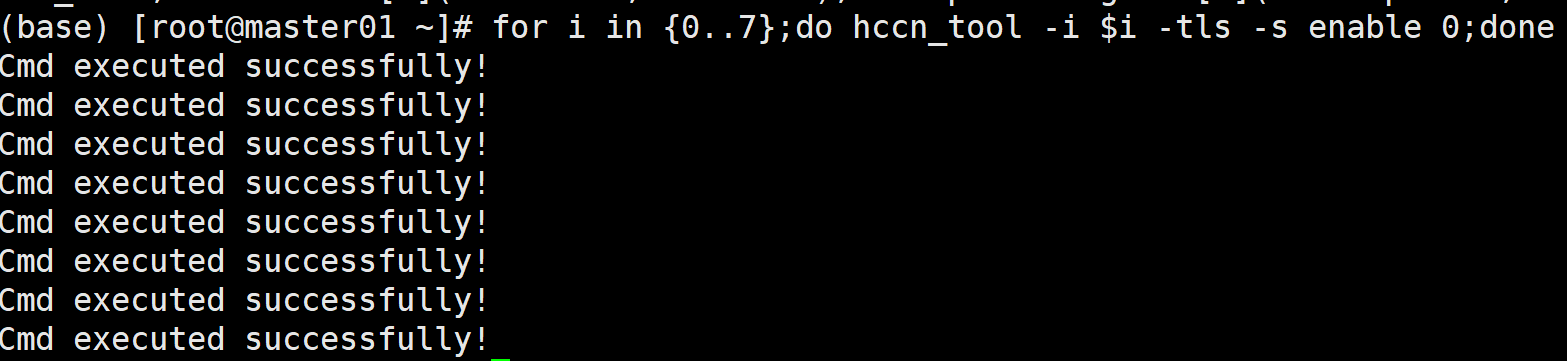
**step7**：检测NPU底层TLS行为一致性，每台机器需要是一样的值，建议全0

|  |
| --- |
| for i in {0..7}; do hccn\_tool -i $i -tls -g ; done | grep switch |



**step8：**NPU底层tls校验行为置0操作

|  |
| --- |
| for i in {0..7};do hccn\_tool -i $i -tls -s enable 0;done |



### 4.3.2 检测机器间互联情况【宿主机】

机器间互联检测可以通过本机每张npu卡ping其他主机的npu卡ip地址，能ping通表示正常。

可使用脚本检测机器互联情况

|  |
| --- |
| ./net\_check.sh --check-connection |

**Demo：**

现在有机器A与机器B，机器每张卡的ip地址可以通过 **4.3.1** 这一章的 **step6** 的方式进行获取。

**Step1:** 获取主机A的每张卡ip地址 - 10.20.0.12~10.20.0.19

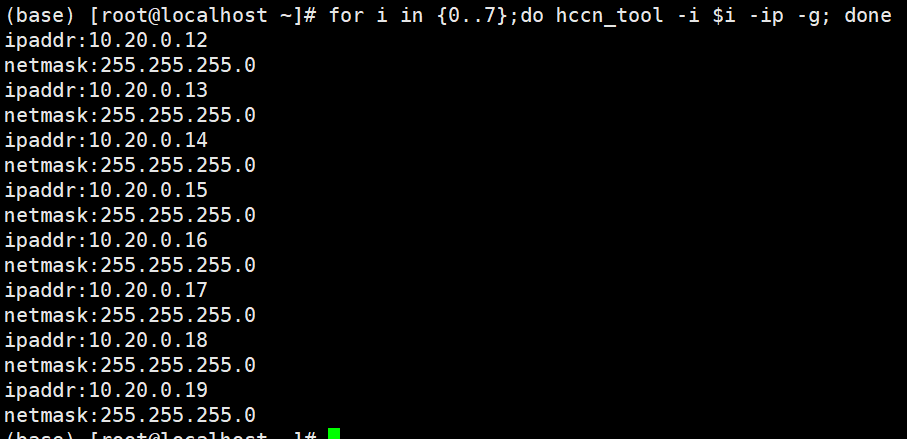


图. 主机A每张卡的ip地址

**Step2:** 获取主机B的每张卡ip地址 - 10.20.0.22~10.20.0.29

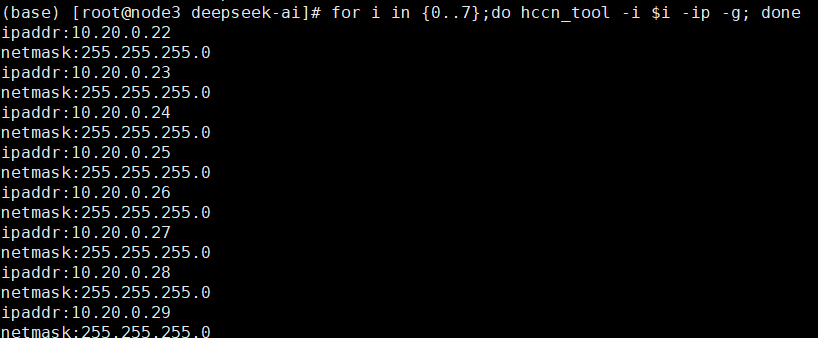


图. 主机B每张卡的ip地址

**Step3:** 用主机A的每张卡ping主机B的每张卡

卡与卡之间互相ping的指令如下：

|  |
| --- |
| hccn\_tool -i [card\_idx] -ping -g address [ip\_address]  # card\_idx: 本机的第几张卡  # ip\_address: npu卡的ip address |

比如，主机A的3卡去ping主机B的7卡(10.20.0.29)：

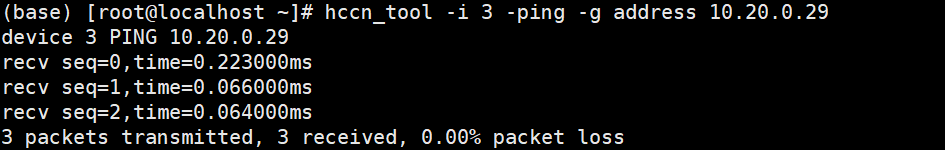


图. 主机A的3卡ping通主机B的7卡截图

根据以上指令，我们可以设置一个循环来将主机A的每张卡去ping主机B的每张卡

|  |
| --- |
| for j in {0..7};      do for i in {22..29};          do hccn\_tool -i ${j} -ping -g address 10.20.0.${i};      done;  done |
| 注意：请自行修改第二行与第三行的ip地址信息（标黄部分） |

输出如下图：

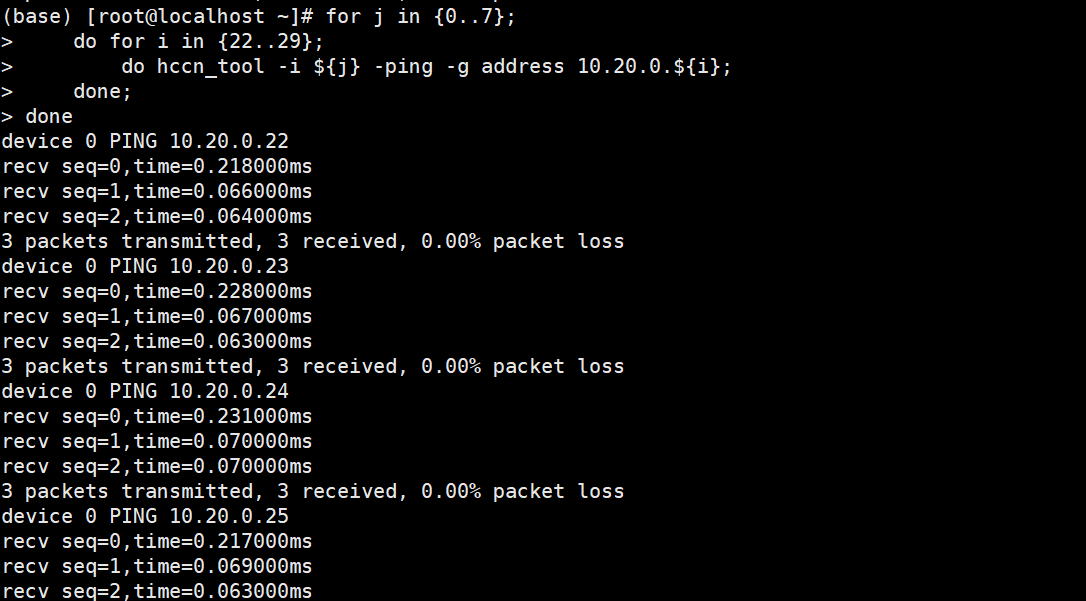


图. 主机A每张卡ping通主机B每张卡截图

**Step4:** 用主机B的每张卡ping主机A的每张卡

保险起见，我们需要在主机B上做一次与Step3相同的操作，指令如下：

|  |
| --- |
| for j in {0..7};      do for i in {12..19};          do hccn\_tool -i ${j} -ping -g address 10.20.0.${i};      done;  done |

|  |  |
| --- | --- |
| **说明** | |
| 1 | 可能会出现同一台机器每张卡的ip地址格式不一样的情况，比如0卡是10.2.0.11，1卡是10.3.5.23。这种情况下，无法使用上述的循环指令测试机器间的连通性，请手动ping通测试。 |

## 4.4 环境配置

该步骤在容器内执行，需在所有节点执行

**Step1:** 可使用环境配置脚本，一键设置环境变量，以及调整配置文件：

|  |
| --- |
| # 需在所有节点执行该脚本，改脚本包含调整配置文件，设置环境变量两个步骤  ./lib/set\_env.sh |

该脚本运行时，会将环境变量写入/root/.bashrc，若该文件中已有openeuler\_deepseek\_env\_config字段，则视为环境变量已存在，会跳过环境变量设置流程。

## 4.5 ray集群启动

该步骤在容器内执行，需在所有节点执行

**Step1:** 可使用ray启动脚本，一键启动ray，调整MASTER\_IP为主节点ip：

|  |
| --- |
| # 主节点先执行  ./lib/ray\_start.sh  # 从节点后执行  ./lib/ray\_start.sh $MASTER\_IP |

使用该脚本前，请确保主节点上，配置文件config.cfg中RAY\_PORT项对应端口空闲

## 4.6 服务启动

该步骤在容器内执行，仅在主节点执行

**Step1:** 可在主节点中启动服务，使用前请务必配置好config.cfg：

|  |
| --- |
| #主节点执行 该脚本启动服务的日志，会存于ds.log中  ./lib/start\_ds.sh |

# 5、服务化测试

## 5.4 使用Benchmark测试【容器内】

使用ascend-vllm性能测试工具

|  |
| --- |
| python benchmark\_parallel.py --backend openai --host [主节点IP] --port [服务端口] --tokenizer [模型路径] --epochs 1 --parallel-num 192 --prompt-tokens 256 --output-tokens 256 |

也可使用vllm开源的性能测试工具

<https://github.com/vllm-project/vllm/tree/main/benchmarks>

## 5.5 使用客户端测试

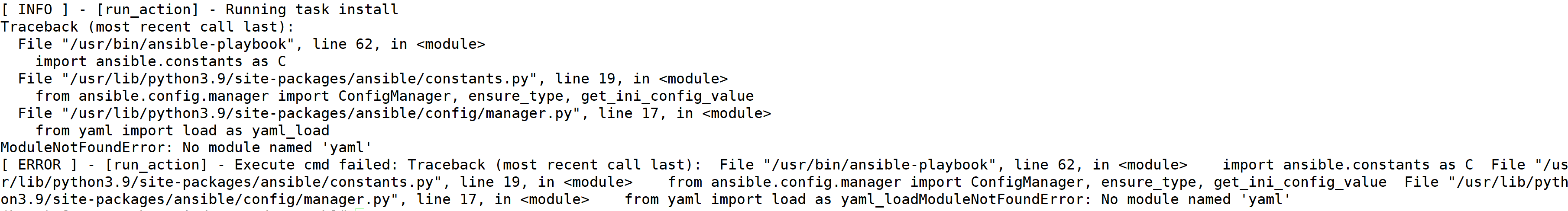
再打开一个主节点客户端，向服务端发送请求

|  |  |
| --- | --- |
| **Deepseek-V3请求样例(请按照配置的LLM\_PORT变量调整请求端口)** | |
| 请求 | curl http://localhost:8000/v1/completions -H "Content-Type: application/json" -d '{"model": "/path/to/model", "prompt": "I love Beijing, because", "max\_tokens": 128, "temperature": 0, "top\_p": 1.0, "top\_k": 1, "repetition\_penalty":1.0}' |
| 结果 |  |

|  |  |
| --- | --- |
| **Deepseek-R1请求样例(请按照配置的LLM\_PORT变量调整请求端口)** | |
| 请求 | curl http://localhost:8000/v1/completions -H "Content-Type: application/json" -d '{"model": "/workspace/deepseekr1", "prompt": "I love Beijing, because", "max\_tokens": 128, "temperature": 0, "top\_p": 1.0, "top\_k": 1, "repetition\_penalty":1.0}' |
| 结果 |  |

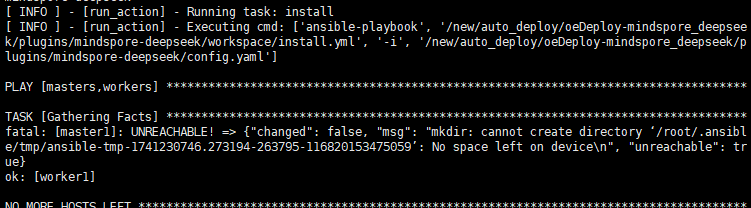
# FAQ

## 运行oedp run install报错找不到库



Oedp兼容性bug，原因为环境上安装了多个py，互相干扰，建议在只有py3.9的环境上执行部署命令

## 6.2 运行部署时显示空间不足



请确保根目录有足够空间

# 7、环境变量说明：

|  |  |
| --- | --- |
| **环境变量** | **功能说明** |
| MS\_ENABLE\_LCCL=off | 关闭多机lccl |
| HCCL\_OP\_EXPANSION\_MODE=AIV | 通信下发优化 |
| vLLM\_MODEL\_BACKEND=MindFormers | 指定使用mindformers模型 |
| vLLM\_MODEL\_MEMORY\_USE\_GB=50 | 性能优化相关 |
| MS\_DEV\_RUNTIME\_CONF="parallel\_dispatch\_kernel:True" | 性能相关 |
| MS\_ALLOC\_CONF="enable\_vmm:False" | 电信特有 |
| ASCEND\_RT\_VISIBLE\_DEVICES=0,1,2,3,4,5,6,7 | 指定可使用的ascend卡 |
| GLOO\_SOCKET\_IFNAME=网卡名称 | ray组网需要 |
| TP\_SOCKET\_IFNAME=网卡名称 | ray组网需要 |
| MINDFORMERS\_MODEL\_CONFIG=yaml路径 | 指定mindformers要使用的模型yaml |