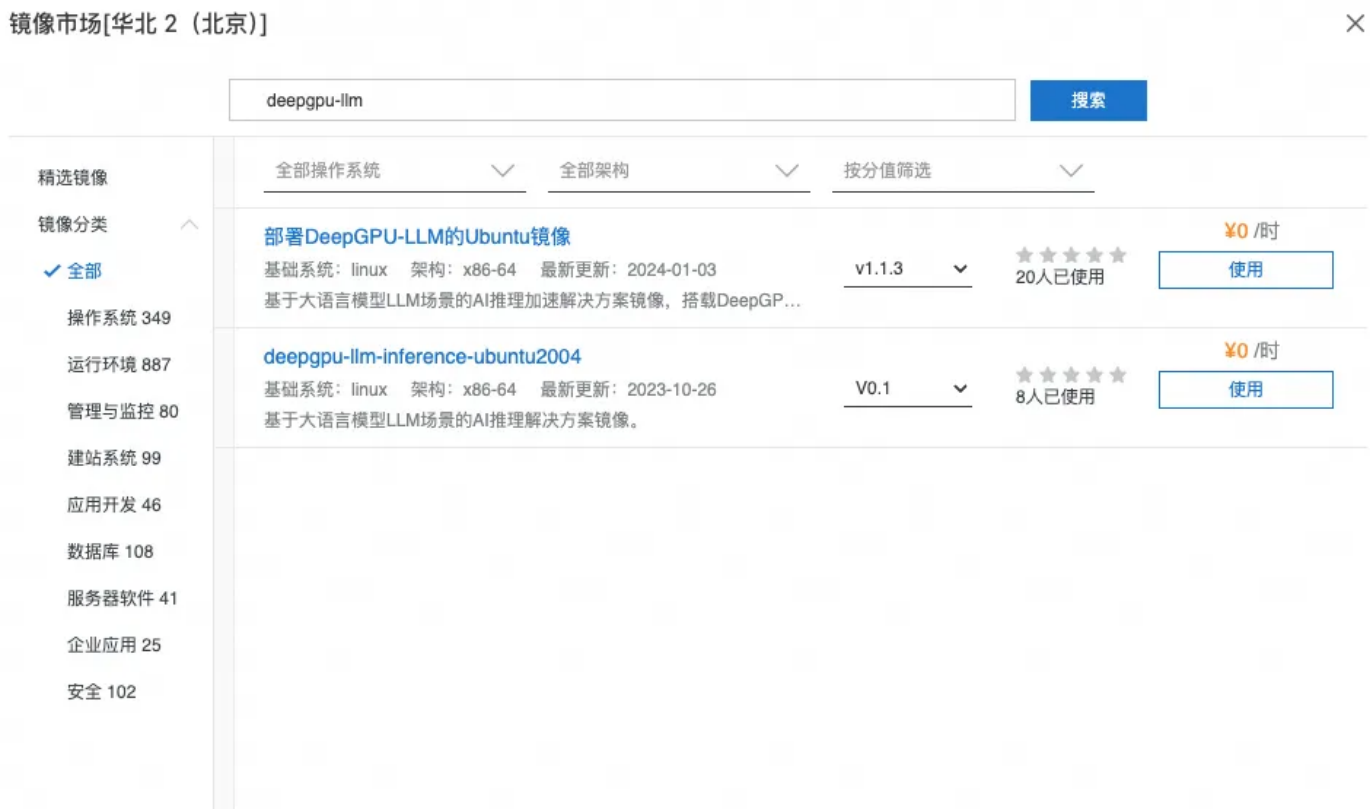
如何在GPU云服务器上快速部署deepgpu-llm

1 离线部署：使用云市场镜像

1.1 方式1：从ECS购买页进入

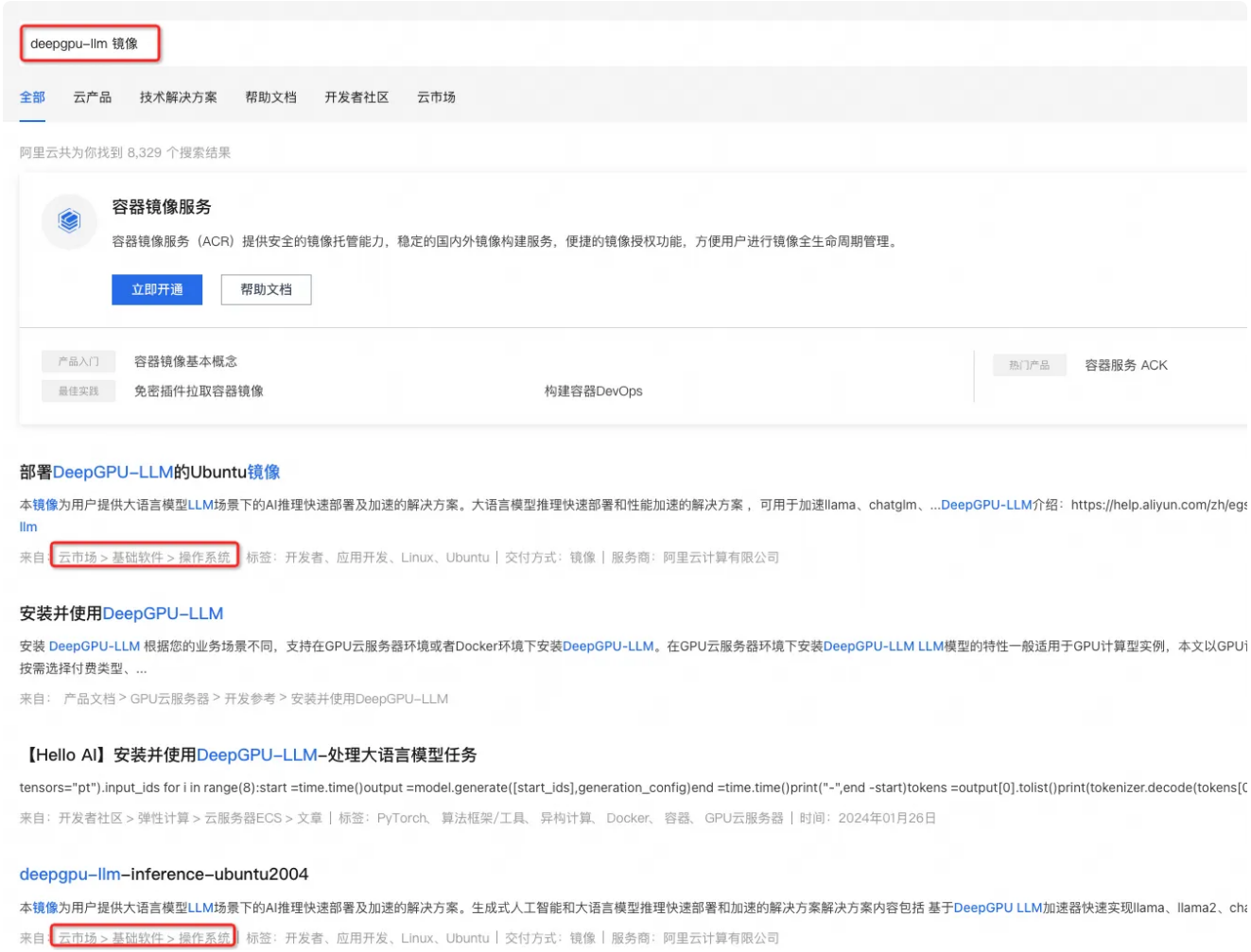
进入ECS实例购买页，按需配置付费类型、地域、网络、可用区、实例类型等；在镜像栏目选择“云市场镜像”–搜索“deepgpu–llm”选择所需镜像。配置好所有配置项目点击下单开通机器。

注：下面是老版本截图，建议选择最新更新的版本即可。



1.2 方式2：官网直接搜索“deepgpu–llm”进入购买

在阿里云官网右上方搜索框，直接搜索“deepgpu–llm 镜像”，在搜索结果中查看“来自：云市场 基础软件 操作系统”的页面，选择更新时间最新的版本点击进入。



在具体镜像的详情页里，点击购买即可进入ECS购买页，deepgpu-llm镜像本身都是免费的。按ECS购买页提示填写相关信息，确认镜像栏目选中所选镜像。由于镜像启动方式的不同，一个镜像无法适配所有机型，若无法匹配，可返回选择其他deepgpu-llm镜像。

1.3 等待镜像安装成功

ssh登陆成功即完成部署。

具体操作，请参见通过密码或密钥认证登录Linux实例。

1.4 升级安装deepgpu-llm（可选）

根据5.1节方法检查deepgpu-llm的安装状态和版本，若觉得版本过低，可以根据本节方法来升级安装deepgpu-llm。

首先从DeepGPU-LLM加速安装包下载需要升级的安装包到对应目录，可使用wget命令进行下载。然后直接用pip命令安装即可。参考命令如下所示。

```
Plain Text
1  wget https://aiacc-inference-public-v2.oss-cn-hangzhou.aliyuncs.com/aiacc-inference-llm/deepgpu_llm-24.3%2Bpt2.1cu121-py3-none-any.whl
2  pip install deepgpu_llm-24.3+pt2.1cu121-py3-none-any.whl
```

2 在线部署：基于ubuntu 22.04系统

2.1 进入ECS购买页，选择所需镜像，开通机器

按需配置付费类型、地域、网络、可用区、实例类型等，在镜像栏目选择“公共镜像”-“Ubuntu”-“Ubuntu 22.04 64位”，可选直接**勾选安装GPU驱动、CUDA和CUDNN**，如下图所示。

2.2 等待系统安装和自动安装项目安装

ssh登陆可查看GPU驱动、CUDA、CUDNN等安装进展。

2.3 安装deepgpu-llm

逐一输入下面命令，在线安装deepgpu-llm的部分依赖项。

```
Plain Text
```

```
apt-get update
apt-get -y install python3.10 python3-pip openmpi-bin libopenmpi-dev curl v
im
```

选择需要安装的版本，在线安装deepgpu-llm及其依赖项（耗时较长，耐心等待下载和安装）。其中xx.x应替换为您实际的DeepGPU-LLM版本号。如何获取最新DeepGPU-LLM版本号，请参见DeepGPU-LLM加速安装包。

```
Plain Text
```

```
# for PyTorch 1.13
pip3 install deepgpu_llm==xx.x+pt1.13cu117 \
    -f https://aiacc-inference-public-v2.oss-cn-hangzhou.aliyuncs.com/aiac
c-inference-llm/deepgpu_llm.html

# for PyTorch 2.0
pip3 install deepgpu_llm==xx.x+pt2.0cu117 \
    -f https://aiacc-inference-public-v2.oss-cn-hangzhou.aliyuncs.com/aiac
c-inference-llm/deepgpu_llm.html

# for PyTorch 2.1
pip3 install deepgpu_llm==xx.x+pt2.1cu121 \
    -f https://aiacc-inference-public-v2.oss-cn-hangzhou.aliyuncs.com/aiac
c-inference-llm/deepgpu_llm.html
```

2.4 补充：ubuntu 20.04安装注意事项

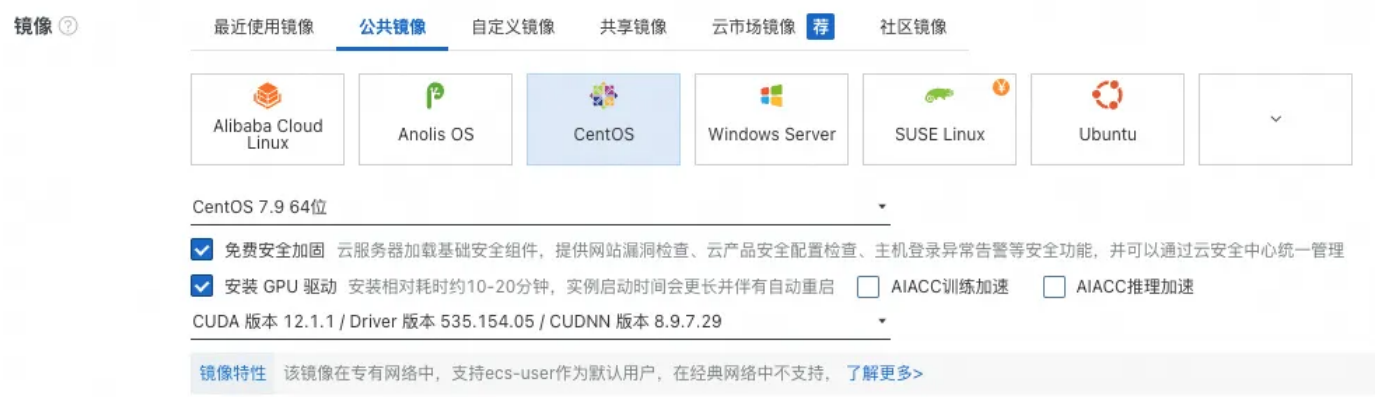
ubuntu 20.04系统上安装deepgpu-llm时，由于apt无法直接安装python 3.10，需要改成安装python 3.9，然后将系统自带的python版本改成默认python 3.9。具体安装python 3.9和调整默认python3版本的命令如下所示。

```
apt update
apt install software-properties-common
apt install python3.9
update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.8 1
update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.9 2
update-alternatives --config python3
python3 --version
```

3 在线部署：基于Centos 7.9系统

3.1 进入ECS购买页，选择所需镜像，开通机器

按需配置付费类型、地域、网络、可用区、实例类型等，在镜像栏目选择“公共镜像”–“CentOS”–“CentOS 7.9 64位”，可选直接**勾选安装GPU驱动+CUDA+CUDNN**，如下图所示。



3.2 等待系统安装和自动安装项目安装

ssh登陆可查看GPU驱动、CUDA、CUDNN等安装进展。

3.3 安装deepgpu-llm

逐一输入下面命令，完成deepgpu-llm部分依赖项目的在线安装。

```
yum install epel-release
yum update
yum install openmpi3 openmpi3-devel curl
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
chmod +x Miniconda3-latest-Linux-x86_64.sh
./Miniconda3-latest-Linux-x86_64.sh
```

其中openmpi3库安装成功后，需要修改环境变量（可添加至~/.bashrc）

```
export PATH=/usr/lib64/openmpi3/bin:$PATH
export LD_LIBRARY_PATH=/usr/lib64/openmpi3/lib:$LD_LIBRARY_PATH
```

其中miniconda环境需要按下面步骤进行使能和配置，使shell进入python 3.10环境（python 3.9也可，自行配置）。

```
/root/miniconda3/bin/conda init
source ~/.bashrc
conda create -n py310 python=3.10
conda activate py310
```

利用命令在线安装deepgpu-llm及其依赖项（耗时较长，耐心等待下载和安装）。也可参考2.3方法指定某个版本进行安装。

```
pip3 install deepgpu_llm -f https://aiacc-inference-public-v2.oss-cn-hangzhou.aliyuncs.com/aiacc-inference-llm/deepgpu_llm.html
```

3.4 补充：centos 8.5安装注意事项

centos 8.5系统中，yum或者dnf无法安装openmpi3和openmpi3-devel，这里我们改为安装openmpi和openmpi-devel。

```
yum install openmpi openmpi-devel
```

然后配置到环境变量里的openmpi相关目录也要改一下，改成下面这个。其他安装步骤和命令不变。

```
export PATH=/usr/lib64/openmpi/bin:$PATH
export LD_LIBRARY_PATH=/usr/lib64/openmpi/lib:$LD_LIBRARY_PATH
```

4 在线部署：基于容器docker

4.1 安装nvidia docker

若系统中已经安装了支持nvidia GPU的docker环境，本节可跳过。

第一步：安装或者升级docker-ce

基于ubuntu操作系统，可以参考下面命令进行操作。

```
apt update
apt remove docker docker-engine docker-ce docker.io containerd runc
apt install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
curl -fsSL https://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-key add -
apt-key fingerprint 0EBFCD88
add-apt-repository "deb [arch=amd64] https://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb_release -cs) stable"
apt update
apt install docker-ce
docker -v
```

在centos操作系统上，参考命令如下。

```
yum remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-engine
yum install -y yum-utils
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
yum install docker-ce docker-ce-cli containerd.io
systemctl start docker
systemctl enable docker
```

第二步：安装nvidia container

ubuntu系统中，可参考下面命令完成nvidia container的安装。

```
$ curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
  && curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-toolkit.list | \
    sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \
    sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list \
  && \
    sudo apt-get update

$ apt-get install -y nvidia-container-toolkit
$ nvidia-ctk runtime configure --runtime=docker
$ systemctl restart docker
```

centos系统上的安装命令有些区别，其命令罗列在下面。

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.repo | sudo tee /etc/yum.repos.d/nvidia-docker.repo
yum clean expire-cache
yum install -y nvidia-docker2
systemctl restart docker
```

4.2 下载并启动docker images

以下两个images都可以顺利安装deepgpu-llm，任选一个即可。

```
pytorch/pytorch:2.1.0-cuda12.1-cudnn8-devel
nvidia/cuda:12.1.0-devel-ubuntu22.04
```

docker下载可参考下面命令，以pytorch/pytorch:2.1.0-cuda12.1-cudnn8-devel为例。

```
docker pull pytorch/pytorch:2.1.0-cuda12.1-cudnn8-devel
```

docker启动命令如下

```
docker run -ti --gpus all --name="deepgpu_llm" --network=host \
           -v /root/workspace:/root/workspace \
           --shm-size 5g pytorch/pytorch:2.1.0-cuda12.1-cudnn8-devel
```

其中关键参数说明如下：

参数项	说明
--shm-size	指定容器的共享内存大小，其大小会影响服务器部署。 例如：--shm-size 5g表示将共享内存大小设置为5GB。您可以根据需要调整此值，以满足您的模型转换所需的内存需求。 （模型越大，模型转换需要的共享内存越大；若硬件限制无法扩展，可按5.4节方法提前转换模型，再进行部署）
-v /root/workspace:/root/workspace	将主机目录映射到Docker中的相应目录，使得主机和Docker之间可以共享文件，请根据自己实际环境情况进行映射。
nvidia/cuda:12.1.0-devel-ubuntu22.04	PyTorch的Docker映像标签
pytorch/pytorch:2.1.0-cuda12.1-cudnn8-devel	

4.3 安装deepgpu-llm

安装步骤与其他环境基本一致，具体命令如下所示。

```
apt-get update
apt-get -y install python3.10 python3-pip openmpi-bin libopenmpi-dev curl vim
pip3 install deepgpu_llm -f https://aiacc-inference-public-v2.oss-cn-hangzhou.aliyuncs.com/aiacc-inference-llm/deepgpu_llm.html
```

5 运行：利用deepgpu-llm运行qwen模型

5.1 检查deepgpu-llm安装状态和版本

利用以下命令查看deepgpu-llm安装版本信息。

```
pip list | grep deepgpu-llm
```

```
root@iZbp1iyvb2cr96l6jh1f47Z:~/deepgpu/app# pip list | grep deepgpu-llm
deepgpu-llm                   24.3+pt2.1cu121
```

这个命令可以查询更多deepgpu-llm相关信息。

```
pip show -f deepgpu-llm
```

5.2 下载开源模型：modelscope

modelscpoe是阿里达摩院提供的开源模型平台，下载modelscope格式的模型有如下两种方式。本步骤以通义千问Qwen模型为例。

- **git lfs clone命令方式**
  i. 进入modelscope官网，搜索模型名称（例如qwen）。
  ii. 在搜索页面的模型库区域，单击通义千问-7B-Chat。
  iii. 找到modelscope的专属模型名并复制模型ID。

iv. 执行以下命令，构建下载命令并下载模型。

```
git-lfs clone https://modelscope.cn/qwen/Qwen-7B-Chat.git
```

- **ModelScope库中的snapshot_download方式**
  i. 进入modelscope官网，搜索模型名称（例如qwen）。
  ii. 在搜索页面的**模型库**区域，单击**通义千问-7B-Chat**。
  iii. 找到modelscope的专属模型名并复制模型ID。

iv. 准备download_from_modelscope.py脚本。

```python
import argparse
import shutil
from modelscope.hub.snapshot_download import snapshot_download
parser = argparse.ArgumentParser(description='download from modelscope')
parser.add_argument('--model_name', help='the download model name')
parser.add_argument('--version', help='the model version')
args = parser.parse_args()
base_dir = '/root/deepgpu/modelscope'
model_dir = snapshot_download(args.model_name, cache_dir=base_dir,revision=args.version)
print(model_dir)
```

v. 执行以下命令，下载模型。模型下载前，您需要在**通义千问-7B-Chat**页面的**模型文件**页签下查看模型版本号。本命令以模型版本号为v.1.1.7为例。

Python

```
python3 download_from_modelscope.py --model_name qwen/Qwen-7B-Chat --version v1.1.7
```

5.3 运行deepgpu-llm：从原始模型目录加载

deepgpu-llm提供了llama_cli、qwen_cli、baichuan_cli和chatglm_cli等脚本帮助用户直接运行相应类别的LLM模型，可以查看“--help”获取帮助，查询具体配置项及其功能。

例如，可以通过下面命令来运行qwen_cli脚本加载qwen-7b-chat模型推理进行对话。

Plain Text

```
qwen_cli --model_dir /root/deepgpu/models/Qwen-7B-Chat --tp_size 1 --precision fp16
```

```
root@iZj6c97rvxtrv2379pt4u5Z:~# qwen_cli --model_dir /root/deepgpu/models/qwen-7b --tp_size 1 --precision fp16

=============== Argument ===============
model_dir: /root/deepgpu/models/qwen-7b
tp_size: 1
precision: fp16
tokenizer_dir: None
version: v1
========================================
Converting HF model: 100%|
running in fp16 mode, cb is False
欢迎使用 DeepGPU 加速版 Qwen 模型，输入内容即可进行对话，clear 清空对话历史，stop 终止程序

用户：你好，你是谁？
我是通义千问，由阿里云开发的AI助手。我可以回答各种问题、提供信息和与用户进行对话。有什么我可以帮助你的吗？

Cost time: 0.72 s
Throughput: 48.55 tokens/s
```

若需运行qwen1.5-7b-chat模型对话，请使用下面的命令。

Plain Text

```
qwen_cli --model_dir /root/deepgpu/models/Qwen1.5-7B-Chat --tp_size 1 --precision fp16
```

```
root@iZbp15nfxqxpeq6mcmn9raZ:~# qwen_cli --model_dir /root/deepgpu/models/Qwen1.5-7B-Chat --tp_size 1 --precision fp16

=============== Argument ===============
model_dir: /root/deepgpu/models/Qwen1.5-7B-Chat
tp_size: 1
precision: fp16
tokenizer_dir: None
========================================
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
Converting HF model: 100%|
running in fp16 mode, cb is False
欢迎使用 DeepGPU 加速版 Qwen 模型，输入内容即可进行对话，clear 清空对话历史，stop 终止程序

用户：你好，你是谁？
我是阿里云推出的一种超大规模语言模型，我叫通义千问
Cost time: 0.34 s
Throughput: 50.54 tokens/s

用户：介绍下杭州的旅游景区？
杭州是中国浙江省的省会，拥有丰富的自然风光和人文景观，以下是一些著名的旅游景区：

1. 西湖：西湖被誉为“人间天堂”，是中国最著名的风景名胜区之一。它包括了苏堤春晓、断桥残雪、雷峰夕照、三潭印月等多个景点，湖面如镜，四季景色各异。

2. 龙井茶园：位于杭州市西部的龙井村，是绿茶的重要产地，可以参观茶园，品尝新鲜茶叶，了解中国茶文化。

3. 千岛湖：位于杭州市淳安县，是一个大型的人工湖，有1078个岛屿，湖光山色非常美丽，适合游玩和度假。

4. 梵净山：距离杭州市较远，位于贵州省铜仁市，是中国著名的佛教名山，山上有许多寺庙和佛教文化遗产。

5. 西溪湿地：位于杭州市区西部，是一个大型的生态湿地公园，有着丰富的水生植物和野生动物，可以乘船游览，感受自然生态。

6. 宋城：以宋朝文化为主题的主题公园，重现了宋代的生活场景，有精彩的表演和各种互动体验。

7. 九溪烟树：位于杭州市西部的淳安县，九溪溪流蜿蜒，山峦叠翠，是杭州近郊的著名山水景观。

8. 杭州宋城千古情：大型实景演出，讲述了一段跨越千年的爱情故事，结合高科技舞台效果，十分震撼。

9. 浙江博物馆：展示了浙江地区的丰富历史文物和艺术珍品，是了解浙江文化和历史的好去处。

10. 城隍山：位于杭州市中心，是一座历史悠久的山，山上有城隍庙，可以登高望远，俯瞰城市风光。

以上只是杭州众多景区的一部分，每个地方都有独特的魅力和特色，可以根据个人兴趣选择前往
Cost time: 4.39 s
Throughput: 89.91 tokens/s
```

5.4 运行deepgpu-llm：转换模型并运行

上一节中直接运行的方法，会在初始化过程中在线转换模型，且该过程对CPU内存/shm_size有一定需求。对于某些受限场景，可以提前做好模型转换，然后部署运行即可。针对不同模型开发了不同的转换命令，其中llama系列使用huggingface_llama_convert，chatglm模型使用huggingface_glm_convert，chatglm2和chatglm3使用huggingface_chatglm2_convert，baichuan系列模型使用huggingface_baichuan_convert，qwen v1版本模型转换使用huggingface_qwen_convert，qwen v1.5版本模型转换使用huggingface_qwen15_convert。

以qwen1.5-7b-chat模型为例，其转换命令如下所示，-in_file指向下载的模型目录，-saved_dir指向生成的模型目录，-infer_gpu_num设定推理运行的GPU数量（即模型切分份数），-weight_data_type设置模型权重使用的数据类型，与预期计算的类型一致，可选fp16、bf16和fp32，-model_name为模型名称。

Plain Text

```
huggingface_qwen15_convert -in_file /root/deepgpu/models/Qwen1.5-7B-Chat -saved_dir /root/deepgpu/models/deepgpu/qwen1.5-7b-chat -infer_gpu_num 1 -weight_data_type fp16 -model_name qwen1.5-7b-chat
```

模型转换好后，按下面命令运行推理。其中--tp_size配置的参数需要与转换时-infer_gpu_num设定的参数一致；--precision用于设置是否要对权重进行量化，可选fp16、int8和int4。

Plain Text

```
qwen_cli --tokenizer_dir /root/deepgpu/models/Qwen1.5-7B-Chat --model_dir /root/deepgpu/models/deepgpu/qwen1.5-7b-chat/1-gpu/  --tp_size 1 --precision fp16
```

```
root@iZj6c97rvxtrv2379pt4u5Z:~# qwen_cli --tokenizer_dir /root/deepgpu/models/Qwen1.5-7B-Chat --model_dir /root/deepgpu/models/deepgpu/qwen1.5-7b-chat/1-gpu/  --tp_size 1 --precision fp16

============== Argument ==============
model_dir: /root/deepgpu/models/deepgpu/qwen1.5-7b-chat/1-gpu/
tp_size: 1
precision: fp16
tokenizer_dir: /root/deepgpu/models/Qwen1.5-7B-Chat
version: v1
======================================
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
running in fp16 mode, cb is False
欢迎使用 DeepGPU 加速版 Qwen 模型，输入内容即可进行对话，clear 清空对话历史，stop 终止程序

用户：你好，你是谁？
我是阿里云推出的一种超大规模语言模型，我叫通义千问
Cost time: 0.36 s
Throughput: 47.16 tokens/s
```