

llm 接口高可用工程实践 课程大纲

课程介绍

本文基于 Chinese-LLaMA-Alpaca-3（<https://github.com/ymcui/Chinese-LLaMA-Alpaca-3>）

项目，介绍如何通过搭建 2 个 llama3 私有化 api 和 oneapi 网关，实现大模型接口的高可用方

案。

（1）上节课，我们进行了 llama3-8b 模型的本地化部署及仿 openai api 接口开放;

（2）本次课基于上节课的基础，进行 llama3-8b 本地化及接口的高可用；

学习知识：

（1）windows 终端 ssh 服务器、windows 版本客户端安装；

（2）windows 作为客户端的 cpu/gpu 大模型的部署和应用：

（3）基于容器(docker)安装 oneapi 及配置，使用 oneapi 连接已部署的 cpu/gpu 版本大模型;

（4）基于 oneapi 学习高可用测试；

（5）一些常用指令操作;

实验环境

本文的基础环境如下

操作系统：ubuntu20.04

CUDA：12.2

英伟达显卡驱动版本：535.183.01

显卡型号：本实验用英伟达 3090（全模型（未开量化）需要单张显存大于 20GB 的英伟达显

卡；

\*\*开量化之后的模型可以使用 11GB 显存的英伟达显卡，基本市面上大部分消费级显卡可以满

足实验要求,

大家根据自己的显卡进行选择实验）

客户端演示环境: windows

实验环境准备

MacOS

命令行工具

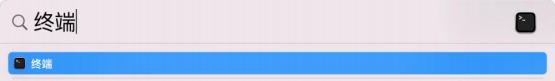
如果您使用的是苹果操作系统，可以直接使用系统自带的终端软件来连接服务器。具体操作步

骤如下，首先按 Command+空格打开聚焦搜索界面：



然后在聚焦搜索框中，输入终端两个字，如下图：





回车即可打开终端：

大模型聊天客户端

我们使用 ChatGPTNextWeb 工具测试我们的接口，如果您没有下载客户端，可以通过下面地址

下载：

Mac：[https://github.com/ChatGPTNextWeb/ChatGPT-Next-](https://github.com/ChatGPTNextWeb/ChatGPT-Next-Web/releases/download/v2.14.2/NextChat_2.14.2_universal.dmg)

[Web/releases/download/v2.14.2/NextChat\_2.14.2\_universal.dmg](https://github.com/ChatGPTNextWeb/ChatGPT-Next-Web/releases/download/v2.14.2/NextChat_2.14.2_universal.dmg)

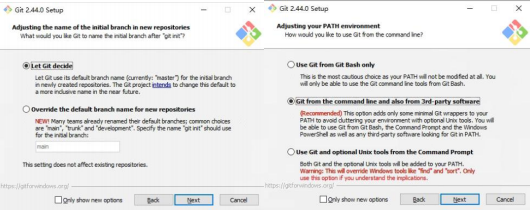
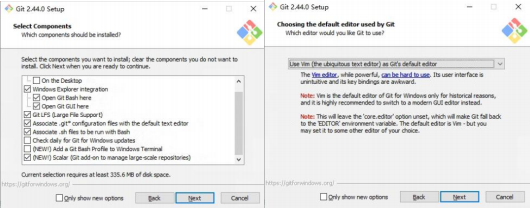
Windows10/11

命令行工具

在 windows 系统中，我们使用 Git Bash 来连接服务器，下面是具体安装步骤，首先我们通过

如下地址下载安装包：[https://github.com/git-for-](https://github.com/git-for-windows/git/releases/download/v2.46.0.windows.1/Git-2.46.0-64-bit.exe)

[windows/git/releases/download/v2.46.0.windows.1/Git-2.46.0-64-bit.exe](https://github.com/git-for-windows/git/releases/download/v2.46.0.windows.1/Git-2.46.0-64-bit.exe)，下载好以 后，双击安装包，出现如下画面：

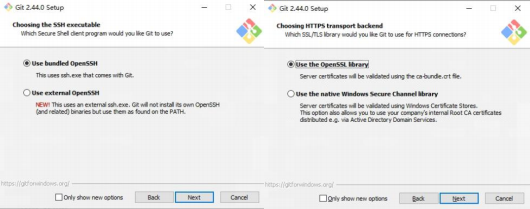


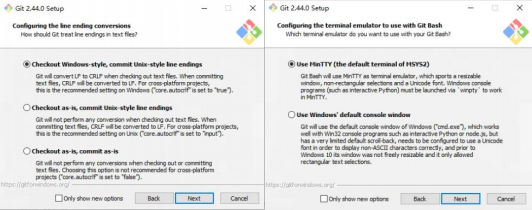


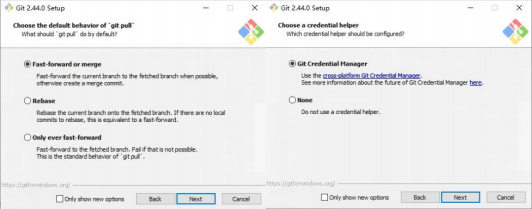
可以将 Only show new options 选项去掉，以看到更多的安装选项。点击 Next，将看到如下

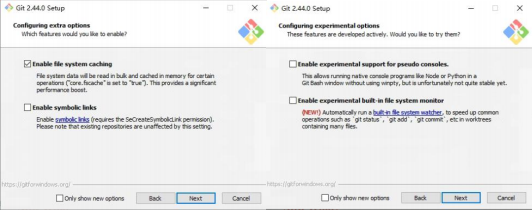
图：

继续点击 Next，出现如下图界面：



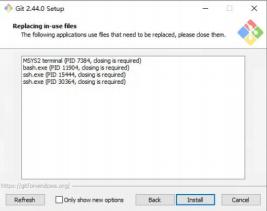








在上述界面中，一直点击 Next 即可。最后点击 install:



大模型聊天客户端

我们使用 ChatGPTNextWeb 工具测试我们的接口，如果您没有下载客户端，可以通过下面地址

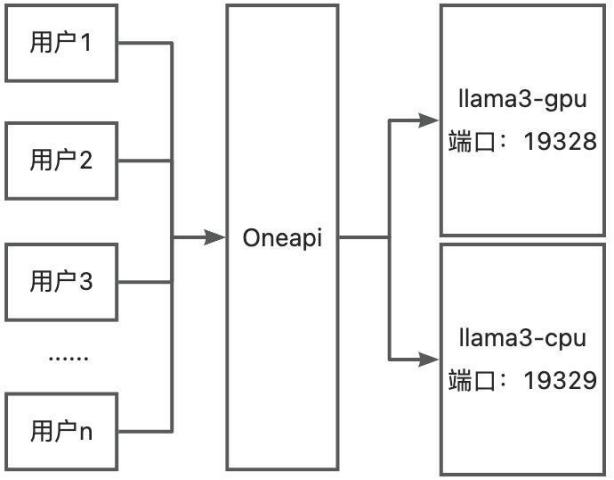
下载：

Windows：[https://github.com/ChatGPTNextWeb/ChatGPT-Next-](https://github.com/ChatGPTNextWeb/ChatGPT-Next-Web/releases/download/v2.14.2/NextChat_2.14.2_x64-setup.exe)

[Web/releases/download/v2.14.2/NextChat\_2.14.2\_x64-setup.exe](https://github.com/ChatGPTNextWeb/ChatGPT-Next-Web/releases/download/v2.14.2/NextChat_2.14.2_x64-setup.exe)

下载后，双击安装包运行，如果出现如下画面，点击更多信息：

然后一直点击 Next 即可安装。



开始实验

本文默认您在操作系统的用户名为：llm\_course，如果您使用的是其他用户名，请更改涉及到

用户名的地址。基本的搭建架构图如下：

我们通过人工停掉 llama3-cpu，保持 llama3-gpu 运行，来模拟节点故障。

安装 Docker（如有可跳过）

#移除 Docker 相关的软件包和容器运行时

apt-get remove docker-ce docker-ce-cli containerd.io docker-compose-plugin docker docker-

engine docker.io containerd runc

# 切换国内源

sed -i -r 's#http://(archive|security).ubuntu.com#https ://mirrors.aliyun.com#g'

/etc/apt/sources.list && apt-get update

### 安装 docker

# 安装 ca-certificates curl gnupg lsb-release

apt-get install ca-certificates curl gnupg lsb-release

#下载 Docker 的 GPG 密钥，并将其添加到 apt-key 中

curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | apt-key add -

# 为 Docker 添加阿里源

add-apt-repository "deb [arch=amd64] http://mirrors.aliyun.com/docker-ce/linux/ubuntu

$(lsb\_release -cs) stable"

# 更新系统的软件包

apt -y update

# 安装 docker 相关的包

apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

兼容多端口启动

因为我们要基于一份代码，启动两个服务，我们需要在原有的代码中加入端口参数，打开下面 文件：~/Chinese-LLaMA-Alpaca-3-3.0/scripts/oai\_api\_demo/openai\_api\_server.py，做如

下修改：

|  |
| --- |
| parser = argparse.ArgumentParser () # 找到改行代码，在改行代码下添加一行  parser.add\_argument('--port', default=19327, type=int) # 新添加一行  # 找到这行代码  uvicorn.run (app, host="0.0.0.0", port=19327, workers=1, log\_config=log\_config)  # 改为如下内容  uvicorn.run (app, host="0.0.0.0", port=args.port, workers=1, log\_config=log\_config) |

启动 llama3 gpu 和 cpu 版本

我们假设您已在上次课《中文 llama3 仿 openai api 实践》中搭建好了相应的环境。如果您还

未搭建对应的环境，可以参考之前的课程文档。

部署 oneapi

创建 docker-compose.yam 文件

启动 gpu 版本

|  |
| --- |
| conda activate chinese\_llama\_alpaca\_3 # 激活虚拟环境  cd ~/Chinese-LLaMA-Alpaca-3-3.0/scripts/oai\_api\_demo/ # 进入脚本目录  python openai\_api\_server.py --gpus 0 --base\_model  /home/llm\_course/.cache/modelscope/hub/ChineseAlpacaGroup/llama-3-chinese-8b-instruct-v3 |

默认情况下，程序将监听 19327 端口。

启动 cpu 版本

|  |
| --- |
| conda activate chinese\_llama\_alpaca 3 cpu # 激活虚拟环境  cd ~/Chinese-LLaMA-Alpaca-3-3.0/scripts/oai\_api\_demo/ # 进入脚本目录  python openai\_api\_server.py --only\_cpu --port 19328 --base\_model  /home/llm\_course/.cache/modelscope/hub/ChineseAlpacaGroup/llama-3-chinese-8b-instruct-v3 |

|  |
| --- |
| mkdir -p ~/oneapi-compose  cd ~/oneapi-compose  touch docker-compose.yaml |

在 docker-compose.yaml 文件中添加如下内容（命令行中可以使用 nano docker-

compose.yaml 打开）：

version: '3.8'

services :

oneapi:

image: m.daocloud.io/ghcr.io/songquanpeng/one-api:v0.6.7

使用如下命令启动容器：docker compose up -d

修改默认密码

打开刚才部署的 oneapi 页面，通过 ifconfig 命令可以查看您的服务器地址，我们这里假设您 的服务器地址是：192.168.9.96，在浏览器中打开：http://192.168.9.96:3030，可以看到如

下界面：

container\_name: oneapi

restart : always

ports:

- 3030:3000

networks:

- oneapi\_llm\_course

environment:

- TZ=Asia/Shanghai

volumes :

- ./data:/data

networks :

oneapi\_llm\_course :

启动服务



点击登录，当您第一次登录 oneapi 时，默认用户名是 root，默认密码是 123456，登录后需要

立即修改密码：



我们这里将密码设置为：Llm\_course，您可以设置您自己认为安全的密码。

添加渠道

添加 llama3-gpu 渠道





再次点击渠道菜单，可以看到如下页面：



添加 llama3-cpu 渠道



再次点击渠道菜单，可以看到如下页面：



添加 Token

我们需要添加一个 Token 用于后续测试：



输入令牌详情：

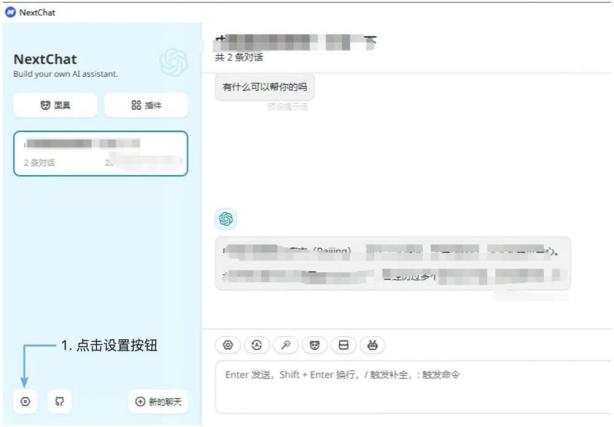


复制刚才创建的 Token 在后续高可用测试中将用到该 Token：

测试高可用

配置 ChatGPTNextWeb

我们使用 ChatGPTNextWeb 来进行测试，首先我们需要配置该软件。如下图：



设置渠道信息：



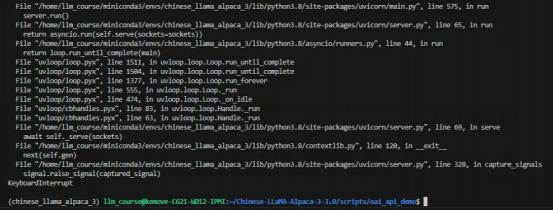
双渠道可用

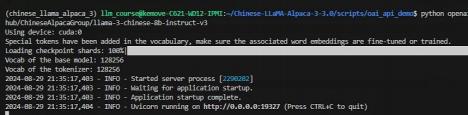
我们输入同样的问题两次，可以观察回答一次慢，一次快，说明是第一次是调用了 CPU 渠道，

第二次是调用了 GPU 渠道，您在测试时，有可能是第一次快，第一次慢，是正常的现象。

CPU 可用 GPU 不可用

我们切换到 GPU 渠道的命令行运行界面：





我们按 Ctrl+c 结束掉 GPU 渠道，如下所示：

我们再次回到 ChatGPTNextWeb 窗口，发送问题，依然可以得到回答，只是回答得非常慢，说 明 CPU 渠道正在提供服务，我们到 oneapi 的渠道页面，对 gpu 渠道进行测试，应该会得到如

下信息：

GPU 可用 CPU 不可用

我们回到 GPU 渠道的命令行界面，再次启动 GPU 渠道服务：





然后切换到 CPU 渠道命令行页面，按 Ctrl+c 结束掉 CPU 渠道，如下所示：



我们再次回到 ChatGPTNextWeb 窗口，发送问题，依然可以得到回答，这次的回答非常快，说 明 GPU 渠道正在提供服务，我们到 oneapi 的渠道页面，对 cpu 渠道进行测试，应该会得到如

下信息：

常见问题

oneapi 怎么做到高可用

以火山引擎为例，我们可以在多台 ECS（云服务器）中部署 oneapi，然后设置一个域名映射到

多个 ip 即可：



作业：

复现本次实验，并提交实验报告