vllm安装  
# (Recommended) Create a new conda environment.

conda create -n qwen python=3.10 -y

conda activate qwen

# Install vLLM with CUDA 12.1.

pip install vllm

服务部署：

vllm serve /root/autodl-fs/Qwen/Qwen2\_\_\_5-32B-Instruct --dtype auto --api-key token-abc123 --host 0.0.0.0 --port 6006 --tensor-parallel-size 4  
#将端口设置成6006的原因就是autodl那边会将实例中的6006端口映射到公网可供访问的ip:port

vllm serve /root/autodl-tmp/Qwen/Qwen2\_\_\_5-7B-Instruct --dtype auto --api-key token-abc123 --host 0.0.0.0 --port 8000

vllm serve /root/autodl-fs/models--unsloth--Qwen2.5-32B-Instruct/snapshots/3656157659e7794e48ce97583273bef618354b45

--dtype auto --api-key token-abc123 --host 0.0.0.0 --port 8000 --tensor-parallel-size 4

Vllm 测试性能指标的命令：  
--input\_length 模型输入的文本长度

----output\_length 模型输出的文本长度

--num\_threads 单位时间内发送的请求数，也可以称batch

python vllm\_inference.py --input\_length 4000 --output\_length 800 --num\_threads 64

Tensorrt\_llm安装命令：

mkdir /root/autodl-tmp/tmp

export TMPDIR=/root/autodl-tmp/tmp

安装对应的cudann，我这里安装 的是12.4.2版本

wget <https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64/cuda-ubuntu2204.pin>

sudo mv cuda-ubuntu2204.pin /etc/apt/preferences.d/cuda-repository-pin-600

wget <https://developer.download.nvidia.com/compute/cuda/12.4.1/local_installers/cuda-repo-ubuntu2204-12-4-local_12.4.1-550.54.15-1_amd64.deb>

sudo dpkg -i cuda-repo-ubuntu2204-12-4-local\_12.4.1-550.54.15-1\_amd64.deb

sudo cp /var/cuda-repo-ubuntu2204-12-4-local/cuda-\*-keyring.gpg /usr/share/keyrings/

sudo apt-get update

sudo apt-get -y install cuda-toolkit-12-4

#删除已经安装的包（进入到var目录下，删除相对应的包）

rm -rf /var/cudnn-local-repo-ubuntu2204-9.0.0/

rm -rf /var/cudnn-local-repo-ubuntu2204-8.9.4.25/

cd /usr/local目录下，删除旧的cudann包，比如旧的cudann是12.2版本的，你就删除掉。

rm -rf cuda12.1

conda create -n tensorrt python=3.10

conda activate tensorrt

conda install -c conda-forge ucx ucc nccl

conda install mpi4py openmpi

conda install cuda-cudart cuda-version=12

apt-get update && apt-get -y install openmpi-bin libopenmpi-dev git git-lfs

pip install tensorrt\_llm==0.13.0 --extra-index-url https://pypi.nvidia.com

pip cache purge

# Build Qwen-14B-Chat using 2-way tensor parallelism.

git clone https://github.com/NVIDIA/TensorRT-LLM.git

python convert\_checkpoint.py --model\_dir /root/autodl-fs/Qwen/Qwen2\_\_\_5-32B-Instruct --output\_dir /root/autodl-fs/tllm\_checkpoint\_4gpu\_tp4 --dtype bfloat16 --tp\_size 2

trtllm-build --checkpoint\_dir /root/autodl-fs/tllm\_checkpoint\_4gpu\_tp4 --output\_dir /root/autodl-fs/qwen/32B/trt\_engines/fp16/4-gpu --gpt\_attention\_plugin bfloat16 --gemm\_plugin bfloat16 --kv\_cache\_type paged

测试模型的命令：  
# Run 32B model with -gpu

mpirun -n 2 --allow-run-as-root python ../run.py --input\_text "请对下面文章总结出50个字的摘要，文章如下：沾衣欲湿杏花雨，吹面不寒杨柳风。”春风吹吹，吹动发丝为她伴舞，吹动柳丝为她摇曳，吹动鲜花为她绽放--春风拂面，轻轻地，轻轻地亲吻着你的脸颊，痒酥稣的，不知不觉便泛起了抹抹红晕。醉了，那么彻底，那么忘我的醉————是风在作祟。看，春风驾着五彩祥云姗姗而来，浑身散发着缕缕沁人的清香。她化作娇嫩的草芽，从坚实的大地中钻了出来：她化作北归的大雁，从遥远的天际飞了回来;她化作姑娘身上薄如蝉翼的丝巾从熙攘的人群中款款走来--和煦的春风是春的使者。看，她飞过了山脉大川，留下了丝丝绿意，掠过了大地，留下了阵阵清爽，飘过了翠绿的田野，留下了点点翠光。春风娓娓讲述着缠绵悱恻的絮语。“不知细叶谁裁出，二月春风似剪刀。”那纤细的柳叶，清晰地脉络，果然出自是春风的妙笔。“春风又绿江南岸，明月何时照我还。”吹绿了江南的春风，你何时又成为了信使，将游子的乡愁传递。春风啊，你又为何不度玉门关，在羌笛声中回旋飘荡。桃花依旧，以灿烂的微笑欢迎着春风，将笑魇如花的美尽情绽放。春风拂面的季节，依偎在大自然的怀抱中，徜徉在泼墨山水的诗情画意中，何其洒脱，何其开怀。春风的爱抚使“乱花渐欲迷人眼，浅草才能没马蹄”，更使“春潮带雨晚来急。”春雨霏霏，在春风的邀请下也来参加这场盛大的春之宴会。于是，春雨在风姑娘的的陪伴下，流星般滑落天际，奏出优美动听的旋律，灵动的音符随风飘荡。颤动的七彩五线谱斜挂在树梢上，如姹紫嫣红的繁花在抖动的枝叶上绚烂着，舞蹈着。不只是哪位情趣高雅的诗人脱口而出“随风潜入夜，润物细无声”的妙语，应和着优扬的乐曲。春风化雨，春雨戏风。他们嬉闹着，欢笑着，不知不觉荡涤了污浊的心灵。同时将纯洁的种子撒向大地，使纯真无邪的童心在世间开花结果。春风飘飘，带着人们的期盼与希望，携着万物的祝福飞向了远方。但那春风拂面的感觉依然很是清晰，依然令人陶醉。高中散文精选800字：叹中华赋无韵之离骚”的他" --max\_output\_len=200 --tokenizer\_dir /root/autodl-fs/Qwen/Qwen2\_\_\_5-32B-Instruct --engine\_dir=/root/autodl-fs/qwen/32B/trt\_engines/fp16/4-gpu

**下面用tensorrtllm\_backend来搭建服务**

1.git clone <https://github.com/triton-inference-server/tensorrtllm_backend.git>

ENGINE\_DIR=/root/autodl-fs/qwen/32B/trt\_engines/fp16/4-gpu

TOKENIZER\_DIR=/root/autodl-fs/Qwen/Qwen2\_\_\_5-32B-Instruct

MODEL\_FOLDER=/triton\_model\_repo

TRITON\_MAX\_BATCH\_SIZE=4

INSTANCE\_COUNT=1

MAX\_QUEUE\_DELAY\_MS=0

MAX\_QUEUE\_SIZE=0

FILL\_TEMPLATE\_SCRIPT=/root/tensorrtllm\_backend/tools/fill\_template.py

DECOUPLED\_MODE=false

python3 ${FILL\_TEMPLATE\_SCRIPT} -i ${MODEL\_FOLDER}/ensemble/config.pbtxt triton\_max\_batch\_size:${TRITON\_MAX\_BATCH\_SIZE}

python3 ${FILL\_TEMPLATE\_SCRIPT} -i ${MODEL\_FOLDER}/preprocessing/config.pbtxt tokenizer\_dir:${TOKENIZER\_DIR},triton\_max\_batch\_size:${TRITON\_MAX\_BATCH\_SIZE},preprocessing\_instance\_count:${INSTANCE\_COUNT}

python3 ${FILL\_TEMPLATE\_SCRIPT} -i ${MODEL\_FOLDER}/tensorrt\_llm/config.pbtxt triton\_backend:tensorrtllm,triton\_max\_batch\_size:${TRITON\_MAX\_BATCH\_SIZE},decoupled\_mode:${DECOUPLED\_MODE},engine\_dir:${ENGINE\_DIR},max\_queue\_delay\_microseconds:${MAX\_QUEUE\_DELAY\_MS},batching\_strategy:inflight\_fused\_batching,max\_queue\_size:${MAX\_QUEUE\_SIZE}

python3 ${FILL\_TEMPLATE\_SCRIPT} -i ${MODEL\_FOLDER}/postprocessing/config.pbtxt tokenizer\_dir:${TOKENIZER\_DIR},triton\_max\_batch\_size:${TRITON\_MAX\_BATCH\_SIZE},postprocessing\_instance\_count:${INSTANCE\_COUNT},max\_queue\_size:${MAX\_QUEUE\_SIZE}

python3 ${FILL\_TEMPLATE\_SCRIPT} -i ${MODEL\_FOLDER}/tensorrt\_llm\_bls/config.pbtxt triton\_max\_batch\_size:${TRITON\_MAX\_BATCH\_SIZE},decoupled\_mode:${DECOUPLED\_MODE},bls\_instance\_count:${INSTANCE\_COUNT}

#### Serving with Triton

# 'world\_size' is the number of GPUs you want to use for serving. This should

# be aligned with the number of GPUs used to build the TensorRT-LLM engine.

MODEL\_FOLDER=/triton\_model\_repo

python3 /tensorrtllm\_backend/scripts/launch\_triton\_server.py --world\_size=2 --model\_repo=${MODEL\_FOLDER}