TensorRT-LLM保姆级教程（二）-离线环境搭建、模型量化及推理

吃果冻不吐果冻皮

随着大模型的爆火，投入到生产环境的模型参数量规模也变得越来越大（从数十亿参数到千亿参数规模），从而导致大模型的推理成本急剧增加。因此，市面上也出现了很多的推理框架，用于降低模型推理延迟以及提升模型吞吐量。

本系列将针对TensorRT-LLM推理进行讲解。本文为该系列第二篇，将基于Bloom进行模型量化及推理。

> 另外，我撰写的**大模型相关的博客及配套代码**均整理放置在Github：**llm-action**，有需要的朋友自取。

环境搭建

基础配置：

- CUDA：12.2
- 镜像：nvcr.io/nvidia/pytorch:...

由于服务器无法访问外网，只能预先准备好镜像，安装包、编译源码等，接下来准备安装TensorRT-LLM，推荐使用 Docker 构建和运行 TensorRT-LLM，整个安装步骤参考 TensorRT-LLM 中构建 Docker 镜像的**步骤**。

首先，进入Docker容器。

```
docker run -dt --name tensorrt_llm_lgd \
--restart=always \
--gpus all \
--network=host \
--shm-size=4g \
-m 64G \
-v /home/guodong.li/workspace:/workspace \
-w /workspace \
nvcr.io/nvidia/pytorch:23.10-py3 \
/bin/bash

docker exec -it tensorrt_llm_lgd bash
```

安装PyTorch、TensorRT、mpi4py等：

```
# 卸载TensorRT
pip uninstall -y tensorrt*
pip uninstall -y torch-tensorrt

pip install mpi4py -i http://nexus3.xxx.com/repository/pypi/simple --trusted-host nexu
pip install polygraphy-0.48.1-py2.py3-none-any.whl -i http://nexus3.xxx.com/repository

# 重新安装PyTorch
pip install torch==2.1.0 -i http://nexus3.xxx.com/repository/pypi/simple --trusted-hos
pip uninstall transformer-engine


# 重新安装TensorRT
tar -xf /tmp/TensorRT.tar -C /usr/local/
```

知乎 首发于 动手学大模型

配置环境变量 `/etc/profile`：

```
ENV LD_LIBRARY_PATH=/usr/local/tensorrt/lib:${LD_LIBRARY_PATH}
```

构建 TensorRT-LLM：

```
python3 ./scripts/build_wheel.py --clean --trt_root /usr/local/tensorrt --cuda_archite
```

由于离线构建，需修改配置文件：

1. 修改pip源：`https://github.com/NVIDIA/TensorRT-LLM/blob/release/0.5.0/scripts/build_wheel.py#L65` 。
2. 修改git远程仓库地址：`https://github.com/NVIDIA/TensorRT-LLM/blob/release/0.5.0/cpp/tests/CMakeLists.txt#L19`

安装TensorRT-LLM：

```
pip install ./build/tensorrt_llm*.whl -i http://nexus3.xxx.com/repository/pypi/simple
```

至此，整个环境搭建就完成了。

基于 Bloom 模型开发实践简介

接下来以Bloom模型为例，进行 TensorRT-LLM 开发实践。

Bloom 示例中主要文件：

- `build.py`：用于构建 TensorRT 引擎来运行Bloom模型。
- `run.py`：模型推理。
- `summarize.py`：使用模型来总结 CNN Dailymail 数据集中的文章。
- `hf_bloom_convert.py`：将HF格式的模型进行转换。

TensorRT-LLM 中，目前针对 Bloom 模型支持的特性如下：

- 支持 FP16
- 支持 INT8 & INT4 仅权重量化
- 支持 INT8 KV CACHE 量化
- 支持SmoothQuant 量化
- 支持张量并行

关于大模型量化之前的文章：**大模型量化概述** 进行过简要概述，后续有时间更详细的梳理常见的一些大模型量化技术。

数据与模型下载

下载Bloom模型，本文基于bloomz-3b进行量化和推理。

```
# 需先安装git-lfs，通常情况下前面已经安装过了。
# git lfs install

# 下载模型
```

下载数据集，本文会用到 CNN Dailymail 数据集和 LAMBADA 数据集。

- huggingface.co/datasets...
- huggingface.co/datasets...

构建 TensorRT 引擎

TensorRT-LLM 基于 HF 上 Bloom 的 checkpoint 构建 TensorRT 引擎。 如果未指定 checkpoint 目录，TensorRT-LLM 将使用虚拟权重构建引擎。

下面使用 build.py 脚本来构建TensorRT 引擎；通常，build.py 仅需单个 GPU，但如果您有推理所需的所有 GPU，则可以通过添加 --parallel_build 参数来启用并行构建，以使引擎构建过程更快。

注意：目前parallel_build功能仅支持单节点。

`hf_bloom_convert.py` 脚本常用参数说明：

- out_dir：模型格式转化之后的输出路径。
- in_file：原始模型路径。
- tensor_parallelism：模型推理时的张量并行度
- calibrate_kv_cache：生成 KV 缓存的缩放因子。 以 INT8 存储 KV Cache 时使用。
- smoothquant：使用Smoothquant对模型进行量化时设置α参数， ，并输出int8权重。第一次尝试最好是 0.5。 该参数必须在 [0, 1] 之间。
- storage_type：设置模型参数存储的数据类型。

`build.py` 脚本常用参数说明：

- model_dir：指定原始HF模型目录。
- bin_model_dir：SmoothQuant 或 KV CACHE 量化时，指定模型转换后的二进制文件。
- dtype：指定模型数据类型。
- use_gemm_plugin：设置gemm数据类型。
- use_gpt_attention_plugin：设置attention的数据类型。
- output_dir：引擎输出目录。
- use_layernorm_plugin：设置layernorm的数据类型。
- use_weight_only：设置仅权重量化，将各种 GEMM 的权重量化为 INT4/INT8。。
- weight_only_precision：设置仅权重量化时的权重精度。必须使用use_weight_only时该参数才会生效。
- use_smooth_quant：使用 SmoothQuant 方法量化各种 GEMM 的激活和权重。更细粒度的量化选项，使用 --per_channel 和 --per_token 参数选型。
- per_channel：默认情况下，对 GEMM 结果使用单个静态缩放因子。per_channel 相反，它为每个通道使用不同的静态缩放因子。后者通常更准确，但速度稍慢。
- per_token：默认情况下，我们使用单个静态缩放因子来缩放 int8 范围内的激活。per_token 在运行时为每个token选择一个自定义缩放因子。后者通常更准确，但速度稍慢。
- int8_kv_cache：默认情况下，使用 dtype 进行 KV 缓存。 int8_kv_cache为KV选择int8量化。
- use_parallel_embedding：默认情况下，嵌入并行被禁用。通过设置此参数，可以启用嵌入并行。
- embedding_sharding_dim：尝试通过在两层之间共享嵌入查找表来减小引擎大小。注意：当不满足条件时，该参数可能不会生效。

FP16

使用 HF 权重**基于单 GPU 及 float16 精度构建引擎**。使用 use_gemm_plugin 来防止准确性问题。

```
python build.py --model_dir /workspace/model/bloomz-3b \
                --dtype float16 \
```

```
                --output_dir /workspace/model/bloomz-3b_trt_engines/fp16/1-gpu/
```

输出模型引擎文件：

```
> tree -h  /workspace/model/bloomz-3b_trt_engines/fp16/1-gpu/
├── [6.8G]  bloom_float16_tp1_rank0.engine
├── [1.2K]  config.json
└── [327K]  model.cache
```

仅 INT8 权重量化（W8A16）

使用单 GPU 和仅 INT8 权重量化构建引擎：

```
python build.py --model_dir /workspace/model/bloomz-3b \
                --dtype float16 \
                --use_gemm_plugin float16 \
                --use_gpt_attention_plugin float16 \
                --use_weight_only \
                --output_dir /workspace/model/bloomz-3b_trt_engines/int8_weight_only/1
```

输出模型引擎文件：

```
> tree -h /workspace/model/bloomz-3b_trt_engines/int8_weight_only/1-gpu/

├── [4.6G]  bloom_float16_tp1_rank0.engine
├── [1.2K]  config.json
└── [317K]  model.cache
```

FP16 + 2路张量并行

使用2路张量并行构建引擎：

```
python build.py --model_dir /workspace/model/bloomz-3b \
                --dtype float16 \
                --use_gemm_plugin float16 \
                --use_gpt_attention_plugin float16 \
                --output_dir /workspace/model/bloomz-3b_trt_engines/fp16/2-gpu/ \
                --world_size 2
```

输出模型引擎文件：

```
> tree -h  /workspace/model/bloomz-3b_trt_engines/fp16/2-gpu/

├── [4.0G]  bloom_float16_tp2_rank0.engine
├── [4.0G]  bloom_float16_tp2_rank1.engine
├── [1.2K]  config.json
└── [327K]  model.cache
```

仅 INT8 权重量化 & INT8 KV CACHE 量化

下面**使用仅 INT8 权重量化及 INT8 KV CACHE 量化**：

对于 INT8 KV 缓存，hf_bloom_convert.py 脚本中有 `--calibrate-kv-cache` 、 `-kv` 选项。设置 `-kv` 将校准模型，然后导出 INT8 KV CACHE推理所需的缩放因子（scaling factors）。

```
-o /workspace/model/bloom-c-model/int8_kv_cache/3b \
--calibrate-kv-cache -t float16
```

输出结果：

```
> tree -h /workspace/model/bloom-c-model/int8_kv_cache/3b
/workspace/model/bloom-c-model/int8_kv_cache/3b
└── [ 28K]  1-gpu
    ├── [2.1K]  config.ini
    ├── [5.0K]  model.final_layernorm.bias.bin
    ├── [5.0K]  model.final_layernorm.weight.bin
    ├── [5.0K]  model.layers.0.attention.dense.bias.bin
    ├── [ 12M]  model.layers.0.attention.dense.weight.0.bin
    ├── [ 15K]  model.layers.0.attention.query_key_value.bias.0.bin
    ├── [   4]  model.layers.0.attention.query_key_value.scale_y_quant_orig.bin
    ├── [ 38M]  model.layers.0.attention.query_key_value.weight.0.bin
    ├── [5.0K]  model.layers.0.input_layernorm.bias.bin
    ├── [5.0K]  model.layers.0.input_layernorm.weight.bin
    ├── [5.0K]  model.layers.0.mlp.dense_4h_to_h.bias.bin
    ├── [ 50M]  model.layers.0.mlp.dense_4h_to_h.weight.0.bin
    ├── [ 20K]  model.layers.0.mlp.dense_h_to_4h.bias.0.bin
    ├── [ 50M]  model.layers.0.mlp.dense_h_to_4h.weight.0.bin
    ├── [5.0K]  model.layers.0.post_attention_layernorm.bias.bin
    ├── [5.0K]  model.layers.0.post_attention_layernorm.weight.bin
    ├── [5.0K]  model.layers.10.attention.dense.bias.bin
    ├── [ 12M]  model.layers.10.attention.dense.weight.0.bin
    ├── [ 15K]  model.layers.10.attention.query_key_value.bias.0.bin
    ├── [   4]  model.layers.10.attention.query_key_value.scale_y_quant_orig.bin
    ├── [ 38M]  model.layers.10.attention.query_key_value.weight.0.bin
    ├── [5.0K]  model.layers.10.input_layernorm.bias.bin
    ├── [5.0K]  model.layers.10.input_layernorm.weight.bin
    ├── [5.0K]  model.layers.10.mlp.dense_4h_to_h.bias.bin
    ├── [ 50M]  model.layers.10.mlp.dense_4h_to_h.weight.0.bin
    ├── [ 20K]  model.layers.10.mlp.dense_h_to_4h.bias.0.bin
    ├── [ 50M]  model.layers.10.mlp.dense_h_to_4h.weight.0.bin
    ├── [5.0K]  model.layers.10.post_attention_layernorm.bias.bin
    ├── [5.0K]  model.layers.10.post_attention_layernorm.weight.bin
...
    ├── [5.0K]  model.word_embeddings_layernorm.bias.bin
    ├── [5.0K]  model.word_embeddings_layernorm.weight.bin
    └── [1.2G]  model.wpe.bin
```

组合仅 INT8 权重量化及 INT8 KV CACHE 量化构建引擎：

```
# Build model with both INT8 weight-only and INT8 KV cache enabled

python build.py --bin_model_dir=/workspace/model/bloom-c-model/int8_kv_cache/3b/1-gpu
                --dtype float16 \
                --use_gpt_attention_plugin float16 \
                --use_gemm_plugin float16 \
                --use_layernorm_plugin \
                --int8_kv_cache \
                --output_dir /workspace/model/bloom-3b-c-model/int8_kv_cache/ \
                --use_weight_only
```

运行结果：

```
tree -h /workspace/model/bloom-3b-c-model/int8_kv_cache/
/workspace/model/bloom-3b-c-model/int8_kv_cache/
├── [4.6G]  bloom_float16_tp1_rank0.engine
├── [1.2K]  config.json
```

```
0 directories, 3 files
```

SmoothQuant 量化（W8A8）

与 FP16 构建引擎处理 HF 权重并直接加载到 TensorRT-LLM 不同，SmoothQuant 需要加载 INT8 权重，该权重应在构建引擎之前进行预处理。

```
python3 hf_bloom_convert.py \
-i /workspace/model/bloomz-3b \
-o /workspace/model/bloom-3b-c-model/smooth/ \
--smoothquant 0.5 \
--tensor-parallelism 1 \
--storage-type float16
```

运行结果：

```
> tree -h /workspace/model/bloom-3b-c-model/smooth/
/workspace/model/bloom-3b-c-model/smooth/
└── [100K]  1-gpu
    ├── [2.1K]  config.ini
    ├── [5.0K]  model.final_layernorm.bias.bin
    ├── [5.0K]  model.final_layernorm.weight.bin
    ├── [5.0K]  model.layers.0.attention.dense.bias.bin
    ├── [   4]  model.layers.0.attention.dense.scale_w_quant_orig.bin
    ├── [ 10K]  model.layers.0.attention.dense.scale_w_quant_orig.col.bin
    ├── [   4]  model.layers.0.attention.dense.scale_x_orig_quant.bin
    ├── [   4]  model.layers.0.attention.dense.scale_y_accum_quant.bin
    ├── [ 10K]  model.layers.0.attention.dense.scale_y_accum_quant.col.bin
    ├── [   4]  model.layers.0.attention.dense.scale_y_quant_orig.bin
    ├── [ 10K]  model.layers.0.attention.dense.smoother.0.bin
    ├── [ 12M]  model.layers.0.attention.dense.weight.0.bin
    ├── [6.2M]  model.layers.0.attention.dense.weight.int8.0.bin
    ├── [6.2M]  model.layers.0.attention.dense.weight.int8.col.0.bin
    ├── [ 15K]  model.layers.0.attention.query_key_value.bias.0.bin
    ├── [ 30K]  model.layers.0.attention.query_key_value.scale_w_quant_orig.bin
    ├── [ 30K]  model.layers.0.attention.query_key_value.scale_w_quant_orig.col.0.bin
    ├── [   4]  model.layers.0.attention.query_key_value.scale_x_orig_quant.bin
    ├── [ 30K]  model.layers.0.attention.query_key_value.scale_y_accum_quant.bin
    ├── [ 30K]  model.layers.0.attention.query_key_value.scale_y_accum_quant.col.0.bin
    ├── [   4]  model.layers.0.attention.query_key_value.scale_y_quant_orig.bin
    ├── [ 38M]  model.layers.0.attention.query_key_value.weight.0.bin
    ├── [ 19M]  model.layers.0.attention.query_key_value.weight.int8.0.bin
    ├── [ 19M]  model.layers.0.attention.query_key_value.weight.int8.col.0.bin
    ├── [5.0K]  model.layers.0.input_layernorm.bias.bin
    ├── [5.0K]  model.layers.0.input_layernorm.weight.bin
    ├── [5.0K]  model.layers.0.mlp.dense_4h_to_h.bias.bin
    ├── [   4]  model.layers.0.mlp.dense_4h_to_h.scale_w_quant_orig.bin
    ├── [ 10K]  model.layers.0.mlp.dense_4h_to_h.scale_w_quant_orig.col.bin
    ├── [   4]  model.layers.0.mlp.dense_4h_to_h.scale_x_orig_quant.bin
    ├── [   4]  model.layers.0.mlp.dense_4h_to_h.scale_y_accum_quant.bin
    ├── [ 10K]  model.layers.0.mlp.dense_4h_to_h.scale_y_accum_quant.col.bin
    ├── [   4]  model.layers.0.mlp.dense_4h_to_h.scale_y_quant_orig.bin
    ├── [ 40K]  model.layers.0.mlp.dense_4h_to_h.smoother.0.bin
    ├── [ 50M]  model.layers.0.mlp.dense_4h_to_h.weight.0.bin
    ├── [ 25M]  model.layers.0.mlp.dense_4h_to_h.weight.int8.0.bin
    ├── [ 25M]  model.layers.0.mlp.dense_4h_to_h.weight.int8.col.0.bin
    ├── [ 20K]  model.layers.0.mlp.dense_h_to_4h.bias.0.bin
    ├── [   4]  model.layers.0.mlp.dense_h_to_4h.scale_w_quant_orig.bin
    ├── [ 40K]  model.layers.0.mlp.dense_h_to_4h.scale_w_quant_orig.col.0.bin
    ├── [   4]  model.layers.0.mlp.dense_h_to_4h.scale_x_orig_quant.bin
    ├── [   4]  model.layers.0.mlp.dense_h_to_4h.scale_y_accum_quant.bin
    ├── [ 40K]  model.layers.0.mlp.dense_h_to_4h.scale_y_accum_quant.col.0.bin
    ├── [   4]  model.layers.0.mlp.dense_h_to_4h.scale_y_quant_orig.bin
```

```
    ├── [ 25M]  model.layers.0.mlp.dense_h_to_4h.weight.int8.col.0.bin
    ├── [5.0K]  model.layers.0.post_attention_layernorm.bias.bin
    ├── [5.0K]  model.layers.0.post_attention_layernorm.weight.bin
...
    ├── [5.0K]  model.word_embeddings_layernorm.bias.bin
    ├── [5.0K]  model.word_embeddings_layernorm.weight.bin
    └── [1.2G]  model.wpe.bin
```

通过 `--use_smooth_quant` 选型启动 INT8 量化。默认情况下，使用逐层量化（ `_per_tensor_` ）构建引擎：

```
# Build model for SmoothQuant in the _per_tensor_ mode.
python3 build.py --bin_model_dir=/workspace/model/bloom-3b-c-model/smooth/1-gpu \
                 --use_smooth_quant \
                 --output_dir "/workspace/model/bloom-3b-c-model/smooth-quant" \
                 --use_gpt_attention_plugin float16
```

运行结果：

```
> tree -h /workspace/model/bloom-3b-c-model/smooth-quant
/workspace/model/bloom-3b-c-model/smooth-quant
├── [3.4G]  bloom_float16_tp1_rank0.engine
├── [1.2K]  config.json
└── [516K]  model.cache

0 directories, 3 files
```

同时，支持使用逐通道量化（ `_per_token_` + `_per_channel_` ）构建引擎：

```
# Build model for SmoothQuant in the _per_token_ + _per_channel_ mode
python3 build.py --bin_model_dir=/workspace/model/bloom-3b-c-model/smooth/1-gpu \
                 --use_smooth_quant \
                 --use_gpt_attention_plugin float16 \
                 --output_dir "/workspace/model/bloom-3b-c-model/smooth-quant-channel-
                 --per_token \
                 --per_channel
```

运行结果：

```
tree -h /home/guodong.li/workspace/model/bloom-3b-c-model/smooth-quant-channel-token
/home/guodong.li/workspace/model/bloom-3b-c-model/smooth-quant-channel-token
├── [4.6G]  bloom_float16_tp1_rank0.engine
├── [1.2K]  config.json
└── [516K]  model.cache

0 directories, 3 files
```

注意：

- 目前需要为 SmoothQuant 启用 GPT 注意力插件（ `--use_gpt_attention_plugin` ）。
- 使用 `--bin_model_dir` 而不是 `--model_dir` ，是因为 SmoothQuant 量化时，模型需要二进制文件中的 INT8 权重和各种缩放（scales）。

模型推理

summarize.py 脚本常用参数说明：

- hf_model_location：指定HF模型和词表地址
- test_hf：测试HF
- test_trt_llm：测试TensorRT-LLM
- data_type：指定数据类型，该参数指定test_hf时使用，将模型参数转换成半精度
- dataset_path：指定数据集缓存目录
- engine_dir：指定引擎目录

FP16

```
python summarize.py --test_trt_llm \
                    --hf_model_location /workspace/model/bloomz-3b \
                    --data_type fp16 \
                    --engine_dir /workspace/model/bloomz-3b_trt_engines/fp16/1-gpu/
```

仅 INT8 权重量化

```
python summarize.py --test_trt_llm \
                    --hf_model_location /workspace/model/bloomz-3b \
                    --data_type fp16 \
                    --engine_dir /workspace/model/bloomz-3b_trt_engines/int8_weight_on
```

运行过程：

```
[11/14/2023-09:54:48] [TRT-LLM] [I] Load tokenizer takes: 0.6626021862030029 sec
[11/14/2023-09:54:54] [TRT] [I] Loaded engine size: 4708 MiB
[11/14/2023-09:54:55] [TRT] [I] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +8,
[11/14/2023-09:54:55] [TRT] [I] [MemUsageChange] Init cuDNN: CPU +2, GPU +10, now: CPU
[11/14/2023-09:54:55] [TRT] [W] TensorRT was linked against cuDNN 8.9.4 but loaded cuD
[11/14/2023-09:54:55] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in engine
[11/14/2023-09:54:55] [TRT] [I] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +8,
[11/14/2023-09:54:55] [TRT] [I] [MemUsageChange] Init cuDNN: CPU +0, GPU +8, now: CPU
[11/14/2023-09:54:55] [TRT] [W] TensorRT was linked against cuDNN 8.9.4 but loaded cuD
[11/14/2023-09:54:56] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in IExecu
[11/14/2023-09:54:56] [TRT] [I] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +8,
[11/14/2023-09:54:56] [TRT] [I] [MemUsageChange] Init cuDNN: CPU +1, GPU +10, now: CPU
[11/14/2023-09:54:56] [TRT] [W] TensorRT was linked against cuDNN 8.9.4 but loaded cuD
[11/14/2023-09:54:57] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in IExecu
[11/14/2023-09:54:58] [TRT-LLM] [I] Load engine takes: 9.880424976348877 sec
/workspace/TensorRT-LLM/examples/bloom/summarize.py:165: UserWarning: To copy construc
  [torch.tensor(line_encoded[i], dtype=torch.int32), pad],
[11/14/2023-09:54:59] [TRT-LLM] [I] ----------------------------------------------------
[11/14/2023-09:54:59] [TRT-LLM] [I] TensorRT-LLM Generated :
[11/14/2023-09:54:59] [TRT-LLM] [I]  Article : ['(CNN)James Best, best known for his p
[11/14/2023-09:54:59] [TRT-LLM] [I]
 Highlights : ['James Best, who played the sheriff on "The Dukes of Hazzard," died Mon
[11/14/2023-09:54:59] [TRT-LLM] [I]
 Summary : [[' Actor James Best, best known for his role as bumbling sheriff Rosco P.
[11/14/2023-09:54:59] [TRT-LLM] [I] ----------------------------------------------------
/workspace/TensorRT-LLM/examples/bloom/summarize.py:165: UserWarning: To copy construc
  [torch.tensor(line_encoded[i], dtype=torch.int32), pad],
[11/14/2023-09:55:10] [TRT-LLM] [I] TensorRT-LLM (total latency: 10.436434745788574 se
[11/14/2023-09:55:10] [TRT-LLM] [I] TensorRT-LLM beam 0 result
[11/14/2023-09:55:11] [TRT-LLM] [I]   rouge1 : 30.60846842935061
[11/14/2023-09:55:11] [TRT-LLM] [I]   rouge2 : 11.315593160478784
[11/14/2023-09:55:11] [TRT-LLM] [I]   rougeL : 24.043680494718327
[11/14/2023-09:55:11] [TRT-LLM] [I]   rougeLsum : 26.250663629946125
```

```
mpirun -n 2 --allow-run-as-root \
    python summarize.py --test_trt_llm \
                        --hf_model_location /workspace/model/bloomz-3b \
                        --data_type fp16 \
                        --engine_dir /workspace/model/bloomz-3b_trt_engines/fp16/2-gpu
```

运行过程：

```
[11/14/2023-09:58:13] [TRT-LLM] [MPI_Rank 1] [I] Load tokenizer takes: 0.4274311065673
[11/14/2023-09:58:13] [TRT-LLM] [MPI_Rank 0] [I] Load tokenizer takes: 0.4551923274993
[11/14/2023-09:58:17] [TRT] [I] Loaded engine size: 4094 MiB
[11/14/2023-09:58:18] [TRT] [I] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +8,
[11/14/2023-09:58:18] [TRT] [I] [MemUsageChange] Init cuDNN: CPU +1, GPU +10, now: CPU
[11/14/2023-09:58:18] [TRT] [W] TensorRT was linked against cuDNN 8.9.4 but loaded cuD
[11/14/2023-09:58:19] [TRT] [I] Loaded engine size: 4094 MiB
[11/14/2023-09:58:20] [TRT] [I] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +8,
[11/14/2023-09:58:20] [TRT] [I] [MemUsageChange] Init cuDNN: CPU +1, GPU +10, now: CPU
[11/14/2023-09:58:20] [TRT] [W] TensorRT was linked against cuDNN 8.9.4 but loaded cuD
[11/14/2023-09:58:23] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in engine
[11/14/2023-09:58:23] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in engine
[11/14/2023-09:58:23] [TRT] [I] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +8,
[11/14/2023-09:58:23] [TRT] [I] [MemUsageChange] Init cuDNN: CPU +0, GPU +8, now: CPU
[11/14/2023-09:58:23] [TRT] [W] TensorRT was linked against cuDNN 8.9.4 but loaded cuD
[11/14/2023-09:58:23] [TRT] [I] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +8,
[11/14/2023-09:58:23] [TRT] [I] [MemUsageChange] Init cuDNN: CPU +0, GPU +8, now: CPU
[11/14/2023-09:58:23] [TRT] [W] TensorRT was linked against cuDNN 8.9.4 but loaded cuD
[11/14/2023-09:58:23] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in IExecu
[11/14/2023-09:58:23] [TRT] [I] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +1, GPU +8,
[11/14/2023-09:58:23] [TRT] [I] [MemUsageChange] Init cuDNN: CPU +0, GPU +10, now: CPU
[11/14/2023-09:58:23] [TRT] [W] TensorRT was linked against cuDNN 8.9.4 but loaded cuD
[11/14/2023-09:58:24] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in IExecu
[11/14/2023-09:58:24] [TRT] [I] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +8,
[11/14/2023-09:58:24] [TRT] [I] [MemUsageChange] Init cuDNN: CPU +0, GPU +10, now: CPU
[11/14/2023-09:58:24] [TRT] [W] TensorRT was linked against cuDNN 8.9.4 but loaded cuD
[11/14/2023-09:58:24] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in IExecu
[11/14/2023-09:58:24] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in IExecu
[11/14/2023-09:58:25] [TRT-LLM] [MPI_Rank 0] [I] Load engine takes: 11.81023645401001
[11/14/2023-09:58:25] [TRT-LLM] [MPI_Rank 1] [I] Load engine takes: 11.762826204299927
/workspace/TensorRT-LLM/examples/bloom/summarize.py:165: UserWarning: To copy construc
  [torch.tensor(line_encoded[i], dtype=torch.int32), pad],
/workspace/TensorRT-LLM/examples/bloom/summarize.py:165: UserWarning: To copy construc
  [torch.tensor(line_encoded[i], dtype=torch.int32), pad],
[11/14/2023-09:58:27] [TRT-LLM] [MPI_Rank 0] [I] ------------------------------------
[11/14/2023-09:58:27] [TRT-LLM] [MPI_Rank 0] [I] TensorRT-LLM Generated :
[11/14/2023-09:58:27] [TRT-LLM] [MPI_Rank 0] [I]  Article : ['(CNN)James Best, best kn
[11/14/2023-09:58:27] [TRT-LLM] [MPI_Rank 0] [I]
 Highlights : ['James Best, who played the sheriff on "The Dukes of Hazzard," died Mon
[11/14/2023-09:58:27] [TRT-LLM] [MPI_Rank 0] [I]
 Summary : [[' Actor James Best, best known for his role as bumbling sheriff Rosco P.
[11/14/2023-09:58:27] [TRT-LLM] [MPI_Rank 0] [I] ------------------------------------
/workspace/TensorRT-LLM/examples/bloom/summarize.py:165: UserWarning: To copy construc
  [torch.tensor(line_encoded[i], dtype=torch.int32), pad],
/workspace/TensorRT-LLM/examples/bloom/summarize.py:165: UserWarning: To copy construc
  [torch.tensor(line_encoded[i], dtype=torch.int32), pad],
[11/14/2023-09:58:42] [TRT-LLM] [MPI_Rank 0] [I] TensorRT-LLM (total latency: 14.92856
[11/14/2023-09:58:42] [TRT-LLM] [MPI_Rank 0] [I] TensorRT-LLM beam 0 result
[11/14/2023-09:58:43] [TRT-LLM] [MPI_Rank 0] [I]   rouge1 : 27.12991734291884
[11/14/2023-09:58:43] [TRT-LLM] [MPI_Rank 0] [I]   rouge2 : 8.273487794146279
[11/14/2023-09:58:43] [TRT-LLM] [MPI_Rank 0] [I]   rougeL : 21.08356714989421
[11/14/2023-09:58:43] [TRT-LLM] [MPI_Rank 0] [I]   rougeLsum : 23.51165220383353
```

逐层量化：

```
python summarize.py --test_trt_llm \
                    --hf_model_location /workspace/model/bloomz-3b \
                    --data_type fp16 \
                    --engine_dir /workspace/model/bloom-3b-c-model/smooth-quant
```

逐通道量化：

```
python summarize.py --test_trt_llm \
                    --hf_model_location /workspace/model/bloomz-3b \
                    --data_type fp16 \
                    --engine_dir /workspace/model/bloom-3b-c-model/smooth-quant-channe
```

总结

本文简要介绍了TensorRT-LLM环境搭建，同时，基于Bloom进行模型量化及推理。码字不易，如果觉得有帮助，欢迎点赞收藏加关注。

参考文档

- github.com/NVIDIA/Tenso...
- github.com/NVIDIA/Tenso...
- github.com/NVIDIA/Tenso...

发布于 2023-11-18 21:41 · IP 属地四川

TensorRT 大模型推理 大模型

知乎 首发于 **动手学大模型**

07-09 · 上海 回复 喜欢

提伯斯xxxH
大佬，对cuda版本有要求吗？比如我这个是12.0的,其他步骤都和您一样，但是就是会报错：AttributeError: module 'tensorrt' has no attribute 'int64'
2023-12-05 · 北京 回复 喜欢

Devin
大佬，jetson目前能跑起来吗
2023-11-25 · 广东 回复 喜欢

理性发言，友善互动

动手学大模型

推荐阅读

CUDA上的量化深度学习模型的自动化优化
CUDA上的量化深度学习模型的自动化优化 深度学习已成功应用于各种任务。在诸如自动驾驶汽车推理之类的实时场景中，模型的推理速度至关重要。网络量化是加速深度学习模型的有效方法。在量化...
吴建明wujianming

$$K, V) = \text{softmax}$$

LLMs量化系列|LLMs Quantization Need What ?
回溯的猫

AI 模型量化
量化是在模型部署阶段非常重要的步骤，主要的目的是为了节省模型inference的时间，简单来说就是将原来4字节的float类型的数据通过预先定义好的缩放规则(可以是均匀缩放，也可以是非均匀缩放...
Garfield

关于CNN 的量化训练
CNN模型的量化研究应该是
的重点，确实能够在保证精度
提下带来明显的性能提升，
一些dsp上，能够加速7～10
年2019，量化真正能大量投
的时机已经到来。目前，用
magic