知乎

基于TensorRT-LLM的大模型部署(速通笔记)

nghuyong
略懂些NLP | 非典型程序员

关注他

36 人赞同了该文章

- 【更新 2023.12.18】踩坑最新v0.6.1版本+W8A8量化。
  - 基于W8A8量化可以基本不损失效果，同时大幅提升推理速度（**x1.5+**），同时**降低一半显存**

TensorRT-LLM是NVIDIA官方的大模型部署方案。本文是一个初步踩坑后的笔记总结。

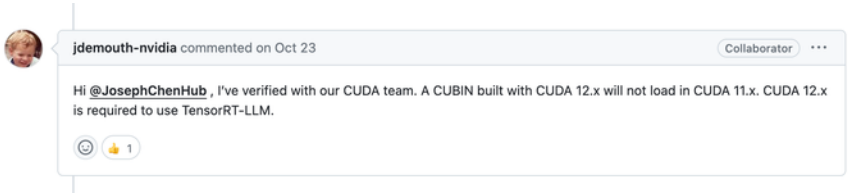
特性

TensorRT-LLM包括以下一系列的特性，是当前大模型部署必备神器:

- 模型转换: 提供了常见大模型示例，包括LLaMA、ChatGLM，Baichuan，Bloom
- 模型量化: 支持多种量化方案，包括**W8A8 (vLLM, lmdeploy等均不支持W8A8量化)**
- 算子优化: 对大模型中主要算子进行了优化，包括Attention
- 多机多卡: 支持多机多卡部署超大规模的模型
- 连续批处理: 即vLLM中的continuous batching，可大幅提高模型吞吐
- 支持Triton: 官方提供了Triton的backend，方便快速接入Triton框架

环境准备

确保CUDA版本为**12.x**，否则先进行升级



建议自己build tritonserver+TensorRT-LLM最新的镜像（官方的镜像还没有v0.6.1版本）

下面基于v0.6.1版本进行编译，参考这里 github.com/triton-infer...

```
git clone -b v0.6.1 https://github.com/triton-inference-server/tensorrtllm_backend.git
cd tensorrtllm_backend
git lfs install
git submodule update --init --recursive
DOCKER_BUILDKIT=1 docker build -t triton_trt_llm -f dockerfile/Dockerfile.trt_llm_back
```

build完毕后，进入容器

```
docker run --gpus all --shm-size=1g triton_trt_llm:latest bash
```

容器内环境包括: v0.6.1版本的tensorrt-llm 以及 tritonserver

```
# tensorrt-llm已经安装
root@5b7ed30b15c5:/app/tensorrt_llm# pip list | grep tensorrt-llm
tensorrt-llm             0.6.1             /app/tensorrt_llm

# tritonserver已经安装
root@5b7ed30b15c5:/opt/tritonserver/bin# ls
tritonserver
```

知乎

```
3rdparty  README.md   build  docker  examples                      requirements-dev.t
LICENSE   benchmarks  cpp    docs    requirements-dev-windows.txt  requirements-window
```

模型(FP16)转换

这里以Baichuan7B-V1-Base为例

```
# 进入Baichuan的例子文件夹
cd tensorrt_llm/examples/baichuan

# 转成Tensorrt engine, 提前下载好 Baichuan-7B
python build.py --model_version v1_7b \
                --model_dir Baichuan-7B \
                --dtype float16 \
                --use_gpt_attention_plugin float16 \
                --use_inflight_batching \
                --paged_kv_cache \
                --max_input_len 32 \
                --max_output_len 32 \
                --max_beam_width 1 \
                --tokens_per_block 32 \
                --output_dir trt_engines/baichuan/
```

```
[12/01/2023-06:16:54] [TRT-LLM] [I] Total time of building baichuan_float16_tp1_rank0.engine: 00:00:26
[12/01/2023-06:16:54] [TRT-LLM] [I] Config saved to ./trt_engines/baichuan/config.json.
[12/01/2023-06:16:54] [TRT-LLM] [I] Serializing engine to ./trt_engines/baichuan/baichuan_float16_tp1_rank0.engine...
[12/01/2023-06:17:46] [TRT-LLM] [I] Engine serialized. Total time: 00:00:52
[12/01/2023-06:17:47] [TRT-LLM] [I] Timing cache serialized to ./trt_engines/baichuan/model.cache
[12/01/2023-06:17:47] [TRT-LLM] [I] Total time of building all 1 engines: 00:01:33
```

模型(W8A8)转换

参考: github.com/NVIDIA/Tenso...

```
# 模型量化
python3 hf_baichuan_convert.py \
-i Baichuan-7B \
-o Baichuan-7B-W8A8 \
-sq 0.8 \
--tensor-parallelism 1 \
--storage-type fp16

# 转成Tensorrt engine
python3 build.py \
--model_version v1_7b \
--bin_model_dir=Baichuan-7B-W8A8 \
--use_smooth_quant \
--use_gpt_attention_plugin float16 \
--per_token \
--per_channel \
--output_dir trt_engines/baichuan_W8A8/
```

为了提高量化后的模型效果，可以进行下面一些操作

- 标定数据集默认是cnn_dailymail，可替换成自己的数据集

修改: github.com/NVIDIA/Tenso...

```
# 删除并修改成加载自己数据集
from datasets import load_dataset
dataset_cnn = load_dataset("ccdv/cnn_dailymail", '3.0.0')
```

知乎

修改: github.com/NVIDIA/Tenso...

```
# 设置成全部训练数据
num_samples = len(train_data)
```

- 标定数据默认进行了长度截断，可取消

修改: github.com/NVIDIA/Tenso...

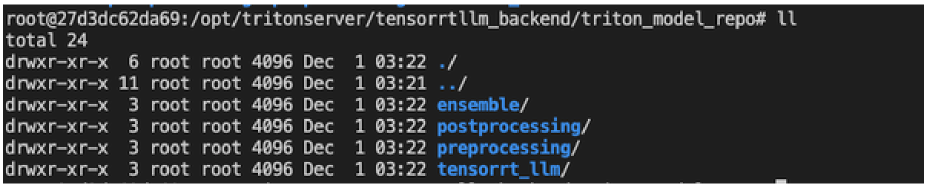
```
# 删除
line_encoded = line_encoded[:, -test_token_num:]
```

模型部署

结合triton进行模型的部署，首先准备相关文件

```
cd /path/to/tensorrtllm_backend
mkdir triton_model_repo
cp -r all_models/inflight_batcher_llm/* triton_model_repo/
cp tensorrt_llm/examples/baichuan/trt_engines/baichuan/* triton_model_repo/tensorrt_ll
cd triton_model_repo/
```

现在 `triton_model_repo` 目录的结构，如下所示:



```
root@27d3dc62da69:/opt/tritonserver/tensorrtllm_backend/triton_model_repo# ll
total 24
drwxr-xr-x  6 root root 4096 Dec  1 03:22 ./
drwxr-xr-x 11 root root 4096 Dec  1 03:21 ../
drwxr-xr-x  3 root root 4096 Dec  1 03:22 ensemble/
drwxr-xr-x  3 root root 4096 Dec  1 03:22 postprocessing/
drwxr-xr-x  3 root root 4096 Dec  1 03:22 preprocessing/
drwxr-xr-x  3 root root 4096 Dec  1 03:22 tensorrt_llm/
```

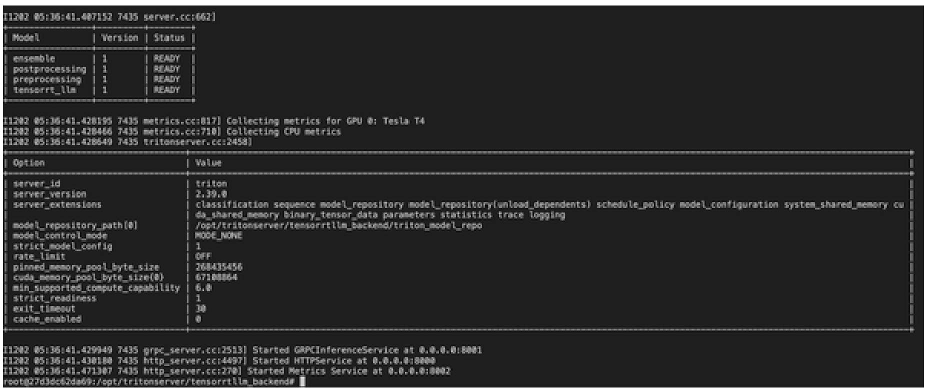
- preprocessing: 用于encode，tokenizer将输入的prompt转换成intpu_id
- tensorrt_llm: 模型推理
- postprocess: 用于decode，tokenizer将输出的token id转换成token
- ensemble: 用来串联以上三个模型: preprocessing->tensorrt_llm->postprocess

修改preprocessing, tensorrt_llm, postprocess中的config.pbtxt

可参考: github.com/triton-infer...

最后启动triton服务

```
python3 scripts/launch_triton_server.py \
--model_repo=/tensorrtllm_backend/triton_model_repo
```



```
I1202 05:36:41.407152 7435 server.cc:662]
| Model          | Version | Status |
| ensemble       | 1       | READY  |
| postprocessing | 1       | READY  |
| preprocessing  | 1       | READY  |
| tensorrt_llm   | 1       | READY  |

I1202 05:36:41.428195 7435 metrics.cc:817] Collecting metrics for GPU 0: Tesla T4
I1202 05:36:41.428466 7435 metrics.cc:710] Collecting CPU metrics
I1202 05:36:41.428649 7435 tritonserver.cc:2458]
| Option                           | Value
| server_id                        | triton
| server_version                   | 2.39.0
| server_extensions                | classification sequence model_repository model_repository(unload_dependents) schedule_policy model_configuration system_shared_memory cu
|                                  | da_shared_memory binary_tensor_data parameters statistics trace logging
| model_repository_path[0]         | /opt/tritonserver/tensorrtllm_backend/triton_model_repo
| model_control_mode               | MODE_NONE
| strict_model_config              | 1
| rate_limit                       | OFF
| pinned_memory_pool_byte_size     | 268435456
| cuda_memory_pool_byte_size{0}    | 67108864
| min_supported_compute_capability | 6.0
| strict_readiness                 | 1
| exit_timeout                     | 30
| cache_enabled                    | 0

I1202 05:36:41.429949 7435 grpc_server.cc:2513] Started GRPCInferenceService at 0.0.0.0:8001
I1202 05:36:41.430180 7435 http_server.cc:4497] Started HTTPService at 0.0.0.0:8000
I1202 05:36:41.471307 7435 http_server.cc:270] Started Metrics Service at 0.0.0.0:8002
root@27d3dc62da69:/opt/tritonserver/tensorrtllm_backend#
```

知乎

直接用curl请求ensemble的HTTP接口进行测试:

```
curl -X POST localhost:8000/v2/models/ensemble/generate \
-d '{"text_input": "北京是", "max_tokens": 20, "bad_words": "", "stop_words": ""}'
```

返回如下json格式的结果

```
{
    "model_name":"ensemble",
    "model_version":"1",
    "sequence_end":false,
    "sequence_id":0,
    "sequence_start":false,
    "text_output":"北京是我国的首都,也是我国的政治中心,北京的气候类型是( )\n解答: "
}
```

参考

- github.com/triton-infer...
- github.com/NVIDIA/Tenso...

编辑于 2023-12-18 18:13 · IP 属地北京

大模型 TensorRT LLM

理性发言，友善互动

19 条评论 默认 最新

chernzy
config.pbtxt有没有现成的参考
02-27 · 广东 回复 喜欢

nghuyong 作者
官方repo里面就有～
02-29 · 北京 回复 喜欢

知乎用户pbpcg2
原来是先进行docker run --gpus all --shm-size=1g triton_trt_llm:latest bash 操作启动容器，然后再继续部署模型吗？GitHub的没看懂😳😳
02-05 · 北京 回复 喜欢

nghuyong 作者
对的，先build好tensorrt的环境
02-06 · 江苏 回复 1

行者说
请教一下，怎么样才能使返回的答案更智能一些？根据设置的max_tokens输出，出现两个问题：答案很长时，会被截断；答案过短时，会重复。谢谢，期待楼主大哥的解答。
2023-12-28 · 广东 回复 喜欢

nghuyong 作者
max tokens 可以设置足够大，比如达到模型最大长度，因为是 continuous batching 也不会拖慢整体速度
2023-12-28 · 北京 回复 喜欢

残败虽傲
你好，我采用你得curl请求ensemble的HTTP接口进行测试: 返回报错：

知乎

2023-12-18 · 广东 回复 喜欢

残败虽傲 ▸ **nghuyong**
tensorrtllm_backend 如果下载最新的，模型这些应该不一样了吧 我看到有tensorrt_llm_bls这个，博主这块更新没有实测把
2023-12-19 · 广东 回复 喜欢

nghuyong 作者 ▸ **残败虽傲**
改成从本地加载就行，可以看看datasets库的API
2023-12-18 · 北京 回复 喜欢

展开其他 1 条回复 》

天涯
目前这个最新版本的镜像nvcr.io/nvidia/tritonse...下载了好久，网速太慢了，简单配置了http代理没起作用，有啥好的推荐方法不
2023-12-07 · 中国台湾 回复 喜欢

nghuyong 作者
文章更新了，可以直接自己build镜像
2023-12-18 · 北京 回复 喜欢

没救了没救了
请问目前支持baichuan2-13b-chat吗?
2023-12-05 · 北京 回复 喜欢

没救了没救了 ▸ **nghuyong**
谢谢您!
2023-12-06 · 北京 回复 喜欢

nghuyong 作者
支持的
2023-12-05 · 北京 回复 喜欢

霁月
会存在一些解码容易乱码的情况吗
2023-12-02 · 广东 回复 喜欢

nghuyong 作者
tokenzier正确就不会吧
2023-12-02 · 北京 回复 喜欢

Ethan Yan
有和vllm吞吐量上的对比吗
2023-12-02 · 河北 回复 喜欢

nghuyong 作者
官方没出，有一些民间的
github.com/NVIDIA/Tenso...
github.com/NVIDIA/Tenso...

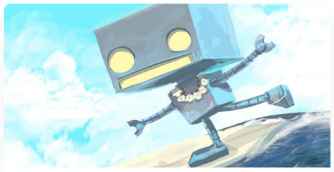
理性发言，友善互动

推荐阅读

万字长文解读NVIDIA TensorRT-LLM部署LoRA...

长文详解--LLM高效预训练(一)

【推荐文章】 MoE： MoE模型的前世今生 DeepSeek-V2和MLA 昆仑万维-SkyworkMoE 成本10w刀的JetMoE MoE的top-p routing 对MoE模型的一些观察 从dense到MoE -- sparse upcycling MoE...



LLM 系列超详细解读 (九)：MobileLLM：优化 1B 参数...

关于LLM RAG的思考

RAG和Agent是两个被寄予厚
LLM落地路径。相比Agent，
阶段RAG更具可行性，并已经
通用的实现方案。 目标/动机
望RAG能够带来如下几个方面
升：解决幻觉、私域知识、知

知乎