大模型的好伙伴，浅析推理加速引擎FasterTransformer

吃果冻不吐果冻皮 关注他

最近几个月，随着ChatGPT的现象级表现，大模型如雨后春笋般涌现。而模型推理⁺是抽象的算法模型触达具体的实际业务的最后一公里。

但是在这个环节中，仍然还有很多已经是大家共识的痛点和诉求，比如：

- 任何线上产品的用户体验都与服务的响应时长成反比，复杂的模型如何极致地压缩请求时延？
- 模型推理通常是资源常驻型服务，如何通过提升服务单机性能从而增加QPS，同时大幅降低资源成本？
- 端-边-云是现在模型服务发展的必然趋势，如何让离线训练的模型“瘦身塑形”从而在更多设备上快速部署使用？

因此，模型推理的加速优化成为了AI界的重要研究领域。

本文给大家分享大模型推理加速引擎FasterTransformer的基本使用。

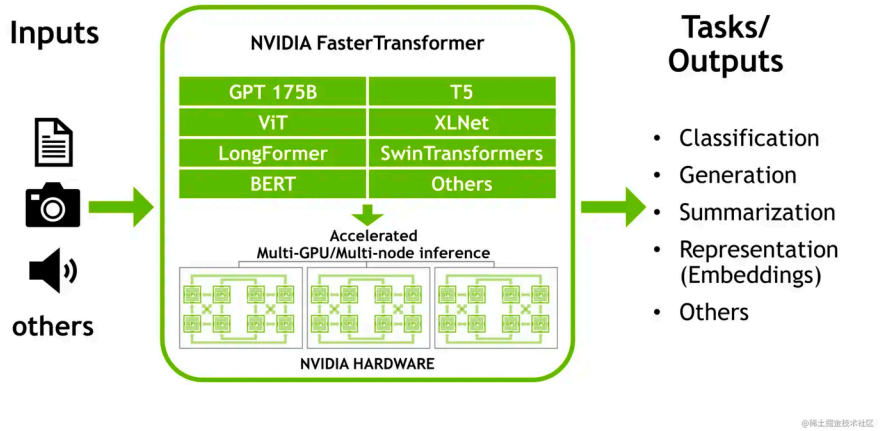


image.png

FasterTransformer简介

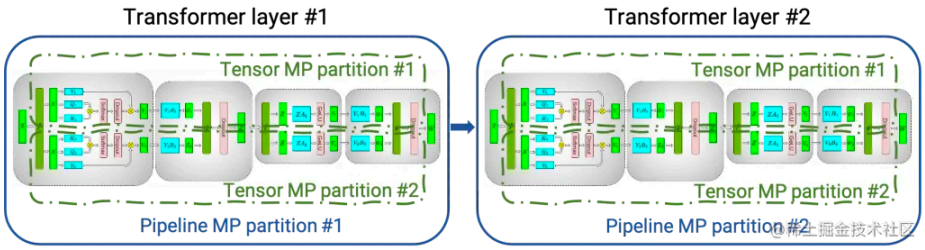
NVIDIA FasterTransformer (FT) 是一个用于实现基于Transformer的神经网络推理的加速引擎。 它包含Transformer块的高度优化版本的实现，其中包含编码器和解码器部分。使用此模块，您可以运行编码器-解码器架构模型（如：T5）、仅编码器架构模型（如：BERT）和仅解码器架构模型（如： GPT）的推理。

FT框架是用C++/CUDA编写的，依赖于高度优化的 cuBLAS、cuBLASLt 和 cuSPARSELt 库，这使您可以在 GPU 上进行快速的 Transformer 推理。

与NVIDIA TensorRT等其他编译器相比，FT 的最大特点是它支持以分布式方式进行 Transformer 大模型推理⁺。

下图显示了如何使用张量并行 (TP) 和流水线并行 (PP) 技术将基于Transformer架构的神经网络拆分到多个 GPU 和节点上。

- 当每个张量被分成多个块时，就会发生张量并行，并且张量的每个块都可以放置在单独的 GPU 上。在计算过程中，每个块在不同的 GPU 上单独并行处理；最后，可以通过组合来自多个 GPU 的结果来计算最终张量。
- 当模型被



在底层，节点间或节点内通信依赖于 MPI 、 NVIDIA NCCL、Gloo等。因此，使用 FasterTransformer，您可以在多个 GPU 上以张量并行运行大型Transformer，以减少计算延迟。同时，TP 和 PP 可以结合在一起，在多 GPU 节点环境中运行具有数十亿、数万亿个参数的大型 Transformer 模型。

除了使用 C ++ 作为后端部署，FasterTransformer 还集成了 TensorFlow（使用 TensorFlow op）、PyTorch （使用 Pytorch op）和 Triton作为后端框架进行部署。当前，TensorFlow op 仅支持单 GPU，而 PyTorch op 和 Triton 后端都支持多 GPU 和多节点。

目前，FT 支持了 Megatron-LM GPT-3、GPT-J、BERT、ViT、Swin Transformer、Longformer、T5 和 XLNet 等模型。您可以在 GitHub 上的 **FasterTransformer**库中查看最新的支持矩阵。

FT 适用于计算能力 >= 7.0 的 GPU，例如 V100、A10、A100 等。

下图展示了 GPT-J 6B 参数的模型推断加速比较：

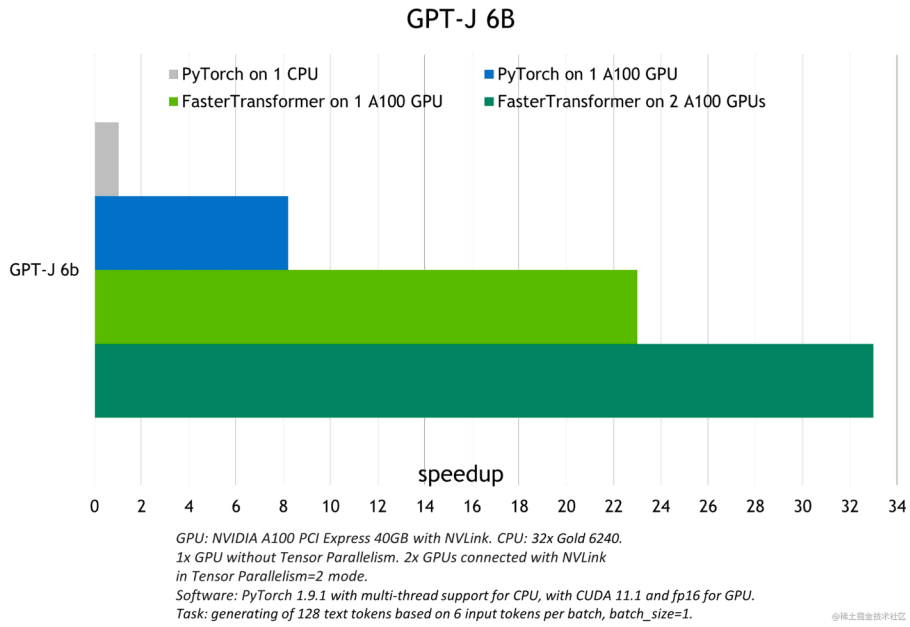


image.png

> 随着ChatGPT的爆火，大语言模型✦(LLM)得到了空前的关注。模型需要哪些核心技术，有没有代码实践教程？针对这些问题，推荐大家学习深蓝学院✦的《生成式预训练语言模型：理论与实战》课程，课程注重理论思想与代码实践相结合，最终带你从0到1制作自己的mini-ChatGPT。

FasterTransformer 中的优化技术✦

与深度学习训练的通用框架相比，FT 使您能够获得更快的推理流水线以及基于 Transformer 的神经网络具有更低的延迟和更高的吞吐量。 FT 对 GPT-3✦ 和其他大型Transformer模型进行的一些优化技术包括：

1. 层融合（Layer fusion）

这是预处理
（kernel）

算。 例如， multi-head attention+ 块中的所有操作都可以合并到一个核（kernel）中。

1. 自回归模型的推理优化(激活缓存)

为了防止通过Transformer重新计算每个新 token 生成器的先前键和值，FT 分配了一个缓冲区来在每一步存储它们。

虽然需要一些额外的内存使用，但 FT 可以节省重新计算的成本。该过程如下图所示。相同的缓存机制用于 NN 的多个部分。

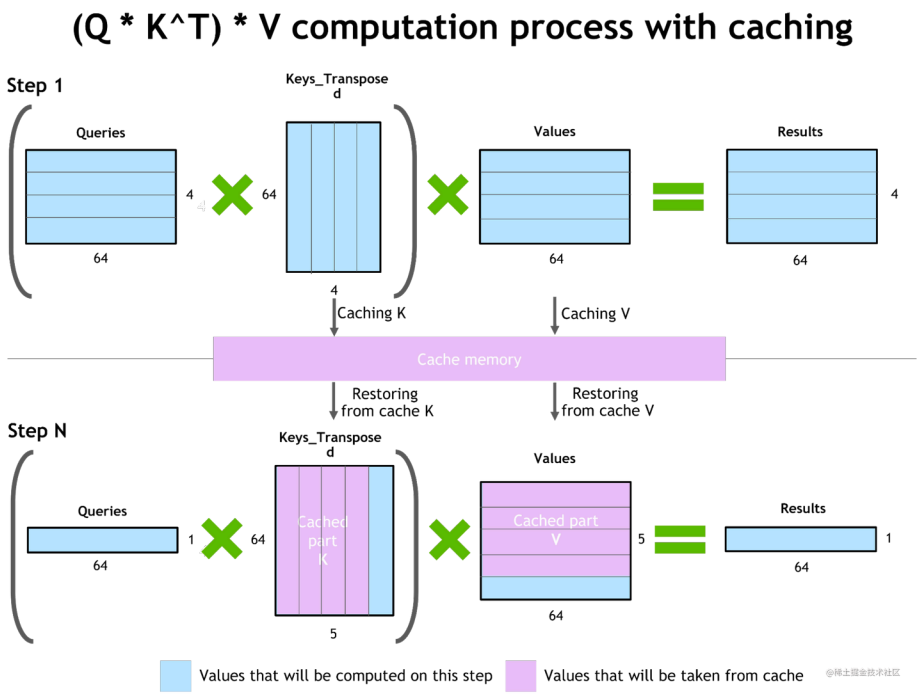


image.png

1. 内存优化

与 BERT 等传统模型不同，大型 Transformer 模型具有多达数万亿个参数，占用数百 GB 存储空间。即使我们以半精度存储模型，GPT-3 175b 也需要 350 GB。因此有必要减少其他部分的内存使用。

例如，在 FasterTransformer 中，我们在不同的解码器层重用了激活/输出的内存缓冲（buffer）。由于 GPT-3 中的层数为 96，因此我们只需要 1/96 的内存量用于激活。

1. 使用 MPI 和 NCCL 实现节点间/节点内通信并支持模型并行

FasterTransormer 同时提供张量并行和流水线并行。 对于张量并行，FasterTransformer 遵循了 **Megatron** 的思想。 对于自注意力块和前馈网络块，FT 按行拆分第一个矩阵的权重，并按列拆分第二个矩阵的权重。 通过优化，FT 可以将每个 Transformer 块的归约（reduction）操作减少到两次。

对于流水线并行，FasterTransformer 将整批请求拆分为多个微批，隐藏了通信的空泡（bubble）。 FasterTransformer 会针对不同情况自动调整微批量大小。

1. MatMul 核自动调整（GEMM 自动调整）

矩阵乘法是基于Transformer的神经网络中最主要和繁重的操作。 FT 使用来自 CuBLAS 和 CuTLASS 库的功能来执行这些类型的操作。 重要的是要知道 MatMul 操作可以在“硬件”级别使用不同的底层（low-level）算法以数十种不同的方式执行。

`[GemmBatchedEx](https://docs.nvidia.com/cuda/cublas/index.html#cublas-GemmBatchedEx)` 函数实现了 MatMul 操作，并以 `cublasGemmAlgo_t` 作为输入参数。 使用此参数，您可以选择不同的底层算法进行操作。

FasterTransformer 库使用此参数对所有底层算法进行实时基准测试，并为模型的参数和您的输入数据（注意层的大小、注意头的数量、隐藏层的大小）选择最佳的一个。 此外，FT 对网络的某些部分使用硬件加速的底层函数，例如： `__expf` 、 `__shfl_xor_sync` 。

1. 低精度推理

FT 的核（kernels）支持使用 fp16 和 int8 等低精度输入数据进行推理。 由于较少的数据传输量和所需的内存，这两种机制都会加速。 同时，int8 和 fp16 计算可以在特殊硬件上执行，例如：Tensor Core（适用于从 Volta 开始的所有 GPU 架构）和即将推出的 Hopper GPU 中的Transformer引擎。

除此之外还有**快速的 C++ BeamSearch 实现**、当模型的权重部分分配到八个 GPU 之间时，**针对 TensorParallelism 8 模式优化的 all-reduce**。

上面简述了FasterTransformer，下面将演示针对 Bloom 模型以 PyTorch 作为后端使用FasterTransformer。

FasterTransformer GPT 简介

下文将会使用BLOOM模型进行演示，而 BLOOM 是一个利用 **ALiBi**(用于添加位置嵌入) 的 GPT 模型的变体，因此，本文先简要介绍一下 GPT 的相关工作。GPT是仅解码器架构模型的一种变体，没有编码器模块，使用GeLU作为激活。

FasterTransformer GPT 工作流程

下图展示了 FasterTransformer GPT 的工作流程。 与 BERT（仅编码器结构） 和编码器-解码器结构不同，GPT 接收一些输入 id 作为上下文，并生成相应的输出 id 作为响应。 在此工作流程中，主要瓶颈是 GptDecoderLayer（transformer块），因为当我们增加层数时，时间会线性增加。在 GPT-3 中，GptDecoderLayer 占用了大约 95% 的总时间。

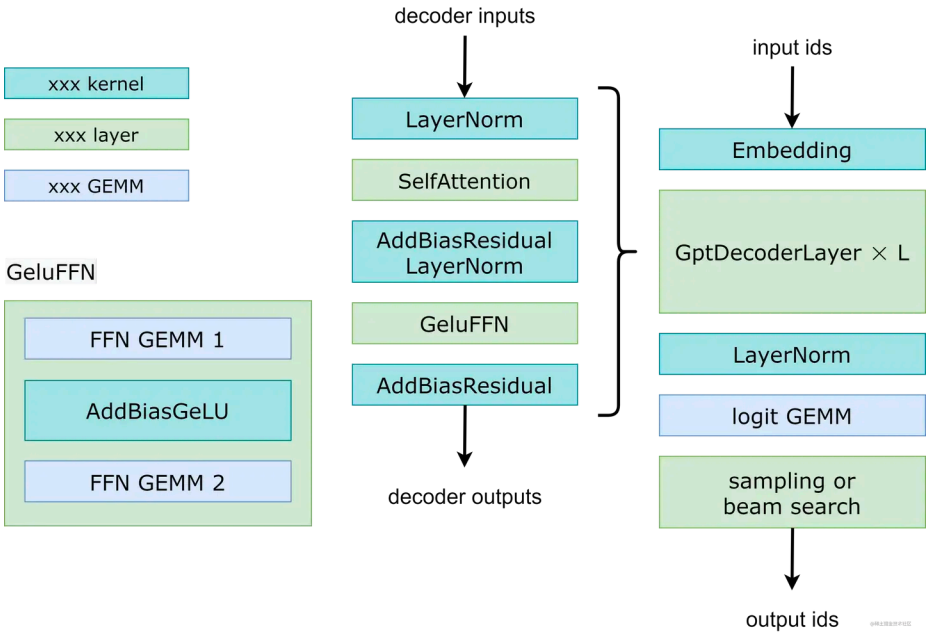
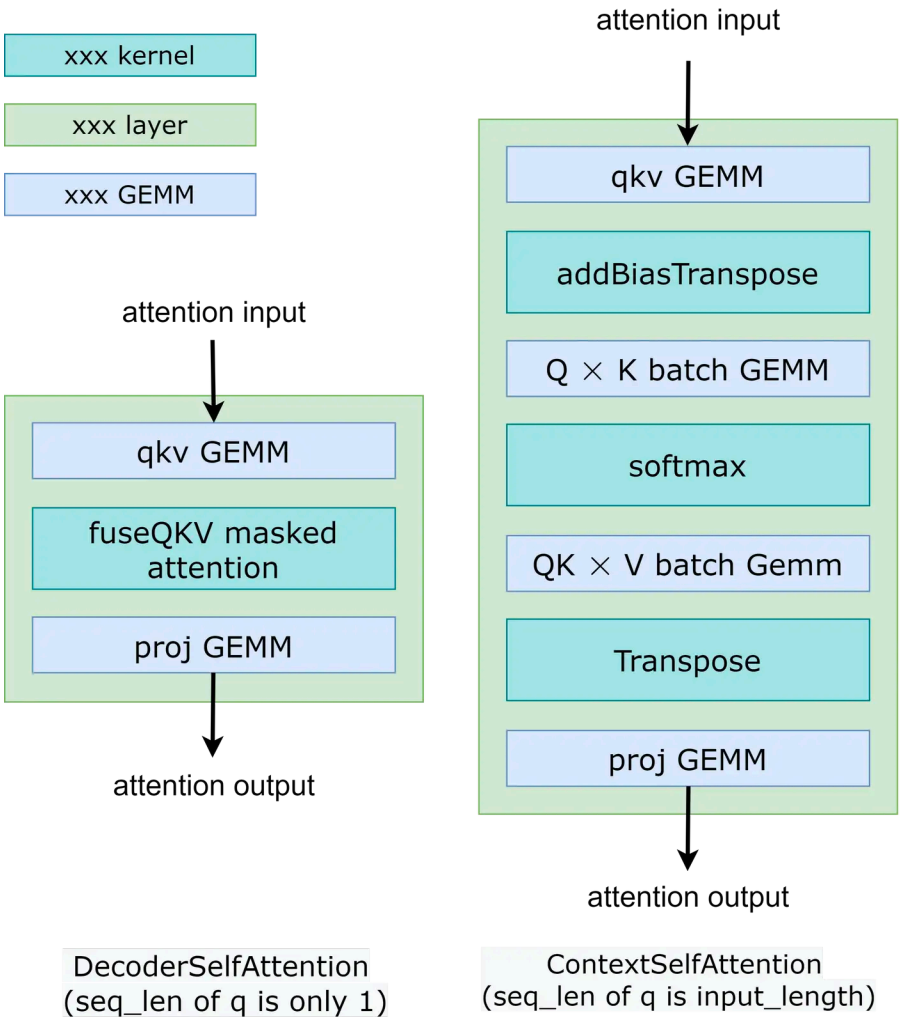


image.png

FasterTransformer 将整个工作流程分成两部分。

- 第一部分是“计算上下文（输入 ids）的 k/v 缓存”。
- 第二部分是“自回归生成输出 ids”。

这两部分的操作类似，只是SelfAttention中张量的形状不同。 因此，我们使用 2 种不同的实现来处理两种不同的情况，如下图所示。



Two types of SelfAttentions

image.png

在 DecoderSelfAttention 中，查询的序列长度始终为 1，因此我们使用自定义的 fused masked multi-head attention kernel 来处理。 另一方面，ContextSelfAttention 中查询的序列长度是最大输入长度，因此我们使用 cuBLAS 来利用tensor core。

以下的示例演示了如何运行多 GPU 和多节点的 GPT 模型。

- `examples/cpp/multi_gpu_gpt_example.cc` ： 它使用MPI来组织所有的GPU。
- `examples/cpp/multi_gpu_gpt_triton_example.cc` ： 它在节点内使用线程，在节点间使用 MPI。 此示例还演示了如何使用基于 FasterTransformer 的 Triton 后端 API 来运行 GPT 模型。
- `examples/pytorch/gpt/multi_gpu_gpt_example.py` ： 这个例子和 `examples/cpp/multi_gpu_gpt_example.cc` 类似，但是通过PyTorch OP封装了FasterTransformer的实例。

总之，运行 GPT 模型的工作流程是：

1. 通过 MPI 或线程初始化 NCCL 通信并设置张量并行和流水线并行的ranks
2. 按张量并行、流水线并行和其他模型超参数的ranks加载权重。
3. 通过张量并行、流水线并行和其他模型超参数的ranks创建ParalelGpt实例。
4. 接收来自客户端的请求并将请求转换为 ParallelGpt 的输入张量格式。
5. 运行forward
6. 将 ParallelGpt 的输出张量转换为客户端的响应并返回响应。

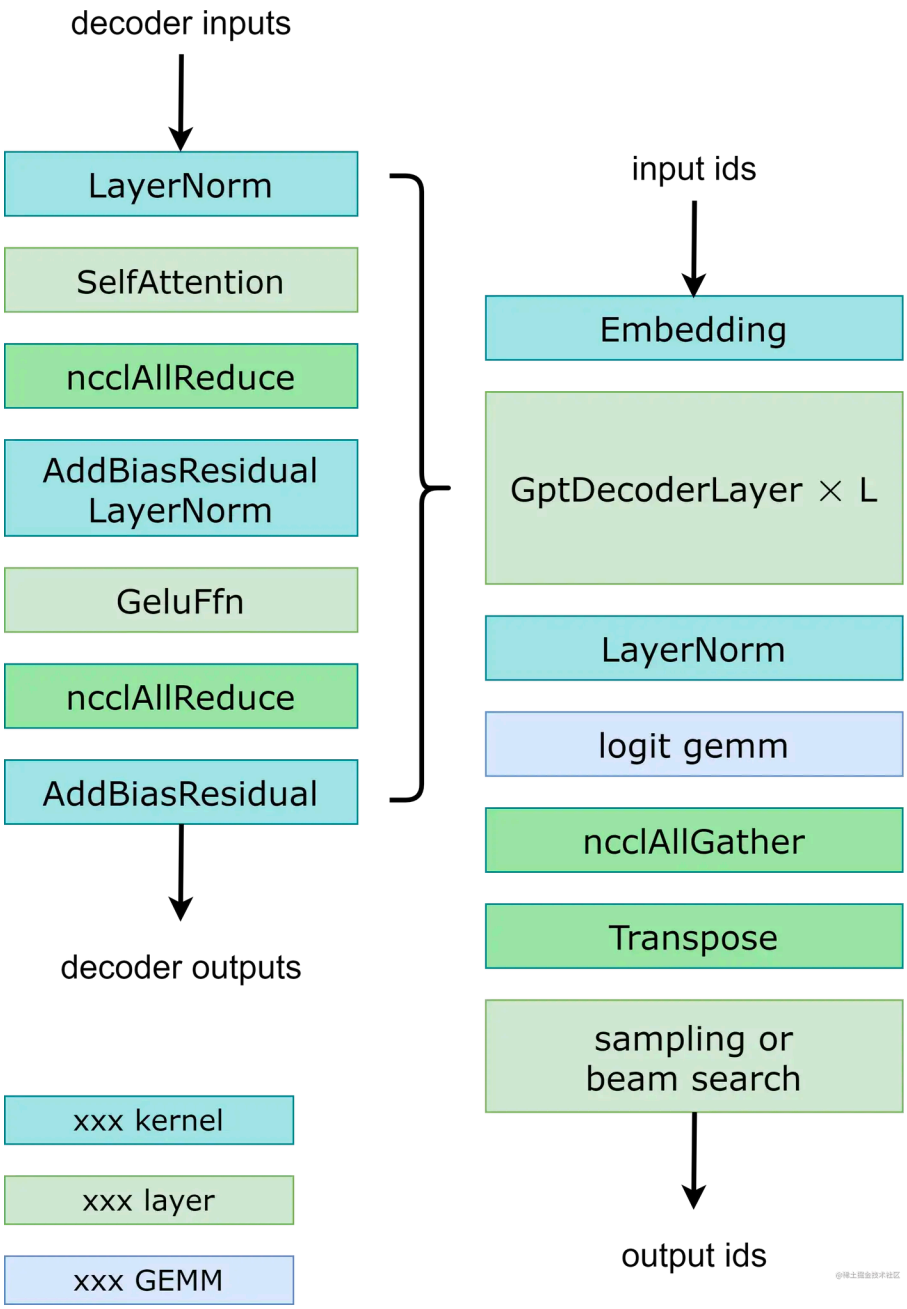
在C++示例代码中，我们跳过第4步和第6步，通过 `examples/cpp/multi_gpu_gpt/start_ids.csv` 加载该请求

有从步骤 1 到步骤 6 的完整示例。

源代码放在 `src/fastertransformer/models/multi_gpu_gpt/ParallelGpt.cc` 中。其中，GPT的构造函数参数包括head_num、num_layer、tensor_para、pipeline_para等，GPT的输入参数包括input_ids、input_lengths、output_seq_len等；GPT的输出参数包括output_ids（包含input_ids 和生成的 id）、sequence_length、output_log_probs、cum_log_probs、context_embeddings。

FasterTransformer GPT 优化

- 核优化：很多核都是基于已经高度优化的解码器和解码码模块的核。为了防止重新计算以前的键和值，我们将在每一步分配一个缓冲区来存储它们。 虽然它需要一些额外的内存使用，但我们可以节省重新计算的成本。
- 内存优化：与 BERT 等传统模型不同，GPT-3 有 1750 亿个参数，即使我们以半精度存储模型也需要 350 GB。 因此，我们必须减少其他部分的内存使用。 在 FasterTransformer 中，我们将重用不同解码器层的内存缓冲。 由于 GPT-3 的层数是 96，我们只需要 1/96 的内存。
- 模型并行：在GPT模型中，FasterTransormer同时提供张量并行和流水线并行。 对于张量并行，FasterTransformer 遵循了 Megatron 的思想。 对于自注意力块和前馈网络块，我们按行拆分第一个矩阵乘法的权重，按列拆分第二个矩阵乘法的权重。 通过优化，我们可以将每个transformer block的归约操作减少到 2 次，工作流程如下图所示。对于流水线并行，FasterTransformer 将整批请求拆分为多个微批并隐藏通信空泡。 FasterTransformer 会针对不同情况自动调整微批量大小。 用户可以通过修改 `gpt_config.ini` 文件来调整模型并行度。 我们建议在节点内使用张量并行，在节点间使用流水线并行，因为，张量并行需要更多的 NCCL 通信。



- 多框架：FasterTransformer除了c上的源代码，还提供了TensorFlow op、PyTorch op和Triton backend。 目前TensorFlow op只支持单GPU，而PyTorch op和Triton backend支持多GPU和多节点。 FasterTransformer 还提供了一个工具，可以将 Megatron 的模型拆分并转换为FasterTransformer二进制文件，以便 FasterTransformer 可以直接加载二进制文件，从而避免为模型并行而进行的额外拆分模型工作。

FasterTransformer GPT 推理选项

FasterTransformer GPT 还提供环境变量以针对特定用途进行调整。

名称	描述	默认值	可接受的值
FMHA_ENABLE	启用融合多头注意力核 (fp16 accumulation)	disabled	ON = enable fmha, otherwise disabled
CONTEXT_ATTENTION_BMM1_HALF_ACCUM	对 qk gemm 使用 fp16 累加，并且只对未融合的多头注意力核产生影响	fp32 accumulation	ON = fp32 accumulation, otherwise fp16 accumulation

环境搭建

基础环境配置

首先确保您具有以下组件：

- NVIDIA Docker 和 NGC 容器
- NVIDIA Pascal/Volta/Turing/Ampere 系列的 GPU

基础组件版本要求：

- CMake: 3.13及以上版本
- CUDA: 11.0及以上版本
- NCCL: 2.10及以上版本
- Python: 3.8.13
- PyTorch: 1.13.0

这些组件在 Nvidia 官方提供的 TensorFlow/PyTorch Docker 镜像中很容易获得。

构建FasterTransformer

推荐使用Nvidia官方提供的镜像，如： `nvcr.io/nvidia/tensorflow:22.09-tf1-py3` 、 `nvcr.io/nvidia/pytorch:22.09-py3` 等，当然也可以使用Pytorch官方提供的镜像。

首先，拉取相应版本的PyTorch镜像。

```
docker pull nvcr.io/nvidia/pytorch:22.09-py3
```

镜像下载完成之后，创建容器，以便后续进行编译和构建FasterTransformer。

```
nvidia-docker run -dti --name bloom_faster_transformer \
--restart=always --gpus all --network=host \
--shm-size 5g \
-v /home/gdong/workspace/code:/workspace/code \
-v /home/gdong/workspace/data:/workspace/data \
-v /home/gdong/workspace/model:/workspace/model \
-v /home/gdong/workspace/output:/workspace/output \
-w /workspace \
nvcr.io/nvidia/pytorch:22.09-py3 \
bash
```

进入容器。

```
docker exec -it bloom_faster_transformer bash
```

下载FasterTransformer代码。

```
cd code
git clone https://github.com/NVIDIA/FasterTransformer.git
cd FasterTransformer/
git submodule init && git submodule update
```

进入build构建FasterTransformer。

```
mkdir -p build
cd build
```

然后，执行 `cmake PATH` 命令生成 Makefile 文件。

```
cmake -DSM=80 -DCMAKE_BUILD_TYPE=Release -DBUILD_PYT=ON -DBUILD_MULTI_GPU=ON ..
```

注意：

第一点：脚本中 `-DMS=xx` 的 `xx` 表示GPU的计算能力。下表显示了常见GPU的计算能力。

GPU	计算能力
P40	60
P4	61
V100	70
T4	75
A100	80
A30	80
A10	86

默认情况下， `-DSM` 设置为 70、75、80 和 86。当用户设置更多类型的 `-DSM` 时，需要更长的编译时间。 因此，我们建议只为您使用的设备设置 `-DSM` 。

第二点：本文使用Pytorch作为后端，因此，脚本中添加了 `-DBUILD_PYT=ON` 配置项。这将构建 TorchScript 自定义类。 因此，请确保 PyTorch 版本大于 1.5.0。

运行过程：

```
-- The CXX compiler identification is GNU 9.4.0
-- The CUDA compiler identification is NVIDIA 11.8.89
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Detecting CUDA compiler ABI info
-- Detecting CUDA compiler ABI info - done
-- Check for working CUDA compiler: /usr/local/cuda/bin/nvcc - skipped
-- Detecting CUDA compile features
-- Detecting CUDA compile features - done
-- Looking for C++ include pthread.h
-- Looking for C++ include pthread.h - found
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Failed
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Found CUDA: /usr/local/cuda (found suitable version "11.8", minimum required is "10
CUDA_VERSION 11.8 is greater or equal than 11.0, enable -DENABLE_BF16 flag
-- Found CUDNN: /usr/lib/x86_64-linux-gnu/libcudnn.so
-- Add DBUILD_CUTLASS_MOE, requires CUTLASS. Increases compilation time
-- Add DBUILD_CUTLASS_MIXED_GEMM, requires CUTLASS. Increases compilation time
-- Running submodule update to fetch cutlass
-- Add DBUILD_MULTI_GPU, requires MPI and NCCL
-- Found MPI_CXX: /opt/hpcx/ompi/lib/libmpi.so (found version "3.1")
-- Found MPI: TRUE (found version "3.1")
-- Found NCCL: /usr/include
-- Determining NCCL version from /usr/include/nccl.h...
-- Looking for NCCL_VERSION_CODE
-- Looking for NCCL_VERSION_CODE - not found
-- Found NCCL (include: /usr/include, library: /usr/lib/x86_64-linux-gnu/libnccl.so.2.
-- NVTX is enabled.
-- Assign GPU architecture (sm=80)
-- Use WMMA
CMAKE_CU
-- COMMO
```

```
-- Found CUDA: /usr/local/cuda (found version "11.8")
-- Caffe2: CUDA detected: 11.8
-- Caffe2: CUDA nvcc is: /usr/local/cuda/bin/nvcc
-- Caffe2: CUDA toolkit directory: /usr/local/cuda
-- Caffe2: Header version is: 11.8
-- Found cuDNN: v8.6.0  (include: /usr/include, library: /usr/lib/x86_64-linux-gnu/lib
-- /usr/local/cuda/lib64/libnvrtc.so shorthash is 672ee683
-- Added CUDA NVCC flags for: -gencode;arch=compute_80,code=sm_80
-- Found Torch: /opt/conda/lib/python3.8/site-packages/torch/lib/libtorch.so
-- USE_CXX11_ABI=True
-- The C compiler identification is GNU 9.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Found Python: /opt/conda/bin/python3.8 (found version "3.8.13") found components: I
-- Configuring done
-- Generating done
-- Build files have been written to: /workspace/code/FasterTransformer/build
```

之后，通过 `make` 使用12个线程去执行编译加快编译速度：

```
make -j12
```

运行过程：

```
[  0%] Building CXX object src/fastertransformer/kernels/cutlass_kernels/CMakeFiles/cu
[  0%] Building CXX object src/fastertransformer/utils/CMakeFiles/nvtx_utils.dir/nvtx_
[  0%] Building CUDA object src/fastertransformer/kernels/CMakeFiles/layernorm_kernels
[  0%] Building CXX object src/fastertransformer/utils/CMakeFiles/cuda_utils.dir/cuda_
[  0%] Building CXX object src/fastertransformer/utils/CMakeFiles/logger.dir/logger.cc
[  1%] Building CXX object 3rdparty/common/CMakeFiles/cuda_driver_wrapper.dir/cudaDriv
[  1%] Building CUDA object src/fastertransformer/kernels/CMakeFiles/custom_ar_kernels
[  1%] Building CUDA object src/fastertransformer/kernels/CMakeFiles/add_residual_kern
[  1%] Building CUDA object src/fastertransformer/kernels/CMakeFiles/activation_kernel
[  1%] Building CUDA object src/fastertransformer/kernels/CMakeFiles/transpose_int8_ke
[  2%] Building CUDA object src/fastertransformer/kernels/CMakeFiles/unfused_attention
[  2%] Building CUDA object src/fastertransformer/kernels/CMakeFiles/bert_preprocess_k
[  2%] Linking CUDA device code CMakeFiles/cuda_driver_wrapper.dir/cmake_device_link.o
[  2%] Linking CXX static library ../../lib/libcuda_driver_wrapper.a
[  2%] Built target cuda_driver_wrapper
...
[100%] Linking CXX executable ../../../bin/gptneox_example
[100%] Built target gptj_triton_example
[100%] Building CXX object examples/cpp/multi_gpu_gpt/CMakeFiles/multi_gpu_gpt_triton_
[100%] Built target gptj_example
[100%] Building CXX object examples/cpp/multi_gpu_gpt/CMakeFiles/multi_gpu_gpt_interac
[100%] Built target gptneox_example
[100%] Linking CXX executable ../../../bin/multi_gpu_gpt_example
[100%] Linking CXX executable ../../../bin/gptneox_triton_example
[100%] Built target multi_gpu_gpt_example
[100%] Built target gptneox_triton_example
[100%] Linking CXX executable ../../../bin/multi_gpu_gpt_triton_example
[100%] Linking CXX static library ../../../../lib/libth_t5.a
[100%] Built target th_t5
[100%] Built target multi_gpu_gpt_triton_example
[100%] Linking CXX executable ../../../bin/multi_gpu_gpt_async_example
[100%] Linking CXX executable ../../../bin/multi_gpu_gpt_interactive_example
[100%] Built target multi_gpu_gpt_async_example
[100%] Linking CXX static library ../../../../lib/libth_parallel_gpt.a
[100%] Built target th_parallel_gpt
[100%] Linking CXX shared library ../../../lib/libth_transformer.so
```

```
[100%] Built target multi_gpu_gpt_interactive_example
[100%] Built target th_transformer
```

至此，构建FasterTransformer完成。

安装依赖包

安装进行模型推理所需要的依赖包。

```
cd /workspace/code/FasterTransformer
pip install -r examples/pytorch/gpt/requirement.txt -i https://pypi.tuna.tsinghua.edu.
```

数据与模型准备

模型

本文使用BLOOM模型进行演示，它不需要学习位置编码，并允许模型生成比训练中使用的序列长度更长的序列。 BLOOM 也具有与 OpenAI GPT 相似的结构。因此，像 OPT 一样，FT 通过 GPT 类提供了 BLOOM 模型作为变体。用户可以使用 `examples/pytorch/gpt/utils/huggingface_bloom_convert.py` 将预训练的 Huggingface BLOOM 模型转换为 fastertransformer 文件格式。

我们使用 `bloomz-560m` 作为基础模型。该模型是基于**bloom-560m**在**xP3**数据集上对多任务进行了微调而得到的。

下载模型：

```
cd /workspace/model
git lfs clone https://huggingface.co/bigscience/bloomz-560m
```

模型文件：

```
> ls -al bloomz-560m
total 2198796
drwxr-xr-x 4 root root       4096 Apr 25 16:50 .
drwxr-xr-x 4 root root       4096 Apr 26 07:06 ..
drwxr-xr-x 9 root root       4096 Apr 25 16:53 .git
-rw-r--r-- 1 root root       1489 Apr 25 16:50 .gitattributes
-rw-r--r-- 1 root root      24778 Apr 25 16:50 README.md
-rw-r--r-- 1 root root        715 Apr 25 16:50 config.json
drwxr-xr-x 4 root root       4096 Apr 25 16:50 logs
-rw-r--r-- 1 root root 1118459450 Apr 25 16:53 model.safetensors
-rw-r--r-- 1 root root 1118530423 Apr 25 16:53 pytorch_model.bin
-rw-r--r-- 1 root root         85 Apr 25 16:50 special_tokens_map.json
-rw-r--r-- 1 root root   14500438 Apr 25 16:50 tokenizer.json
-rw-r--r-- 1 root root        222 Apr 25 16:50 tokenizer_config.json
```

数据集

本文使用Lambada数据集，它是一个NLP（自然语言处理）任务中使用的数据集。它包含大量的英文句子，并要求模型去预测下一个单词，这种任务称为语言建模。Lambada数据集的特点是它的句子长度较长，并且包含更丰富的语义信息。因此，对于语言模型的评估来说是一个很好的测试数据集。

下载LAMBADA测试数据集。

```
cd /workspace/data
wget -c https://github.com/cybertronai/bflm/raw/master/lambada_test.jsonl
```

数据格式如下：

```
{"text": "In my palm is a clear stone, and inside it is a small ivory statuette. A gua
{"text": "Give me a minute to change and I'll meet you at the docks." She'd forced tho
...
{"text": ""Only one source I know of that would be likely to cough up enough money to
{"text": "Helen's heart broke a little in the face of Miss Mabel's selfless courage. S
{"text": "Preston had been the last person to wear those chains, and I knew what I'd s
```

模型格式转换

为了避免在模型并行时，拆分模型的额外工作，FasterTransformer 提供了一个工具，用于将模型从不同格式拆分和转换为 FasterTransformer 二进制文件格式；然后， FasterTransformer 可以直接以二进制格式加载模型。

将Huggingface Transformer模型权重文件格式转换成FasterTransformer格式。

```
cd /workspace/code/FasterTransformer

python examples/pytorch/gpt/utils/huggingface_bloom_convert.py \
    --input-dir /workspace/model/bloomz-560m \
    --output-dir /workspace/model/bloomz-560m-convert \
    --data-type fp16 \
    -tp 1 -v
```

转换过程：

```
python examples/pytorch/gpt/utils/huggingface_bloom_convert.py \
>     --input-dir /workspace/model/bloomz-560m \
>     --output-dir /workspace/model/bloomz-560m-convert \
>     --data-type fp16 \
>     -tp 1 -v

====================== Arguments ======================
 - input_dir..........: /workspace/model/bloomz-560m
 - output_dir.........: /workspace/model/bloomz-560m-convert
 - tensor_para_size....: 1
 - data_type..........: fp16
 - processes..........: 1
 - verbose............: True
 - by_shard...........: False
=======================================================
loading from pytorch bin format
model file num: 1
 - model.wte.....................................: shape (250880, 1024)     | saved
 - model.pre_decoder_layernorm.weight............: shape (1024,)            | saved
 - model.pre_decoder_layernorm.bias..............: shape (1024,)            | saved
 - model.layers.0.input_layernorm.weight.........: shape (1024,)            | saved
 - model.layers.0.input_layernorm.bias...........: shape (1024,)            | saved
 - model.layers.0.attention.query_key_value.weight.: shape (1024, 3, 1024)  s | saved
 - model.layers.0.attention.query_key_value.bias...: shape (3, 1024)        s | saved
 - model.layers.0.attention.dense.weight.........: shape (1024, 1024)     s | saved
 - model.layers.0.attention.dense.bias...........: shape (1024,)            | saved
 - model.layers.0.post_attention_layernorm.weight..: shape (1024,)          | saved
 - model.layers.0.post_attention_layernorm.bias....: shape (1024,)          | saved
 - model.layers.0.mlp.dense_h_to_4h.weight.........: shape (1024, 4096)     s | saved
 - model.layers.0.mlp.dense_h_to_4h.bias..........: shape (4096,)          s | saved
...
rs.22.ml
```

```
 - model.layers.23.input_layernorm.weight.........: shape (1024,)            | saved
 - model.layers.23.input_layernorm.bias...........: shape (1024,)            | saved
 - model.layers.23.attention.query_key_value.weight: shape (1024, 3, 1024)  s | saved
 - model.layers.23.attention.query_key_value.bias..: shape (3, 1024)        s | saved
 - model.layers.23.attention.dense.weight.........: shape (1024, 1024)     s | saved
 - model.layers.23.attention.dense.bias...........: shape (1024,)            | saved
 - model.layers.23.post_attention_layernorm.weight.: shape (1024,)           | saved
 - model.layers.23.post_attention_layernorm.bias...: shape (1024,)           | saved
 - model.layers.23.mlp.dense_h_to_4h.weight.......: shape (1024, 4096)     s | saved
 - model.layers.23.mlp.dense_h_to_4h.bias.........: shape (4096,)          s | saved
 - model.layers.23.mlp.dense_4h_to_h.weight.......: shape (4096, 1024)     s | saved
 - model.layers.23.mlp.dense_4h_to_h.bias.........: shape (1024,)            | saved
 - model.final_layernorm.weight...................: shape (1024,)            | saved
 - model.final_layernorm.bias.....................: shape (1024,)            | saved
Checkpoint conversion (HF >> FT) has done (elapsed time: 17.07 sec)
```

转换成FasterTransformer格式后的文件如下所示：

```
> tree bloomz-560m-convert/
bloomz-560m-convert/
└── 1-gpu
    ├── config.ini
    ├── model.final_layernorm.bias.bin
    ├── model.final_layernorm.weight.bin
    ├── model.layers.0.attention.dense.bias.bin
    ├── model.layers.0.attention.dense.weight.0.bin
    ├── model.layers.0.attention.query_key_value.bias.0.bin
    ├── model.layers.0.attention.query_key_value.weight.0.bin
    ├── model.layers.0.input_layernorm.bias.bin
    ├── model.layers.0.input_layernorm.weight.bin
    ├── model.layers.0.mlp.dense_4h_to_h.bias.bin
    ├── model.layers.0.mlp.dense_4h_to_h.weight.0.bin
    ├── model.layers.0.mlp.dense_h_to_4h.bias.0.bin
    ├── model.layers.0.mlp.dense_h_to_4h.weight.0.bin
    ├── model.layers.0.post_attention_layernorm.bias.bin
    ├── model.layers.0.post_attention_layernorm.weight.bin
    ├── model.layers.1.attention.dense.bias.bin
    ...
    ├── model.layers.8.post_attention_layernorm.weight.bin
    ├── model.layers.9.attention.dense.bias.bin
    ├── model.layers.9.attention.dense.weight.0.bin
    ├── model.layers.9.attention.query_key_value.bias.0.bin
    ├── model.layers.9.attention.query_key_value.weight.0.bin
    ├── model.layers.9.input_layernorm.bias.bin
    ├── model.layers.9.input_layernorm.weight.bin
    ├── model.layers.9.mlp.dense_4h_to_h.bias.bin
    ├── model.layers.9.mlp.dense_4h_to_h.weight.0.bin
    ├── model.layers.9.mlp.dense_h_to_4h.bias.0.bin
    ├── model.layers.9.mlp.dense_h_to_4h.weight.0.bin
    ├── model.layers.9.post_attention_layernorm.bias.bin
    ├── model.layers.9.post_attention_layernorm.weight.bin
    ├── model.pre_decoder_layernorm.bias.bin
    ├── model.pre_decoder_layernorm.weight.bin
    └── model.wte.bin
```

模型基准测试

下面使用官方提供的样例进行基准测试对比下Huggingface Transformers和FasterTransformer的响应时长。

Huggingface Transformers基准测试

运行命令：

```
# Run HF benchmark
CUDA_VISIBLE_DEVICES=1 python examples/pytorch/gpt/bloom_lambada.py \
    --tokenizer-path /workspace/model/bloomz-560m \
    --dataset-path /workspace/data/lambada_test.jsonl \
    --lib-path bulid/lib/libth_transformer.so \
    --test-hf \
    --show-progress
```

运行过程：

```
python examples/pytorch/gpt/bloom_lambada.py \
>     --tokenizer-path /workspace/model/bloomz-560m \
>     --dataset-path /workspace/data/lambada_test.jsonl \
>     --lib-path bulid/lib/libth_transformer.so \
>     --test-hf \
>     --show-progress

================== Arguments ==================
 - num_heads.................: None
 - size_per_head.............: None
 - inter_size................: None
 - num_layers................: None
 - vocab_size................: None
 - tensor_para_size..........: 1
 - pipeline_para_size........: 1
 - remove_padding............: True
 - shared_contexts_ratio.....: 1.0
 - batch_size................: 8
 - output_length.............: 32
 - beam_width................: 1
 - top_k.....................: 1
 - top_p.....................: 1.0
 - temperature...............: 1.0
 - len_penalty...............: 0.0
 - beam_search_diversity_rate: 0.0
 - start_id..................: 0
 - end_id....................: 2
 - repetition_penalty........: 1.0
 - random_seed...............: None
 - return_cum_log_probs......: 0
 - checkpoint_path...........: None
 - dataset_path..............: /workspace/data/lambada_test.jsonl
 - output_path...............: None
 - tokenizer_path............: /workspace/model/bloomz-560m
 - lib_path..................: bulid/lib/libth_transformer.so
 - test_hf...................: True
 - acc_threshold.............: None
 - show_progress.............: True
 - inference_data_type.......: None
 - weights_data_type.........: None
 - int8_mode.................: 0
================================================
100%|██████████████████████████████████████████████████████████████████████████████
Accuracy: 39.4722% (2034/5153) (elapsed time: 146.7230 sec)
```

FasterTransformer基准测试

运行命令：

```
# Run FT benchmark
python examples/pytorch/gpt/bloom_lambada.py \
    --checkpoint-path /workspace/model/bloomz-560m-convert/1-gpu \
    --to
    --da
```

```
    --lib-path build/lib/libth_transformer.so \
    --show-progress
```

注：还可添加 `--data-type fp16` 以半精度方式加载模型，以减少模型对于显存的消耗。

运行过程：

```
python examples/pytorch/gpt/bloom_lambada.py \
>     --checkpoint-path /workspace/model/bloomz-560m-convert/1-gpu \
>     --tokenizer-path /workspace/model/bloomz-560m \
>     --dataset-path /workspace/data/lambada_test.jsonl \
>     --lib-path build/lib/libth_transformer.so \
>     --show-progress

================== Arguments ==================
 - num_heads.................: None
 - size_per_head.............: None
 - inter_size................: None
 - num_layers................: None
 - vocab_size................: None
 - tensor_para_size..........: 1
 - pipeline_para_size........: 1
 - remove_padding............: True
 - shared_contexts_ratio.....: 1.0
 - batch_size................: 8
 - output_length.............: 32
 - beam_width................: 1
 - top_k.....................: 1
 - top_p.....................: 1.0
 - temperature...............: 1.0
 - len_penalty...............: 0.0
 - beam_search_diversity_rate: 0.0
 - start_id..................: 0
 - end_id....................: 2
 - repetition_penalty........: 1.0
 - random_seed...............: None
 - return_cum_log_probs......: 0
 - checkpoint_path...........: /workspace/model/bloomz-560m-convert/1-gpu
 - dataset_path..............: /workspace/data/lambada_test.jsonl
 - output_path...............: None
 - tokenizer_path............: /workspace/model/bloomz-560m
 - lib_path..................: build/lib/libth_transformer.so
 - test_hf...................: False
 - acc_threshold.............: None
 - show_progress.............: True
 - inference_data_type.......: None
 - weights_data_type.........: None
 - int8_mode.................: 0
================================================
[FT][INFO] Load BLOOM model
 - head_num..................: 16
 - size_per_head.............: 64
 - layer_num.................: 24
 - tensor_para_size..........: 1
 - vocab_size................: 250880
 - start_id..................: 1
 - end_id....................: 2
 - weights_data_type.........: fp16
 - layernorm_eps.............: 1e-05
 - inference_data_type.......: fp16
 - lib_path..................: build/lib/libth_transformer.so
 - pipeline_para_size........: 1
 - shared_contexts_ratio.....: 1.0
 - int8_mode.................: 0
[WARNING] gemm_config.in is not found; using default GEMM algo
[FT][WAR
[FT][INF
```

```
100%|████████████████████████████████████████████████████████████████████████████████
Accuracy: 39.4722% (2034/5153) (elapsed time: 13.0032 sec)
```

对比Huggingface Transformers和FasterTransformer

```
HF:    Accuracy: 39.4722% (2034/5153) (elapsed time: 146.7230 sec)
FT:    Accuracy: 39.4722% (2034/5153) (elapsed time: 13.0032 sec)
```

可以看到它们的准确率一致，但是FasterTransformer比Huggingface Transformers的推理速度更加快速。

模型并行推理（多卡）

对于像GPT3（175B）、OPT-175B这样的大模型，单卡无法加载整个模型，因此，我们需要以分布式（模型并行）方式进行大模型推理。模型并行推理有两种方式：张量并行和流水线并行，前面已经进行过相应的说明，这里不再赘述。

张量并行

模型转换

如果想使用张量并行 (TP) 技术将模型拆分多个GPU进行推理，可参考如下命令将模型转换到2个GPU上进行推理。

```
python examples/pytorch/gpt/utils/huggingface_bloom_convert.py \
--input-dir /workspace/model/bloomz-560m \
--output-dir /workspace/model/bloomz-560m-convert \
--data-type fp16 \
-tp 2 -v
```

转换成张量并行度为2的FasterTransformer格式后的文件如下所示：

```
tree /workspace/model/bloomz-560m-convert/2-gpu
/workspace/model/bloomz-560m-convert/2-gpu
├── config.ini
├── model.final_layernorm.bias.bin
├── model.final_layernorm.weight.bin
├── model.layers.0.attention.dense.bias.bin
├── model.layers.0.attention.dense.weight.0.bin
├── model.layers.0.attention.dense.weight.1.bin
├── model.layers.0.attention.query_key_value.bias.0.bin
├── model.layers.0.attention.query_key_value.bias.1.bin
├── model.layers.0.attention.query_key_value.weight.0.bin
├── model.layers.0.attention.query_key_value.weight.1.bin
├── model.layers.0.input_layernorm.bias.bin
├── model.layers.0.input_layernorm.weight.bin
├── model.layers.0.mlp.dense_4h_to_h.bias.bin
├── model.layers.0.mlp.dense_4h_to_h.weight.0.bin
├── model.layers.0.mlp.dense_4h_to_h.weight.1.bin
├── model.layers.0.mlp.dense_h_to_4h.bias.0.bin
├── model.layers.0.mlp.dense_h_to_4h.bias.1.bin
├── model.layers.0.mlp.dense_h_to_4h.weight.0.bin
├── model.layers.0.mlp.dense_h_to_4h.weight.1.bin
├── model.layers.0.post_attention_layernorm.bias.bin
├── model.layers.0.post_attention_layernorm.weight.bin
...
├── model.layers.9.attention.dense.bias.bin
├── model.layers.9.attention.dense.weight.0.bin
├── mode
├── mode
```

```
├── model.layers.9.attention.query_key_value.bias.1.bin
├── model.layers.9.attention.query_key_value.weight.0.bin
├── model.layers.9.attention.query_key_value.weight.1.bin
├── model.layers.9.input_layernorm.bias.bin
├── model.layers.9.input_layernorm.weight.bin
├── model.layers.9.mlp.dense_4h_to_h.bias.bin
├── model.layers.9.mlp.dense_4h_to_h.weight.0.bin
├── model.layers.9.mlp.dense_4h_to_h.weight.1.bin
├── model.layers.9.mlp.dense_h_to_4h.bias.0.bin
├── model.layers.9.mlp.dense_h_to_4h.bias.1.bin
├── model.layers.9.mlp.dense_h_to_4h.weight.0.bin
├── model.layers.9.mlp.dense_h_to_4h.weight.1.bin
├── model.layers.9.post_attention_layernorm.bias.bin
├── model.layers.9.post_attention_layernorm.weight.bin
├── model.pre_decoder_layernorm.bias.bin
├── model.pre_decoder_layernorm.weight.bin
└── model.wte.bin

0 directories, 438 files
```

张量并行模型推理

运行命令：

```
mpirun -n 2 --allow-run-as-root python examples/pytorch/gpt/bloom_lambada.py \
    --checkpoint-path /workspace/model/bloomz-560m-convert/2-gpu \
    --tokenizer-path /workspace/model/bloomz-560m \
    --dataset-path /workspace/data/lambada_test.jsonl \
    --lib-path build/lib/libth_transformer.so \
    --tensor-para-size 2 \
    --pipeline-para-size 1 \
    --show-progress
```

运行过程：

```
mpirun -n 2 --allow-run-as-root python examples/pytorch/gpt/bloom_lambada.py \
>     --checkpoint-path /workspace/model/bloomz-560m-convert/2-gpu \
>     --tokenizer-path /workspace/model/bloomz-560m \
>     --dataset-path /workspace/data/lambada_test.jsonl \
>     --lib-path build/lib/libth_transformer.so \
>     --tensor-para-size 2 \
>     --pipeline-para-size 1 \
>     --show-progress

================== Arguments ==================
 - num_heads.................: None
 - size_per_head.............: None
 - inter_size................: None
 - num_layers................: None
 - vocab_size................: None
 - tensor_para_size..........: 2
 - pipeline_para_size........: 1
 - remove_padding............: True
 - shared_contexts_ratio.....: 1.0
 - batch_size................: 8
 - output_length.............: 32
 - beam_width................: 1
 - top_k.....................: 1
 - top_p.....................: 1.0
 - temperature...............: 1.0
 - len_penalty...............: 0.0
 - beam_search_diversity_rate: 0.0
 - start_id..................: 0
 - end_i
 - repet
```

```
 - random_seed.............: None
 - return_cum_log_probs.....: 0
 - checkpoint_path..........: /workspace/model/bloomz-560m-convert/2-gpu
 - dataset_path.............: /workspace/data/lambada_test.jsonl
 - output_path..............: None
 - tokenizer_path...........: /workspace/model/bloomz-560m
 - lib_path.................: build/lib/libth_transformer.so
 - test_hf..................: False
 - acc_threshold............: None
 - show_progress............: True
 - inference_data_type......: None
 - weights_data_type........: None
 - int8_mode................: 0
================================================

================== Arguments ==================
 - num_heads................: None
 - size_per_head............: None
 - inter_size...............: None
 - num_layers...............: None
 - vocab_size...............: None
 - tensor_para_size.........: 2
 - pipeline_para_size.......: 1
 - remove_padding...........: True
 - shared_contexts_ratio....: 1.0
 - batch_size...............: 8
 - output_length............: 32
 - beam_width...............: 1
 - top_k....................: 1
 - top_p....................: 1.0
 - temperature..............: 1.0
 - len_penalty..............: 0.0
 - beam_search_diversity_rate: 0.0
 - start_id.................: 0
 - end_id...................: 2
 - repetition_penalty.......: 1.0
 - random_seed..............: None
 - return_cum_log_probs.....: 0
 - checkpoint_path..........: /workspace/model/bloomz-560m-convert/2-gpu
 - dataset_path.............: /workspace/data/lambada_test.jsonl
 - output_path..............: None
 - tokenizer_path...........: /workspace/model/bloomz-560m
 - lib_path.................: build/lib/libth_transformer.so
 - test_hf..................: False
 - acc_threshold............: None
 - show_progress............: True
 - inference_data_type......: None
 - weights_data_type........: None
 - int8_mode................: 0
================================================
[FT][INFO] Load BLOOM model
 - head_num.................: 16
 - size_per_head............: 64
 - layer_num................: 24
 - tensor_para_size.........: 2
 - vocab_size...............: 250880
 - start_id.................: 1
 - end_id...................: 2
 - weights_data_type........: fp16
 - layernorm_eps............: 1e-05
 - inference_data_type......: fp16
 - lib_path.................: build/lib/libth_transformer.so
 - pipeline_para_size.......: 1
 - shared_contexts_ratio....: 1.0
 - int8_mode................: 0
[FT][INFO] Load BLOOM model
 - head_
 - size_
```

```
 - layer_num.................: 24
 - tensor_para_size.........: 2
 - vocab_size...............: 250880
 - start_id.................: 1
 - end_id...................: 2
 - weights_data_type........: fp16
 - layernorm_eps............: 1e-05
 - inference_data_type......: fp16
 - lib_path.................: build/lib/libth_transformer.so
 - pipeline_para_size.......: 1
 - shared_contexts_ratio....: 1.0
 - int8_mode................: 0
world_size: 2
world_size: 2
[WARNING] gemm_config.in is not found; using default GEMM algo
[WARNING] gemm_config.in is not found; using default GEMM algo
[FT][INFO] NCCL initialized rank=0 world_size=2 tensor_para=NcclParam[rank=0, world_si
[FT][INFO] Device NVIDIA A800 80GB PCIe
[FT][INFO] NCCL initialized rank=1 world_size=2 tensor_para=NcclParam[rank=1, world_si
[FT][INFO] Device NVIDIA A800 80GB PCIe
/workspace/code/FasterTransformer/examples/pytorch/gpt/utils/gpt.py:221: SyntaxWarning
  assert(self.pre_embed_idx < self.post_embed_idx, "Pre decoder embedding index should
  0%|          | 0/645 [00:00<?, ?it/s]/workspace/code/FasterTransformer/examples/pyto
  assert(self.pre_embed_idx < self.post_embed_idx, "Pre decoder embedding index should
100%|██████████| 645/645 [00:20<00:00, 31.11it/s]Accuracy: 39.4527% (2033/5153) (elaps
100%|██████████| 645/645 [00:20<00:00, 31.21it/s]Accuracy: 39.4527% (2033/5153) (elaps
```

流水线并行

模型转换

如果仅使用流水线并行，不使用张量并行，则 `tp` 设置为1即可，如果需要同时进行张量并行和流水线并行，则需要将 `tp` 设置成张量并行度大小。具体命令参考前面的模型转换部分。

流水线并行模型推理

运行命令：

```
CUDA_VISIBLE_DEVICES=1,2 mpirun -n 2 --allow-run-as-root python examples/pytorch/gpt/b
    --checkpoint-path /workspace/model/bloomz-560m-convert/1-gpu \
    --tokenizer-path /workspace/model/bloomz-560m \
    --dataset-path /workspace/data/lambada_test.jsonl \
    --lib-path build/lib/libth_transformer.so \
    --tensor-para-size 1 \
    --pipeline-para-size 2 \
    --batch-size 1 \
    --show-progress
```

运行过程：

```
CUDA_VISIBLE_DEVICES=1,2 mpirun -n 2 --allow-run-as-root python examples/pytorch/gpt/b
>     --checkpoint-path /workspace/model/bloomz-560m-convert/1-gpu \
>     --tokenizer-path /workspace/model/bloomz-560m \
>     --dataset-path /workspace/data/lambada_test.jsonl \
>     --lib-path build/lib/libth_transformer.so \
>     --tensor-para-size 1 \
>     --pipeline-para-size 2 \
>     --batch-size 1 \
>     --show-progress

========
 - num_h
```

```
 - size_per_head.............: None
 - inter_size................: None
 - num_layers................: None
 - vocab_size................: None
 - tensor_para_size..........: 1
 - pipeline_para_size........: 2
 - remove_padding............: True
 - shared_contexts_ratio.....: 1.0
 - batch_size................: 1
 - output_length.............: 32
 - beam_width................: 1
 - top_k.....................: 1
 - top_p.....................: 1.0
 - temperature...............: 1.0
 - len_penalty...............: 0.0
 - beam_search_diversity_rate: 0.0
 - start_id..................: 0
 - end_id....................: 2
 - repetition_penalty........: 1.0
 - random_seed...............: None
 - return_cum_log_probs......: 0
 - checkpoint_path...........: /workspace/model/bloomz-560m-convert/1-gpu
 - dataset_path..............: /workspace/data/lambada_test.jsonl
 - output_path...............: None
 - tokenizer_path............: /workspace/model/bloomz-560m
 - lib_path..................: build/lib/libth_transformer.so
 - test_hf...................: False
 - acc_threshold.............: None
 - show_progress.............: True
 - inference_data_type.......: None
 - weights_data_type.........: None
 - int8_mode.................: 0
================================================

=================== Arguments ===================
 - num_heads.................: None
 - size_per_head.............: None
 - inter_size................: None
 - num_layers................: None
 - vocab_size................: None
 - tensor_para_size..........: 1
 - pipeline_para_size........: 2
 - remove_padding............: True
 - shared_contexts_ratio.....: 1.0
 - batch_size................: 1
 - output_length.............: 32
 - beam_width................: 1
 - top_k.....................: 1
 - top_p.....................: 1.0
 - temperature...............: 1.0
 - len_penalty...............: 0.0
 - beam_search_diversity_rate: 0.0
 - start_id..................: 0
 - end_id....................: 2
 - repetition_penalty........: 1.0
 - random_seed...............: None
 - return_cum_log_probs......: 0
 - checkpoint_path...........: /workspace/model/bloomz-560m-convert/1-gpu
 - dataset_path..............: /workspace/data/lambada_test.jsonl
 - output_path...............: None
 - tokenizer_path............: /workspace/model/bloomz-560m
 - lib_path..................: build/lib/libth_transformer.so
 - test_hf...................: False
 - acc_threshold.............: None
 - show_progress.............: True
 - inference_data_type.......: None
 - weigh
 - int8_
```

```
================================================
[FT][INFO] Load BLOOM model
 - head_num.................: 16
 - size_per_head............: 64
 - layer_num................: 24
 - tensor_para_size.........: 1
 - vocab_size...............: 250880
 - start_id.................: 1
 - end_id...................: 2
 - weights_data_type........: fp16
 - layernorm_eps............: 1e-05
 - inference_data_type......: fp16
 - lib_path.................: build/lib/libth_transformer.so
 - pipeline_para_size.......: 2
 - shared_contexts_ratio....: 1.0
 - int8_mode................: 0
[FT][INFO] Load BLOOM model
 - head_num.................: 16
 - size_per_head............: 64
 - layer_num................: 24
 - tensor_para_size.........: 1
 - vocab_size...............: 250880
 - start_id.................: 1
 - end_id...................: 2
 - weights_data_type........: fp16
 - layernorm_eps............: 1e-05
 - inference_data_type......: fp16
 - lib_path.................: build/lib/libth_transformer.so
 - pipeline_para_size.......: 2
 - shared_contexts_ratio....: 1.0
 - int8_mode................: 0
world_size: 2
world_size: 2
[WARNING] gemm_config.in is not found; using default GEMM algo
[WARNING] gemm_config.in is not found; using default GEMM algo
[FT][INFO] NCCL initialized rank=0 world_size=2 tensor_para=NcclParam[rank=0, world_si
[FT][INFO] NCCL initialized rank=1 world_size=2 tensor_para=NcclParam[rank=0, world_si
[FT][INFO] Device NVIDIA A800 80GB PCIe
[FT][INFO] Device NVIDIA A800 80GB PCIe
100%|██████████| 5153/5153 [01:51<00:00, 46.12it/s] current process id: 47861   Accura
current process id: 47862   Accuracy: 39.4527% (2033/5153) (elapsed time: 102.3391 sec
```

单卡、流水线并行、张量并行对比

下面在BatchSize为1的情况下，对单卡、张量并行、流水线并行进行了简单的测试，仅供参考（由于测试时，有其他训练任务也在运行，可能对结果会产生干扰）。

TP=1、PP=1、BZ=1：

累积响应时长：

```
100%|████████████████████████████████████████████████████████████████████████████████
current process id: 47645   Accuracy: 39.4527% (2033/5153) (elapsed time: 132.2274 sec
```

显存占用：

```
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      8356      C   python                            1740MiB |
+-----------------------------------------------------------------------------+
```

TP=2、PI

累积响应时长：

```
100%|██████████| 5153/5153 [00:35<00:00, 144.80it/s]current process id: 49111   Accura
current process id: 49112   Accuracy: 39.4916% (2035/5153) (elapsed time: 26.5110 sec)
```

显存占用：

```
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    1   N/A  N/A     41339      C   python                           1692MiB |
|    2   N/A  N/A     41340      C   python                           1692MiB |
+-----------------------------------------------------------------------------+
```

TP=1、PP=2、BZ=1：

累积响应时长：

```
100%|██████████| 5153/5153 [00:33<00:00, 153.92it/s]current process id: 48755   Accura
current process id: 48754   Accuracy: 39.4527% (2033/5153) (elapsed time: 24.4391 sec)
```

显存占用：

```
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    1   N/A  N/A      4001      C   python                           1952MiB |
|    2   N/A  N/A      4002      C   python                           1952MiB |
+-----------------------------------------------------------------------------+
```

TP=1、PP=3、BZ=1：

累积响应时长：

```
100%|██████████| 5153/5153 [00:33<00:00, 152.46it/s]current process id: 48220   Accura
100%|██████████| 5153/5153 [00:33<00:00, 153.63it/s]current process id: 48219   Accura
current process id: 48221   Accuracy: 39.4527% (2033/5153) (elapsed time: 24.3489 sec)
```

显存占用：

```
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A     57588      C   python                           1420MiB |
|    1   N/A  N/A     57589      C   python                           1468MiB |
|    2   N/A  N/A     57590      C   python                           1468MiB |
+-----------------------------------------------------------------------------+
```

结语

本文给大家简要介绍了FasterTransformer的基本概念以及如何使用FasterTransformer进行单机及分布式模型推理，希望能够帮助大家快速了解FasterTransformer。

参考文档：

- **Accelerated Inference for Large Transformer Models Using NVIDIA Triton Inference Server**
- **FasterT**

编辑于 2023-08-11 23:28 · IP 属地四川

大模型 推理引擎 人工智能

理性发言，友善互动

15 条评论 默认 最新

胡润

使用v100-32g显卡测试bloom-560m和bloom-3b模型，ft推理的acc均为0，模型输出结果为"model_answer": "-\u00e0-vis","output_ids": [90610]。不知道有没有遇到过类似的情况

2023-07-31 · 广东 回复 1

一杆梅子酒

========== Arguments ==========
- input_dir.....: ../model/bloomz-560m
- output_dir.....: ../model/bloomz-560m-convert
- tensor_para_size.....: 1
- quant_file.....:
- data_type.....: fp16
- processes.....: 1
- verbose.....: True
- by_shard.....: False
==========
loading from safetensors format
model file num: 1
model file path: ../model/bloomz-560m/model.safetensors
begin to process quant file.
Traceback (most recent call last):
File "/root/Mzp/ai_inference/Fastertransformer/examples/pytorch/gpt/utils/huggingface_bloom_convert.py", line 550, in <module>
main()
File "/root/Mzp/ai_inference/Fastertransformer/examples/pytorch/gpt/utils/huggingface_bloom_convert.py", line 542, in main
process_by_model_param(args.input_dir, dtype, tp_size, save_dir, args.quant_file, args.processes)
File "/root/Mzp/ai_inference/Fastertransformer/examples/pytorch/gpt/utils/huggingface_bloom_convert.py", line 422, in process_by_model_param
_process_quant_file(model_config, Path(quant_file), dtype, tp_size, save_dir)
File "/root/Mzp/ai_inference/Fastertransformer/examples/pytorch/gpt/utils/huggingface_bloom_convert.py", line 477, in _process_quant_file
state_dict = torch.load(model_file, map_location="cpu")
^^
File "/root/miniconda3/envs/llamatest/lib/python3.12/site-packages/torch/serialization.py", line 997, in load
with _open_file_like(f, 'rb') as opened_file:
^^^^^^^^^^^^^^^^^^^^^^^^^^
File "/root/miniconda3/envs/llamatest/lib/python3.12/site-packages/torch/serialization.py", line 444, in _open_file_like
return _open_file(name_or_buffer, mode)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "/root/miniconda3/envs/llamatest/lib/python3.12/site-packages/torch/serialization.py", line 425, in __init__
super().__init__(open(name, mode))
^^^^^^^^^^^^^^^^^^
IsADirectoryError: [Errno 21] Is a directory: '.' 这是哪里的问题呀 各位佬。 就是模型转换的时候

08-19 · 浙江 回复 喜欢

nobody

用[redacted]

20[redacted]

DigmonSeeker

想请问跑FT多卡推理的时候抱这个错怎么解决啊，目前没有找到方案

2023-08-25 · 北京 回复 喜欢

一杆梅子酒

大哥单卡跑通了吗 Traceback (most recent call last):
File "examples/pytorch/gpt/bloom_lambada.py", line 409, in <module>
main()
File "/usr/local/lib/python3.8/dist-packages/torch/autograd/grad_mode.py", line 27, in decorate_context
return func(*args, **kwargs)
File "examples/pytorch/gpt/bloom_lambada.py", line 304, in main
model, tokenizer = get_model_and_tokenizer(args)
File "examples/pytorch/gpt/bloom_lambada.py", line 264, in get_model_and_tokenizer
model = bloom.Bloom(**model_args)
File "/root/Mzp/ai_inference/Fastertransformer/examples/pytorch/gpt/utils/bloom.py", line 245, in __init__
super().__init__(
File "/root/Mzp/ai_inference/Fastertransformer/examples/pytorch/gpt/utils/gpt.py", line 528, in __init__
self.rank = dist.get_rank()
File "/usr/local/lib/python3.8/dist-packages/torch/distributed/distributed_c10d.py", line 1044, in get_rank
default_pg = _get_default_group()
File "/usr/local/lib/python3.8/dist-packages/torch/distributed/distributed_c10d.py", line 584, in _get_default_group
raise RuntimeError(
RuntimeError: Default process group has not been initialized, please make sure to call init_process_group 我报错这个。 我有八个a30，但是我只想跑一个卡

08-22 · 浙江 回复 喜欢

一杆梅子酒 ‣ **编号89757**

大哥 帮我看看:Traceback (most recent call last):
File "examples/pytorch/gpt/bloom_lambada.py", line 409, in <module>
main()
File "/usr/local/lib/python3.8/dist-packages/torch/autograd/grad_mode.py", line 27, in decorate_context
return func(*args, **kwargs)
File "examples/pytorch/gpt/bloom_lambada.py", line 304, in main
model, tokenizer = get_model_and_tokenizer(args)
File "examples/pytorch/gpt/bloom_lambada.py", line 264, in get_model_and_tokenizer
model = bloom.Bloom(**model_args)
File "/root/Mzp/ai_inference/Fastertransformer/examples/pytorch/gpt/utils/bloom.py", line 245, in __init__
super().__init__(
File "/root/Mzp/ai_inference/Fastertransformer/examples/pytorch/gpt/utils/gpt.py", line 528, in __init__
self.rank = dist.get_rank()
File "/usr/local/lib/python3.8/dist-packages/torch/distributed/distributed_c10d.py", line 1044, in get_rank
default_pg = _get_default_group()
File "/usr/local/lib/python3.8/dist-packages/torch/distributed/distributed_c10d.py", line 584, in _get_default_group
raise RuntimeError(
RuntimeError: Default process group has not been initialized, please make sure to call init_process_group 我报错这个。 我有八个a30，但是我只想跑一个卡 这个问题在哪呢

08-22 · 浙江 回复 喜欢

这是个名字
请问一下下，执行的时候可以指定使用哪几张卡么
2023-08-10 · 广东 回复 喜欢

rot.cx
虽然需要一些额外的内存使用，但 FT 可以节省重新计算的成本

访存慢 还是 计算慢?
2023-07-26 · 北京 回复 喜欢

乐天
大佬，流水线并行为啥用的是单个GPU转化的FT模型，但是可以加载到多个GPU上?
2023-06-13 · 上海 回复 喜欢

inccc
请问一下，如果要把FasterTransformer打包成whl文件可以调用，要怎么操作？官方文档里貌似没有说明。
2023-06-12 · 浙江 回复 喜欢

Infi-zc
问下大佬，对比 1,3 ， pp = 2 时为啥内存占用还变高了呢？pp = 3 时，内存占用又小了，这个显示的不是 max_memory_usage 吧?
2023-06-09 · 上海 回复 喜欢

Infi-zc ‣ 吃果冻不吐果冻皮
2023-06-09 · 上海 回复 喜欢

吃果冻不吐果冻皮 作者
这应该是截图的时机不一致导致的，不是显示的峰值，还有测试的模型本身相对不大，所以缓存这些可能也有影响。
2023-06-09 · 四川 回复 喜欢

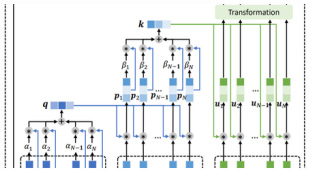
aake
写得不错👍
2023-06-08 · 浙江 回复 喜欢

理性发言，友善互动

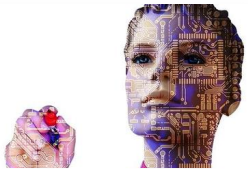
AI工程（MLOps）
AI System/MLOps

动手学大模型

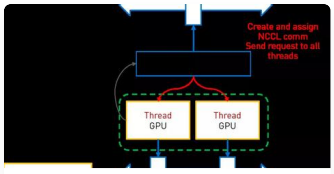
推荐阅读



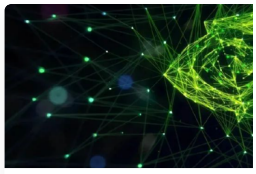
Fastformer-又简单又好用的Transformer变体！清华...
我爱计算机... 发表于我爱计算机...



微信也在用的Transformer加速推理工具，现在腾讯开源了
量子位 发表于量子位



最新 FasterTransformer 4.0 发布，全球首次支持 GPT-3 ...
NVIDIA英伟达中国



使用 FasterTransforme Triton 推理服务器加速大
NVIDIA英伟达中国