# 一、fastertransformer改造

## 准备镜像（1周）

花费很大力气，装环境，最后发现镜像系统版本低不能装高版本libc

无奈只能更换镜像，由于内网各种限制，只能在外网拉好镜像直接push，在此过程中反复准备环境推镜像耗费了很大精力且浪费时间

## 安装FasterTransformer环境（2天）

根据FasterTransformer的教程，拉取git代码，编译等。。

发现教程中的大坑，前面让使用TensorFlow的镜像，后面跑pytorch的推理。。

仔细研究文档，重新配置pytorch环境，安装完成

安装完成后进行cmake编译c++库，cmake文件中又需要拉git代码？？检查后发现是测试代码，直接注释掉

## 安装mpi环境（2天）

编译后执行发现缺少mpi环境，重新安装mpi环境，并配置环境变量

root@a22466da80eb:/opt/hpcx/ompi/bin# echo $PATH

/usr/local/nvm/versions/node/v16.19.1/bin:/usr/local/lib/python3.8/dist-packages/torch\_tensorrt/bin:/usr/local/mpi/bin:/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/ucx/bin:/opt/tensorrt/bin

root@a22466da80eb:/opt/hpcx/ompi/bin# echo $LD\_LIBRARY\_PATH

/usr/local/lib/python3.8/dist-packages/torch/lib:/usr/local/lib/python3.8/dist-packages/torch\_tensorrt/lib:/usr/local/cuda/compat/lib:/usr/local/nvidia/lib:/usr/local/nvidia/lib64:/usr/local/cuda-11/lib64

执行指令

export PATH=/usr/local/nvm/versions/node/v16.19.1/bin:/usr/local/lib/python3.8/dist-packages/torch\_tensorrt/bin:/usr/local/mpi/bin:/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/ucx/bin:/opt/tensorrt/bin

export LD\_LIBRARY\_PATH=/usr/local/lib/python3.8/dist-packages/torch/lib:/usr/local/lib/python3.8/dist-packages/torch\_tensorrt/lib:/usr/local/cuda/compat/lib:/usr/local/nvidia/lib:/usr/local/nvidia/lib64:/usr/local/cuda-11/lib64

## 调整FasterTransformer源码，跑通推理（>1周）

由于FasterTransformer不支持chatglm-6b，因此对于FasterTransformer中的gpt推理代码需要进行一定的调整后才可以正常运行

FasterTransformer中包含python、c++、cuda三部分代码，全部需要根据自研模型推理过程进行调整，研读FasterTransformer和transformers库源码耗费大量时间

## 精度对齐（很久）

### contextdecoder 设置乘7，decoder不乘

token: [129113 129113 129113 129113 129113 129113 129113 129113 129113 129113

129113 129113 129113 129113 129113 129113 129113 129113 129113 129113

129113 129113 129113 129113 129113 129113 129113 129113 129113 129113

129113 129113]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到追到

### contextdecoder 设置不乘，decoder不乘

token: [ 28376 83831 98786 20005 20005 20005 20263 20263 20263 20263

20007 20005 87725 20005 20004 20263 20263 106100 20005 116292

20263 45444 84260 28242 20263 20263 20005 87637 92580 85063

20119 20263]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

preserve与接下来的 TTTT. 的选择

TT标明 互利Tipper城市mosTT 蜂蜜当地的兴tT

### contextdecoder 设置不乘，decoder乘7

token: [115520 20035 20298 20151 86764 90458 86764 120881 20393 86527

94204 86527 90047 20649 85897 86527 86527 105596 20515 48637

20151 87208 48637 97555 20312 86527 20151 20418 86903 20787

32505 86527]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

湿了! Qn彼此扭彼此的那份 c互相傍晚互相五大 g相互互相互相和时间 GMILn碗MIL的组合 right互相n m的日子 ksided互相

### 调整lm\_head前的layernorm

注：contextdecoder 设置不乘，decoder乘7

rank: 0

token: [100505 86319 100129 20298 20298 21865 89802 20169 20298 84412

36054 84345 83836 24097 83886 21865 86556 24563 20018 20005

84784 20222 20354 106777 23638 20298 89157 31749 29687 85851

86287 20298]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

迅顿不放 Q Qce危c Q教nate定对cdot其ce刻CA9 拍BN修长fn Q候PKila弹残 Q

### 设置contextdecoder 设置不乘，decoder不乘

type: <class 'torch.Tensor'>,

token: [107541 20017 20059 20005 20005 20059 20106 20106 20005 20263

108568 20004 116292 20319 23270 20263 20005 20234 106916 138875

20263 116292 106916 99169 20005 20263 20119 93346 21438 20017

106916 20263]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

是不可"... ...ss T自习

互利 TptT D双方都出息T互利双方都所示 Tt便可 host"双方都T

### 发现FasterTransformer没有做rotary embedding，加rotary

token: [107541 20005 91413 86298 20005 91413 113264 84174 113264 113264

102490 52587 116806 140160 20263 142823 121086 133031 133602 133589

52587 23947 116806 86599 118003 84980 145731 48420 127385 23573

24659 20005]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

是不可 附近的轴 附近的内战影响内战内战暖心icker还特别辕T在此过程中权和选择适合自己的就完了记者注意到ickerTV还特别袋会认为堂..."Explain每天喝dycy

### 调整attention输入

token: [107541 20340 92671 20005 20005 20004 20005 20004 20005 20005

88815 21004 85623 93314 23270 20005 20004 87888 93314 87702

20004 83868 21004 85280 20005 20004 20005 87702 87888 85280

21004 20004]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

是不可M的前

寻求by寻找自制pt

谱自制赢

及by复

赢谱复by

### 发现原版自注意力有乘一个layer+1，增加layer+1

token: [107541 118151 84004 20005 20005 90738 21486 83972 20005 85280

84699 85072 92431 20119 84070 20004 93391 20929 114591 20013

20119 84135 20119 88152 22279 99714 112341 42564 20119 23270

20172 20008]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

是不可其事发展 图像em名 复权招坐着t光

梯de例行3t达t反馈ty锚得来tourtpt who0

### contextdecoder 设置不乘，decoder乘7

token: [ 29405 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

xy

### contextdecoder 设置乘7，decoder乘7

token: [150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

### 打印FFN（mlp）中的计算行列数

std::cout << "inter size: " << inter\_size\_ << ", m: " << m << ", hidden-uints\_: " << hidden\_units\_ << std::endl;

inter size: 16384, m: 1, hidden-uints\_: 4096

the 2nd inter size: 16384, m: 1, hidden-uints\_: 4096

### FFN（mlp）中的计算行列数计算，检查，未修改

1-> weight 4096 \* 16384

input n \* 4096

mul n\*4096 X 4096\*16384

m: 16384, n: 1, k: 4096

A = m \* k, B = k \* n

A: weght

B:input

m: 4096, n: 1, k: 16384

m:150528, n: 1, k: 4096, hidden\_units\_offset:0, vocab\_size\_units\_offset:0

std::cout << "m:" << vocab\_size\_padded\_ << ", n: " << local\_batch\_size \* beam\_width << ", k: " << hidden\_units\_ << ", hidden\_units\_offset:" << hidden\_units\_offset << ", vocab\_size\_units\_offset:" << vocab\_size\_units\_offset << std::endl;

                                          vocab\_size\_padded\_,  // n

                                          local\_batch\_size \* beam\_width,

                                          hidden\_units\_,  // k

                                          &alpha,

                                          padded\_embedding\_kernel\_ptr\_,

                                          gemm\_data\_type,

                                          hidden\_units\_,                                   // k

                                          decoder\_output\_final\_buf + hidden\_units\_offset,  // OPT: no final layer norm

150528 \* 4096 X 4096 \* 1

### 修改lm\_head中对A矩阵的转置配置

直接报错

### contextdecoder 设置不乘，decoder乘7

type: <class 'torch.Tensor'>,

token: [100505 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

迅

### contextdecoder 设置不乘，decoder不乘

token: [107541 118151 84004 20005 20005 90738 21486 83972 20005 85280

84699 85072 92431 20119 84070 20004 93391 20929 114591 20013

20119 84135 20119 88152 22279 99714 112341 42564 20119 23270

20172 20008]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

是不可其事发展 图像em名 复权招坐着t光

梯de例行3t达t反馈ty锚得来tourtpt who0

### beam width =2

token: [87707 20005 20005 20004 20005 20005 20005 20005 20005 83837 20004 20004

20004 20004 20005 20008 20004 20004 20005 20005 20005 20005 20119 20005

20005 20004 20005 20005 20005 20005 20004 20005]

[INFO] batch 0, beam 1:

[Context]

你好

[Output]

闪

月

0

t

### beam width=1, contextdecoder 设置乘7，decoder乘7

token: [ 20039 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001]

[INFO] batch 0, beam 1:

[Context]

你好

[Output]

}

### 尝试把layerid传入selfattention，调整query\_key\_layer\_scaling\_coeff

then the loop 1-13 start..

m:150528, n: 1, k: 4096, hidden\_units\_offset:0, vocab\_size\_units\_offset:0

input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-output123456789: Й▒-output123456789: Й▒-then the sec 2 start..

here finish..

forward success !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

generate cost: 2.2622804641723633

rank: -1

type: <class 'torch.Tensor'>,

token: [150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

### 尝试把layerid传入selfattention，调整query\_key\_layer\_scaling\_coeff与python版本一致

then the loop 1-13 start..

m:150528, n: 1, k: 4096, hidden\_units\_offset:0, vocab\_size\_units\_offset:0

input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-input123456789: Й▒-output123456789: Й▒-output123456789: Й▒-then the sec 2 start..

here finish..

forward success !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

generate cost: 2.2622804641723633

rank: -1

type: <class 'torch.Tensor'>,

token: [150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

### 尝试修改gptdecoder为contextdecoder

不对，放弃

### 发现权重值与原版本不一致，尝试重新导出，检查两次导出是否一致

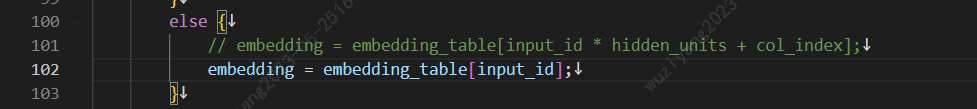
发现导出权重与原版本权重一致。。

### 验证使用导出的权重进行推理，结果是否会变化

未变化。。

### 修改wordembedding取值下标

src/ft/kernels/gpt\_kernel.cu



token: [ 83849 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001 150001 150001 150001 150001 150001 150001 150001 150001

150001 150001]

[INFO] batch 0, beam 0:

[Context]

你好

[Output]

就

### 细查cache内容

FT kv的shape：

self\_k\_cache\_shape: (28,1,32,16,36,8)

self\_v\_cache\_shape: (28,1,32,36,128)

glm6b的kv的shape：

28 \* [4, 1, 32, 128]

...

28 \* [29, 1, 32, 128]

### 调整第一次decode过程

1.input\_layernorm计算似乎不对劲，查看residual，在FT中residual1表示输入（无修改）

2.input\_layernorm对方差的计算似乎不对劲，调整方差计算方式

3.input\_layernorm的grid和block检查

## 输出层层对齐

#### 对齐方法

数据GPU转CPU，然后打印，需要注意变量类型和数据长度

                int input\_cpu[max\_input\_length];

                cudaMemcpy(input\_cpu, tiled\_input\_ids\_buf\_, max\_input\_length \* sizeof(int), cudaMemcpyDeviceToHost);

                //  20005, 94874, 150001, 150004

                std::cout << "input id:";

                for (int i = 0; i < max\_input\_length; i++)

                    std::cout << ", " << input\_cpu[i];

                std::cout << std::endl;

#### 1.input ids

匹配

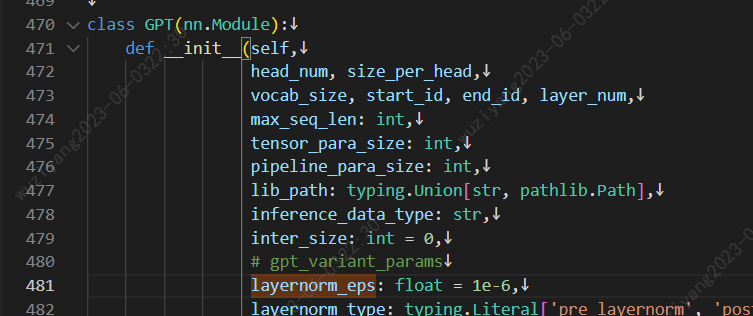
#### 2.第一次word\_embedding结果

之前做了错误的修改，改回之后正确

#### 3.第一次decode（困难重重）

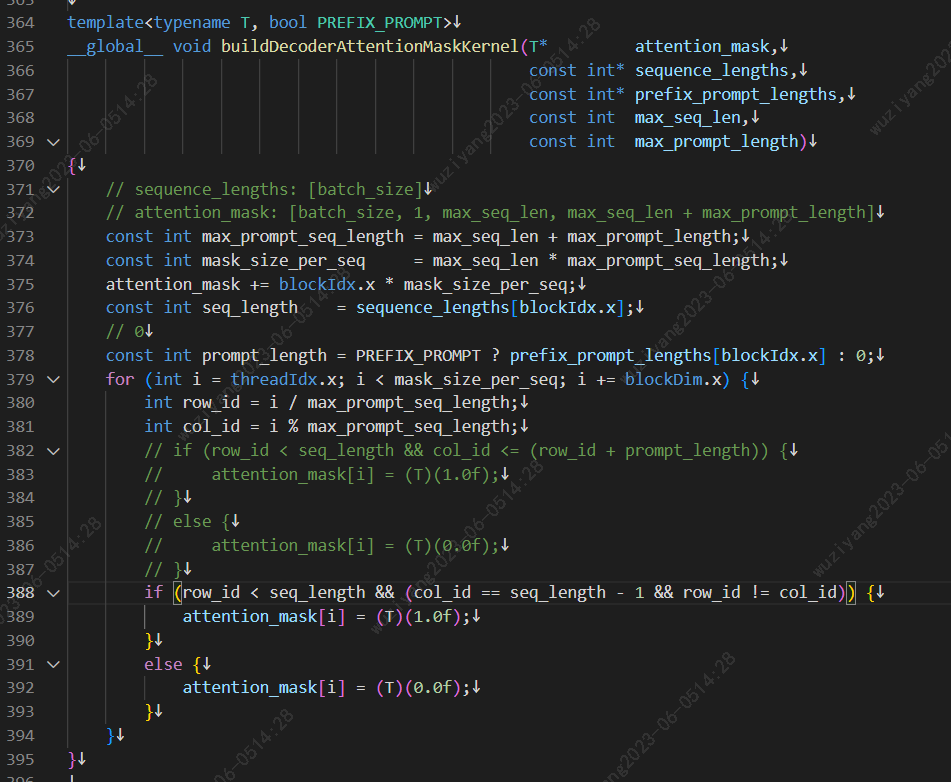
##### 3.1 layernorm\_eps

**发现layernorm\_eps 参数不对，调整gpt.py，input\_layernorm的eps应为1e-5，调整后layernorm依然未完全对齐，但误差较小，暂时略过**



##### 3.2 attention mask

**发现输入的attention mask值不对，调整kernels/gpt\_kernels.cu中mask计算方法**(chatglm对于mask的计算方法为：输入所有id mask最后一个id，最后一个id不mask)



qkv ln层：weight与bias匹配

##### 3.3 qkv矩阵乘

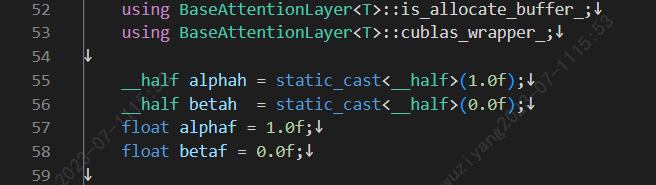
**qkv矩阵乘结果不正确，调试（4天）：**

对cuda矩阵乘有了一定理解，发现错误原因是部分参数变量类型与权重不匹配导致（fp16）

原始代码可能是因为没有配置对应参数，导致默认参数不正确，修改/layers/attention\_layers/GptContextAttentionLayer.cc如下：



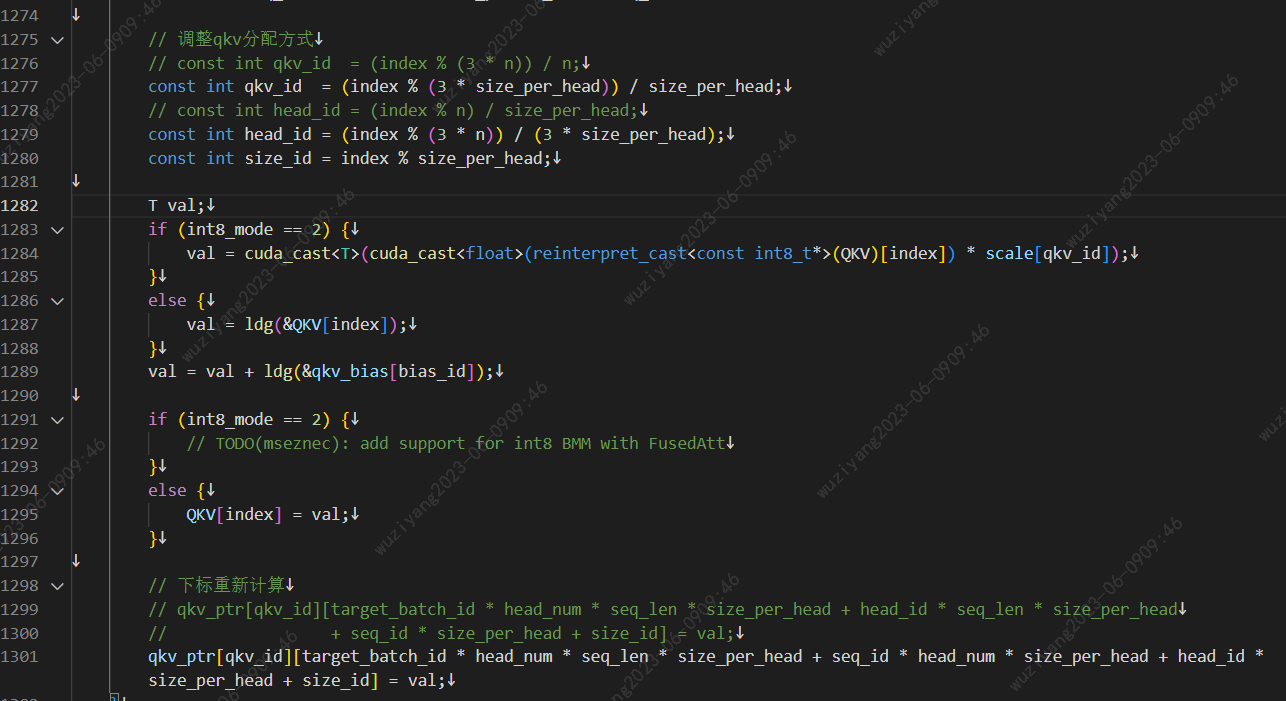
由于矩阵乘中变量类型为half，需要在头文件/layers/attention\_layers/GptContextAttentionLayer.h定义half类型的参数：



##### 3.4 qkv split

**qkv split过程未对齐**

qkv矩阵乘的结果size为12288 \* 4096，但是qkv分布是[[128, 128, 128], [128, 128, 128]... ]格式，因此原cuda对于qkv分布时计算的id不正确，导致数据split错误。修改/kernels/unfused\_attention\_kernels.cu如下：



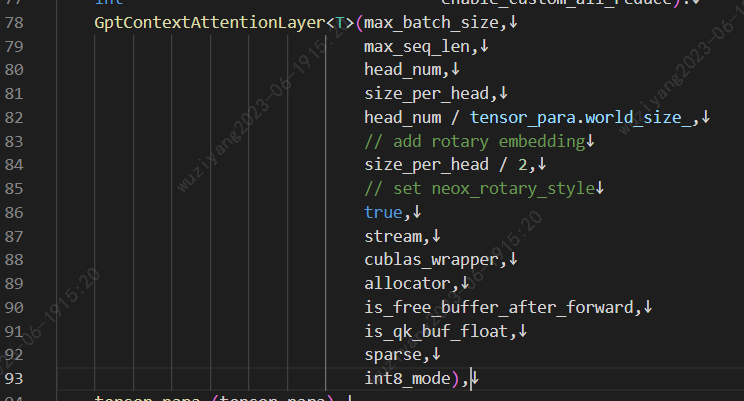
##### 3.5 rotary embedding

**rotary embedding未对齐**

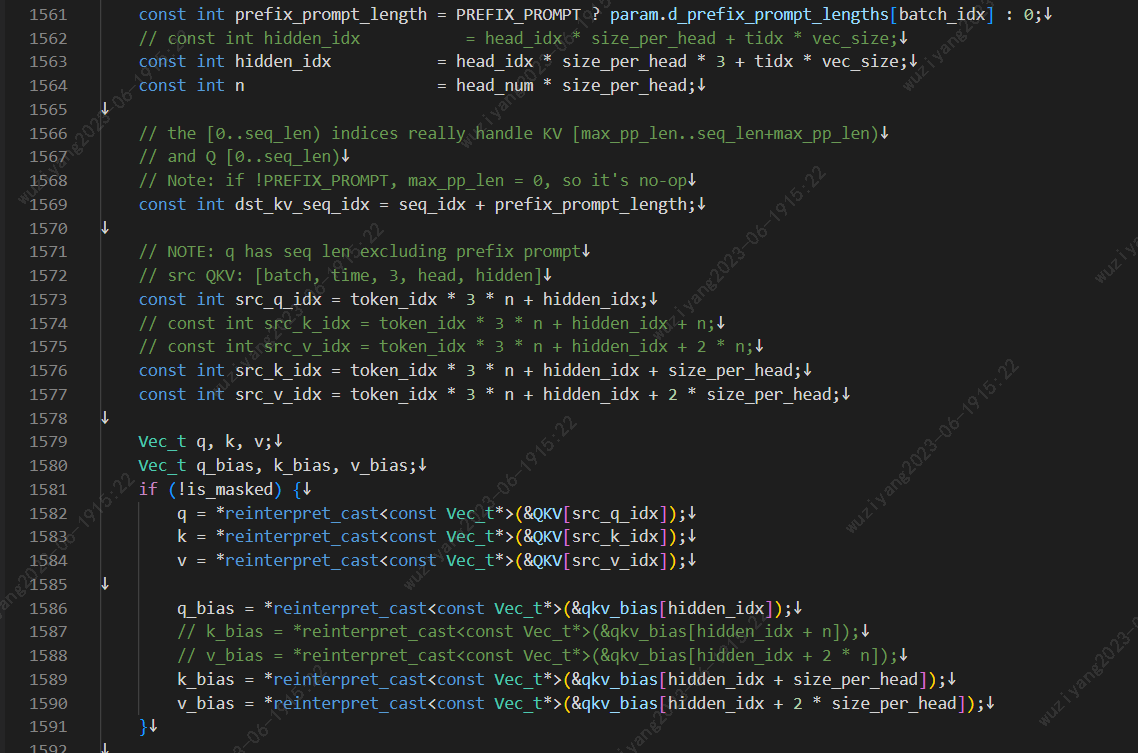
**原方法路径没做rotary embedding，调整rotary embedding参数，进入另一路径，重新调整qkv split和rotary embedding计算**

修改内容：

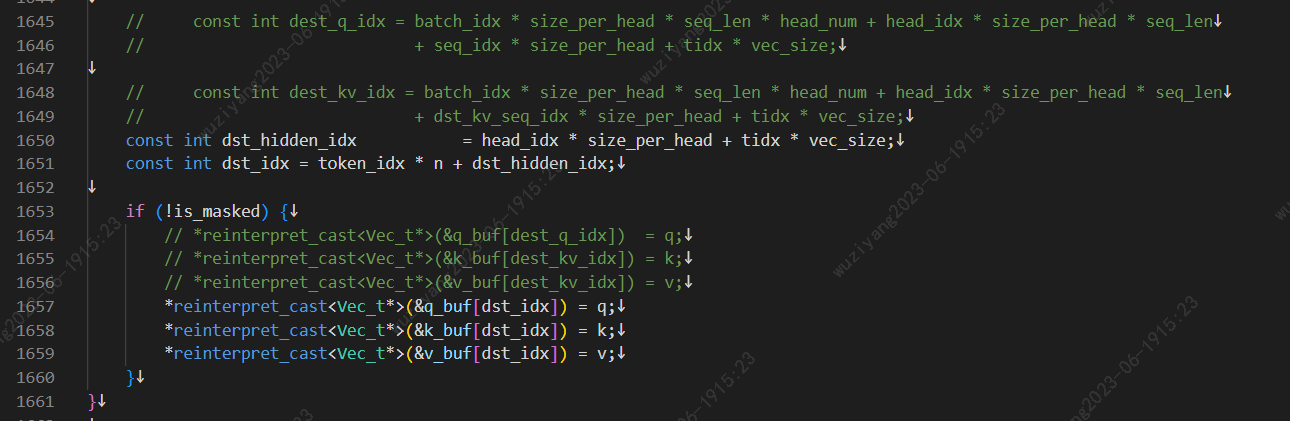
1. /layers/attention\_layers/TensorParallelGptContextAttentionLayer.cc，调整类初始化方法，增加两个参数，使其推理时进入rotary embedding路径



2.调整/kernels/unfused\_attention\_kernels.cu，调整cuda计算中id计算方式，使qkv计算与取值符合正确顺序



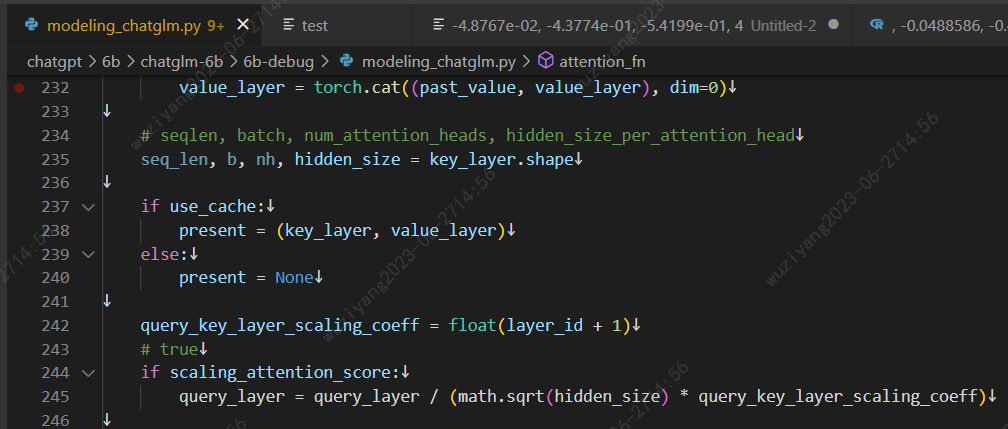
3.调整/kernels/unfused\_attention\_kernels.cu，调整结果qkv的存储id计算方式，使存储结果顺序正确



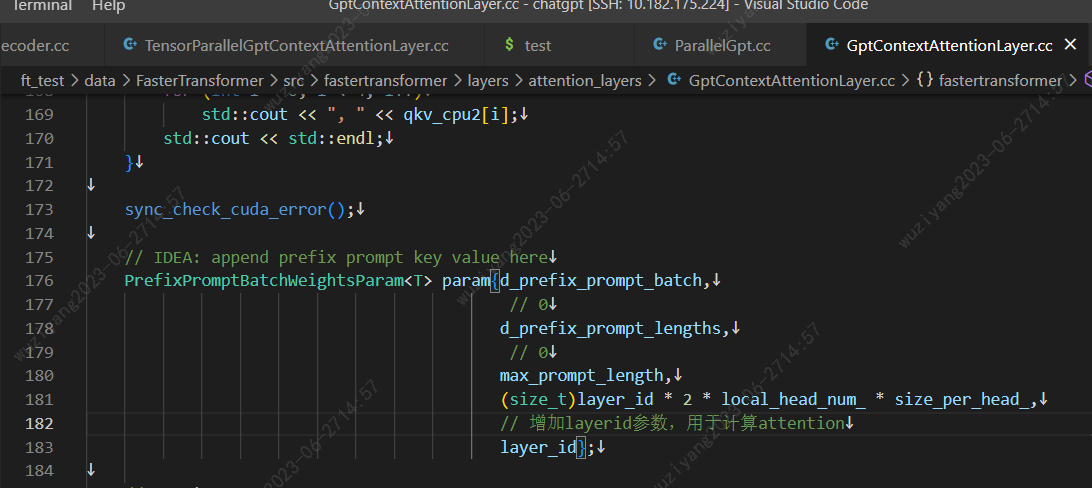
##### 3.6 attention计算

**attention计算未对齐**

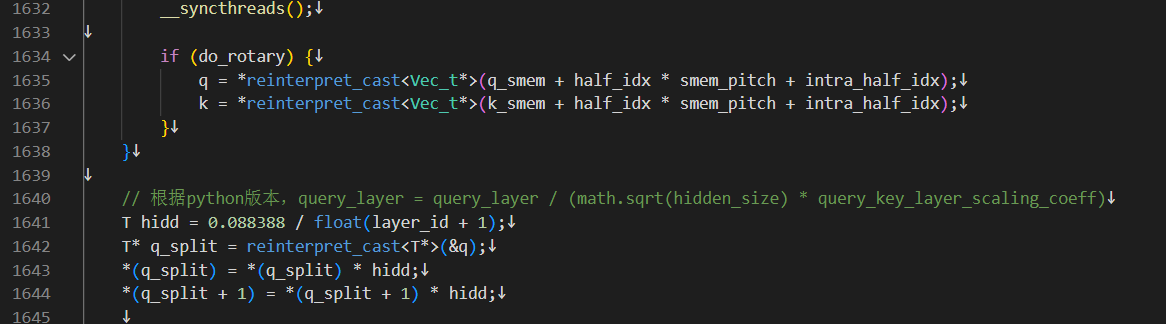
attention计算中，需要除以layerid和hidden（如下图所示），FasterTransformer没有提供此功能，需手动添加；另外在qkv注意力计算过程中，还需要一个softmax过程和最终的shape调整



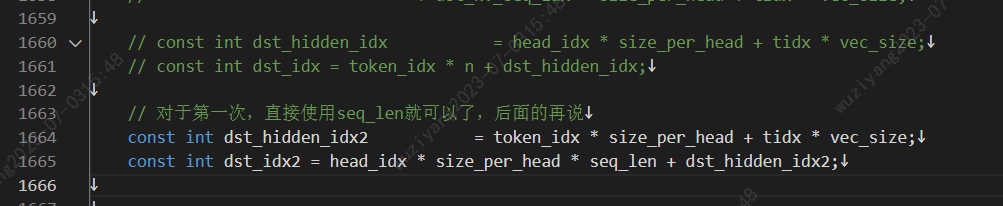
1.传入layerid参数，调整/layers/attention\_layers/GptContextAttentionLayer.cc，增加传入layerid



2.增加除以layerid和hidden，调整/kernels/unfused\_attention\_kernels.cu



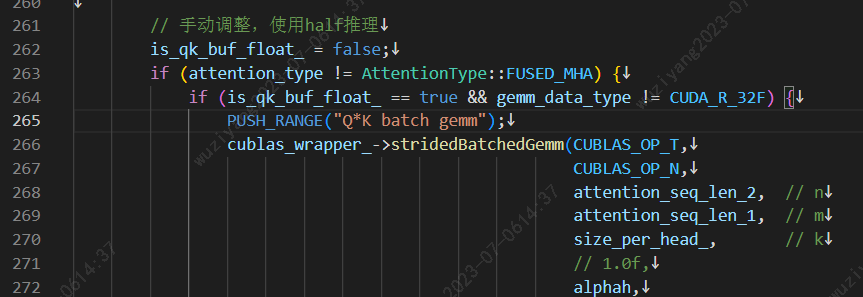
3.由于前面rotary embedding中结果存储格式为n \* 32 \* 128，不适合后续多头注意力矩阵乘计算，因此调整结果存储格式为 32 \* n \* 128，修改/kernels/unfused\_attention\_kernels.cu

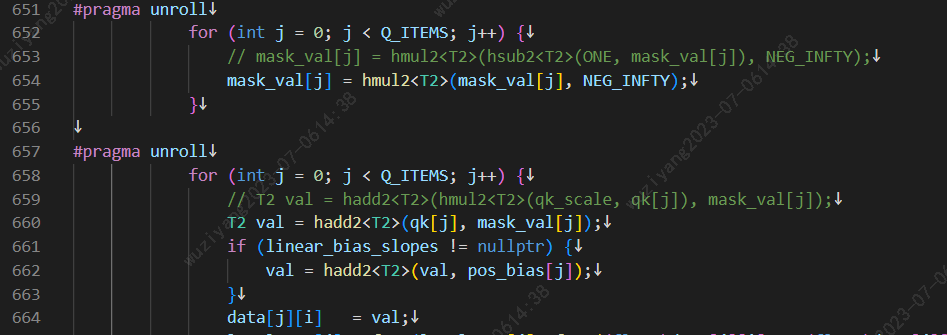


4.调整多头注意力矩阵乘参数，修改/layers/attention\_layers/GptContextAttentionLayer.cc



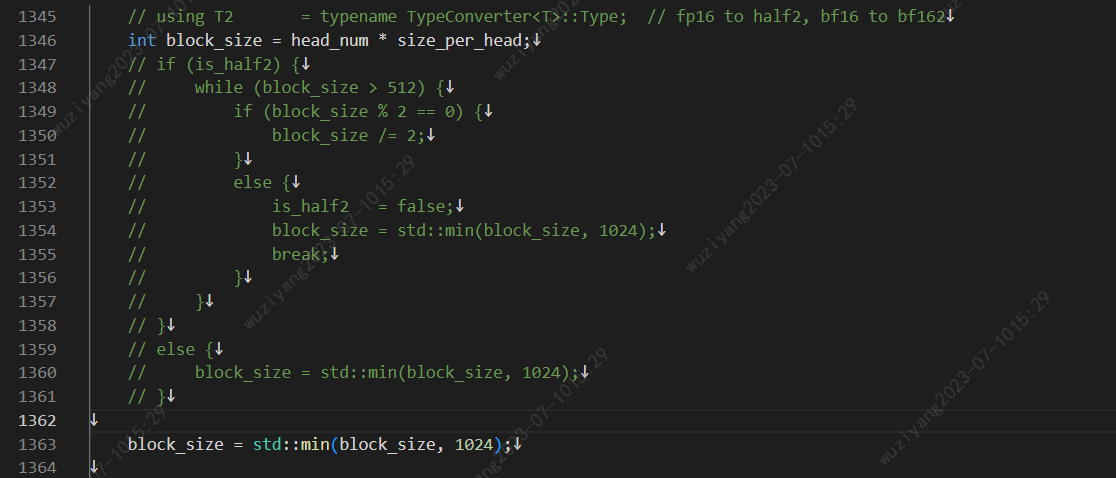
5.attention后的softmax计算有问题，原路径使用float进行计算，导致结果错误；另在softmax中，对mask计算有误，对scale计算需取消，分别修改/layers/attention\_layers/GptContextAttentionLayer.cc和/kernels/unfused\_attention\_kernels.cu

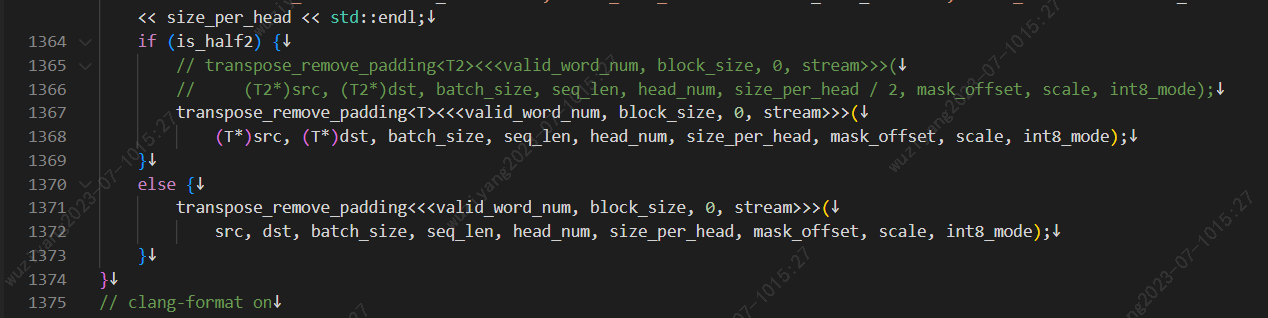




6.softmax后的结果顺序需要重新调整，修改/kernels/unfused\_attention\_kernels.cu，另外此函数使用的变量是T2类型，不适用于当前情况，调整为T

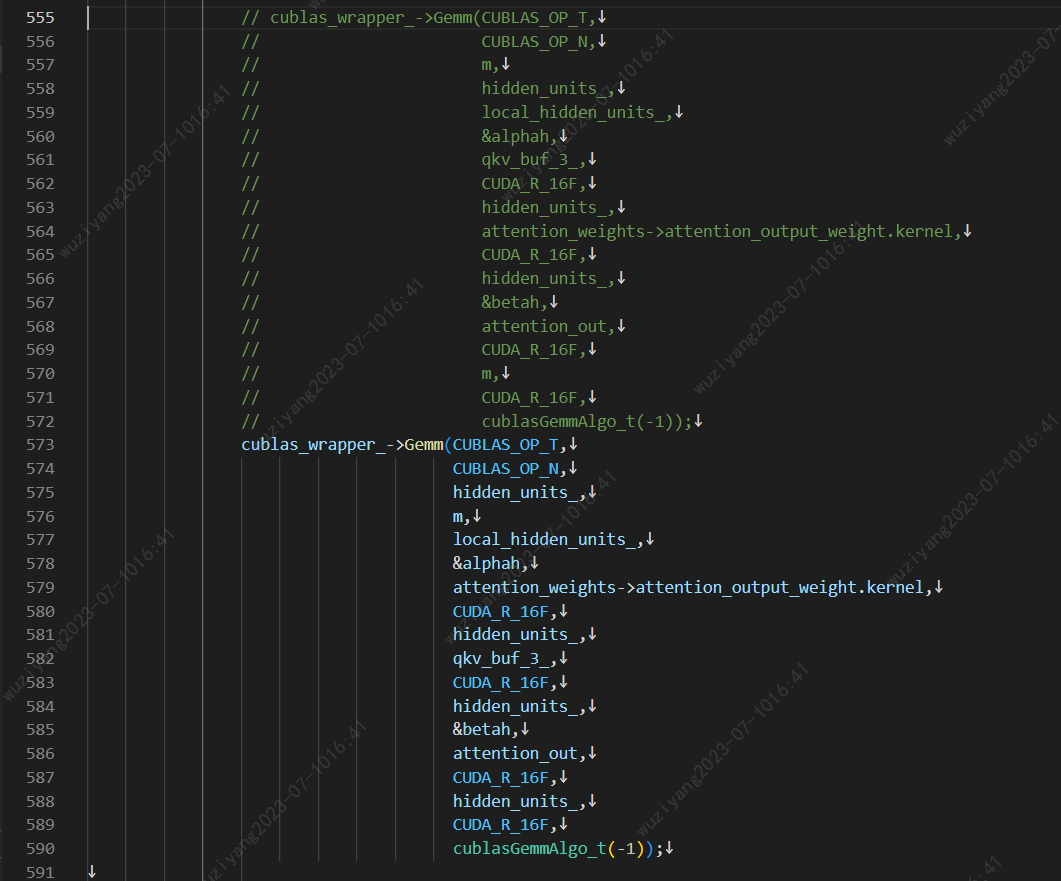






##### 3.7 dense矩阵乘

**dense矩阵乘法未对齐，终于到了较为简单的领域，参考qkv矩阵乘，简单调整下即可，修改layers/attention\_layers/GptContextAttentionLayer.cc**

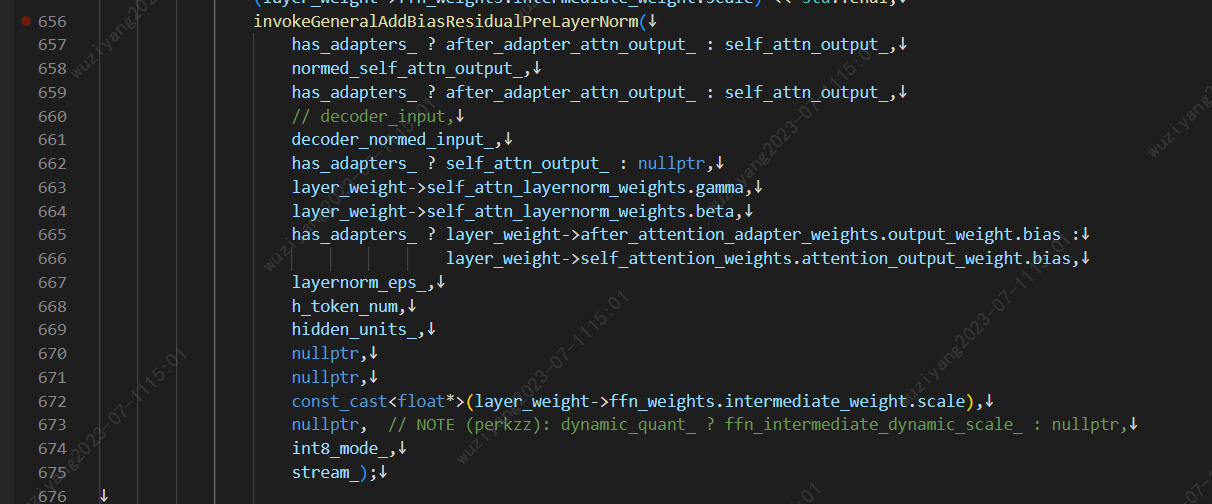


##### 3.8残差连接

残差连接计算中，FasterTransformer未对input做权重增强，另函数输入需修改为norm后的input，修改/kernels/layernorm\_kernels.cu文件如下：



修改models/multi\_gpu\_gpt/ParallelGptContextDecoder.cc文件如下：



##### 3.9 mlp

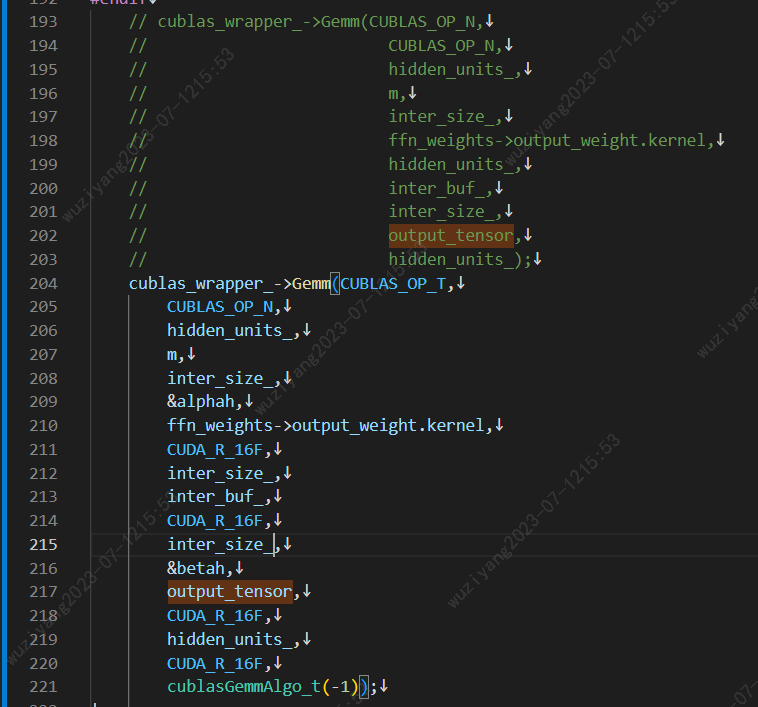
mlp部分包括两个矩阵乘和一个gelu激活，矩阵乘调整方式同上

1.第一个矩阵乘调整/layers/FfnLayer.cc：



2.gelu激活，不需要调整

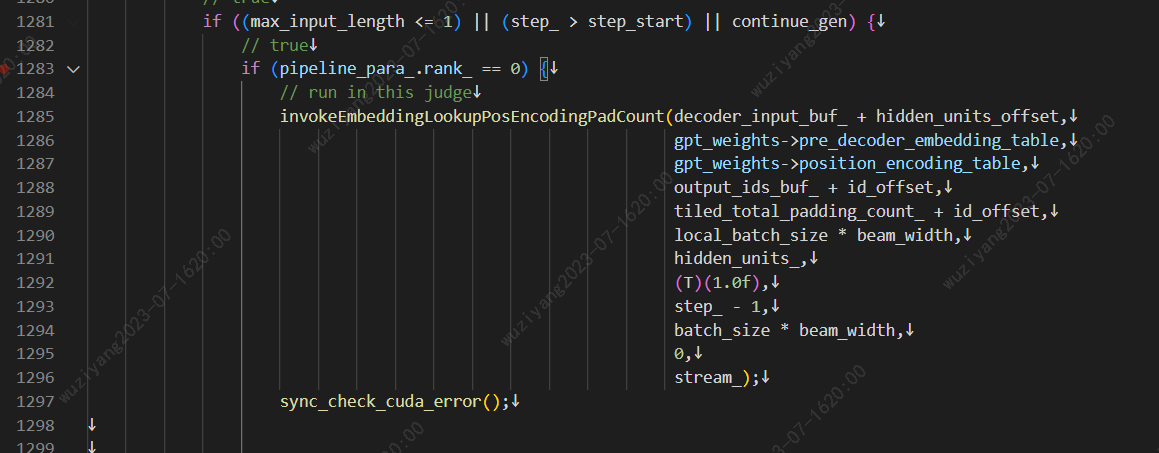
3.第二个矩阵乘，类似修改/layers/FfnLayer.cc：



#### 4.第二轮计算

完成第一轮的对齐后，发现token2结果依然未对齐，继续调查

##### 4.1word embedding

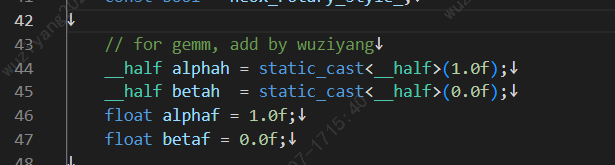
word embedding部分，由于上一个token是正确的，所以结果也是正确的，见/models/multi\_gpu\_gpt/ParallelGpt.cc

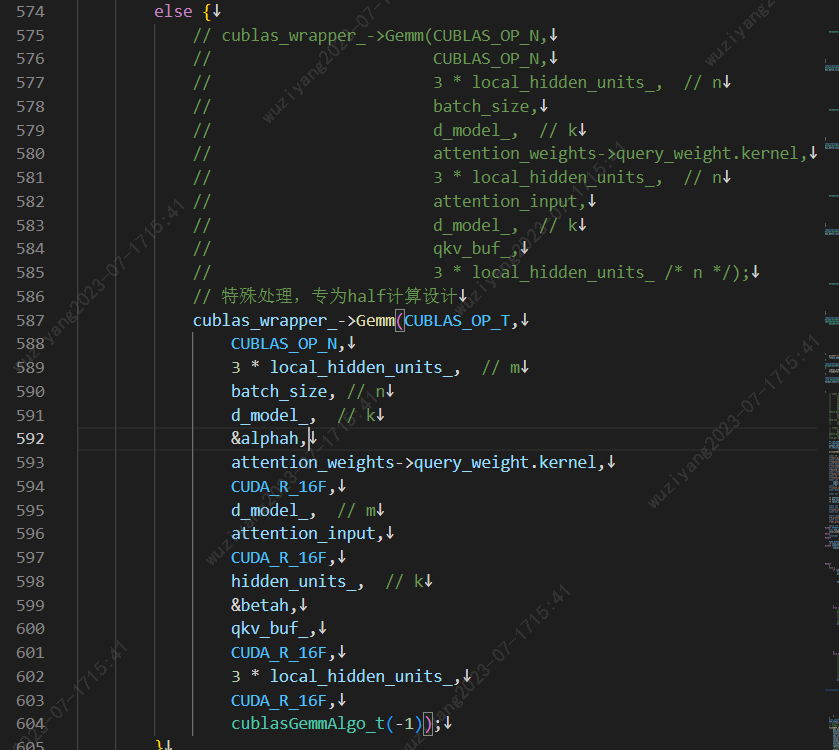
##### 4.2 layernorm

同第一轮，有误差，但基本对齐，暂时略过

##### 4.3 qkv矩阵乘

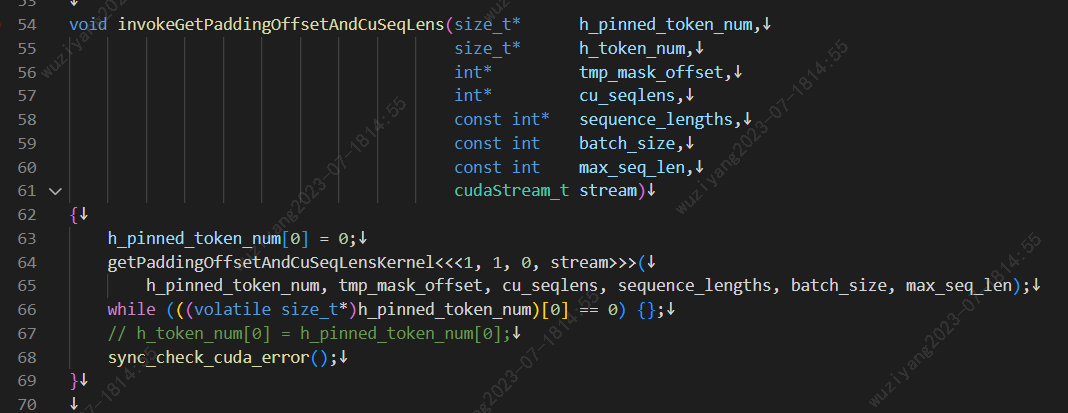
同第一轮，先增加half类型参数，修改/layers/attention\_layers/DecoderSelfAttentionLayer.h；再修改/layers/attention\_layers/DecoderSelfAttentionLayer.cc





##### 4.4 转换思路，使用上一轮的函数修改

修改ft\_test/data/FasterTransformer/src/fastertransformer/kernels/bert\_preprocess\_kernels.cu

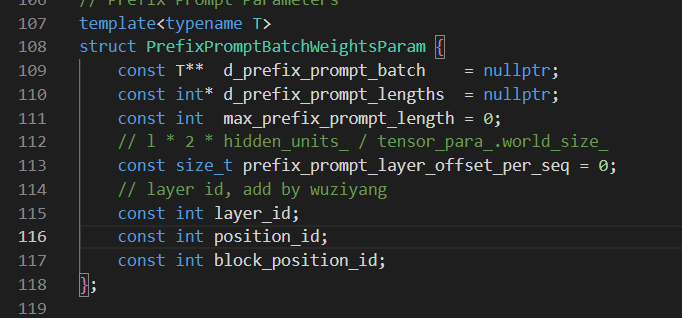


##### 4.5 发现上一轮中的unfused\_attention计算有问题

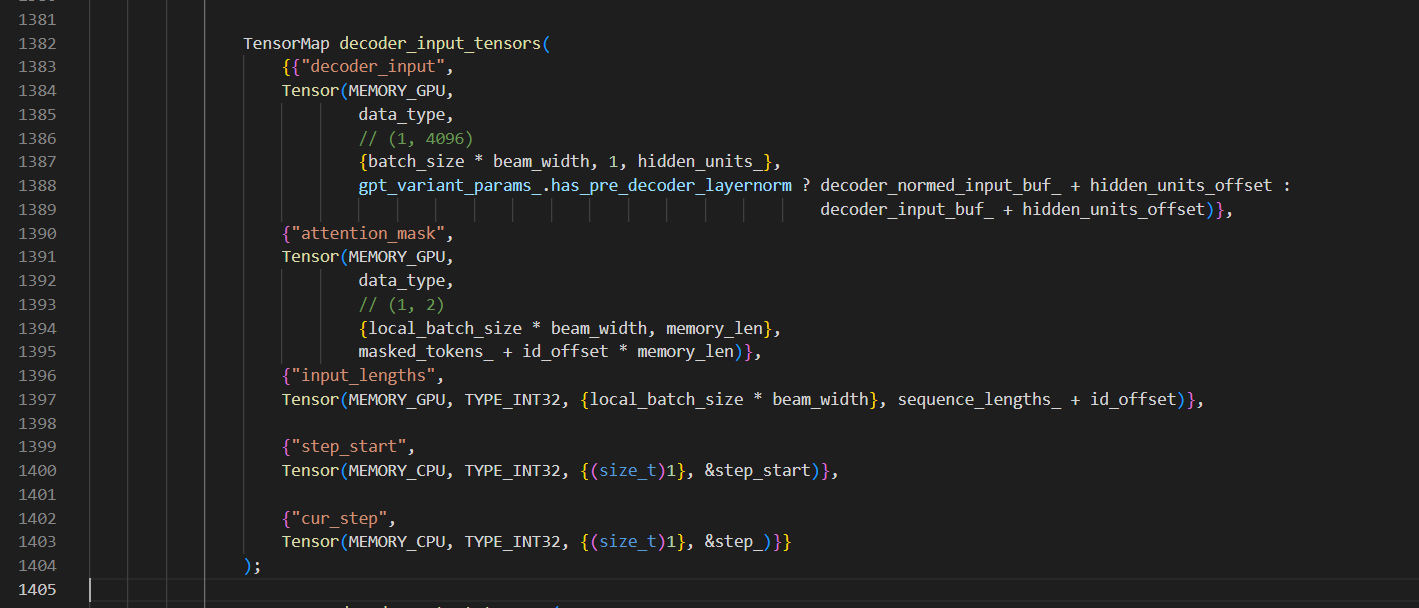
**原python代码中attention计算，对于qkv，应将qk进行split，得到前后两部分[n,32,128] -> 2 \* [n,32,64]，然后对两部分分别做position的rotary embedding，但是发现ft中只对前一部分做了embedding，未对后一部分做，且position不合适，进行调整**

1.增加position id和block position id

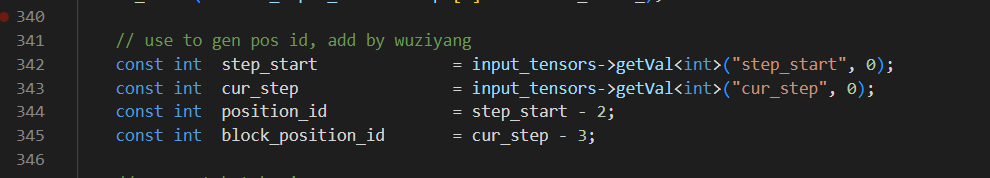
修改/kernels/unfused\_attention\_kernels.h

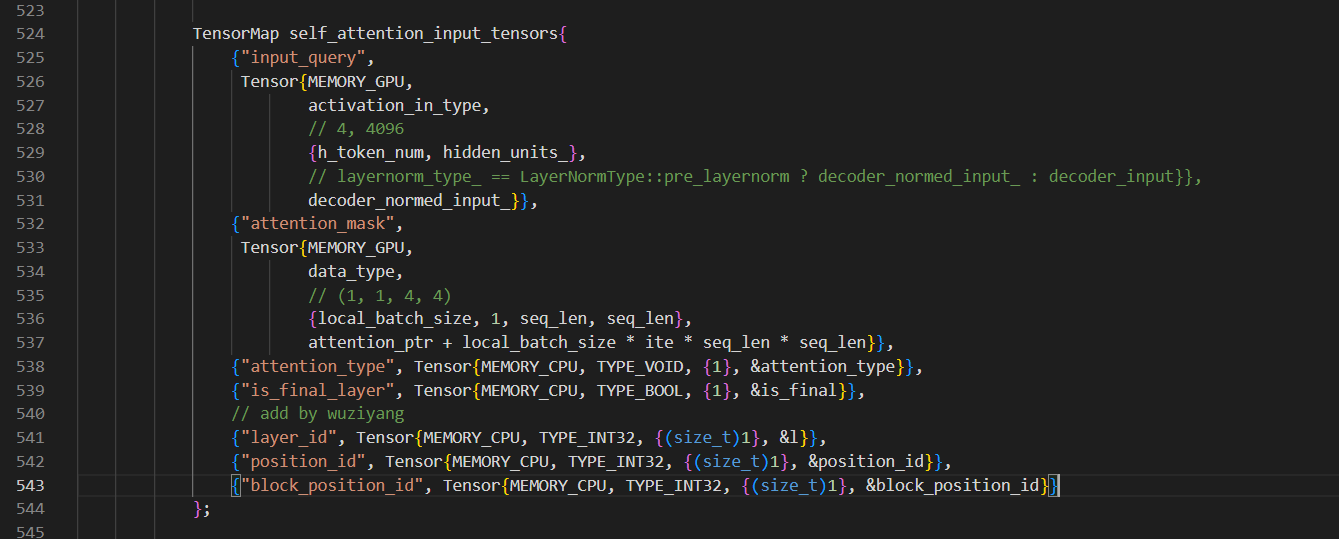


修改/models/multi\_gpu\_gpt/ParallelGpt.cc，以传入输入长度与当前生成长度，用于计算pos id和block pos id

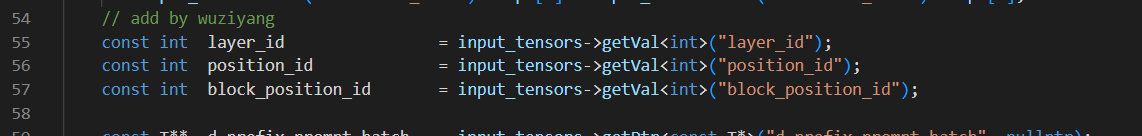


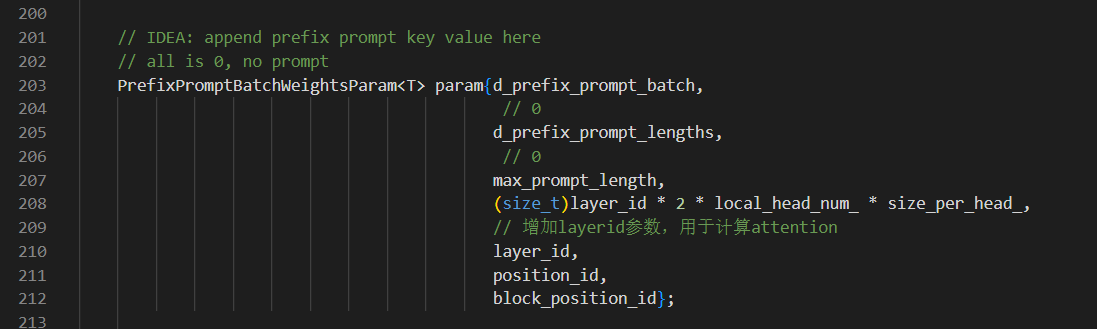
修改/models/multi\_gpu\_gpt/ParallelGptContextDecoder.cc，计算pos id和block pos id并传入attention



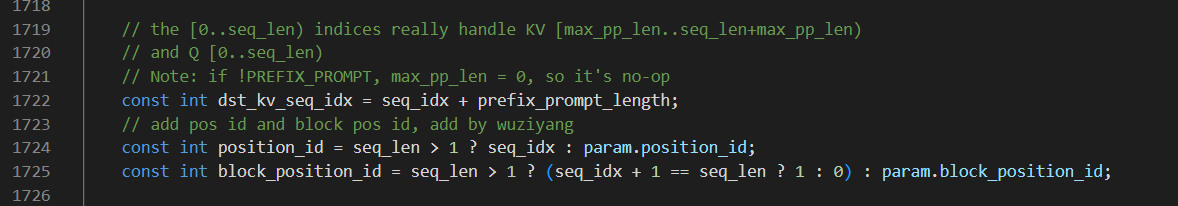


修改/layers/attention\_layers/GptContextAttentionLayer.cc，传入pos id和block pos id

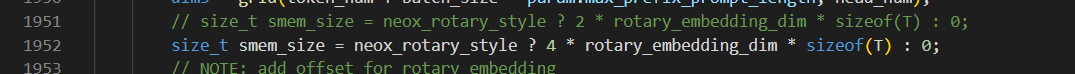




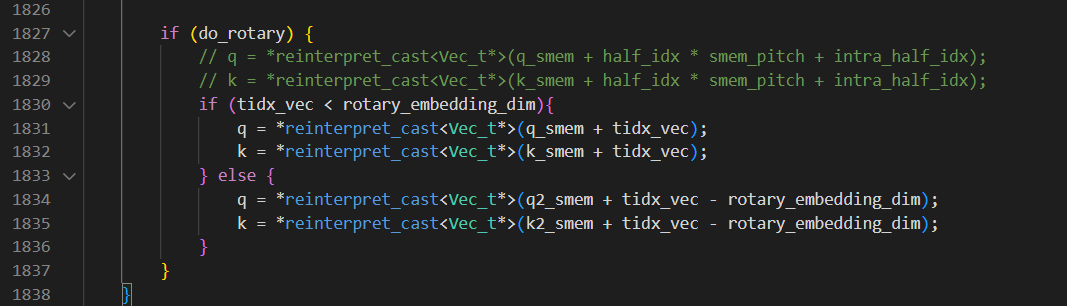
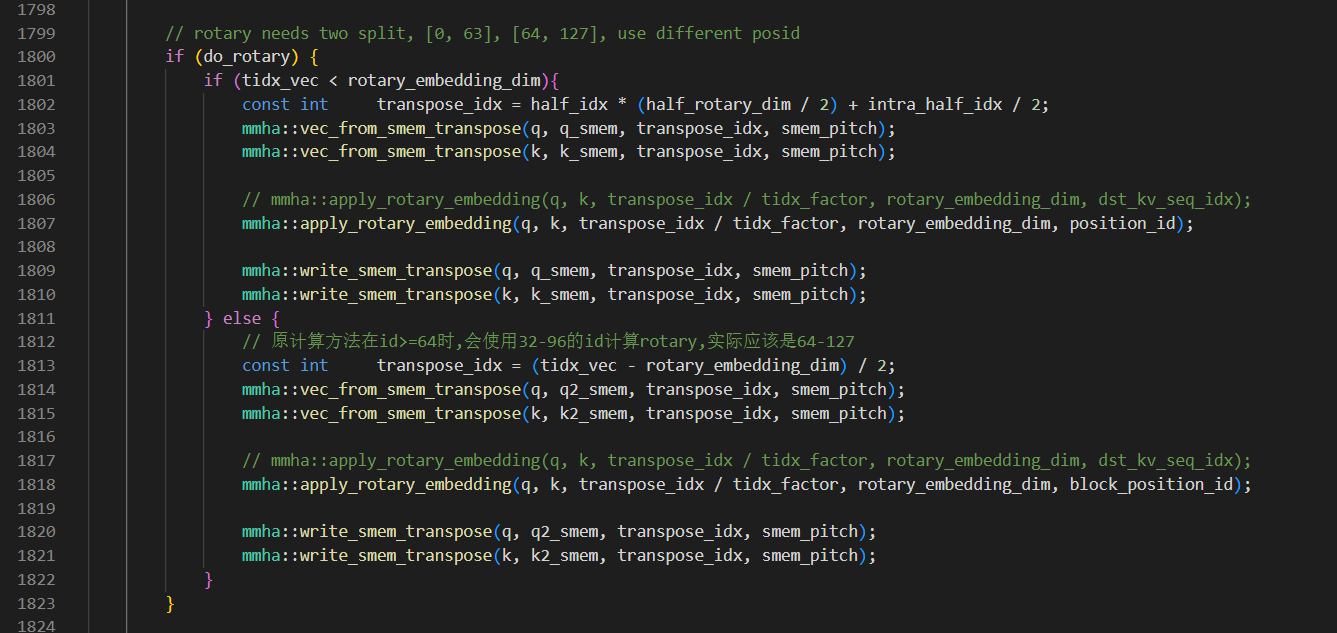
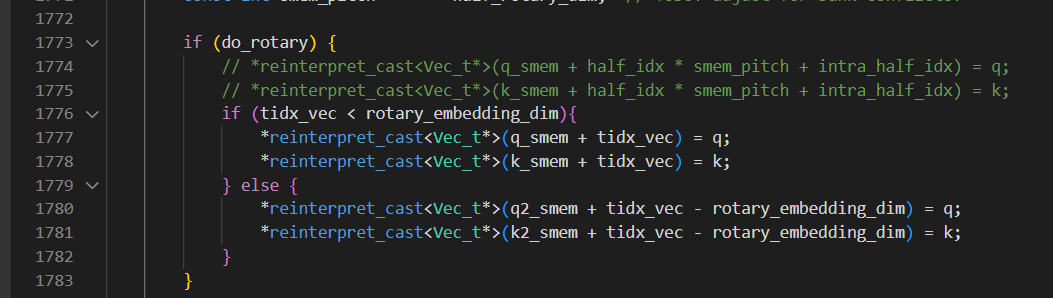
修改/kernels/unfused\_attention\_kernels.cu，计算pos id和block pos id



增大smem，以计算rotary embedding



调整rotary计算，计算后64

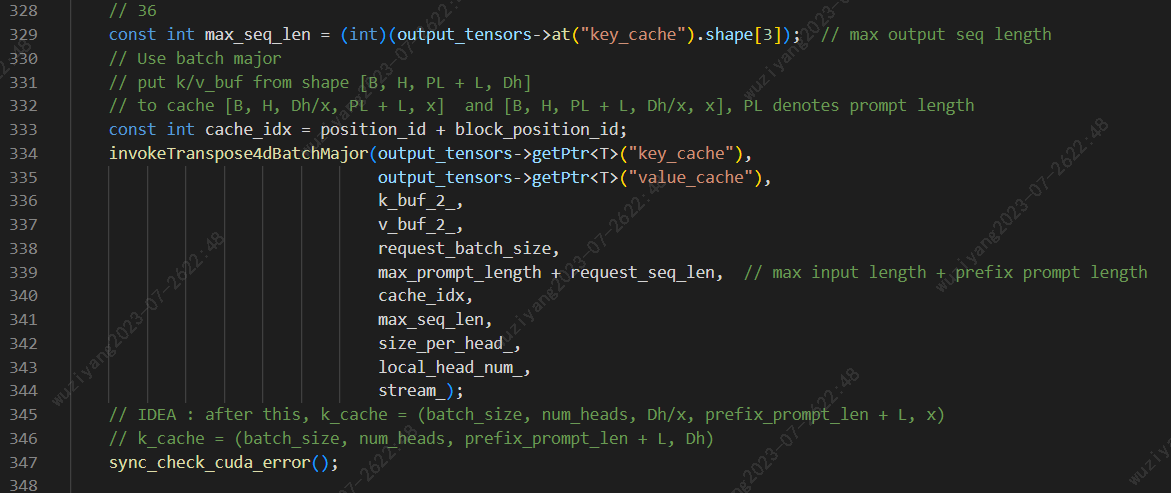


##### 4.6 发现kv cache存储不正确

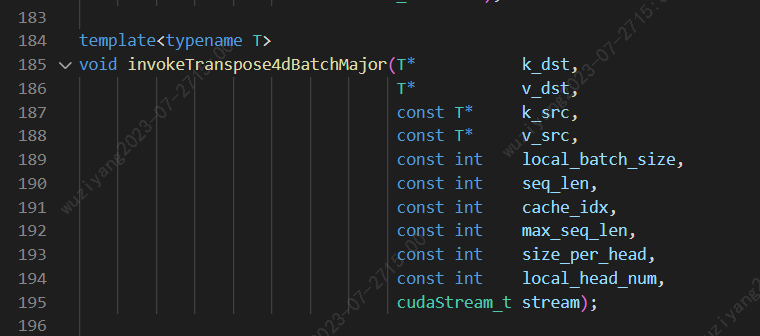
**修复rotary embedding的问题后，发现kv cache未生效，继续排查原因**

1.cache未正确缓存

发现kv cache未正确缓存，其存储及下标计算有误，修改/layers/attention\_layers/GptContextAttentionLayer.cc，增加cache\_idx

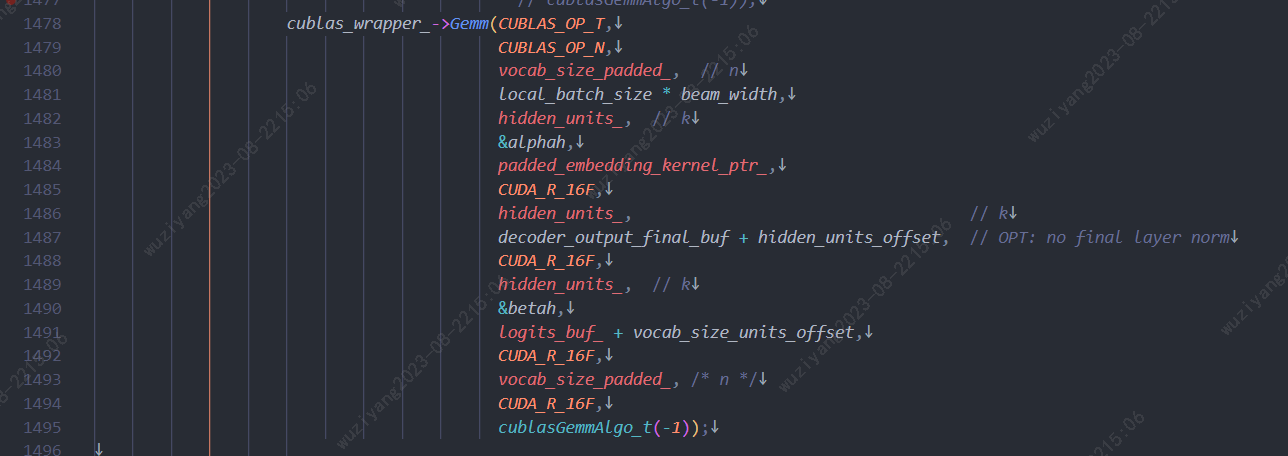


/kernels/unfused\_attention\_kernels.h，增加cache\_idx



##### 4.7发现最终的lm\_head矩阵乘有问题，影响结果

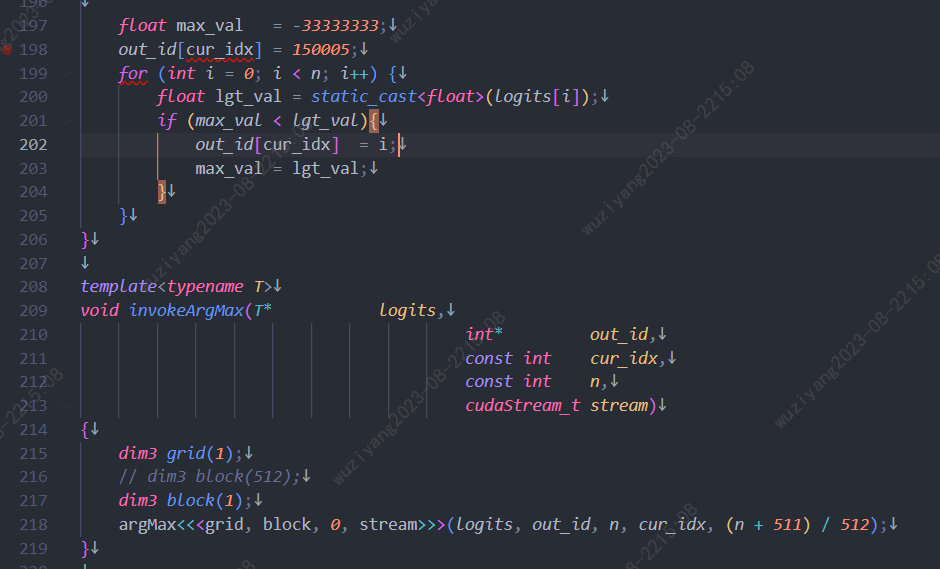
调整代码src/fastertransformer/models/multi\_gpu\_gpt/ParallelGpt.cc，改成正确的fp16矩阵乘



##### 4.8发现decode类有问题，得到logits后计算的id不正确

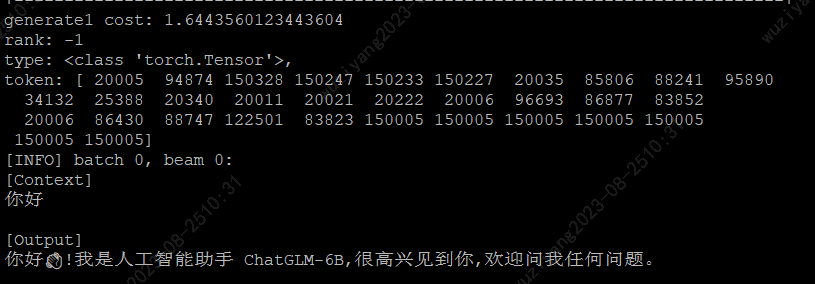
手动实现argmax算子，结果发现argmax的结果是实际结果的1/2，未查明原因，将计算挪到cpu上后正常，修改src/fastertransformer/models/multi\_gpu\_gpt/ParallelGpt.cc



算子实现在/src/fastertransformer/kernels/ban\_bad\_words.cu

##### 4.9 stop标志没有生效

##### 4.10 完成对齐



# 二、fastertransformer优化

## 2.1优化思路

1) 第一次layernorm直接计算出来，缓存本地，减少计算量，相对于cuda gemm也能提高精度

2) mask生成部分，直接cpu生成后传gpu是否比在gpu生成快？

3) rotary embedding中的部分内容是可以预先计算的

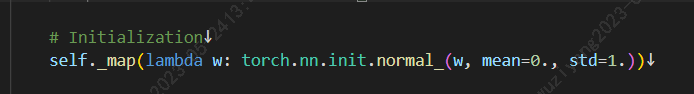
4) ParallelGpt.cc中的 is\_context\_qk\_buf\_float\_参数，似乎会对精度和速度有影响

5) 缓存rotary embedding中的rotary\_embedding结果部分(cos, sin)，并一直存储

## 2.2优化措施

### 1.权重保存与读取速度慢

检查发现这一段无意义代码耗时10min左右！！！注释之后大大提升模型加载速度与debug效率，提升至10s左右！！！



### 2.调整所有cuda算子的block等参数（完成精度对齐后进行）

### 3.删除冗余代码，不必要的输出，计算等

## 2.3优化结果

### 1.最初版

完成最初版本的精度对齐后，性能如下：

|  |  |  |
| --- | --- | --- |
|  | pytorch | FasterTransformer |
| 耗时/ms/token | 69.91631 | 62.5 |

### 2.冷启动优化

冷启动优化后，性能如下：

|  |  |  |  |
| --- | --- | --- | --- |
|  | pytorch | 上一版 | 最新版本 |
| 耗时/ms/token | 70.0 | 62.5 | 57.9 |

问题：

无法生成总长度超过64的序列

# 三、工程化

## 3.1流式输出接口修改

为提升用户体验，将接口修改为流式输出

### 1.模型流式输出修改

### 2.接口websocket修改

### 3.接口sse修改

### 4.接口redis队列修改

## 3.2显存占用测试

### 1.chatglm6b推理显存测试

在A100机器，测试chatglm6b模型显存占用，结果如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| token | 0（模型初始化） | 25（你好） | 330（我准备买套房子，你有什么建议？） | 408（用pytorch实现RNN） | 827(为我生成一篇关于中国1950至2000年之间政府政策变化以及对社会、国家和国际上造成的影响 的详细报告，字数在2000字以上) | 904（带history） | 2048（估计） |
| 显存峰值 | 12882MiB | 13294MiB | 13706MiB | 13762MiB | 14502MiB | 15192MiB | 17388Mib |
| 推理结束显存 | **-** | 13262Mib | 13246Mib | 13248Mib | 13252MiB | 13252MiB | 13252MiB |
| 增加显存（每token） | 12882 | 16.48 | 1.345 | 1.265 | 1.516 | 2.146 | 2.2 |