FlashDecoding++用户使用文档-v1.0

基本信息

本文档为安装包***FlashDecodingPlusAE-v1.0.zip***的对应说明文档。安装包的文件结构如下：

- **Llama2/**
  - model/：Llama2模型文件夹
  - inference.py：Llama2推理测试文件
  - test.py：Llama2吞吐测试文件
  - convert.py：Llama2模型转换文件
- **ChatGLM2/**
  - model/：ChatGLM2模型文件夹
  - inference.py：ChatGLM2推理测试文件
  - test.py：ChatGLM2吞吐测试文件
- **inficom-0.0.1-cp310-cp310-linux_x86_64.whl**：算子库二进制文件
- **script/**
  - ops：包含单算子的python测试文件
- **README.md**：英文版README

安装

环境需求：

- Python 3.10.
- CUDA 11.8 或以上.
- PyTorch 2.0.0 或以上.
- transformers==4.34.0.
- tokenizers==0.14.1.
- accelerate==0.23.0.
- hiq-python==1.1.9.
- sentencepiece==0.1.99

- `batchsize=1` 需要NVIDIA Volta架构或更新的GPU，`batchsize=1~8` 需要NVIDIA Ampere架构或更新的GPU.

安装命令：

```
pip install inficom-0.0.1-cp310-cp310-linux_x86_64.whl
```

或者直接使用安装脚本：

```
bash install.sh
```

使用

模型测试

本安装包支持模型包括：

- Llama2-7B
- Llama2-13B
- ChatGLM2-6B

本安装包目前仅支持单卡推理测试。

Llama2-7B/13B

请首先从官方渠道下载模型权重。为了下载checkpoints和tokenizer文件，需要填写这个表格。请求通过后，会收到下载相关的链接。对于13B模型，官方权重文件默认 `MP=2` ，需要转换成 `MP=1` 后才能在单卡上进行推理。

1. 权重转换（仅13B模型）

   执行以下命令，可以将原本 `.pth` 文件转化为 `MP=1` ，新的 `.pth` 文件将会保存在自定义的文件夹 `${Llama-2-13b-chat-new` 中：

   ```
   cd Llama2
   python convert.py --src-dir ${Llama-2-13b-chat} --dst-dir ${Llama-2-13b-chat-new}
   cp ${Llama-2-13b-chat}/checksum.ch ${Llama-2-13b-chat}/params.json ${Llama-2-13b-chat}/tokenizer.model ${Llama-2-13b-chat}/tokenizer_checklist.chk ${Llama-2-13b-chat-new}
   ```

2. 语句生成（以7B模型为例）

```
python inference.py --ckpt-dir ${Llama-2-7b-chat} --tokenizer-path ${Llama-2-7b-chat/tokenizer.model}
```

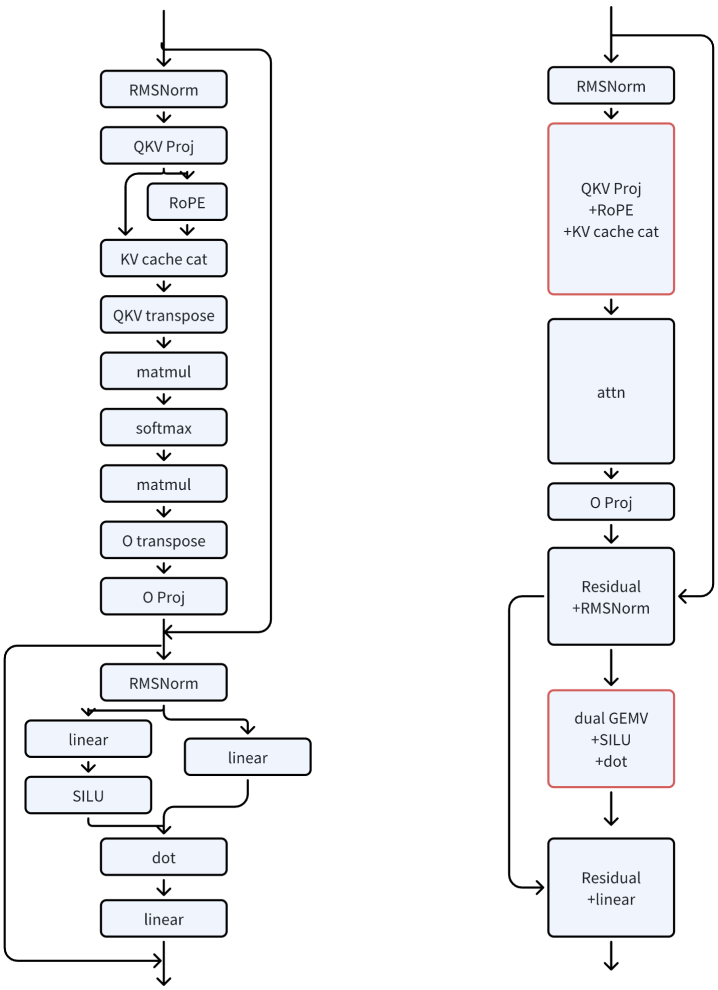
对于以上命令，可以传入`--batch-size`来设置`batchsize`的大小（注意此安装包目前仅支持`batchsize={1, 2, ..., 8}`）。传入`--engine-use`来开启优化的模型推理，如果不传入默认使用baseline版本的模型进行计算。

3. 吞吐测试（以7B模型为例）

```
python test.py --ckpt-dir ${Llama-2-7b-chat}
```

对于以上命令，可以传入`--batch-size`，`--input-length`，`--output-length`来改变测试配置。传入`--backend 'hf'`将会运行HuggingFace的实现来作为baseline，传入`--backend 'ours'`将会运行优化后的模型。

4. 模型搭建参考示意图



左图为最原始的基于PyTorch搭建Llama2模型中的1个Transformer Block，**右图**为我们搭建的Llama2模型的1个Transformer Block，**经过算子融合后，1个Transformer Block仅包含7个大算子**。其中，**红框**内是对性能提升影响较大的融合部分。

ChatGLM2-6B

请从HuggingFace上下载模型权重。

1. 语句生成

```
python inference.py --ckpt-dir ${THUDM/chatglm2-6b}
```

对于以上命令，可以传入`--batch-size`来设置`batchsize`的大小（注意此安装包目前仅支持`batchsize={1, 2, ..., 8}`）。传入`--engine-use`来开启优化的模型推理，如果不传入默认使用baseline版本的模型进行计算。

2. 吞吐测试

```
python test.py --ckpt-dir ${THUDM/chatglm2-6b}
```

对于以上命令，可以传入`--batch-size`，`--input-length`，`--output-length`来改变测试配置。传入`--backend 'hf'`将会运行HuggingFace的实现来作为baseline，传入`--backend 'ours'`将会运行优化后的模型。

部分吞吐测试结果

1. Llama2-7B

batchsize	input length	output length	token/s
1	128	128	113.169
1	2k	128	97.25
8	128	128	780.002
8	2k	128	515.61

2. Llama2-13B

batchsize	input length	output length	token/s
1	128	128	62.872
1	2k	128	56.147
8	128	128	456.085
8	2k	128	320.668

3. ChatGLM2-6B

batchsize	input length	output length	token/s
1	128	128	120.092
1	2k	128	106.92
8	128	128	868.402
8	2k	128	647.955

算子测试

本安装包同时提供**部分算子**接口，具体请参考算子清单部分的详细说明。

算子清单

Decode attention

> **decode_mha_with_async_softmax**(*at::Tensor Q, at::Tensor K, at::Tensor V, const int max_len, const int len, const float scale, const float attn_max*) -> at::Tensor

说明:

使用异步softmax计算的attention算子，适用于decode阶段。返回保存了计算结果的tensor。

参数:

- Q - query tensor, (batchsize, 1, head num, head size)
- K - key cache, (batchsize, max length, head num, head size)
- V - value cache, (batchsize, max length, head num, head size)
- max_len - max length, int
- len - current length, int
- scale - attention scale, float

- attn_max - the unified "max" value to compute async softmax, default: 8.0f

---

> **decode_mha_fall_back**(*at::Tensor Q, at::Tensor K, at::Tensor V, const int max_len, const int len, const float scale, const float attn_max*) -> at::Tensor

说明:

未使用异步softmax计算的attention算子，适用于decode阶段。返回保存了计算结果的tensor。

参数:

- Q - query tensor, (batchsize, 1, head num, head size)
- K - key cache, (batchsize, max length, head num, head size)
- V - value cache, (batchsize, max length, head num, head size)
- max_len - max length, int
- len - current length, int
- scale - attention scale, float

Residual

> **add_residual**(at::Tensor R, at::Tensor X) -> at::Tensor

说明:

残差计算算子。返回保存了计算结果的tensor。

参数:

- R - residual tensor, (batchsize, 1, hidden dim)
- X - hidden states, (batchsize, 1, hidden dim)

Norm

> **rmsnorm**(at::Tensor X, at::Tensor RW) -> at::Tensor

说明:

计算RMSNorm。返回保存了计算结果的tensor。

参数:

- X - hidden states, (batchsize, 1, hidden dim)
- RW - RMSNorm的权重, (hidden dim)

> **layernorm**(at::Tensor X, at::Tensor RW, at::Tensor RB) -> at::Tensor

说明:

计算LayerNorm。返回保存了计算结果的tensor。

参数:

- X - hidden states, (batchsize, 1, hidden dim)
- RW - LayerNorm的权重, (hidden dim)
- RB - LayerNorm的偏置, (hidden dim)

Linear

> **gemv_acc_fp16**(at::Tensor X, at::Tensor W) -> at::Tensor

说明:

以FP16累加的矩阵-向量乘法计算。返回保存了计算结果的tensor。

参数:

- X - 向量, 即hidden states, (batchsize, 1, dim_0)
- W - 矩阵, 即权重, (dim_1, dim_0), 注意是转置的

> **gemv_acc_fp32**(at::Tensor X, at::Tensor W) -> at::Tensor

说明:

以FP32累加的矩阵-向量乘法计算。返回保存了计算结果的tensor。

参数:

- X - 向量, 即hidden states, (batchsize, 1, dim_0)
- W - 矩阵, 即权重, (dim_1, dim_0), 注意是转置的

> **flat_gemm_m8n256k32x8_bz1**(at::Tensor X, at::Tensor W) -> at::Tensor

说明:

矮胖形的矩阵-矩阵乘法计算。仅支持*M*维度最大为8。返回保存了计算结果的tensor。

参数:

- X - 矩阵, (*M*, *K*)
- W - 矩阵, (*N*, *K*), 注意是转置的

> **flat_gemm_mix_for_decode**(at::Tensor X, at::Tensor W) -> at::Tensor

说明：

矮胖形的矩阵-矩阵乘法计算。仅支持*M*维度最大为16。返回保存了计算结果的tensor。

参数：

- X - 矩阵, (*M*, *K*)
- W - 矩阵, (*N*, *K*), 注意是转置的

MLP_fuse

> **dual_linear_silu_dot_fwd**(at::Tensor X, at::Tensor W1, at::Tensor W2) -> at::Tensor

说明：

完成MLP(FFN)中的部分结果计算的融合算子，以Llama2-hugging face中的实现为例，其完成的计算是

```
down_proj = self.down_proj(self.act_fn(self.gate_proj(x)) * self.up_proj(x))
# 其中
# self.act_fn(self.gate_proj(x)) * self.up_proj(x)
# (act_fn使用silu)等价于
# dual_linear_silu_dot_fwd(x, self.gate_proj.weight, self.up_proj.weight)
```

目前仅支持batchsize维度最大为8，返回保存了计算结果的tensor。

参数：

- X - hidden states, (batchsize, 1, hidden_dim_1)
- W1 - FFN中的权重1, 在torch.nn.linear的存储是转置的状态, (hidden_dim_2, hidden_dim_1)
- W2 - FFN中的权重2, 在torch.nn.linear的存储是转置的状态, (hidden_dim_2, hidden_dim_1)

返回值：

- O - (batchsize, 1, hidden_dim_2)

---

attn_proj_rope_kv_cat_fuse

> **attn_proj_rope_kv_cat_fwd**(at::Tensor X, at::Tensor WQ, at::Tensor WK, at::Tensor WV, at::Tensor K, at::Tensor V, at::Tensor freq, const int max_len, int len) -> at::Tensor

说明：

完成self attention中的QKV projection+rope+kv cache cat的融合算子

目前仅支持batchsize维度最大为8，返回Q projection计算结果的tensor。

参数：

- X - hidden states, (batchsize, 1, hidden_dim)
- WQ - attention中q_proj的权重, (hidden_dim, hidden_dim)
- WK - attention中k_proj的权重, (hidden_dim, hidden_dim)
- WV - attention中v_proj的权重, (hidden_dim, hidden_dim)
- K - K cache, (batchsize, max_len, num_heads, head_dim)
- V - V cache, (batchsize, max_len, num_heads, head_dim)
- freq - 预先计算好在rope中需要用到的将位置信息与维度信息结合起来的频率数组，实现与meta-llama版本一致，与hugging-face版本的实现不同，具体计算方式可以参考算子测试 `/script/ops/attn_proj_rope_kv_cat_fwd_test.py` 中的 `precompute_freqs_cis` 函数
- max_len - 预分配的kv cache最大上下文长度
- len - 当前KV cache的上下文长度

返回值：

- Q - (batchsize, 1, hidden_dim)

使用方法 - 以layernorm算子为例

```
import torch
import torch.nn as nn
from inficom import layernorm

# 自定义随机输入
x = torch.empty((2, 1, 4096), dtype=torch.float16,
device="cuda").normal_(mean=0., std=0.5)
# 定义LayerNorm层
layer = nn.LayerNorm(4096)
# 参照输出
ref_out = layer(x)
# 算子输出
out = layernorm(x, layer.weight)
# 比较结果
all_close = torch.allclose(ref_out, out, atol=1e-2, rtol=1e-4)
print(all_close)
```

算子测试

注意，由于实际模型搭建过程中大量使用了融合算子，单算子测试性能与模型测试性能并不等价，仅供参考。单算子测试文件的目的是为了对算子接口进行说明。

算子测试文件在 `/script/ops/` 文件夹中，首先进入该文件夹：

```
cd script/ops
```

该文件夹中包含6个随机数测试文件，接下来将介绍其使用方法：

1. decode_attn_test.py

需求：需安装FlashAttention, xformers等算子库作为baseline。

传参：

- --batch-size
- --head-num
- --head-size：目前仅支持128
- --cache-len：KV cache的长度
- --xformers-decoder：是否测试xformers中基于纯CUDA Core的decode attention实现

输出：正确性对比，单算子用时

---

2. flat_gemm_test.py

传参：(需在.py文件最上方中手动修改)

- Z：batchsize, *M*维度
- DIM1： *K*维度
- DIM2：*N维度*

输出：正确性对比，单算子用时

---

3. gemv_test.py

传参：(需在.py文件最上方中手动修改)

- Z：batchsize=1, *M*维度
- DIM1： *K*维度
- DIM2：*N维度*

输出：正确性对比，单算子用时

---

4. rmsnorm_test.py

传参：(需在.py文件最上方中手动修改)

- Z：batchsize
- DIM： hidden dim

输出：正确性对比，单算子用时

---

5. add_residual_test.py

传参：(需在.py文件最上方中手动修改)

- Z：batchsize
- DIM： hidden dim

输出：正确性对比，单算子用时

---

6. residual_rmsnorm_test.py

传参：(需在.py文件最上方中手动修改)

- Z：batchsize
- DIM： hidden dim

输出：正确性对比，单算子用时

---

7. dual_linear_silu_dot_fwd_test.py

传参：(需在.py文件最上方中手动修改)

- Z：batchsize
- DIM1： hidden dim 1
- DIM2： hidden dim 2

输出：正确性对比，单算子用时

---

8. attn_proj_rope_kv_cat_fwd_test.py

传参：(需在.py文件最上方中手动修改)

- Z：batchsize

- DIM： hidden dim
- LEN：sequence length
- MAX_LEN: KV cache max sequence length
- HN：num of heads in multi-head-attention
- HS：dimension per head in multi-head-attention (DIM / HN)

输出：正确性对比，单算子用时

附件

安装包：*FlashDecodingPlusAE-v1.0.zip*