ByteTransformer: A High-Performance Transformer Boosted for Variable-Length Inputs

Yujia Zhai, Chengquan Jiang, Leyuan Wang, Xiaoying Jia, Shang Zhang, Zizhong Chen, Xin Liu, Yibo Zhu
Important metrics for LLM Inference

- Throughput = query / s – maximize for batch job speed to allow more users

- Latency = s / token – minimize for user experience
## LLM Inference Is Actually Slow

<table>
<thead>
<tr>
<th>model id</th>
<th>tok out</th>
<th>sec</th>
<th>model name</th>
<th>ms/token</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpt-3.5-turbo-1106</td>
<td>3772</td>
<td>24.3</td>
<td>latest gpt-3.5</td>
<td>6.5</td>
</tr>
<tr>
<td>gpt-4-1106-preview</td>
<td>4096</td>
<td>74.7</td>
<td>gpt-4 turbo</td>
<td>18.2</td>
</tr>
<tr>
<td>gpt-3.5-turbo-0613</td>
<td>3800</td>
<td>79.5</td>
<td>old gpt-3.5</td>
<td>20.9</td>
</tr>
<tr>
<td>gpt-4-0613</td>
<td>4141</td>
<td>400.0</td>
<td>old gpt-4</td>
<td>96.6</td>
</tr>
</tbody>
</table>

Table Credit: https://www.taivo.ai/__a-wild-speed-up-from-openai-dev-day/
Tensorflow XLA, PyTorch JIT
- Leverage the domain-specific just-in-time compilation technique to boost performance
- Does not support kernel fusion or variable length input

FasterTransformer
- Support variable length input by batching requests with similar sequence lengths
- Partially support fused MHA ($\leq 512$)

TurboTransformer
- Support variable length input by batching requests with similar sequence lengths

ByteTransformer
- Memory-Bound Kernel Fusion
- Variable Length Support
- Fused Multi-Head Attention (MHA)

A lot of redundant memory & computation for batching requests with different sequence length!
Bert Transformer Architecture

- Batch Size (bs): 16
- Head Number: 12
- Head Size: 64

\[ k = 12 \times 64 = 768 \]

m: sequence length

<table>
<thead>
<tr>
<th>Number of Computation</th>
<th>Gemm #0</th>
<th>Gemm #1</th>
<th>Gemm #2</th>
<th>Gemm #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 6mk^2 )</td>
<td>( m^2 )</td>
<td>( 2mk^2 )</td>
<td>( 8mk^2 )</td>
<td></td>
</tr>
</tbody>
</table>

(a) Sequence lengths 256

(b) Sequence lengths 1024
Fusing Memory-bound Operations

compute packed (Q,K,V) in one GEMM

add bias (Q,K,V)

batched GEMM Q x K

softmax

batched GEMM QK x V

transpose

GEMM

fused add bias & layernorm

fused GEMM with add bias & activation

GEMM

fused add bias & layernorm

Fused Add bias & layernorm

LayerNorm (Max_seq * batch size)

Hidden

+ CTA

Hidden

CTA
Fusing Memory-bound Operations

Fused Gemm with add bias & activation

```c
#define GEMM_TYPE(BLOCK_M, BLOCK_N, BLOCK_K, WARP_M, WARP_N, WARP_K, INST_M, INST_N, INST_K, \ 
    NUM_STAGES) \ 
    using ShapeMMAThreadBlock_##BLOCK_M####BLOCK_N####BLOCK_K = \ 
        cutlass::GemmShape<BLOCK_M, BLOCK_N, BLOCK_K>;
using ShapeMMAWarp_##WARP_M####WARP_N####WARP_K = \ 
        cutlass::GemmShape<WARP_M, WARP_N, WARP_K>;
using ShapeMMAp_##INST_M####INST_N####INST_K = \ 
        cutlass::GemmShape<INST_M, INST_N, INST_K>;
using Gemm_##BLOCK_M####BLOCK_K####WARP_M####WARP_K####INST_M####INST_N####INST_K = \ 
        cutlass::Gemm<
            ElementInputA, LayoutInputA, ElementInputB, LayoutInputB, \ 
            ElementOutput, LayoutOutput, ElementAccumulator, MMAPp, SmArch, \ 
            ShapeMMAThreadBlock_##BLOCK_M####BLOCK_K, \ 
            ShapeMMAWarp_##WARP_M####WARP_K, \ 
            ShapeMMAp_##INST_M####INST_N####INST_K, \ 
            EpilogueOp, \ 
            SwizzleThreadBlock, NUM_STAGES>;
```

However, there is still computation redundancy!
Zero Padding Algorithm

- Compute prefix sum & zero padding
- Compute packed (Q,K,V) in one GEMM
- Fused rebuild padding & add bias
- Batched GEMM Q x K
  - Softmax
  - Batched GEMM QK x V
  - Fused zero padding & transpose
  - GEMM
  - Fused add bias & layernorm
  - Fused GEMM with add bias & activation
  - GEMM
  - Fused add bias & layernorm

- Threadmap
- Mask matrix
- Use prefix sum to compute offset info
- Original input tensor
- Packing with offset info
- Hidden dim
- Batch size
- Packed input tensor
- Hidden dim
- Input tensor (if padded)
Zero Padding Algorithm

MHA cannot benefit from zero padding without modification
Fused Multi Head Attention: Short Sequence

Launch grid={head_num, seq_len / split_seq_len, batch_size}
Fused Multi Head Attention: Long Sequence

Grouped GEMM
Every block handles one tile, suppose we have a tile_id = t, we need to know:
1. problem_id
2. Tile_offset inside that problem

To get the Q, K, V pointer and offset in Q:

- Compute number of tile
- Prefix Sum

Which problem has the tile_id = t (sum of tild ≥ tild_id)

Warp sync problem_id, tile_start, tile_end inside the problem
Fused Multi Head Attention: Long Sequence

1. $max_j = \max(x_0, \ldots, x_n)$
2. $sum_j = e^{x_0 - max_j} + \ldots + e^{x_n - max_j}$
3. $softmax_j = \frac{e^{xi} - max}{sum}$

# of problems = batch sz * head num
Stepwise Optimization

- **Grouped Gemm Q x K**
  - Reduction
  - Grouped Gemm Q x V
- **Grouped Gemm Q x K**
  - Softmax
  - Batched Gemm Q x V
- **Batched Gemm Q x K**
  - Transpose
  - Gemm
  - Add bias & activation
  - Gemm
  - Layer norm
- **Batched Gemm QK x V**
  - Fused zero padding & transpose
  - Gemm
  - Add bias & layer norm
  - Gemm
  - Layer norm
- **Batched Gemm QK x V**
  - Fused add bias & layer norm
  - Gemm
  - Add bias & layer norm
  - Gemm
  - Layer norm

### Table

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Zero Padding</th>
<th>Zero Padding + fused MHA</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEMM0</td>
<td>6mk^2</td>
<td>6(α - m)k^2</td>
<td>6(α - m)k^2</td>
</tr>
<tr>
<td>MHA</td>
<td>(\frac{4m^2}{l_2}k)</td>
<td>(\frac{4m^2}{l_2}k)</td>
<td>(\frac{4(α - m)^2}{l_2}k)</td>
</tr>
<tr>
<td>GEMM1</td>
<td>2mk^2</td>
<td>2(α - m)k^2</td>
<td>2(α - m)k^2</td>
</tr>
<tr>
<td>GEMM2</td>
<td>8mk^2</td>
<td>8(α - m)k^2</td>
<td>8(α - m)k^2</td>
</tr>
<tr>
<td>GEMM3</td>
<td>8mk^2</td>
<td>8(α - m)k^2</td>
<td>8(α - m)k^2</td>
</tr>
</tbody>
</table>
Evaluation: Fusion Kernel

- Compute packed (Q,K,V) in one GEMM
- Add bias (Q,K,V)
- Batched GEMM Q x K
  - Softmax
  - Batched GEMM QK x V
  - Transpose
  - GEMM
  - Fused add bias & layernorm
  - Fused GEMM with add bias & activation
  - GEMM
  - Fused add bias & layernorm
Evaluation: Fusion MHA

Fig. 11: Fused MHA for short sequences.

Fig. 12: Fused MHA for long sequences.
Evaluation: Fusion MHA

Fig. 13: Comparisons of our FMHA with FlashAttention.
Evaluation: End-To-End

Fig. 16: End-to-end benchmark for other BERT-like models.
ByteTransformer provides a high-performance implementation that supports variable sequence length input and achieves an average of 50% speedup end-to-end on different models.

However, there are other possible optimizations to concern:

- Tail Effect