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

| model id           | tok out | sec  | model name        | ms/token |
|--------------------|---------|------|-------------------|----------|
| gpt-3.5-turbo-1106 | 3772    | 24.3 | latest gpt-3.5    | 6.5      |
| gpt-4-1106-preview | 4096    | 74.7 | gpt-4 turbo       | 18.2     |
| gpt-3.5-turbo-0613 | 3800    | 79.5 | old gpt-3.5       | 20.9     |
| gpt-4-0613         | 4141    | 400.0| old gpt-4         | 96.6     |

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

| Number of Computation | Gemm #0 | Gemm #1 | Gemm #2 | Gemm #3 |
|-----------------------|--------|---------|---------|---------|
| Gemm                  | 6mk^2  | 2mk^2  | 8mk^2  | 8mk^2  |
| MHA                   | \( \frac{m^2}{bs} k \) | \( \frac{m^2}{bs} k \) | \( \frac{m^2}{bs} k \) | \( \frac{m^2}{bs} k \) |

(a) Sequence lengths 256

(b) Sequence lengths 1024
Fusing Memory-bound Operations

- Fused Add bias & layernorm

LayerNorm (Max_seq * batch size)

- LayerNorm
- CTA
- Hidden

- GEMM
  - fused add bias & layernorm
  - fused GEMM with add bias & activation
  - transposen
  - batched GEMM QK x V
  - softmax
  - add bias (Q,K,V)
  - batched GEMM Q x K
  - compute packed (Q,K,V) in one GEMM
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>;
#define GEMM_TYPE(BLOCK_M, BLOCK_N, BLOCK_K, WARP_M, WARP_N, WARP_K, INST_M, INST_N, INST_K) \
    using ShapeMMAWarp_##WARP_M##_##WARP_N##_##WARP_K = \
        cutlass::GemmShape<WARP_M, WARP_N, WARP_K>;
#define GEMM_TYPE(INST_M, INST_N, INST_K) \
    using ShapeMMAop_##INST_M##_##INST_N##_##INST_K = \
        cutlass::GemmShape<INST_M, INST_N, INST_K>;
#define GEMM_TYPE(INST_M, INST_N, INST_K) \
    using Gemm_##BLOCK_M##_##BLOCK_N##_##BLOCK_K##_##WARP_M##_##WARP_N##_##WARP_K##_##INST_M##_##INST_N##_##INST_K = \
        cutlass::Gemm<
            ElementInputA, LayoutInputA, ElementInputB, LayoutInputB, 
            ElementOutput, LayoutOutput, ElementAccumulator, MMAop, SmArch, 
            ShapeMMAThreadBlock_##BLOCK_M##_##BLOCK_N##_##BLOCK_K, 
            ShapeMMAWarp_##WARP_M##_##WARP_N##_##WARP_K, 
            ShapeMMAop_##INST_M##_##INST_N##_##INST_K, 
            SwizzleThreadBlock, NUM_STAGES>>, 
```

However, there is still computation redundancy!
Zero Padding Algorithm

- **Batch Size**: A warp to deal with a whole sequence.
- **Threadmap**: Mask matrix.
- **Original Input Tensor**: Packing with offset info.
- **Hidden Dim**: Hidden dim.
- **Input Tensor**: (If padded).

**Diagram Steps**:
1. Compute prefix sum & zero padding.
2. Compute packed (Q,K,V) in one GEMM.
3. Fused rebuild padding & add bias.
4. Batched GEMM Q x K.
5. Softmax.
6. Batched GEMM QK x V.
7. Fused zero padding & transpose.
8. GEMM.
9. Fused add bias & layernorm.
10. Fused GEMM with add bias & activation.
11. GEMM.
12. Fused add bias & layernorm.
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
Fused Multi Head Attention: Long Sequence

Tile Scheduling

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. $\text{max}_j = \text{max}(x_0, \ldots, x_n)$
2. $\text{sum}_j = e^{x_0 - \text{max}_j} + \ldots + e^{x_n - \text{max}_j}$

Partial reduction:

1. Calculate $\text{max}_j$ and $\text{sum}_j$ for each head.
2. Reduce the partial results to a single value.
3. Apply fused element-wise operations to obtain the final output.

# of problems = batch sz * head num
Stepwise Optimization

- Compute packed \((Q, K, V)\) in one GEMM
  - Add bias \((Q, K, V)\)
  - Batched GEMM \(Q \times K\)
    - Softmax
  - Batched GEMM \(QK \times V\)
    - Transpose

- Grouped Gemm \(Q \times K\)
  - Compute prefix sum & zero padding
  - Batched GEMM \(Q \times K\)
  - Softmax
  - Batched GEMM \(QK \times V\)
    - Fused zero padding & transpose

|                  | Baseline                  | Zero Padding              | Zero Padding + fused MHA |
|------------------|---------------------------|---------------------------|--------------------------|
| GEMM0            | \(6mk^2\)                 | \(6(\alpha \cdot m)k^2\)  | \(6(\alpha \cdot m)k^2\) |
| MHA              | \(4\frac{m^2}{b_k}k\)    | \(4\frac{m^2}{b_k}k\)    | \(4\frac{(\alpha \cdot m)^2}{b_k}k\) |
| GEMM1            | \(2mk^2\)                 | \(2(\alpha \cdot m)k^2\)  | \(2(\alpha \cdot m)k^2\)  |
| GEMM2            | \(8mk^2\)                 | \(8(\alpha \cdot m)k^2\)  | \(8(\alpha \cdot m)k^2\)  |
| GEMM3            | \(8mk^2\)                 | \(8(\alpha \cdot m)k^2\)  | \(8(\alpha \cdot m)k^2\)  |
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 GEMM with add bias & activation
  - GEMM
  - Fused add bias & layernorm

![Graph 1](image1.png)

- Baseline
- Optimized
- Speedup

![Graph 2](image2.png)

- GEMM
- Add-bias & GELU
- Fused
  - Speedup

Sequence Length:
- 128
- 256
- 384
- 512
- 640
- 768
- 896
- 1024

Scaled Execution Time:
- 0
- 0.5
- 1
- 1.5
- 2
- 2.5

Improvement Ratio:
- 0%
- 20%
- 40%
- 60%
- 80%
- 120%
Evaluation: Fusion MHA

Fig. 11: Fused MHA for short sequences.

Fig. 12: Fused MHA for long sequences.
Fig. 13: Comparisons of our FFMHA 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