LLM Inference FOM

ACM Reference Format:
. 2025. LLM Inference FOM. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Contents

1 INTRODUCTION

Large Language Models (LLMs) have emerged as a transformative force in AI, revolutionizing Natural Language Processing (NLP) and text generation. These models, such as GPT and LLaMA, have risen to prominence due to their ability to understand and generate human-like text across various tasks. *LLM Inference* is a critical aspect of various modern applications, which refers to using a trained LLM to generate responses or make predictions. As LLMs continue to grow in size and complexity, optimizing inference becomes increasingly crucial to balance performance with computational resources, energy consumption, and response times.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 TRANSFORMER BLOCK

Table 1 illustrates all the notations within a Large Language Model used in this report.

2.1 Self-Attention

Query, Key, and Value Projections:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

where $Q \in \mathbb{R}^{T_{\text{in}} \times d_{\text{model}}}$, $K \in \mathbb{R}^{T_{\text{in}} \times \frac{d_{\text{model}}}{G}}$, and $V \in \mathbb{R}^{T_{\text{in}} \times \frac{d_{\text{model}}}{G}}$.

Attention Scores:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{\frac{d_{\text{model}}}{G}}}\right)V$$

Output Projection:

$$O = \text{Attention}(Q,K,V)W_O$$

where $O \in \mathbb{R}^{T_{\text{in}} \times d_{\text{model}}}$.

2.2 Feedforward Network (MLP) - Dense

Feedforward Network: - Gate, Up and Down FC Layers:

$$G = XW_{\text{gate}}$$

$$U = XW_{\text{up}}$$

$$Z = \text{ReLU}(U)G$$

$$D = ZW_{\text{down}}$$

2.3 Mixture of Experts (MoE)

Gating Function The gating mechanism determines the top A experts for each token:

$$L = \text{softmax}(XW_{expertgating})$$

Top-A Expert Selection The gate selects the top A experts for each token and computes their weights:

$$w_{\text{top-}A} = \text{Top-}A(L)$$

Expert Computation Each selected expert i processes its assigned tokens independently:

$$Z_{expert_i} = MLP(X)$$

Combining Expert Outputs The outputs from all experts are combined based on their gating weights:

$$X_{MoE} = \sum Z_{expert_i} w_i$$

2.4 Residual Connections

$$X' = \text{LayerNorm}(X + O)$$

2.5 Layer Normalization

$$X'' = \text{LayerNorm}(X' + Z)$$

2.6 Positional Embeddings

Token and Positional Embedding Addition:

$$X = W_E + W_P$$

where $W_E \in \mathbb{R}^{V \times d_{\text{model}}}$ and $W_P \in \mathbb{R}^{L \times d_{\text{model}}}$.

3 MEMORY REQUIREMENT - LLM INFERENCE

The total memory required for LLM inference can be broken down into three different parts: **Model Memory**, **Maximum Activation Memory**, and **Maximum KV Cache Memory**.

3.1 Model Memory

The model memory represents the storage required for all the trainable parameters of the model. This includes weight matrices and embedding matrices.

$$\text{Model Memory} = 2 \cdot N \cdot \left(2 \cdot d_{\text{model}}^2 + 2 \cdot H \cdot \frac{d_{\text{model}}^2}{G} + 2 d_{\text{model}} + 2 \cdot d_{\text{model}} \cdot d_{\text{ff}} + d_{\text{ff}} \cdot d_{\text{model}} \right) + V \cdot d_{\text{model}} + L \cdot d_{\text{model}}$$

Here:

- The factor of 2 accounts for 4 bytes per parameter (assuming FP16 precision).
- The terms correspond to:
  - N: Number of layers, with $2d_{\text{model}}$ accounting for LayerNorm parameters per layer.

3.2 Maximum Activation Memory

Activation memory is required to store intermediate activations during forward propagation. For inference, this is dominated by the largest layer's activations.

$$\text{Activation Memory} = 2 \cdot B \cdot T_{\text{in}} \cdot d_{\text{model}}$$

- B: Batch size.
- T_{in}: Input sequence length.
- d_{model}: Hidden dimension of the model.

3.3 Maximum KV Cache Memory

During inference, key-value (KV) pairs from self-attention are cached to avoid recomputation. The memory required for KV caching is:

$$\text{KV Cache Memory} = 2 \cdot B \cdot (T_{\text{in}} + T_{\text{out}}) \cdot H \cdot 2 \cdot \frac{d_{\text{model}}}{G}$$

- The factor of 2 accounts for both keys and values.
- H: Number of attention heads.
- G: Number of query groups in GQA

4 FLOP CALCULATION - PREFILL PHASE

4.1 Transformer Block - Dense

Consider a single dense transformer block processing a batch of B input sequences, each of token length T_{in} and model's hidden dimension d_{model}, and dimension of the intermediate feed-forward layer d_{ff}. Below is the breakdown of the FLOPs for one forward pass through this Transformer block.

4.1.1 Layer Normalization FLOPs: Each Transformer block applies two layer normalization operations (one before the attention sub-block and one before the MLP sub-block). Layer normalization over a vector of length d_{model} has linear cost in d_{model}. The number of FLOPs per token for a single layer norm (accounting for mean & variance computation and the per-element scale/shift) is $4d_{\text{model}}$. For batch size B and sequence length T_{in}, one layer norm costs about $4BT_{\text{in}}d_{\text{model}}$ FLOPs. With two layer norm operations per block, the total is approximately:

$$\text{FLOPs}_{\text{LN(total)}} = 8 \cdot B \cdot T_{\text{in}} \cdot d_{\text{model}}$$

4.1.2 Self-Attention FLOPs (FlashAttention v2). The self-attention sub-block consists of several steps:

- **QKV Projection:** The input is linearly projected to Query, Key, and Value matrices. This involves three dense transforms of size $d_{\text{model}} \to d_{\text{model}}$. The FLOPs for these projections is
  $$6 \cdot B \cdot T_{\text{in}} \cdot d_{\text{model}}^2$$
  since each linear projection costs $2 \cdot B \cdot T_{\text{in}} \cdot d_{\text{model}} \cdot d_{\text{model}}$ (multiply-adds) and there are three of them (Q, K, V).
- **Attention Score Computation:** For each head, we compute dot-products between queries and keys to form attention scores. With H heads (and $d_{\text{model}} = H \times d_{\text{head}}$), computing all $T_{\text{in}} \times T_{\text{in}}$ dot-products per head costs $2T_{\text{in}}^2 d_{\text{head}}$ FLOPs. Across all heads this is equal to
  $$2BT_{\text{in}}^2 d_{\text{model}}$$
  FlashAttention fuses the softmax normalization with this step, avoiding extra passes.
- **Weighted Value Computation:** The attention probabilities are multiped with the value (V) vectors. This corresponds to multiplying the $T_{\text{in}} \times T_{\text{in}}$ attention matrix by the Value matrix. The FLOPs are similar to the score computation, about
  $$2BT_{\text{in}}^2 d_{\text{model}}$$
- **Output Projection:** The concatenated attention outputs (of dimension d_{model} per token) are passed through a final linear projection back to d_{model}. This costs
  $$2BT_{\text{in}} d_{\text{model}}^2$$

Adding up all these FLOPs together, the self-attention block requires:

$$\text{FLOPs}_{\text{attention}} = 6BT_{\text{in}}d_{\text{model}}^2 + 2BT_{\text{in}}^2 d_{\text{model}} + 2BT_{\text{in}}^2 d_{\text{model}} + 2BT_{\text{in}} d_{\text{model}}^2 .$$

$$\text{FLOPs}_{\text{attention}} = 8BT_{\text{in}}d_{\text{model}}^2 + 4BT_{\text{in}}^2 d_{\text{model}} .$$

Parameter	Description
W_Q	Query weight matrix (dimensions: $d_{model} \times d_{model}$)
W_K	Key weight matrix in GQA (dimensions: $d_{model} \times \frac{d_{model}}{G}$)
W_V	Value weight matrix in GQA (dimensions: $d_{model} \times \frac{d_{model}}{G}$)
W_O	Output projection matrix (dimensions: $d_{model} \times d_{model}$)
W_{gate}	Gating projection matrix in MLP (dimensions: $d_{model} \times d_{ff}$)
W_{up}	Up projection matrix in MLP (dimensions: $d_{model} \times d_{ff}$)
W_{down}	Down projection matrix in MLP (dimensions: $d_{ff} \times d_{model}$)
$W_{expertgating}$	MoE Gate expert routing matrix (dimensions: $d_{model} \times E$)
W_E	Token embedding matrix (dimensions: $V \times d_{model}$)
W_P	Positional embedding matrix (dimensions: $L \times d_{model}$)
d_{model}	Hidden dimension of the model
d_{ff}	Feedforward layer dimension
N	Number of Transformer Layers
V	Vocabulary size
L	Maximum input sequence length
H	Number of attention heads
E	Number of experts in MoE layers
B	Batch size
T_{in}	Input sequence length
T_{out}	Output sequence length
G	Number of query groups in Grouped-Query Attention (GQA)
A	Number of activated experts per token in MoE layers
P	Number of GPUs per node
Q	Total Number of Nodes

Table 1: Weight matrices and inference-related parameters of a large language model.

4.1.3 MLP Dense Block FLOPs. The MLP module consists of the following four core operations: Gate projection (Projects input from dimension d_{model} to d_{ff}), Up projection (Projects input from dimension d_{model} to d_{ff}), Element-wise multiplication (Multiplies the outputs of the gate and up projections (activation function step)), Down projection (Projects from dimension d_{ff} back to d_{model})

FLOPs Breakdown by Operation

Gate projection:

$$\text{FLOPs}_{gate} = 2 \times (B \times T_{in}) \times d_{model} \times d_{ff}$$

Up projection:

$$\text{FLOPs}_{up} = 2 \times (B \times T_{in}) \times d_{model} \times d_{ff}$$

Element-wise multiplication (activation step):

$$\text{FLOPs}_{mult} = B \times T_{in} \times d_{ff}$$

Down projection:

$$\text{FLOPs}_{down} = 2 \times (B \times T_{in}) \times d_{ff} \times d_{model}$$

Total FLOPs for the MLP Block: By adding all the above components, we get:

$$\begin{aligned}\text{FLOPs}_{MLP} &= \text{FLOPs}_{gate} + \text{FLOPs}_{up} + \text{FLOPs}_{mult} + \text{FLOPs}_{down} \\ &= 6BT_{in}d_{model}d_{ff} + BT_{in}d_{ff} \\ &= BT_{in}d_{ff}(6d_{model} + 1)\end{aligned}$$

4.1.4 Residual Connection FLOPs. Finally, the block has two residual skip connections (one adding the attention output to the original input, and one adding the MLP output to the post-attention representation). Each residual addition operates on $BT_{\text{in}}d_{\text{model}}$ elements (a single add per element). The FLOPs for both additions sum to:

$$\text{FLOPs}_{\text{residuals}} = 2BT_{\text{in}}d_{\text{model}} .$$

4.1.5 Total FLOPs per Transformer Block. Summing all contributions from above (layer norms, attention, MLP, and residual adds), we get the total FLOPs for one Transformer block:

$$\begin{aligned}\text{FLOPs}_{\text{total}} &= \text{FLOPs}_{\text{LN}} + \text{FLOPs}_{\text{attention}} + \text{FLOPs}_{\text{MLP}} + \text{FLOPs}_{\text{residuals}} \\ &= 8BT_{\text{in}}d_{\text{model}} + \left(8BT_{\text{in}}d_{\text{model}}^2 + 4BT_{\text{in}}^2 d_{\text{model}}\right) + \\ &\quad 4BT_{\text{in}}d_{\text{model}}d_{\text{ff}} + 2BT_{\text{in}}d_{\text{model}}\end{aligned}$$

4.2 Transformer Block - MoE

We compute the total FLOPs for a single Transformer block with a Mixture of Experts feedforward network. The block consists of four sub modules: Layer normalization (before attention and before MoE MLP) - Self-attention (FlashAttention v2) - MoE feedforward network (FFN with top-A experts active per token) - Residual connections

4.2.1 Layer Normalization FLOPs. Each normalization operation has a cost of approximately $4 \cdot d_{model}$ FLOPs per token. Since there are two normalization layers (before attention and before the MoE MLP), the total FLOPs for normalization is:

$$\text{FLOPs}_{\text{LN}} = 8 \cdot B \cdot T_{in} \cdot d_{model}$$

4.2.2 Self-Attention FLOPs (FlashAttention v2). The self-attention mechanism includes:

QKV Projection:

$$\text{FLOPs}_{\text{QKV}} = 6 \cdot B \cdot T_{in} \cdot d_{model}^2$$

Attention Score Computation (QK^T):

$$\text{FLOPs}_{QK^TV} = 2 \cdot B \cdot H \cdot T_{in}^2 \cdot \frac{d_{model}}{H} = 2 \cdot B \cdot T_{in}^2 \cdot d_{model}$$

Weighted Value Computation:

$$\text{FLOPs}_{\text{QKV}} = 2 \cdot B \cdot T_{in}^2 \cdot d_{model}$$

Output Projection:

$$\text{FLOPs}_{O} = 2 \cdot B \cdot T_{in} \cdot d_{model}^2$$

Summing all attention-related FLOPs:

$$\text{FLOPs}_{\text{attention}} = 8 \cdot B \cdot T_{in} \cdot d_{model}^2 + 4 \cdot B \cdot T_{in}^2 \cdot d_{model}$$

4.2.3 MoE Feedforward Network FLOPs. The MoE MLP consists of:
Router Computation:

$$\text{FLOPs}_{\text{router}} = B \cdot T_{in} \cdot d_{model} \cdot E$$

Expert Selection and Dispatching:

$$\text{FLOPs}_{\text{selection}} = B \cdot T_{in} \cdot E \cdot \log_2 E$$

Expert Computation:

$$\begin{aligned}
\text{FLOPs}_{\text{gate}} &= B \cdot T_{in} \cdot A \cdot d_{model} \cdot d_{ff} \\
\text{FLOPs}_{\text{up}} &= B \cdot T_{in} \cdot A \cdot d_{model} \cdot d_{ff} \\
\text{FLOPs}_{\text{GeLU}} &= B \cdot T_{in} \cdot A \cdot d_{ff} \cdot C_{GeLU} \\
\text{FLOPs}_{\text{gating}} &= B \cdot T_{in} \cdot A \cdot d_{ff} \\
\text{FLOPs}_{\text{down}} &= B \cdot T_{in} \cdot A \cdot d_{ff} \cdot d_{model}
\end{aligned}$$

Expert Output Combination:

$$\text{FLOPs}_{\text{combination}} = B \cdot T_{in} \cdot A \cdot d_{model}$$

Summing all MoE FLOPs:

$$\begin{aligned}
\text{FLOPs}_{\text{MoE}} = {} & B \cdot T_{in} \cdot d_{model} \cdot E + B \cdot T_{in} \cdot E \cdot \log_2 E \\
& + 2 \cdot B \cdot T_{in} \cdot A \cdot d_{model} \cdot d_{ff} + B \cdot T_{in} \cdot A \cdot d_{ff} \cdot C_{GeLU} \\
& + B \cdot T_{in} \cdot A \cdot d_{ff} + B \cdot T_{in} \cdot A \cdot d_{ff} \cdot d_{model} \\
& + B \cdot T_{in} \cdot A \cdot d_{model}
\end{aligned}$$

4.2.4 Residual Connection FLOPs. Each residual connection involves an element-wise addition over d_{model} per token. With two residuals (one after attention and one after MoE MLP), we get:

$$\text{FLOPs}_{\text{residuals}} = 2 \cdot B \cdot T_{in} \cdot d_{model}$$

4.2.5 Total FLOPs for a Transformer Block with MoE. Summing all contributions:

$$\begin{aligned}
\text{FLOPs}_{\text{total}} = {} & \text{FLOPs}_{\text{LN}} + \text{FLOPs}_{\text{attention}} + \text{FLOPs}_{\text{MoE}} + \text{FLOPs}_{\text{residuals}} \\
= {} & 8 \cdot B \cdot T_{in} \cdot d_{model} + (8 \cdot B \cdot T_{in} \cdot d_{model}^2 + 4 \cdot B \cdot T_{in}^2 \cdot d_{model}) \\
& + \Big(B \cdot T_{in} \cdot d_{model} \cdot E + B \cdot T_{in} \cdot E \cdot \log_2 E \\
& + 2 \cdot B \cdot T_{in} \cdot A \cdot d_{model} \cdot d_{ff} + B \cdot T_{in} \cdot A \cdot d_{ff} \cdot C_{GeLU} \\
& + B \cdot T_{in} \cdot A \cdot d_{ff} + B \cdot T_{in} \cdot A \cdot d_{ff} \cdot d_{model} + B \cdot T_{in} \cdot A \cdot d_{model} \Big) \\
& + 2 \cdot B \cdot T_{in} \cdot d_{model}
\end{aligned}$$

This provides the total computational complexity of a Transformer block incorporating Mixture of Experts.

Theoretical Compute Time:

$$T_{\text{compute}} = \frac{\text{FLOPs}_{\text{total}}}{\text{TFLOPS} \times 10^{12}} \tag{1}$$

5 FLOP CALCULATION - DECODE PHASE

5.1 Transformer Block - Dense

5.2 Transformer Block - MoE

6 COMMUNICATION

We use Tensor Parallelism (TP) for all layers, including experts in MoE layers within a single transformer block and Pipeline parallelism (PP) to split layers across different nodes. TP is a technique to shard individual model weights across multiple GPUs, splitting tensors along each head (for self-attention layers) and a group of neurons (in the MLP layer) to distribute memory and computation. During forward pass, GPUs communicate partial results through synchronized operations like all-reduce to combine outputs. We divide the communication cost into inter-GPU and intra-node. All the FC layers within a transfer block (self-attention and MLP) are present on the same node and inter GPU communication occurs between weights and activations. Different transformer layers ar are present on different nodes and hence only activations are sent from one node to another.

6.1 Prefill

During the prefill phase, all the input tokens are processed in parallel. The total communication cost (in bytes) between GPUs for tensor parallelism for the input sequence of token length T_{in} can be expressed as:

$$t_{\text{comms\_prefill}} = \frac{2 \cdot 2 \cdot 2 \cdot B \cdot T_{\text{in}} \cdot d_{\text{model}} \cdot (P-1)}{P} \tag{2}$$

There exists one broadcast operation to send input activation to all GPUs and one all-reduce operation to collect activations from all GPUs. The first factor 2 is for two bytes (FP16), the second factor 2 is for self-attention and MLP block and the third factor 2 is for all reduce and broadcast operations.

6.2 Decode

During the decode phase, only one token is processed throughout the layer. Therefore, the total communication for one token can be expressed

$$t_{\text{comms\_single\_token}} = \frac{2 \cdot 2 \cdot 2 \cdot B \cdot d_{\text{model}} \cdot (P-1)}{P} \tag{3}$$

The overall communication between GPUs during the decode phase for all the T_{out} tokens is equal to :

$$t_{\text{comms\_decode}} = \frac{2 \cdot 2 \cdot 2 \cdot B \cdot T_{\text{out}} \cdot d_{\text{model}} \cdot (P-1)}{P} \tag{4}$$

6.3 Intra Node - Prefill

During the prefill phase, we move only the output activations once, in the case of data transfer from one node to the other node, which is equal to the following in bytes (FP16):

$$t_{\text{comms\_prefill}}(node2node) = 2 \cdot B \cdot T_{\text{in}} \cdot d_{\text{model}} \tag{5}$$

6.4 Intra Node - Decode

During the decode phase, only one token is processed and hence the communication cost between nodes

$$t_{\text{comms\_decode}} = 2 \cdot B \cdot T_{\text{in}} \cdot d_{\text{model}} \tag{6}$$

The total communication cost to generate T_{out} tokens for a single layer is given as follows:

$$t_{\text{comms\_decode}} = 2 \cdot B \cdot T_{\text{out}} \cdot d_{\text{model}} \tag{7}$$

7 FIGURE OF MERIT (FOM)

7.1 Performance Metrics

(1) **Time to First Token (TTFT)** is the amount of time required to produce the first output token after receiving an input prompt. It represents how quickly the users can see the LLM's output after submitting their query. We measure TTFT by setting the maximum output to one token and recording the time to generate this output.

(2) **Inter Token Latency (ITL)** refers to the average time interval between generating consecutive tokens. It measures how quickly the model can produce each subsequent token after the previous one.

$$\text{ITL} = \frac{(\text{End-to-End Latency} - \text{TTFT})}{\text{Batch Size x (Output Tokens-1)}} \tag{8}$$

(3) **Throughput** is a key indicator of a hardware's processing efficiency. It provides insight into the model's capacity to handle sequences and batches. Throughput can be defined as the total number of tokens (both input and output) processed by the hardware per second. We first calculate the end-to-end latency, the time elapsed between the input prompt provided to LLM, and the generation of the final output token. We convert latency to throughput using Equation 9.

$$\text{throughput} = \frac{\text{Batch Size} \times (\text{Input} + \text{Output Tokens})}{\text{End-to-End Latency}} \tag{9}$$

7.2 LLM - Dense

7.3 LLM - MoE

REFERENCES