LLM Inference FOM

ACM Reference Format:
. 2025. LLM Inference FOM. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 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
© 2025 Association for Computing Machinery.
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:

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

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:

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

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.

$$\text{Model Memory} = q\cdot N\cdot\left(2\cdot d_{\text{model}}^2 + 2\cdot H\cdot\frac{d_{\text{model}}^2}{G} + 2d_{\text{model}} + 3\cdot d_{\text{model}}\cdot d_{\text{ff}}\right) + q\cdot(V\cdot d_{\text{model}} + L\cdot d_{\text{model}})$$

Here:

- The factor of q accounts for quantization per parameter (for FP16 precision, q = 2 and for Int8, q = 1).
- The terms correspond to:
  - N: Number of layers.
  - $2d_{\text{model}}$ accounting for LayerNorm parameters per layer.
  - $2\cdot d_{\text{model}}^2 + 2\cdot H\cdot\frac{d_{\text{model}}^2}{G}$ accounts for self-attention layer
  - $3\cdot d_{\text{model}}\cdot d_{\text{ff}}$ accounts for MLP block
  - $V\cdot d_{\text{model}}$ account for input embedding
  - $L\cdot d_{\text{model}}$ accounts for output lm-head

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.

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

- The factor of q accounts for quantization per parameter (for FP16 precision, q = 2 and for Int8, q = 1).
- 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:

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

- The factor of q accounts for quantization per parameter (for FP16 precision, q = 2 and for Int8, q = 1).
- The factor of 2 accounts for both keys and values.
- H: Number of attention heads.
- G: Number of query groups in GQA

3.4 Cmemory

$$C_{memory} = \frac{DeviceMemory}{ModelMemory + ActivationMemory + KVCacheMemory}$$

4 FIGURE OF MERIT (FOM) METRICS

4.1 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.

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

(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 the equation below:

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

5 FOM EQUATIONS

5.1 Option 1 (Including Memory and Communication)

If T is the end-to-end latency or time to solution, Figure of Merit can be defined as follows:

$$\text{FOM} = \frac{\alpha C_{memory} + \beta C_{communication}}{T}$$

5.2 Option 2 (Excluding Memory and Communication)

Prefill phase computation.

$$C_{\text{prefill}} = B\times N\times T_{\text{in}}\left(4d_{\text{model}}^2 + \frac{8d_{\text{model}}^2}{h} + 2d_{\text{model}}T_{\text{in}}\right) \quad (1)$$

B batch size (concurrent requests)
N number of Transformer layers
T_{in} number of input tokens
d_{model} model/hidden dimension
h number of attention heads

(1) **Feed-forward network (FFN)** $4d_{\text{model}}^2$: first projection $d_{\text{model}} \rightarrow 4d_{\text{model}}$ and second projection $4d_{\text{model}} \rightarrow d_{\text{model}}$.

(2) **Multi-head attention (MHA)** $\frac{8d_{\text{model}}^2}{h}$: Q,K,V projections ($3d_{\text{model}}^2$), output projection ($1d_{\text{model}}^2$), plus inter-head overhead ($\sim 4d_{\text{model}}^2/h$).

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_{gating}	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.

(3) **Attention scores** $2d_{\text{model}}T_{\text{in}}$: computing $QK^{\top}$ for every token pair.

Figure-of-Merit (FOM).

$$\text{FOM} = \frac{C_{\text{prefill}} + \text{Total}_{C_{\text{decode}}}}{\text{End-to-end Latency}} \tag{2}$$

Total$_{C_{\text{decode}}}$ sum of C_{decode} over all output tokens.
End-to-end Latency wall-clock time from receiving the prompt to finishing generation.

Key insights.

- Prefill cost grows linearly with input length.
- Decode cost rises as the KV cache expands.
- $2d_{\text{model}}T_{\text{in}}$ dominates prefill attention.
- $2d_{\text{model}}(T_{\text{in}}+j)$ dominates decode attention.
- FOM highlights the need to optimize both phases.

Expanding Total$_{C_{decode}}$.

$$\text{Total}_{C_{\text{decode}}} = \sum_{i=0}^{T_{\text{out}}-1} C_{\text{decode}}(i),$$

$$C_{\text{decode}}(i) = BN\left(4d_{\text{model}}^2 + \frac{8d_{\text{model}}^2}{h} + 2d_{\text{model}}(T_{\text{in}}+i)\right).$$

$$\text{Total}_{C_{\text{decode}}} = BN\sum_{i=0}^{T_{\text{out}}-1}\left(4d_{\text{model}}^2 + \frac{8d_{\text{model}}^2}{h} + 2d_{\text{model}}(T_{\text{in}}+i)\right)$$

$$= BNT_{\text{out}}\left(4d_{\text{model}}^2 + \frac{8d_{\text{model}}^2}{h} + 2d_{\text{model}}T_{\text{in}}\right) + BN2d_{\text{model}}\sum_{i=0}^{T_{\text{out}}-1} i.$$

Because

$$\sum_{i=0}^{T_{\text{out}}-1} i = \frac{T_{\text{out}}(T_{\text{out}}-1)}{2},$$

we have

$$\text{Total}_{C_{\text{decode}}} = BNT_{\text{out}}\left(4d_{\text{model}}^2 + \frac{8d_{\text{model}}^2}{h} + 2d_{\text{model}}T_{\text{in}}\right)$$

$$+ \text{B N d}_{\text{model}}T_{\text{out}}(T_{\text{out}}-1)$$

$$= \text{B N T}_{\text{out}}\left(4d_{\text{model}}^2 + \frac{8d_{\text{model}}^2}{h} + 2d_{\text{model}}T_{\text{in}} + d_{\text{model}}(T_{\text{out}}-1)\right)$$

$$= BNT_{\text{out}}\left(4d_{\text{model}}^2 + \frac{8d_{\text{model}}^2}{h} + 2d_{\text{model}}T_{\text{in}} + d_{\text{model}}T_{\text{out}} - d_{\text{model}}\right).$$

$$\begin{aligned}\mathrm{FOM} &= \frac{C_{\text{prefill}} + \text{Total}_{C_{\text{decode}}}}{\text{End-to-end Latency}} \\ &= \frac{BNT_{\text{in}}\left(4d_{\text{model}}^2 + \frac{8d_{\text{model}}^2}{h} + 2d_{\text{model}}T_{\text{in}}\right)}{\text{End-to-end Latency}} \\ &\quad + \frac{BNT_{\text{out}}\left(4d_{\text{model}}^2 + \frac{8d_{\text{model}}^2}{h} + 2d_{\text{model}}T_{\text{in}} + d_{\text{model}}T_{\text{out}} - d_{\text{model}}\right)}{\text{End-to-end Latency}}\end{aligned} \tag{3}$$

5.3 Simplified Version

$$C_{\text{prefill}} = BN\left(6d_{\text{model}}^2 T_{\text{in}} + d_{\text{model}}T_{\text{in}}^2\right)$$

$$C_{\text{decode}} = BNT_{\text{out}} \cdot 6d_{\text{model}}^2$$

$$\mathrm{FOM} = \frac{C_{\text{prefill}} + C_{\text{decode}}}{End-to-EndLatency}$$

REFERENCES