Performance modeling and projection of Transformer Large Language model

Huihuo Zheng

April 8, 2025

Abstract

Large language models (LLM) has become one of the important technics in science and engineering, as well as social science and our daily life. It is not too much to anticipate that it will eventually penetrate every area of science discipline according to the current trend of development and application. We will see more ALCF users using LLM in their scientific campaign. Therefore, we select LLM as one of the ALCF-4 AI benchmarks. The goal of this study is to build performance modeling and projection for LLM pretraining. In this study, we systematically investigate the computation, communication, memory access pattern of LLMs and provide a detailed performance modeling and performance projection. The conclusion of this study will be very useful for designing future supercomputers to support LLM workloads.

1 Terms and definition

There are a lot of terms and variables for model configuration, system setup, hyper-parameters for training. We list below the ones that are relevant to the performance modeling.

- **Embedding dimension (d)**: each token is represented as a vector in a high dimensional space. The dimension of that space is called embedding dimension.
- **Sequence length (s)**: each sample in the training is composed of a continuous list of tokens. The number of token in a sample is called sequence length.
- **Input (X)**: a input sample is a 2-D tensor of $[s, d]$
- **Number of attention head (h)**: this is the number of Q, K, V matrices in the multihead attention; the dimension of each matrix is $[d, d_k]$ where $d_k = d/h$.
- **Tensor parallelism size TP**: number of ranks in the tensor parallel group.
- **Pipeline parallelism size PP**: number of ranks in the pipeline parallel group.
- **Data parallelism size DP**: number of ranks in the data parallel group.

2 Forward propagation

In each of the transformer layer, there are two parts (1) self-attention layer and (2) multilayer perceptrons (MLP).

2.1 Self-attention

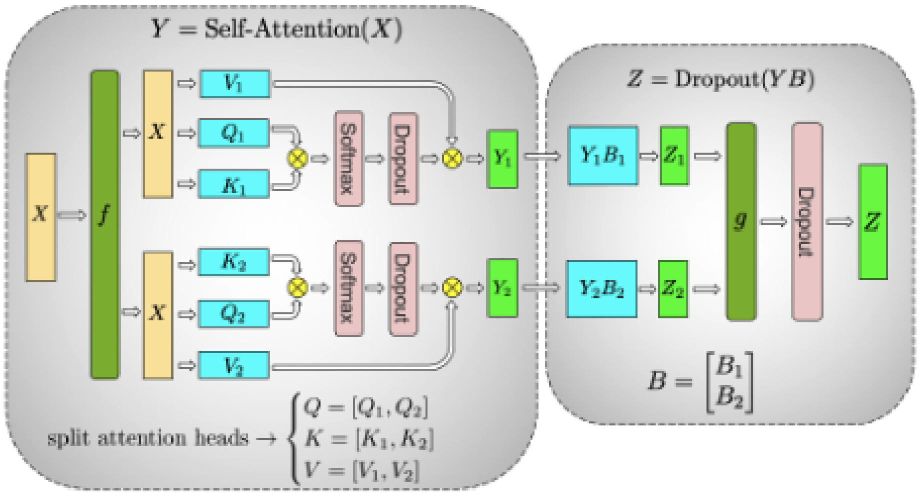

Figure 1: Self-attention layer

We assume the input matrix is X of $s \times d$ where s is the sequence length and d is the hidden dimension. In the self-attention layer, we have

$$Y = \text{softmax}(QK^T)V\,, \tag{1}$$

where Q, K, V are the query, key, and value matrices,

$$V = XW_V, \quad Q = XW_Q, \quad K = XW_K\,. \tag{2}$$

In multi-head attention, instead of one single Q, K, V matrix of dimension $[d, d]$, we have h sets of matrices Q_i, K_i, and V_i, each is of dimension $s \times d_k$, where $d_k = d/h$, which is the result of XW_M, for $M = Q, K, V$.

$$[s, d] \times [d, d_k] \equiv [s, d_k] \tag{3}$$

Therefore, in this step, we have the total number of Gemm operation is

$$\#FLOP = 3 \times s \times d \times d/h \times h/TP = \frac{3sd^2}{TP} \tag{4}$$

The matmul size is $[s, d] \times [d, d/h]$, and we have $3 \times h/TP$ of them. Note that in computing Q_i, K_i, and V_i, different heads are distributed to different TP ranks, each owns h/TP heads.

For the softmax part, for computing $Q_iK_i^T$, we have sd_ks, and we have h/TP of them per rank. The total FLOPs from gemm operation is $sd_ks \times h/TP = s^2d/TP$.

Now, in multiplying V_i, we have another s^2d/TP, because of matrix multiplication: $[s, s] \times [s, d_k]$ The output is

$$Y_i = \text{softmax}(Q_i K_i^T) V_i \tag{5}$$

The dimension of Y_i is $[s, d_k]$. Then the next step is go through a fully connected layer,

$$[Y_1, Y_2, ...Y_h] \times \begin{bmatrix} B_1 \\ B_2 \\ ... \\ B_h \end{bmatrix} = \sum_i^h Y_i B_i \tag{6}$$

Note that the Y matrix is distributed to different ranks in the tensor parallel group in column wise way, where as the weight matrix B for the fully connected layer is distributed to different ranks in a row wise way. The summation is an all reduce operations with matrix size $[s, d]$ in the TP group. The gemm operations involved is $[s, d_k] \times [d_k, d]$, in which we have h/TP of them. The total flop is

$$\#FLOP = s \times d_k \times d \times h/TP = \frac{sd^2}{TP} \tag{7}$$

Now let us summarize the computation / communication involved in the self-attention Total number of flops is $4sd^2/TP + 2s^2d/TP$

stage	operation	FLOP	matrix
Generating Q_i, K_i, V_i	gemm	$3sd^2/TP$	$[s, d] \times [d, d_k]$
$Q_i K_i^T$	gemm	s^2d/TP	$[s, d_k] \times [d_k, s]$
softmax	exp	s^2h/TP	[s, s]
softmax $\times V$	gemm	s^2d/TP	$[s, s] \times [s, d_k]$
FCL	gemm	sd^2/TP	$[s, d_k] \times [d_k, d]$
FCL	allreduce	N/A	$[s, d]$

Table 1: Operations involved in self-attention

2.2 MLP

The output Z from self-attention layer in Fig. 1 will be passed to the multilayer perceptrons in Fig. 2 as X. The input dimension is $[s, d]$, where s is the sequence length and d is the hidden dimension.

The shapes of A and B are $A \equiv [d, 4d]$, and $B \equiv [4d, d]$ respectively. We use column parallelism to distributed A across different accelerators, $A_i \equiv [d, 4d/TP]$; whereas, we use row parallelism to distribute B: $B_i \equiv [4d/TP, d]$. For the first hidden layer

$$Y_i = \text{GeLU}(XA_i) \equiv [s, 4d/TP]\,. \tag{8}$$

It involves a matrix multiplication of $[s, d] \times [d, 4d/TP]$, which has $\#FLOP = 4sd^2/TP$. For the second step, we have

$$Z = [Y_1, Y_2, ...Y_{TP}] \times \begin{bmatrix} B_1 \\ B_2 \\ ... \\ B_{TP} \end{bmatrix} = \sum_i Y_i B_i \equiv [s, d]\,. \tag{9}$$

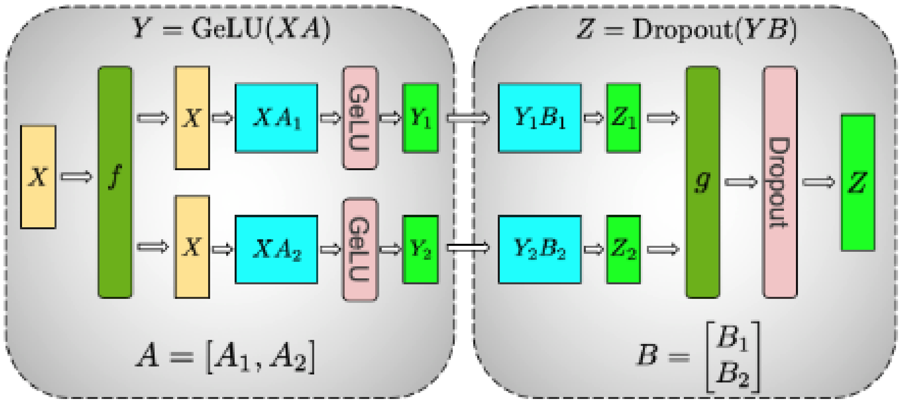

Figure 2: MLP layer

For each rank, the gemm operation involved is a multiplication of $[s, 4d/TP] \times [4d/TP, d] \equiv [s, d]$, which as $\#FLOP = 4sd^2/TP$. We also need to perform an allreduce to sum over Z_i. The message for the allreduce operation is a matrix of $[s, d]$.

The total number of flops for MLP is $8sd^2/TP$. Adding the flops from self-attention layer, we get the total flops for the forward propagation of a transformer layer: $(12sd^2 + 2s^2d)$/TP.

3 Backward propagation

3.1 Backpropagation algorithm

For backward pass, let us first consider in general case, how to compute the gradient of the loss function with respect to a weight matrix, $\frac{\partial L}{\partial W}$ assuming the forward propagation is as follows,

$$\mathbf{X^l} = \sigma(\mathbf{X^{l-1}W^l} + \mathbf{b}) \,, \tag{10}$$

where l is the layer index, and σ is the activation function which could be sigmoid, ReLU, GeLU, etc. With the help of chain rule, one can get the gradient with respect to W^l from the gradient with respect to the activations x^l,

$$\frac{\partial L}{\partial W^l} = \frac{\partial L}{\partial X^l}\frac{\partial X^l}{\partial W^l} \,. \tag{11}$$

So in calculating the derivative, it involves two steps corresponding to calculating the derivatives with respect to the activations and the weight matrices respectively.

Derivative with respect to activations

Let us first figure out the gradient with respect to the activation x^l. By using the chain rule, we have,

$$\frac{\partial L}{\partial x^{l-1}_{ij}} = \sum_k \frac{\partial L}{\partial x^l_{ik}}\frac{\partial x^l_{ik}}{\partial x^{l-1}_{ij}} = \frac{\partial L}{\partial x^l_j}\sigma' W_{jk} \,. \tag{12}$$

For simplicity, let us define $\underline{x}_{ij} = \frac{\partial L}{\partial x_{ij}}$, we can rewritten Eq. (12) as

$$\underline{x}^{l-1} = \sigma' \underline{x}^{l} W^T \, . \tag{13}$$

We find that once we know $\underline{x}^l$, we can obtain $\underline{x}^{l-1}$ from that. The weight matrix is the same as the forward propagation for that layer except that it is transposed as shown in Fig. 3 (a) and (b).

Now for the computational cost for this step, suppose the dimension of X^{l-1} is $[m, n]$, and W is $[n, k]$, then we have the output X^l is of shape $[m, k]$. The number of flops for forward pass is $m \times n \times k$, which is from the matmul of $[m, n] \times [n, k] \equiv [m, k]$ [see Fig. 3(a)]; and the number of flops for the first step of back propagation is $m \times k \times n$, which is from the matmul of $[m, k] \times [k, n] \equiv [m, n]$. We find that this is exactly the same as the forward propagation.

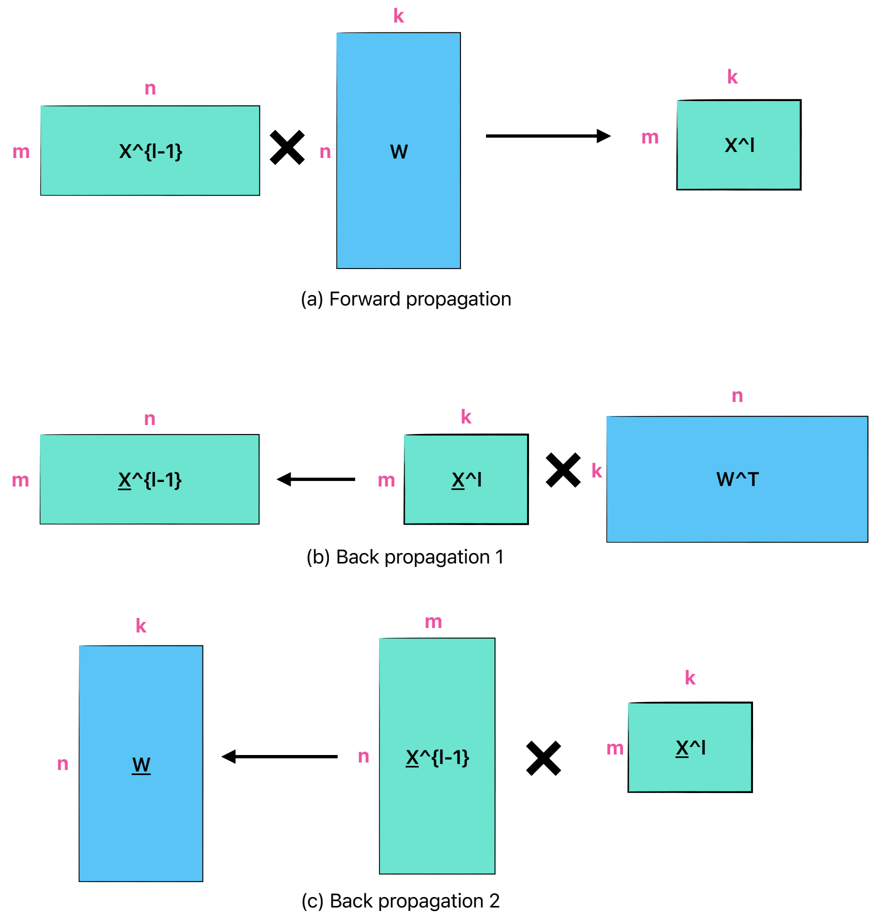

Figure 3: Schematic plot for forward propagation and backward propagation

Derivative with respect to weights

Now let us come to the derivative with respect to weight. This can be obtained from the derivative with respect to the activation by using the chain rule,

$$\frac{\partial L}{\partial W_{ij}} = \sum_k \frac{\partial L}{\partial X^l_{kj}} \frac{\partial X^l_{kj}}{\partial W_{ij}} = \sigma' \sum_k x^{l-1}_{ki} \underline{x}^l_{kj} \, , \tag{14}$$

which can be denoted as,

$$\underline{W} = \sigma' [X^{l-1}]^T \underline{X}^l \, . \tag{15}$$

The matrix mulplication involved is

$$[k, n] \equiv [k, m] \times [m, n] \, . \tag{16}$$

The flops is again $k \times n \times k$.

We conclude that the total cost for backward propagation from Layer l to Layer $l-1$ is $2 \times m \times n \times k$.

To summarize, if the forward propagation is $X^{l-1}[m, n] \times W[n, k] \to X^l[m, k]$.

- Computing $\underline{X}^{l-1}$: $\underline{X}^{l-1}[m, k] \leftarrow \underline{X}^l[m, k] \times W^T[k, n]$
- Computing $\underline{W}$: $\underline{\mathrm{W}}[k, n] \leftarrow [X^{l-1}]^T[k, m] \times \underline{X}^l[m, n]$

The whole process is schematically shown in Fig. 3.

3.2 Back propagation of a transformer layer

3.2.1 MLP

Now let us go backward from MLP to self-attention. For MLP in Fig. 2, the back propagation concerning $Z_i = Y_i B_i$,

$$\frac{\partial L}{\partial Y_i} = \frac{\partial L}{\partial Z} \times \frac{\partial Z}{\partial Y_i} \longrightarrow \underline{Y}_i = \underline{Z}_i B_i^T \longrightarrow [s, \frac{4d}{TP}] = [s, d] \times [d, \frac{4d}{TP}] \, . \tag{17}$$

$$\frac{\partial L}{\partial B_i} = \frac{\partial L}{\partial Z} \times \frac{\partial Z}{\partial B_i} \longrightarrow \underline{B}_i = Y_i^T \underline{Z}_i \longrightarrow [\frac{4d}{TP}, d] = [\frac{4d}{TP}, s] \times [s, d] \tag{18}$$

Following the same approach, we can find out the back propagation for $Y = GeLU(XA_i)$

$$\underline{X} = \sum_i \mathrm{GeLU}' \underline{Y}_i A_i^T \longrightarrow [s, d] \equiv [s, \frac{4d}{TP}] \times [\frac{4d}{TP}, d] \, , \tag{19}$$

$$\underline{A}_i = \mathrm{GeLU}' X^T \underline{Y}_i \longrightarrow [d, \frac{4d}{TP}] \equiv [d, s] \times [s, \frac{4d}{TP}] \, . \tag{20}$$

We need to have an allreduce here to get X by a summation over the results from all ranks.

The total cost for this is $16d^2 s/TP$, and an allreduce of $[s, d]$ over all the ranks in the TP group.

3.2.2 self-attention

For self-attention layer, for the fully connected layer $Z_i = Y_i B_i$, we have

$$\begin{aligned} \underline{Y}_i &= \sigma' \underline{Z} B_i^T \longrightarrow [s, d_k] \equiv [s, d] \times [d, d_k] && (21) \\ \underline{B}_i &= \sigma' Y_i^T \underline{Z} \longrightarrow [d_k, d] \equiv [d_k, s] \times [s, d] && (22) \end{aligned}$$

Now for the complicated part, since $Y_i = \text{softmax}(Q_i K_i^T) V_i$. We have

$$\underline{V}_i = [\text{softmax}(Q_i K_i^T)]^T \underline{Y}_i \tag{23}$$

In order to figure out the derivative with respect to Q_i and K_i, let us define $M = Q_i K_i^T$ and $S = \text{softmax}(M)$. We have

$$\underline{S} = \underline{Y}_i^T V_i^T \, . \tag{24}$$

Let us figure out the softmax

$$S_{ij} = \frac{\exp M_{ij}}{\sum_k \exp M_{ik}} \, . \tag{25}$$

We have

$$\frac{\partial S_{ij}}{\partial M_{ik}} = \delta_{jk} S_{ij} - S_{ij} S_{ik} \tag{26}$$

Therefore, we have

$$\frac{\partial L}{\partial M_{ik}} = \sum_j \frac{\partial L}{\partial S_{ij}} (\delta_{jk} S_{ij} - S_{ij} S_{ik}) \tag{27}$$

We can see how $\underline{M}$ can be obtained from $\underline{S}$. We can also figure out

$$\underline{Q}_i = \underline{M} K_i, \quad \underline{K}_i = \underline{M}^T Q_i \tag{28}$$

And from K_i, Q_i, and V_i, we can get the gradients with respect to W_Q W_K, and W_V.

Overall, we can see that, the back propagation has about two times FLOPs of the forward.

4 Memory Footprint

The model parameters we have is W_K, W_Q, W_V, W_o from the self-attention layer, all of which are $[d, d]$ shape; For MLP, we have two weight matrices $[4, 4d]$, and $[4d, d]$, therefore, the total number of parameters in a transformer layer $12Ld^2$ where L is the number of layers in the model.

One also need to store the activations, this will be proportional to the batch size. All the inputs and activations are listed below:

- X, Y, Z in self-attention (Fig. 1): $s \times d$

- Q, K, V in self-attention (Fig. 1): $s \times d/TP$
- Z in MLP (Fig. 2): $s \times d$
- Y in MLP (Fig. 2): $s \times 4d/TP$

Notice that the total number of parameters are

$$L \times B \times (sd + sd + sd + 3sd/TP + 4sd/TP) = LBsd(3 + 7/TP)\,, \tag{29}$$

where B is the microbatch size, and L is the number of layers. Note that we have avoided the double counting: Z in the self-attention will be X in MLP, and also Z in MLP will be the X in the next layer of self-attention.

For the optimization process, we have to store the gradient and the momentum. In mixed precision training, we need 6 bytes for each model parameter, 4 to save the model in fp32, and 2 to use in computation in fp16. We need 4 bytes per parameter for Optimizer states to save the momentum in fp32 (Adam Optimizer). We need to save one fp32 gradient value for each parameter.

5 Definition of the figure of Merit

5.1 Time to Solution

We have the total compute time should be

$$t = \alpha \frac{bL(36sd^2 + 6s^2d)}{PP \cdot TP} + \beta \frac{4bLsd}{PP}\,, \tag{30}$$

where α is the FLOP/s per GPU, and β is the intranode allreduce bandwidth.

The first term is from the computation, and the second time is from reduction of $[s, d]$ matrix for for 4 times. If we have data parallelism, we also have an allreduce within each DP groups. We will have $TP * PP$ allreduce concurrently in this step. If it is implemented in a good way, the allreduce of previous layer can be overlap with the computation of next layer. Therefore, the DP communication can be neglected.

5.2 Figure of Merit

The FOM can be defined as the ratio between the complexity of the problem and the time to solution.

$$\text{FOM} = \frac{(6sd^2 + s^2d)bL}{T}\,. \tag{31}$$

Notice that here we only consider the computation part in the complexity. The communication comes into play only because we cannot