大模型推理瓶颈及极限理论值分析

原创 喜欢卷卷的瓦力 瓦力算法学研所 2024年07月31日 10:02 上海

◈ 技术总结专栏 ◈

作者：喜欢卷卷的瓦力

本篇讲述大模型推理机制及其极限理论值分析。

大家在做大模型推理优化的时候，可能都会考虑一个核心问题：**推理的极限在哪里?**

本文基于文章 [LLM inference speed of light] 及ArthurChiao的中文版文章做了详细解读，分析了大模型推理的速度瓶颈及量化评估方式，希望对小伙伴们理解大模型推理内部工作机制与推理优化有帮助。

下面是一个快捷目录。由于内容比较多，本篇主要介绍一到四，剩下的部分会在下一篇结合一些落地应用进行讲解。

一、常见的浮点运算单位介绍

这里简单列举一些比较常见的单位：

- FLOPs： floating point of operations的缩写，是浮点运算次数，可以用来衡量算法/模型复杂度。

- 一个GFLOPS（gigaFLOPS）= 每秒十亿（=10^9）次的浮点运算
- 一个TFLOPS（teraFLOPS） = 每秒一万亿（=10^12）次的浮点运算

二、推理机制

1. 语言模型Teacher-forcing的特点：逐token生成，无法并行

当LLM生成文本时，是teacher-forcing，也就是逐个token生成的，我们将其看作一个函数：

- **输入**：一个token
- 输出：一组概率，每个概率对应词汇表中的一个token
- **推理过程**：使用概率来指导抽样，产生（从词汇表中选择）下一个 token 作为最终输出。

此处简单介绍一下词汇表。

词汇表（vocabulary）：通常由单词、单词片段、中文汉字等组成（这些都称为 token），一般会基于训练语料先基于bpe或者sentence piece等算法切分token生成。

文本生成过程就是不断重复以上过程。可以看出，在生成一个文本序列时，没有并行性的可能性。

2. 生成过程建模：矩阵乘法

广义上，当处理一个 token 时，模型执行两种类型的操作：

1） 矩阵-向量乘法

一个大矩阵（例如 8192x8192）乘以一个向量，得到另一个向量。

2） attention 计算

在生成过程中，模型不仅可以看到当前 token 的状态，还可以看到序列中所有之前 token 的内部状态 —— 这些状态被存储在 KV-cache 的结构中， 它本质上是**文本中每个之前位置的 key 向量和 value 向量的集合。**

attention 为当前 token 生成一个 query 向量，计算它与所有之前位置的 key 向量之间的点积， 然后归一化得到的一组标量，并通过对所有之前的 value 向量进行加权求和来计算一个 value 向量，使用点积得到最终得分。

三、瓶颈分析

上述的矩阵-向量乘法和attention 计算两步计算有一个重要的共同特征：从矩阵或 KV-cache 读取的每个元素，只需要进行**非常少量的浮点运算。**

- 矩阵-向量乘法对每个矩阵元素执行一次**乘加运算**（2 FLOPs）；
- attention 对每个 key 执行一次**乘加**，对每个 value 执行一次**乘加。**

1. 一些典型显卡的"算力-带宽"比

现代 CPU/GPU 的 **ALU 操作（乘法、加法）内存 IO*速度要快得多。**例如：

- AMD Ryzen 7950X：67 GB/s 内存带宽和 2735 GFLOPS，Flop:byte = 40:1
- NVIDIA GeForce RTX 4090：1008 GB/s 显存带宽和 83 TFLOPS，Flop:byte = 82:1
- NVIDIA H100 SXM：3350 GB/s 内存带宽和 67 TFLOPS， 对于矩阵乘法，tensor core 提供 ~494 TFLOPS 稠密算力，Flop:byte = 147:1。

对于 FP16/FP8 等精度较低的浮点数，比率更夸张：

- H100 TensorCore 对于 dense FP8 矩阵的理论吞吐量为 1979 TFLOPS，FLOP:byte = 590:1。

在这些场景中，无论是否使用 TensorCore 或使用什么浮点格式，ALU 都非常充足。

2. 瓶颈：访存带宽

因此，transformer 这种**只需要对每个元素执行两次操作**的场景，必定受到访存带宽的限制。所以，基于下面几个因素，

1）模型参数配置

2）KV-cache 大小

3）访存带宽

四、以Mistral-7B为例，计算极限推理延迟

Mistral7B由MistralAI发布，采用分组查询注意力（GQA）和滑动窗口注意力，超越了13B和34B参数模型。同时，Mistral8x7B是首个开源的MoE大模型，具有8个7B参数的专家和高效的稀疏处理。

1. 参数的组成与计算

Mistral-7B 有 72 亿参数（所有矩阵元素的总数是 72 亿个）。具体参数配置如图

Parameter	Value
dim	4096
n_layers	32
head_dim	128
hidden_dim	14336
n_heads	32
n_kv_heads	8
window_size	4096
context_len	8192
vocab_size	32000

参数的组成如下：

1) embedding 矩阵

计算公式：dim * vocab_size

4096 * 32000 = 131M

矩阵-向量乘法中不会使用这整个大矩阵，**每个 token 只读取这个矩阵中的一行**，因此数据量相对很小，后面的带宽计算中将忽略这个

2) 计算与 attention 相关的向量

用的GQA，n_heads是Q的头数，n_kv_heads是K和V的头数

计算公式：n_layers * (window_size * (head_dim * n_heads + head_dim * n_kv_heads * 2))

32 * (4096 * (128 * 32 + 128 * 8 * 2) + 4096 * 128 * 32) = 1342M

3) 计算通过 feed-forward 转换 hidden states的过程

Q, K, V分别算所以乘3

计算公式：n_layers * (dim * hidden_dim * 3)

32 * (4096 * 14336 * 3) = 5637M

4) 将 hidden states 转换为 token 概率

计算公式：dim * vocab_size

4096 * 32000 = 131M

这与 embedding 矩阵不同，会用于矩阵乘法。

以上加起来，大约有 7111M(~7B) “活跃”参数用于矩阵乘法。

2. 计算一个 token 所需加载的数据量

1）总数据量
如果模型使用FP16作为矩阵元素的类型，那**每生成一个 token，需要加载到 ALU 上的数据量**：

7111M params * 2Byte/param = ~14.2 GB

虽然计算下一个 token 时每个矩阵都可以复用，但硬件缓存的大小通常只有几十 MB， 矩阵无法放入缓存中，因此我们可以断定，这个生成（推理）过程的速度不会快于显存带宽。

attention 计算需要读取当前 token 及前面上下文中所有 tokens 对应的 KV-cache， 所以**读取的数据量取决于生成新 token 时模型看到多少前面的 token**， 这包括

- 系统提示词（通常对用户隐藏）
- 用户提示词
- 前面的模型输出
- 可能还包括长聊天会话中多个用户的提示词

2) KV-cache 部分的数据量
对于 Mistral，KV-cache

- 为每层的每个 key 存储 8 个 128 元素向量，
- 为每个层的每个 value 存储 8 个 128 元素向量，

这加起来，32层则每个 token 对应 32 * 128 * 8 * 2 = 65K 个元素；
如果 KV-cache 使用 FP16，那么对于 token number P，我们需要读取 P * 130 KB 的数据。例如， token number 1000 将需要从 KV-cache 读取 130MB 的数据。跟 14.2GB 这个总数据量相比，这 130MB 可以忽略不计了。、

3) 以 RTX 4090 为例，极限延迟计算

根据以上数字，现在可以很容易地计算出推理所需的最小时间。

例如，在 NVIDIA RTX 4090（1008 GB/s）上，

- 14.2GB (fp16) 需要 ~14.1ms 读取，因此可以预期对于位置靠前的 token， 每个 token 大约需要 14.1ms（KV-cache 影响可以忽略不计）。
- 如果使用 8bit权重，需要读取 7.1GB，这需要大约 7.0ms。

这些都是理论下限，代表了生成每个 token 的最小可能时间。

以上根据数学建模和计算得出了一些理论极限数字，那么下一篇继续接着看看这些理论极限的作用以及落地时如何优化。

想要获取技术资料的同学欢迎关注公众号，进群一起交流~

参考文献

[1] LLM inference speed of light—（https://zeux.io/2024/03/15/llm-inference-sol/）

[2] ArthurChiao's blog —（https://arthurchiao.art/blog/llm-inference-speed-zh/）

上一篇
为什么多模态大语言模型最近用BLIP2中Q-Former结构的变少了？

下一篇
视觉面经之一问：为什么DETR不需要NMS后处理？