

# 一、大模型部署:

软件工程领域: 部署通常指的是将研发完毕的软件投入使用的过程

人工智能领域: 模型部署是实现深度学习算法落地应用的关键步骤。简而言之, 就是将训练好的深度学习模型在特定环境中运行的过程

场景: CPU, 单GPU/TPU/NPU, 多卡集群

移动端/边缘端: 移动机器人, 手机...

挑战: 计算量巨大

LMDeploy 量化部署 LLM-VLM 实践

OpenMMLab bilibili

### 大模型部署面临的挑战

上海人工智能实验室  
Shanghai Artificial Intelligence Laboratory

#### 计算量巨大

- 大模型参数量巨大, 前向推理时需要进行大量计算。
- 根据InternLM2技术报告<sup>[1]</sup>提供的模型参数数据, 以及OpenAI团队提供的计算量估算方法<sup>[2]</sup>, 20B模型每生成1个token, 就要进行约406亿次浮点运算; 照此计算, 若生成128个token, 就要进行5.2万亿次运算。
- 20B算是大模型里的“小”模型了, 若模型参数规模达到175B (GPT-3), Batch-Size (BS) 再大一点, 每次推理计算量将达到千万亿量级。
- 以NVIDIA A100为例, 单张理论FP16运算性能为每秒77.97 TFLOPs<sup>[3]</sup> (77万亿), 性能捉襟见肘。

#### 大模型前向推理所需计算量计算公式<sup>[2]</sup>:

$$C_{\text{forward}} = 2N + 2n_{\text{layer}}n_{\text{ctx}}d_{\text{attn}}$$

注: 其中,  $N$  为模型参数量,  $n_{\text{layer}}$  为模型层数,  $n_{\text{ctx}}$  为上下文长度 (默认1024),  $d_{\text{attn}}$  为注意力输出维度。单位: FLOPs per Token

#### 大模型前向推理所需计算量估算 (InternLM2为例)<sup>[1]</sup>:

$N$	$n_{\text{layer}}$	$d_{\text{attn}}$	$C_{\text{forward}}$
1.8 B	24	2048	3.7 GFLOPs
7 B	32	4096	14.2 GFLOPs
20 B	48	6144	40.6 GFLOPs

[1] Cai Z, Cao M, Chen H, et al. InternLM2 Technical Report[J]. arXiv preprint arXiv:2403.17297, 2024.  
[2] Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models[J]. arXiv preprint arXiv:2001.08361, 2020.  
[3] <https://www.nvidia.com/>

二、①

②内存开销巨大

LMDeploy 量化部署 LLM-VLM 实践

OpenMMLab bilibili

## 大模型部署面临的挑战

上海人工智能实验室  
Shanghai Artificial Intelligence Laboratory

### 内存开销巨大

- 以FP16为例，20B模型仅加载参数就需40G+显存，175B模型(如GPT-3)更是需要350G+显存。
- 大模型在推理过程中，为避免重复计算，会将计算注意力(Attention)得到的KV进行缓存。根据InternLM2技术报告<sup>[1]</sup>提供的模型参数数据，以及KV Cache空间估算方法<sup>[2]</sup>，以FP16为例，在batch-size为16、输入512 tokens、输出32 tokens的情境下，仅20B模型就会产生10.3GB的缓存。
- 目前，以NVIDIA RTX 4060消费级显卡为例(参考零售价¥2399<sup>[3]</sup>)，单卡显存仅有8GB；NVIDIA A100单卡显存仅有80GB。

### KV Cache显存占用估算公式<sup>[2]</sup>:

$$M_{kvcache} = 4bn_{layer}d_{atts}(s+n)$$

注：其中，b为batch-size， $n_{layer}$ 为模型层数， $d_{atts}$ 为注意力输出维度，s为输入序列长度，n为输出序列长度。单位：字节(Byte)

### 前向推理KV Cache空间估算(InternLM2为例)<sup>[1]</sup>:

N	$n_{layer}$	$d_{atts}$	b	s	n	$M_{kvcache}$
1.8 B	24	2048	16	512	32	1.7 GB
7 B	32	4096	16	512	32	4.6 GB
20 B	48	6144	16	512	32	10.3 GB

[1] Cai Z, Cao M, Chen H, et al. InternLM2 Technical Report[J]. arXiv preprint arXiv:2403.17297, 2024.  
[2] <https://ghuailan.zhihu.com/question/624740065>  
[3] <https://www.nvidia.cn/enforce/graphics-cards/40-series/rtx-4060-4060ti/>

LMDeploy 量化部署 LLM-VLM 实践

OpenMMLab bilibili

## 大模型部署面临的挑战

上海人工智能实验室  
Shanghai Artificial Intelligence Laboratory

### 访存瓶颈

- 大模型推理是“访存密集”型任务。目前硬件计算速度“远快于”显存带宽，存在严重的访存性能瓶颈。
- 以RTX 4090推理175B大模型为例，BS为1时计算量为6.83TFLOPs，远低于82.58 TFLOPs的FP16计算能力；但访存量高达32.62 TB，是显存带宽每秒处理能力的30倍。

### 动态请求

- 请求量不确定。
- 请求时间不确定。
- Token逐个生成，生成数量不确定。

### GPT3-175B推理阶段计算访存比分析(输入1k, 输出250)<sup>[1]</sup>:

BS	计算量	访存量	计算访存比
1	6.83 TFLOPs	32.62 TB	0.20
8	55.37 TFLOPs	32.67 TB	1.67
16	112.3 TFLOPs	32.73 TB	3.43

### 常见GPU浮点运算性能与内存带宽<sup>[2]</sup>:

GPU	FP16算力	FP32算力	FP64算力	显存带宽	FP16算力/显存带宽
RTX 4090	82.58 TFLOPs	82.58 TFLOPs	1290 GFLOPs	1008 GB/s	81.92
A100 80G	77.97 TFLOPs	19.49 TFLOPs	9.746 TFLOPs	2039 GB/s	38.24
H100 80G	267.6 TFLOPs	66.91 TFLOPs	33.45 TFLOPs	1681 GB/s	159.2

[1] <https://cloud.baidu.com/article/919629>  
[2] <https://www.topcpu.net/>

③ 访存瓶颈

三、部署方法：

④ 模型剪枝：移除模型中不必要的或冗余的组件，比如参数，以使模型更高效，→ 减少存储需求，提高计算效率

非结构化剪枝  
结构化剪枝

②知识蒸馏：一种经典的模型压缩方法，核心思想是通过引导轻量化的学生模型“模仿”性能更好、结构更复杂的教师模型，在不改变学生模型结构的情况下提高模型性能。

③量化：将传统的浮点数的表示方法转化为整数或其他离散方式，以减轻深度学习模型的存储和计算负担

OpenMMLab bilibili

### 量化(Quantization)

量化技术将传统的表示方法中的浮点数转换为整数或其他离散形式，以减轻深度学习模型的存储和计算负担。

**量化感知训练(QAT) LLM-QAT<sup>[1]</sup>**

- 量化目标无缝地集成到模型的训练过程中。这种方法使LLM在训练过程中适应低精度表示。

**量化感知微调(QAF) PEQA<sup>[2]</sup>, QLORA<sup>[3]</sup>**

- QAF涉及在微调过程中对LLM进行量化。主要目标是确保经过微调的LLM在量化为较低位宽后仍保持性能。

**训练后量化(PTQ) LLM.int8<sup>[4]</sup>, AWQ<sup>[5]</sup>**

- 在LLM的训练阶段完成后对其参数进行量化。PTQ的主要目标是减少LLM的存储和计算复杂性，而无需对LLM架构进行修改或进行重新训练。

**通用公式：**

$$ZP = \frac{\min + \max}{2}$$
$$S = \frac{\max - \min}{255}$$

量化:  $q = \text{round}\left(\frac{f - ZP}{S}\right)$

反量化:  $f = q \times S + ZP$

**Reference:**

- [1] Liu Z, Oguz B, Zhao C, et al. Llm-qat: Data-free quantization aware training for large language models[J]. arXiv preprint arXiv:2305.17888, 2023.
- [2] Arshia F Z, Keyvanrad M A, Sadidpour S S, et al. PeQA: A Massive Persian Question-Answering and Chatbot Dataset[C]/2022 12th International Conference on Computer and Knowledge Engineering (ICCKE). IEEE, 2022: 392-397.
- [3] Dettmers T, Pagnoni A, Holtzman A, et al. Qlora: Efficient finetuning of quantized llms[J]. Advances in Neural Information Processing Systems, 2024, 36.
- [4] Dettmers T, Lewis M, Belkada Y, et al. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale[J]. Advances in Neural Information Processing Systems, 2022, 35: 30318-30332.
- [5] Lin J, Tang J, Tang H, et al. Awq: Activation-aware weight quantization for compression and acceleration[J]. arXiv preprint arXiv:2306.00978, 2023.

④. LMDeploy

OpenMMLab bilibili

### LMDeploy核心功能

**模型高效推理** 参考命令: `lmdeploy chat -h`


- TurboMind是LMDeploy团队开发的一款关于LLM推理的高效推理引擎。它的主要功能包括: LLaMa 结构模型的支持, continuous batch推理模式和可扩展的KV缓存管理器。

**模型量化压缩** 参考命令: `lmdeploy lite -h`

- W4A16量化(AWQ): 将FP16的模型权重量化为INT4, Kernel计算时, 访存量直接降为FP16模型的1/4, 大幅降低了访存成本。Weight Only是指仅量化权重, 数值计算依然采用FP16(需要将INT4权重反量化)。

**服务化部署** 参考命令: `lmdeploy serve -h`

- 将LLM封装为HTTP API服务, 支持Triton扩展。

 LMDeploy