五 LMDeploy量化部署笔记等

https://www.bilibili.com/video/BV1tr421x75B/

第五章涉及了LMDeploy的环境部署、模型对话、模型量化、模型服务和Python代码集成的全方位操作，以及利用LMDeploy进行视觉多模态大模型的运行和应用。

一、LMDeploy环境部署

Conda环境配置。

安装LMDeploy包。

二、模型对话

模型托管与推理引擎，包括使用Hugging Face社区，或国内的MindScope社区以及OpenXLab社区。

TurboMind推理引擎支持LLaMa结构模型，连续批处理推理模式和可扩展的KV缓存管理器。

LMDeploy可以自动将HF格式模型转换为TurboMind格式。

介绍了如何使用Transformer库运行InternLM2-Chat-1.8B模型。

展示了使用LMDeploy进行对话的操作，以及其推理速度相较于Transformer库的提升。

三、模型量化

介绍了量化的概念，包括计算密集和访存密集两个概念。

量化策略，包括KV8量化和W4A16量化。

设置最大KV Cache缓存大小，以及如何通过设置KV缓存大小来控制显存占用。

四、模型服务

封装模型推理服务，将其分为模型推理/服务、API Server和Client三个模块。

展示了如何启动API服务器，以及如何使用命令行和网页客户端连接API服务器。

五、Python代码集成

展示了如何将大模型推理集成到Python代码中。

展示了向TurboMind后端传递参数的例子。

六、拓展

使用LMDeploy运行视觉多模态大模型llava，并提供了本地运行和gradio接口运行的代码示例。

介绍了使用LMDeploy运行第三方大模型，以及比较LMDeploy与Transformer库推理速度的方法。

整体的系统介绍了LMDeploy环境的设置和应用，强调了其在模型量化、托管、服务及推理速度方面的优势，为用户提供了从环境部署到模型推理服务全链条的操作流程。还特别提到了如何使用LMDeploy处理多模态数据，并进行视觉任务的模型运行。

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[1]提供的模型参数数据，以及OpenAI团队提供的计算量估算方法[2]，**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[3] (**77万亿**)，性能捉紧。

大模型前向推理所需计算量计算公式[2]：

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

注：其中，N为模型参数量，n_{layer}为模型层数，n_{ctx}为上下文长度（默认1024），d_{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)[1]：

N	n_{layer}	d_{attn}	C_{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.topcpu.net/

量化(Quantization)

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

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[1]

- 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[2]，QLORA[3]

- 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[4]，AWQ[5]

- 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：

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

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

反量化：$f = q \times S + ZP$

Reference：

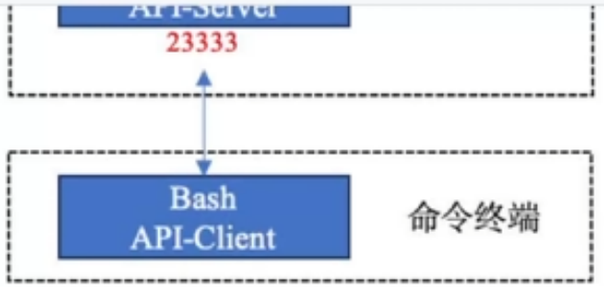
[1] Liu Z, Oguz B, Zhao C, et al. Llm-qat: Data-free quantization aware training for large language models[J]. arXiv preprint arXiv:2305.17888, 2023.

[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 llm compression and acceleration[J]. arXiv preprint arXiv:2306.00978, 2023.



4.3 网页客户端连接API服务器

关闭刚刚的VSCode终端，但服务器端的终端不要关闭。

新建一个VSCode终端，激活conda环境。

```
conda activate lmdeploy
```

使用Gradio作为前端，启动网页客户端。

```
lmdeploy serve gradio http://localhost:23333 \
    --server-name 0.0.0.0 \
    --server-port 6006
```

```
(lmdeploy) root@intern-studio-40059667:~# lmdeploy serve gradio http://localhost:23333 \
>     --server-name 0.0.0.0 \
>     --server-port 6006
server is gonna mount on: http://0.0.0.0:6006
Running on local URL:  http://0.0.0.0:6006
```

运行命令后，网页客户端启动。在电脑本地新建一个cmd终端，新开一个转发端口：

```
ssh -CNg -L 6006:127.0.0.1:6006 root@ssh.intern-ai.org.cn
```

打开浏览器，访问地址 `http://127.0.0.1:6006`

```
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("/root/internlm2-chat-1_8b", trust_remote_co

# Set `torch_dtype=torch.float16` to load model in float16, otherwise it will be loade
model = AutoModelForCausalLM.from_pretrained("/root/internlm2-chat-1_8b", torch_dtype=
model = model.eval()

inp = "hello"
print("[INPUT]", inp)
response, history = model.chat(tokenizer, inp, history=[])
print("[OUTPUT]", response)

inp = "please provide three suggestions about time management"
print("[INPUT]", inp)
response, history = model.chat(tokenizer, inp, history=history)
print("[OUTPUT]", response)
```

```
那里。狐狸犹豫了一下，但是松鼠坚持要帮助它。于是，狐狸和松鼠一起出发了。

当他们到达森林的出口时，狐狸非常感激松鼠的帮助。它意识到，如果没有松鼠的帮助，它就不会找到出口。从那以后，狐狸和松鼠成为了好朋友，它们一起探索森林，分享彼此的知识和经验。

这个故事告诉我们，勇敢和聪明并不是互相排斥的，它们可以相辅相成。如果你需要帮助，不要害怕寻求他人的帮助，因为他们可能比你自己更了解情况。
double enter to end input >>> exit

(lmdeploy) root@intern-studio-40059667:~#
```