

LMDeploy 大模型量化部署实践

官方文档: [InternLM/tutorial/lmdeploy/lmdeploy.md](https://internlm-tutorial.github.io/tutorials/llmdeploy/llmdeploy.md)

视频链接: [LMDeploy 大模型量化部署实践bilibili](#)

基础作业:

- 使用 LMDeploy 以本地对话、网页Gradio、API服务中的一种方式部署 InternLM-Chat-7B 模型, 生成 300 字的小故事

简介

1. 大模型部署背景

大模型部署, 即将训练好的模型在特定的软硬件环境中启动的过程, 使模型能够接受输入并返回预测结果。

为了满足性能和效率的要求, 常常需要对模型进行优化, 例如模型压缩和硬件加速。

大模型特点:

- **内存开销巨大**, 7B 模型仅仅权重就需要 14+G 内存 (指float16)
- 采用自回归生成 token, 需要**缓存历史信息** (如 Attention 的 k/v), 带来巨大的内存开销
- 动态 shape, 请求数不固定, token 逐个生成, 且数量不定
- 相对于视觉模型, LLM 结构简单, Transformers 结构, 大部分是 decoder-only

大模型部署背景

OpenMMLab 书生·浦语

大模型部署挑战

- 设备**
 - 如何应对巨大的存储问题? 低存储设备 (消费级显卡、手机等) 如何部署?
- 推理**
 - 如何加速 token 的生成速度
 - 如何解决动态shape, 让推理可以不间断
 - 如何有效管理和利用内存
- 服务**
 - 如何提升系统整体吞吐量?
 - 对于个体用户, 如何降低响应时间?

大模型部署方案

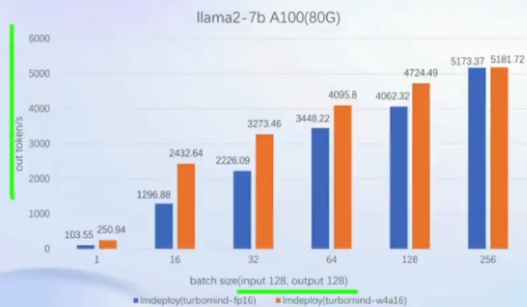
- 技术点**
 - 模型并行
 - 低比特量化
 - Page Attention
 - transformer 计算和访存优化
 - Continuous Batch
 - ...
- 方案**
 - huggingface transformers
 - 专门的推理加速框架
- 云端**
 - lmdeploy
 - vllm
 - tensorrt-llm
 - deepspeed
 - ...
- 移动端**
 - llama.cpp
 - mlc-llm
 - ...

2. LMEeploy简介

推理性能

静态推理性能

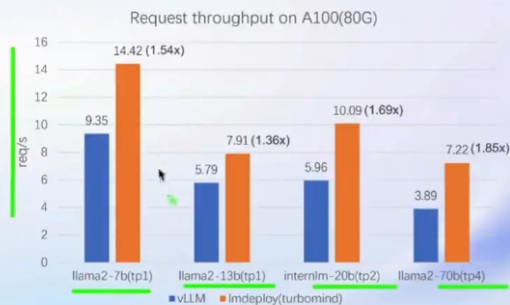
- 固定 batch, 输入/输出 token 数量



LMDeploy w4a16 vs fp16 的性能对比。小 batch 下 (≤ 16), w4a16 的推理性能是 fp16 的 2 倍多

动态推理性能

- 真实对话, 不定长的输入/输出



LMDeploy vs vLLM 的性能对比。计算精度为 FP16、BF16

核心功能 - 量化

为什么要做量化?

- Weight FP16 + KV Cache FP16

模型	权重	KV Cache (tokens=2K) (batch=8)	KV Cache (tokens=8K) (batch=8)	KV Cache (tokens=32K) (batch=8)
Llama 7B	14 GB	8 GB	32 GB	128 GB
Llama 70B	140 GB	5 GB	20 GB	80 GB

- Weight INT4 + KV Cache INT8

模型	权重	KV Cache (tokens=2K) (batch=8)	KV Cache (tokens=8K) (batch=8)	KV Cache (tokens=32K) (batch=8)
Llama 7B	3.5 GB	4 GB	16 GB	64 GB
Llama 70B	35 GB	2.5 GB	10 GB	40 GB

24GB 显存
量化前, 7B 模型, 并发 8, max length 2k
量化后, 7B 模型, 并发 8, max length 8k

80GB 显存
量化前, 70B 模型 oom
量化后, 70B 模型, 并发 8, max length 32k

核心功能 - 量化

为什么做 Weight Only 的量化?

两个基本概念

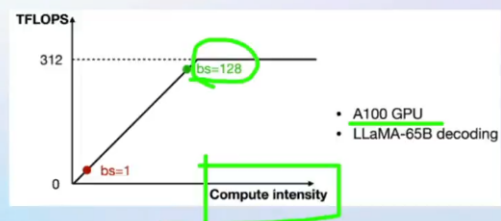
- 计算密集 (compute-bound)**: 推理的绝大部分时间消耗在数值计算上; 针对计算密集场景, 可以通过使用更快的硬件计算单元来提升计算速度, 比如量化为 W8A8 使用 INT8 Tensor Core 来加速计算。
- 访存密集 (memory-bound)**: 推理时, 绝大部分时间消耗在数据读取上; 针对访存密集型场景, 一般是通过提高计算访存比来提升性能。

LLM 是典型的访存密集型任务

常见的 LLM 模型是 Decoder Only 架构。推理时大部分时间消耗在逐 Token 生成阶段 (Decoding 阶段), 是典型的访存密集型场景。如右图, A100 的 FP16 峰值算力为 312 TFLOPS, 只有在 Batch Size 达到 128 这个量级时, 计算才成为推理的瓶颈, 但由于 LLM 模型本身就很大, 推理时的 KV Cache 也会占用很多显存, 还有一些其他的因素影响 (如 Persistent Batch), 实际推理时很难做到 128 这么大的 Batch Size。

Weight Only 量化一举多得

- 4bit Weight Only 量化, 将 FP16 的模型权重量化为 INT4, 访存量直接降为 FP16 模型的 1/4, 大幅降低了访存成本, 提高了 Decoding 的速度。
- 加速的同时还节省了显存, 同样的设备能够支持更大的模型以及更长的对话长度



核心功能 – 量化

如何做 Weight Only 的量化?

- LMDeploy 使用 MIT HAN LAB 开源的 AWQ 算法, 量化为 4bit 模型
- 推理时, 先把 4bit 权重, 反量化回 FP16 (在 Kernel 内部进行, 从 Global Memory 读取时仍是 4bit), 依旧使用的是 FP16 计算
- 相较于社区使用比较多的 GPTQ 算法, AWQ 的推理速度更快, 量化的时间更短

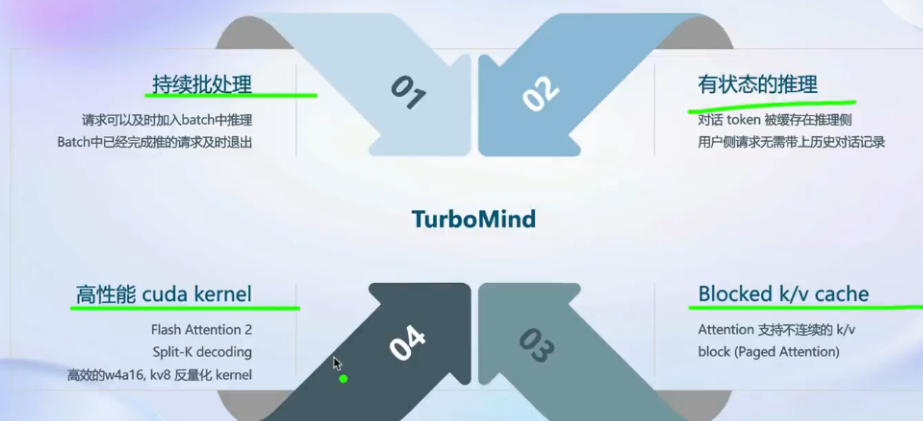
PPL↓		Llama-2			LLaMA			
		7B	13B	70B	7B	13B	30B	65B
FP16	-	5.47	4.88	3.32	5.68	5.09	4.10	3.53
INT3 g128	RTN	6.66	5.52	3.98	7.01	5.88	4.88	4.24
	GPTQ	6.43	5.48	3.88	8.81	5.66	4.88	4.17
	GPTQ-R	6.42	5.41	3.86	6.53	5.64	4.74	4.21
	AWQ	6.24	5.32	3.74	6.35	5.52	4.61	3.95
INT4 g128	RTN	5.73	4.98	3.46	5.96	5.25	4.23	3.67
	GPTQ	5.69	4.98	3.42	6.22	5.23	4.24	3.66
	GPTQ-R	5.63	4.99	3.43	5.83	5.20	4.22	3.66
	AWQ	5.60	4.97	3.41	5.78	5.19	4.21	3.62

Baseline: fp16 weight, fp16 activation

AWQ: int4 weight, fp16 activation

核心功能 - 推理引擎TurboMind

核心功能 – 推理引擎 TurboMind



动手实战

1.环境配置

创建环境

```
/root/share/install_conda_env_internlm_base.sh lmdeploy  
conda activate lmdeploy
```

安装LMDeploy

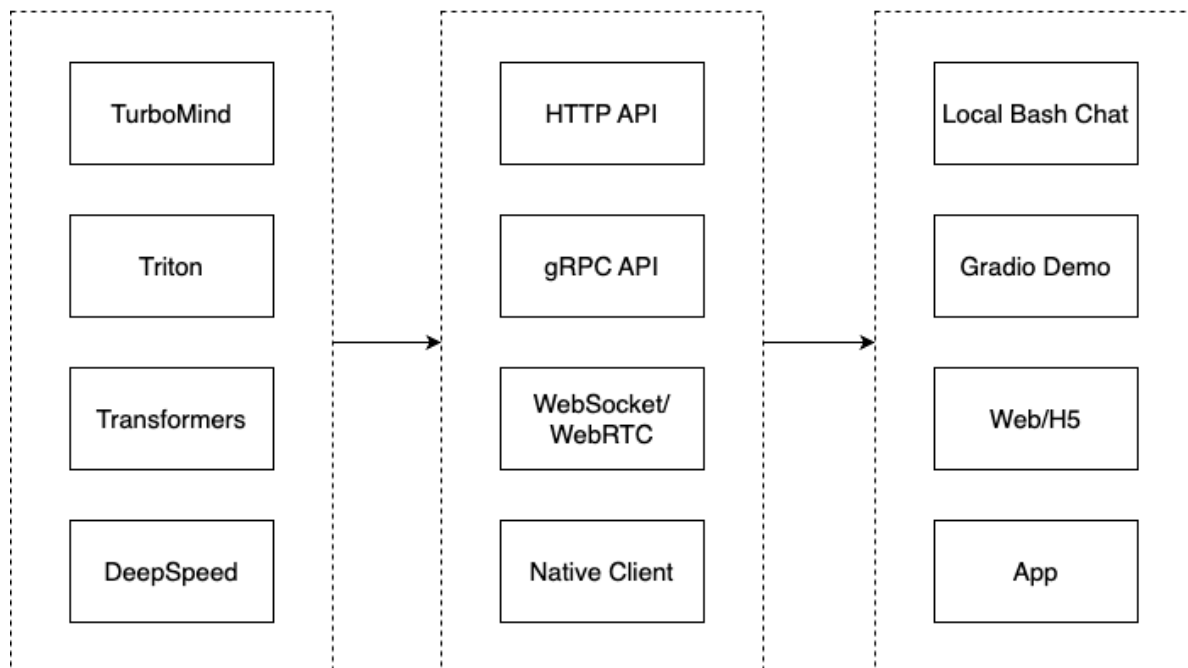
```
# 解决 ModuleNotFoundError: No module named 'packaging' 问题
pip install packaging
# 使用 flash_attn 的预编译包解决安装过慢问题
pip install /root/share/wheels/flash_attn-2.4.2+cu118torch2.0cxx11abiTRUE-cp310-
cp310-linux_x86_64.whl
pip install 'lmdeploy[all]==v0.1.0'
```

2.服务部署

Model Inference/Server

API Server

Client



从架构上把整个服务流程分成下面几个模块。

1. 模型推理/服务。主要提供模型本身的推理，一般来说可以和具体业务解耦，专注模型推理本身性能的优化。可以以模块、API等多种方式提供。
2. Client。可以理解为前端，与用户交互的地方。
3. API Server。一般作为前端的后端，提供与产品和服务相关的数据和功能支持。

2.1 模型转换

使用 TurboMind 推理模型需要先将模型转化为 TurboMind 的格式，目前支持在线转换和离线转换两种形式。在线转换可以直接加载 Huggingface 模型，离线转换需需要先保存模型再加载。

TurboMind 是一款关于 LLM 推理的高效推理引擎，基于英伟达的 [FasterTransformer](#) 研发而成。它的主要功能包括：LLaMa 结构模型的支持，persistent batch 推理模式和可扩展的 KV 缓存管理器。

2.1.1 在线转换

```
# 需要能访问 Huggingface 的网络环境
lmdeploy chat turbomind internlm/internlm-chat-20b-4bit --model-name internlm-
chat-20b
lmdeploy chat turbomind Qwen/Qwen-7B-Chat --model-name qwen-7b

lmdeploy chat turbomind /share/temp/model_repos/internlm-chat-7b/ --model-name
internlm-chat-7b
```

2.1.2 离线转换

离线转换需要在启动服务之前，将模型转为 lmdeploy TurboMind 的格式

```
lmdeploy convert internlm-chat-7b /root/share/temp/model_repos/internlm-chat-7b/
```

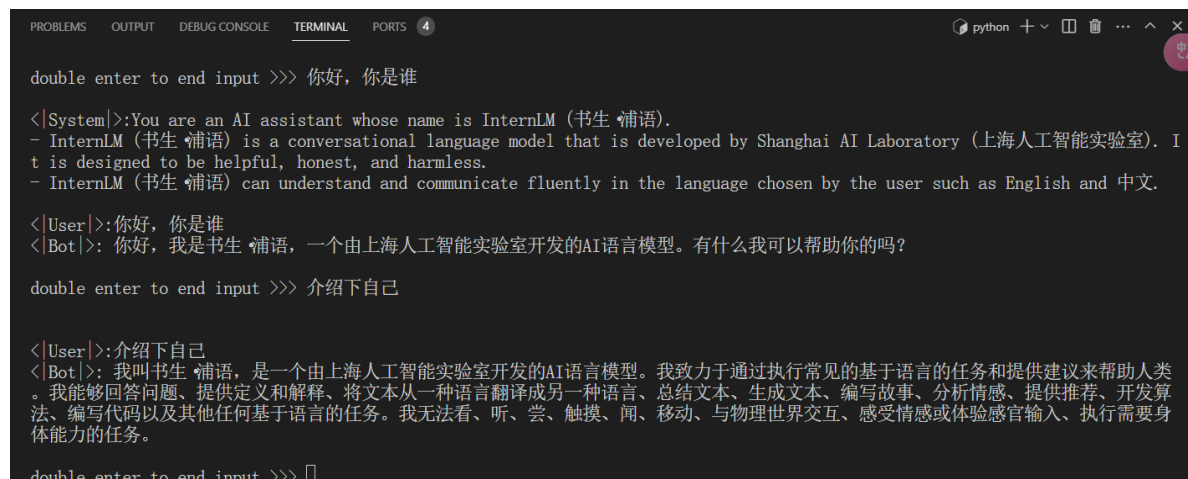
执行完成后将会在当前目录生成一个 workspace 的文件夹。这里面包含的就是 TurboMind 和 Triton “模型推理”需要到的文件。

2.2 TurboMind 推理+命令行本地对话

模型转换完成后，我们就具备了使用模型推理的条件

本地对话 (Bash Local Chat)

```
# Turbomind + Bash Local Chat
lmdeploy chat turbomind ./workspace
```



```
double enter to end input >>> 你好，你是谁

<|System|>:You are an AI assistant whose name is InternLM (书生 补语).
- InternLM (书生 补语) is a conversational language model that is developed by Shanghai AI Laboratory (上海人工智能实验室). It is designed to be helpful, honest, and harmless.
- InternLM (书生 补语) can understand and communicate fluently in the language chosen by the user such as English and 中文.

<|User|>:你好，你是谁
<|Bot|>: 你好，我是书生 补语，一个由上海人工智能实验室开发的AI语言模型。有什么我可以帮助你的吗？

double enter to end input >>> 介绍下自己

<|User|>:介绍下自己
<|Bot|>: 我叫书生 补语，是一个由上海人工智能实验室开发的AI语言模型。我致力于通过执行常见的基于语言的任务和提供建议来帮助人类。我能够回答问题、提供定义和解释、将文本从一种语言翻译成另一种语言、总结文本、生成文本、编写故事、分析情感、提供推荐、开发算法、编写代码以及其他任何基于语言的任务。我无法看、听、尝、触摸、闻、移动、与物理世界交互、感受情感或体验感官输入、执行需要身体能力的任务。

double enter to end input >>> 
```

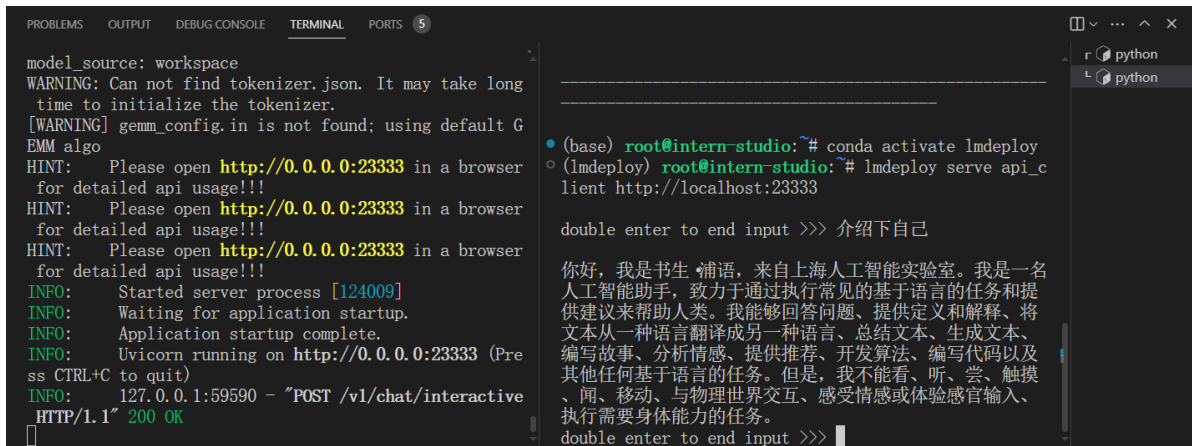
2.3 TurboMind推理+API服务

“模型推理/服务”目前提供了 Turbomind 和 TritonServer 两种服务化方式。此时，Server 是 TurboMind，API Server 可以提供对外的 API 服务

```
# ApiServer+Turbomind  api_server => AsyncEngine => TurboMind
lmdeploy serve api_server ./workspace \
    --server_name 0.0.0.0 \
    --server_port 23333 \
    --instance_num 64 \
    --tp 1
```

然后，我们可以新开一个窗口，执行下面的 Client 命令

```
# ChatApiClient+ApiServer (注意是http协议，需要加http)
lmdeploy serve api_client http://localhost:23333
```

```
model_source: workspace
WARNING: Can not find tokenizer.json. It may take long
time to initialize the tokenizer.
[WARNING] gemm_config.in is not found; using default G
EMM algo
HINT: Please open http://0.0.0.0:23333 in a browser
for detailed api usage!!!
HINT: Please open http://0.0.0.0:23333 in a browser
for detailed api usage!!!
HINT: Please open http://0.0.0.0:23333 in a browser
for detailed api usage!!!
INFO: Started server process [124009]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:23333 (Pre
ss CTRL+C to quit)
INFO: 127.0.0.1:59590 - "POST /v1/chat/interactive
HTTP/1.1" 200 OK

• (base) root@intern-studio:~# conda activate lmdeploy
(lmdeploy) root@intern-studio:~# lmdeploy serve api_c
lient http://localhost:23333

double enter to end input >>> 介绍下自己

你好，我是书生·浦语，来自上海人工智能实验室。我是一名
人工智能助手，致力于通过执行常见的基于语言的任务和提
供建议来帮助人类。我能够回答问题、提供定义和解释、将
文本从一种语言翻译成另一种语言、总结文本、生成文本、
编写故事、分析情感、提供推荐、开发算法、编写代码以及
其他任何基于语言的任务。但是，我不能看、听、尝、触摸
、闻、移动、与物理世界交互、感受情感或体验感官输入、
执行需要身体能力的任务。

double enter to end input >>>
```

刚刚我们启动的是 API Server，自然也有相应的接口。可以直接打开 `http://{host}:23333` 查看
首先做一下SSH转发

```
ssh -CNg -L 23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p <你的ssh端口号>
```

以 `v1/chat/completions` 接口为例，简单试一下。接口请求参数如下：

```
{
  "model": "internlm-chat-7b",
  "messages": "写一个300字的童话故事",
  "temperature": 0.7,
  "top_p": 1,
  "n": 1,
  "max_tokens": 512,
  "stop": false,
  "stream": false,
  "presence_penalty": 0,
  "frequency_penalty": 0,
  "user": "string",
  "repetition_penalty": 1,
  "renew_session": false,
  "ignore_eos": false
}
```

请求结果如下：

