PR #42975 完整报告

vllm-project/vllm

add enqueue all option to throughput benchmark

合并时间：2026-05-20 11:16

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42975

执行摘要

- 一句话：增加预排队选项以提升吞吐基准测试可重复性
- 推荐动作：建议精读该 PR，特别是 enqueue_chat 的设计模式以及睡眠唤醒控制流。值得关注的是如何通过 sleep(level=0) 暂停调度以实现全量入队，这是 vLLM 中一种重要的调度控制手段。

功能与动机

在 run_vllm_chat 内部，llm.chat() 会同时进行请求入队和批量处理，导致批次大小不可预测。本 PR 通过将入队和调度解耦解决这一竞争条件，使基准测试结果更稳定可重复。

实现拆解

1. 在 vllm/entrypoints/llm.py 中新增 enqueue_chat 方法，其签名与 chat 类似但仅调用 _add_chat_requests（从 _run_chat 提取的私有方法）并返回请求 ID 列表。
2. 在 vllm/benchmarks/throughput.py 的 run_vllm 和 run_vllm_chat 中增加 prequeue_requests 参数：当为 True 时，先调用 llm.sleep(level=0) 暂停调度，然后通过 llm.enqueue/enqueue_chat 添加所有请求，最后在 finally 块中用 llm.wake_up(tags=["scheduling"]) 恢复调度，并调用 wait_for_completion 获取结果。
3. 在 argparse 中添加 --prequeue-requests 标志，并添加校验：只支持 vllm/vllm-chat 后端，且不能与异步引擎同时使用。
4. 在 tests/entrypoints/llm/test_mm_processor_kwargs.py 中新增 test_enqueue_chat_forwards_mm_processor_kwargs 和 test_run_chat_forwards_mm_processor_kwargs 以验证 mm_processor_kwargs 参数正确转发。

关键文件：

- vllm/entrypoints/llm.py（模块 LLM 入口；类别 source；类型 core-logic；符号 enqueue_chat, _add_chat_requests）：核心文件，新增 enqueue_chat 公共 API 和 _add_chat_requests 内部方法，是实现预排队功能的核心
- vllm/benchmarks/throughput.py（模块 基准测试；类别 source；类型 core-logic；符号 run_vllm, run_vllm_chat, validate_args, add_cli_args）：基准测试利用新 API 提供 --prequeue-requests 选项以增强测量可重复性

- tests/entrypoints/llm/test_mm_processor_kwargs.py（模块 测试；类别 test；类型 test-coverage；符号 test_enqueue_chat_forwards_mm_processor_kwargs, test_run_chat_forwards_mm_processor_kwargs）：新增测试验证 enqueue_chat 和 _run_chat 的正确参数转发

关键符号：enqueue_chat, _add_chat_requests, run_vllm, run_vllm_chat

关键源码片段

vllm/benchmarks/throughput.py

基准测试利用新 API 提供 --prequeue-requests 选项以增强测量可重复性

```
def run_vllm(
    requests: list[SampleRequest],
    n: int,
    engine_args: EngineArgs,
    do_profile: bool,
    disable_detokenize: bool = False,
    prequeue_requests: bool = False,
) -> tuple[float, list[RequestOutput] | None]:
    from vllm import LLM, SamplingParams
    llm = LLM.from_engine_args(engine_args)
    # ... 省略断言和 prompt 构建 ...
    if not use_beam_search:
        if prequeue_requests:
            llm.sleep(level=0, mode="abort")
        start = time.perf_counter()
        if do_profile:
            llm.start_profile()
        if prequeue_requests:
            try:
                # 在调度恢复前将所有请求入队
                llm.enqueue(
                    prompts,
                    sampling_params,
                    lora_request=lora_requests,
                    use_tqdm=True,
                )
            finally:
                llm.wake_up(tags=["scheduling"])
            outputs = llm.wait_for_completion(
                output_type=RequestOutput, use_tqdm=True
            )
        else:
            outputs = llm.generate(
                prompts, sampling_params, lora_request=lora_requests, use_tqdm=True
            )
        if do_profile:
            llm.stop_profile()
```

```
        end = time.perf_counter()
    return end - start, outputs
```

tests/entrypoints/llm/test_mm_processor_kwargs.py

新增测试验证 enqueue_chat 和 _run_chat 的正确参数转发

```
def test_enqueue_chat_forwards_mm_processor_kwargs() -> None:
    llm = _make_mock_llm()
    mm_processor_kwargs = {"do_pan_and_scan": True}
    sampling_params = SamplingParams(max_tokens=1)
    messages = [{"role": "user", "content": "hello"}]

    # Mock _add_chat_requests to track calls
    llm._add_chat_requests = Mock(return_value=["req-0"])

    request_ids = llm.enqueue_chat(
        messages,
        sampling_params=sampling_params,
        use_tqdm=False,
        mm_processor_kwargs=mm_processor_kwargs,
    )

    assert request_ids == ["req-0"]
    # 验证 mm_processor_kwargs 被正确传递到了 _add_chat_requests
    assert llm._add_chat_requests.call_args.kwargs["mm_processor_kwargs"] == (
        mm_processor_kwargs
    )
```

评论区精华

- DarkLight1337 建议将 chat 和 completion 的预排队标志合并为同一个参数，作者 pmaybank 采纳并在后续提交中实现。
- gemini-code-assist[bot] 建议在 _run_chat 中传递 priority 参数到 _add_chat_requests 以保证一致性。
- 合并预排队标志为统一参数 (design): 作者 pmaybank 采纳，在后续提交中实现单一 --prequeue-requests 标志。
- 在 _run_chat 中传递 priority 到 _add_chat_requests (correctness): 已修复，在后续提交中添加了 priority 参数。

风险与影响

- 风险：
  - 新增 enqueue_chat API 可能会与现有 sleep/wake_up 模式产生交互风险，若调用者不按照文档中的暂停 / 恢复顺序可能导致死锁或请求丢失。
  - benchmark 中的 prequeue 路径使用了 try-finally 确保 wake_up 被调用，但若 _run_engine 内部异常仍可能有一定风险。

    - 修改了内部方法 _add_chat_requests 的提取，可能影响 chat 功能，但测试覆盖了主要路径。
- 影响：
    - 用户：benchmark 用户可通过 --prequeue-requests 获得更稳定可重复的吞吐测量，但可能略微降低测量吞吐值。
    - 系统：新增公共 API enqueue_chat 可供外部使用 sleep/wake_up 模式控制请求流程。
    - 团队：需维护两套并行路径（chat vs enqueue+wait），增加代码复杂性。
    - 风险标记：预排队模式可能因 sleep/wake_up 使用不当导致死锁，新增 API 未覆盖全异常路径测试，基准测试默认行为不变，但新分支引入不同路径需维护

关联脉络

- 暂无明显关联 PR