

# PR #56760 完整报告

vllm-project/vllm

[Bugfix] Measure complete pooling responses

合并时间: 2026-09-15 21:21

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56760>

## 执行摘要

- 一句话: Pooling 基准修复: 度量完整响应体并支持 bytes\_only
- 推荐动作: 值得精读的小型 bugfix。它清晰展示了异步 HTTP 客户端中“延迟测量点”与“响应体消费”之间常见的陷阱, iter\_any() 增量消费避免内存膨胀的做法有借鉴价值; bytes\_only 无 metadata 时回退客户端估算的取舍也可作为后续协议设计的参考。适合 benchmark 工具维护者和性能团队阅读。

## 功能与动机

PR body 明确列出三个问题: Pooling benchmarks 在响应 headers 到达时就记录 E2E 延迟, JSON 响应在解析时被消费, 但二进制 bytes 响应从未被消费, 因此报告的延迟不包含响应传输时间, 且连接可能在服务端仍在写时被关闭; 新增的 bytes\_only 格式被当作 JSON 处理, 每个请求都失败 (报错 Attempt to decode JSON with unexpected mimetype: application/octet-stream); 用确定性延迟服务器实测, main 分支 bytes 格式报告 latency 为 0.0005 秒而实际 wall 时间为 0.25 秒以上, 证明测量完全失真。

## 实现拆解

变更集中在 vllm/benchmarks/lib/endpoint\_request\_func.py 的 \_run\_pooling\_request, 并按以下步骤推进:

1. 调整延迟测量点: 把 output.ttft = output.latency = time.perf\_counter() - st 从“响应 headers 到达后立即记录”移动到“响应 body 消费完成后记录”, 使延迟真正覆盖端到端传输。
2. 按 encoding\_format 分策略消费 body: bytes / bytes\_only 使用 async for \_ in response.content.iter\_any(): pass 增量读取, 避免把 embedding 大数组留在内存中; float 等其他格式使用 await response.read() 完整读入后再解析。
3. token 用量来源三分支: bytes 从响应 metadata header 读取 usage; bytes\_only 因该格式刻意不携带 metadata, 回退为客户端侧 prompt\_len; float 从 JSON body 读取 usage。
4. 调用链配套透传: async\_request\_openai\_embeddings、async\_request\_vllm\_rerank、async\_request\_openai\_embeddings\_chat、async\_request\_infinity\_embeddings、async\_request\_vllm\_pooling 五个入口均新增 prompt\_len=request\_func\_input.prompt\_len 参数传入。
5. 测试配套: 本 PR 未新增测试文件, 作者用既有 tests/benchmarks/test\_endpoint\_request\_func\_timing.py (18 passed) 加 MRV2 E2E 冒烟验证 (/v1/embeddings 与 /pooling 三

种格式共 128 请求全部通过)。

关键文件:

- vllm/benchmarks/lib/endpoint\_request\_func.py (模块 基准工具; 类别 source; 类型 core-logic; 符号 \_run\_pooling\_request, async\_request\_openai\_embeddings, async\_request\_vllm\_rerank, async\_request\_openai\_embeddings\_chat): 唯一变更文件, 修复 pooling 基准的延迟度量点并支持 bytes\_only 格式, 是整个 PR 的核心。

关键符号: \_run\_pooling\_request, async\_request\_openai\_embeddings, async\_request\_vllm\_rerank, async\_request\_openai\_embeddings\_chat, async\_request\_infinity\_embeddings, async\_request\_vllm\_pooling

## 关键源码片段

### vllm/benchmarks/lib/endpoint\_request\_func.py

唯一变更文件, 修复 pooling 基准的延迟度量点并支持 bytes\_only 格式, 是整个 PR 的核心。

```
async def _run_pooling_request(
    session: aiohttp.ClientSession,
    api_url: str,
    payload: dict[str, Any],
    headers: dict[str, Any],
    pbar: tqdm | None = None,
    num_input_sequences: int = 1,
    prompt_len: int = 0,
) -> RequestFuncOutput:
    """发送 pooling 类请求并记录端到端延迟。

    修复前延迟在响应 headers 到达时即被记录, 二进制 body 从未被消费,
    导致延迟被低估且连接可能在服务端仍在写响应时被关闭。
    """
    output = RequestFuncOutput(num_input_sequences=num_input_sequences)
    st = time.perf_counter()
    output.start_time = st
    try:
        async with session.post(url=api_url, headers=headers, json=payload) as response:
            if response.status == 200:
                encoding_format = payload.get('encoding_format', 'float')
                if encoding_format in ('bytes', 'bytes_only'):
                    # 二进制 embedding 体通常较大, 增量消费但不保留数据在内存
                    async for _ in response.content.iter_any():
                        pass
            else:
                # JSON 体相对较小, 直接完整读入再解析
                await response.read()
            # 延迟必须在 body 消费完成后记录, 才是真正的端到端延迟
            output.ttft = output.latency = time.perf_counter() - st

    if encoding_format == 'bytes':
```

```

        # bytes 格式的 token 用量来自响应 metadata header
        metadata = json.loads(response.headers['metadata'])
        usage = metadata.get('usage', {})
    elif encoding_format == 'bytes_only':
        # bytes_only 刻意不携带 metadata，回退到客户端侧 prompt 长度
        usage = {'prompt_tokens': prompt_len}
    else:
        data = await response.json()
        usage = data.get('usage', {})

    output.success = True
    output.generated_text = ''
    output.prompt_len = usage.get('prompt_tokens', 0)
else:
    output.success = False
    output.error = response.reason or ''
except Exception as e:
    output.success = False
    output.error = str(e)

if pbar:
    pbar.update(1)
return output

```

## 评论区精华

本 PR 没有产生技术性 review 讨论。[claude\[bot\]](#) 因该 PR 来自 fork 而禁用自动评审；维护者 [yewentao256](#) 直接批准，评论为“LGTM, thanks for the work!”，无未解决疑虑。PR body 中作者对 [bytes\\_only](#) 的 prompt token 为客户端估算这一权衡做了明确说明，属于有意的设计取舍。

- 维护者审批 (other): 无追加修改即合并。

## 风险与影响

- 风险：

1. 无新增自动化测试：修复依赖既有 18 个用例与作者手工 E2E 冒烟，回归保护较弱。
2. [bytes\\_only](#) token 统计偏差：prompt\_len 来自客户端侧估算，与 tokenizer 真实统计可能有出入，但作者声明不影响请求成功与延迟记录。
3. benchmark 请求将等待完整 body：若服务端响应 body 迟迟不发，benchmark 请求会挂住（原实现会提前关闭连接），这是端到端延迟测量的正确语义，但改变了挂起时的失败模式。
4. 影响面可控：变更只落在 `vllm/benchmarks/lib/endpoint_request_func.py`，不触及生产 serving 代码路径。- 影响：影响范围为性能团队与 benchmark 使用者：pooling / embedding 类基准的 E2E 延迟、成功率与 token 统计从此反映完整响应传输，[bytes\\_only](#) 格式可正常跑通。对生产服务、API 用户无直接影响；对依赖 benchmark 数据做性能对比的团队而言，修复后数据可信度显著提升，但历史基线数据与修复后数据不

可直接对比。 - 风险标记：无新增测试覆盖，bytes\_only 延迟统计为客户端估算，benchmark 请求将等待完整 body

## 关联脉络

- PR #56908 [Model Runner V2] Tolerate lack of pinned memory support: 本 PR 的复现与验证均基于 MRV2 pooling server (VLLM\_USE\_V2\_MODEL\_RUNNER=1)，56908 是对 MRV2 平台能力的补齐，两者同属 MRV2 演进线。