PR #39937 完整报告

vllm-project/vllm

[Model Runner V2] Multiple prompt logprobs support

合并时间：2026-04-21 23:49

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/39937

执行摘要

- 一句话：V2 Model Runner 支持多个 prompt logprob
- 推荐动作：值得精读以实现差异化批处理参数的传递模式。但需留意 gemini 指出的切片缺失问题，建议在合并后续 PR 或生产环境遇到断言错误时补充每个请求的精确截断逻辑。

功能与动机

作为 #39337 的一部分，需要支持多个 prompt logprobs（top-k），并修复在启用 chunked prefill 和 preemption 时 prompt_logprobs 断言失败的问题。PR body 中的测试用例原本因位置只有 1 个 logprob 而报错 AssertionError: Output 0 position 1 has 1 logprobs, expected 2。

实现拆解

1. 状态追踪：在 PromptLogprobsWorker.__init__ 中新增 self.num_prompt_logprobs 数组（np.int32），用于持久化每个请求的 sampling_params.prompt_logprobs。
2. 参数记录：在 add_request 中将 sampling_params.prompt_logprobs or 0 写入对应位置，同时保留 uses_prompt_logprobs 布尔数组以区分请求是否启用（prompt_logprobs=0 属于启用但无需返回）。
3. 批处理最大值计算：在 compute_prompt_logprobs 中从 num_prompt_logprobs 提取当前批的各请求所需数量，计算 max_num_prompt_logprobs：若任一请求为 -1（请求所有 token），则取 -1；否则取整数最大值。
4. 向下游传递：将 max_num_prompt_logprobs 作为额外参数传入 compute_prompt_logprobs_with_chunking，使其在计算时按全局最高数量生成 logprob 张量。该函数返回三元组 prompt_token_ids, prompt_logprobs, prompt_ranks（之前为二元组）。
5. 结果收集：遍历批中请求，根据 is_prompt_chunked 和 start_idx >= end_idx 条件选择构造 LogprobsTensors 或 None；合并分块结果时对 None 作跳过处理。

涉及文件：仅 vllm/v1/worker/gpu/sample/prompt_logprob.py。无测试、配置或 schema 配套变更。

关键文件：

- vllm/v1/worker/gpu/sample/prompt_logprob.py（模块 采样器；类别 source；类型 core-logic；符号 PromptLogprobsWorker.init, PromptLogprobsWorker.add_request, PromptLogprobsWorker.compute_prompt_logprobs）：核心文件，实现多 prompt logprob 的追踪、计算与结果收集。改动集中在 PromptLogprobsWorker 类。

关键符号：PromptLogprobsWorker.init, PromptLogprobsWorker.add_request, PromptLogprobsWorker.compute_prompt_logprobs, compute_prompt_logprobs_with_chunking

关键源码片段

vllm/v1/worker/gpu/sample/prompt_logprob.py

核心文件，实现多 prompt logprob 的追踪、计算与结果收集。改动集中在 PromptLogprobsWorker 类。

```
# vllm/v1/worker/gpu/sample/prompt_logprob.py
# 类初始化：新增 num_prompt_logprobs 数组
class PromptLogprobsWorker:
    def __init__(self, max_num_reqs: int):
        self.max_num_reqs = max_num_reqs
        self.uses_prompt_logprobs = np.zeros(self.max_num_reqs, dtype=bool)
        self.num_prompt_logprobs = np.zeros(self.max_num_reqs, dtype=np.int32) #
        新增：记录每个请求的 prompt_logprobs 数量
        self.in_progress_prompt_logprobs: dict[str, list[LogprobsTensors]] = {}

    def add_request(self, req_id: str, req_idx: int, sampling_params: SamplingParams):
        uses_prompt_logprobs = sampling_params.prompt_logprobs is not None
        self.uses_prompt_logprobs[req_idx] = uses_prompt_logprobs
        # 保存具体数值，0 表示启用但不需要返回（与 uses_prompt_logprobs 协同）
        self.num_prompt_logprobs[req_idx] = sampling_params.prompt_logprobs or 0
        if uses_prompt_logprobs:
            self.in_progress_prompt_logprobs[req_id] = []

    def compute_prompt_logprobs(self, ...):
        # ... ( 省略数据准备 )
        # 获取当前批中各请求的 prompt_logprobs 数量
        num_prompt_logprobs = self.num_prompt_logprobs[idx_mapping_np]
        # ... 过滤出真正需要计算的请求
        requested_num_prompt_logprobs = num_prompt_logprobs[needs_prompt_logprobs]
        # 批最大值：若任一请求为 -1（表示全部），则整体取 -1
        max_num_prompt_logprobs = (
            -1 if np.any(requested_num_prompt_logprobs == -1)
            else int(requested_num_prompt_logprobs.max())
        )
        # 将最大值传入下游函数，确保一次计算产出足够多的 top-k
        prompt_token_ids, prompt_logprobs, prompt_ranks = (
            compute_prompt_logprobs_with_chunking(
                ..., max_num_prompt_logprobs, # 新增参数
            )
```

```
)
# ... 按请求切片，但未对 prompt_logprobs 的第二维进行按需裁剪
# 注意：这里仅切了 token 位置维度，而 top-k 维度始终为 max_num_prompt_logprobs
logprobs = LogprobsTensors(
    logprob_token_ids=prompt_token_ids[start_idx:end_idx],
    logprobs=prompt_logprobs[start_idx:end_idx], # shape: [positions, max_num_prompt_
    logprobs]
    selected_token_ranks=prompt_ranks[start_idx:end_idx],
)
# ...
```

评论区精华

- uses_prompt_logprobs 冗余性（design）：njhill 询问引入 num_prompt_logprobs 后是否可移除 uses_prompt_logprobs；yewentao256 解释因 prompt_logprobs=0 是有效启用状态，需保留。
- 中间变量简化（style）：njhill 建议为 num_prompt_logprobs[needs_prompt_logprobs] 和 is_prompt_chunked[i] 使用中间变量以避免重复索引，均已采纳。
- 切片未达预期（correctness，来自 gemini-code-assist[bot]）：当批处理中请求的 prompt_logprobs 值不同时，当前实现按最大值计算并返回所有位置的完整 top-k，而下游测试断言期望精确数量，可能导致 AssertionError。该评论未获直接回复，最终代码仍保持按批最大值返回，未按请求单独切片。
- uses_prompt_logprobs 是否冗余 (design): 保留 uses_prompt_logprobs，num_prompt_logprobs 补充数值信息。
- 批处理最大值未按请求切片 (correctness): 未直接回复，最终代码未实现切片，风险遗留。
- 代码简化建议 (style): 已采纳，最终代码应用了这些简化。

风险与影响

- 风险：回归 / 正确性风险：混合请求批处理时未根据请求实际 prompt_logprobs 对返回张量进行切片，导致所有请求获得相同最大数量的 top-k logprobs。若下游组件（如测试断言）检查精确数量，可能触发断言失败。PR body 中的测试仅使用了统一值 2，未暴露该问题。 性能风险：引入 max_num_prompt_logprobs 计算和 compute_prompt_logprobs_with_chunking 的额外维度，但影响局限于启用 prompt logprobs 的请求，通常为少数。 兼容性：仅影响 V2 Model Runner（VLLM_USE_V2_MODEL_RUNNER=1），V1 runner 不受影响。函数签名变化（返回三元组）已同步至调用方。
- 影响：用户：V2 Model Runner 用户可指定 sampling_params.prompt_logprobs 为大于 0 的整数来获取多个 prompt logprobs，但混合批处理时可能收到超出预期的 logprobs 数量，需要下游容忍。 系统：新增数组和分支，但仅在请求启用时产生额外计算，影响有限。 团队：代码简化后维护性提升；未解决的切片问题需后续跟踪（如 #39337 后续 PR）。
- 风险标记：混合请求未按需切片，缺少多请求差异化测试覆盖，需下游容忍额外 logprobs

关联脉络

- PR #39337 Multiple prompt logprobs support: 本 PR 是 #39337 的一部分，具体实现在此合并。