PR #40559 完整报告

vllm-project/vllm

[Model Runner V2] Add `logprob_token_ids` support

合并时间：2026-05-02 01:02

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40559

执行摘要

- 一句话：添加 logprob_token_ids 支持到 V2 模型 Runner
- 推荐动作：值得精读，特别是 compute_topk_logprobs 的 fast/slow path 设计和 _fill_logprob_token_ids_kernel 的实现。njhill 的重构也体现了模块化原则。也可以关注 gemini-code-assist 的自动化 review 质量。

功能与动机

该变更是 #39337 的一部分，用于支持 generative scoring 功能。CI 测试 test_generative_scoring_e2e.py 因缺少此支持而失败。

实现拆解

1. 在 vllm/sampling_params.py 中定义 MAX_LOGPROB_TOKEN_IDS 常量，添加 num_logprobs 计算属性（统一 logprobs 与 logprob_token_ids），并在 _validate_logprobs 中增加对 logprob_token_ids 长度的校验。
2. 在 vllm/v1/worker/gpu/sample/logprob.py 中新增 LogprobTokenIdsState 类（采用 staged write 模式），以及 _fill_logprob_token_ids_kernel Triton kernel，用于在 GPU 上高效地构建 token ID 矩阵和有效性掩码。
3. 在 vllm/v1/worker/gpu/sample/sampler.py 的 Sampler 类中实例化 LogprobTokenIdsState，并在 __init__、add_request、apply_staged_writes 和 __call__ 中集成。__call__ 中根据 max_per_req_token_ids 判断是否进入 logprob 计算路径（即使全局 top-k 为 0）。
4. 重写 compute_topk_logprobs 函数：当无自定义 token ID 时走快路径（同原逻辑）；有自定义 token ID 时走慢路径，调用 _fill_logprob_token_ids_kernel 填充矩阵和掩码，再计算 logprobs 并置零无效值。
5. 调整 vllm/v1/engine/logprobs.py 中 LogprobsProcessor.from_new_request 和 vllm/v1/core/sched/scheduler.py 中 update_from_output 使用 sampling_params.num_logprobs 代替直接访问 logprobs 字段，以正确支持自定义 token ID 场景。

关键文件：

- vllm/v1/worker/gpu/sample/logprob.py（模块 采样计算；类别 source；类型 core-logic；符号 _fill_logprob_token_ids_kernel, LogprobTokenIdsState, init, add_request）：核心

改动文件，新增 LogprobTokenIdsState、_fill_logprob_token_ids_kernel，重写 compute_topk_logprobs 以支持自定义 token ID

- vllm/sampling_params.py（模块 采样参数；类别 source；类型 core-logic；符号 num_logprobs）：添加 MAX_LOGPROB_TOKEN_IDS 常量和 num_logprobs 计算属性，更新验证逻辑
- vllm/v1/worker/gpu/sample/sampler.py（模块 采样器；类别 source；类型 dependency-wiring）：集成 LogprobTokenIdsState，调整 __call__中的 logprobs 计算条件
- vllm/v1/engine/logprobs.py（模块 引擎输出；类别 source；类型 data-contract）：使用 sampling_params.num_logprobs 替换直接访问 logprobs 字段
- vllm/v1/core/sched/scheduler.py（模块 调度器；类别 source；类型 data-contract）：使用 sampling_params.num_logprobs 判断是否需要提取 logprobs

关键符号：compute_topk_logprobs, _fill_logprob_token_ids_kernel, Sampler.call, SamplingParams.num_logprobs, LogprobTokenIdsState.add_request, LogprobsProcessor.from_new_request

关键源码片段

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

核心改动文件，新增 LogprobTokenIdsState、_fill_logprob_token_ids_kernel，重写 compute_topk_logprobs 以支持自定义 token ID

```
def compute_topk_logprobs(
    logits: torch.Tensor,
    num_logprobs: int,
    sampled_token_ids: torch.Tensor,
    cu_num_logits: list[int] | None = None,
    logprob_token_ids_state: "LogprobTokenIdsState | None" = None,
    expanded_idx_mapping: torch.Tensor | None = None,
    max_per_req_token_ids: int = 0,
) -> LogprobsTensors:
    """
    计算每个请求的 logprobs。
    如果没有任何请求指定自定义 logprob_token_ids（快速路径），
    则直接取采样 token + top-k 的 logprobs。
    否则（慢速路径），通过 _fill_logprob_token_ids_kernel
    构建 token ID 矩阵和有效性掩码，覆盖 top-k 列。
    """
    assert num_logprobs >= 0
    batch_size, vocab_size = logits.shape

    if max_per_req_token_ids == 0:
        # Fast path: 无自定义 token ID 请求
        logprob_token_ids = sampled_token_ids.unsqueeze(-1)
        if num_logprobs > 0:
            topk_indices = torch.topk(logits, num_logprobs, dim=-1).indices
```

```
            logprob_token_ids = torch.cat((logprob_token_ids, topk_indices), dim=1)
        logprobs = compute_token_logprobs(logits, logprob_token_ids)
    else:
        # Slow path: 至少一个请求指定 logprob_token_ids
        assert logprob_token_ids_state is not None
        assert expanded_idx_mapping is not None
        topk_indices = None
        if num_logprobs > 0:
            topk_indices = torch.topk(logits, num_logprobs, dim=-1).indices

        num_cols = max(num_logprobs, max_per_req_token_ids)
        logprob_token_ids = sampled_token_ids.new_zeros((batch_size, 1 + num_cols))
        valid_mask = torch.zeros_like(logprob_token_ids, dtype=torch.bool)
        # 使用 Triton kernel 填充 token IDs 和有效掩码
        _fill_logprob_token_ids_kernel[(batch_size,)](
            logprob_token_ids,
            logprob_token_ids.stride(0),
            valid_mask,
            valid_mask.stride(0),
            sampled_token_ids,
            topk_indices if topk_indices is not None else logprob_token_ids,
            topk_indices.stride(0) if topk_indices is not None else 0,
            # ... 其他参数（省略以保持简洁）
        )
        logprobs = compute_token_logprobs(logits, logprob_token_ids)
        # 将无效位置的 logprob 置零
        logprobs = logprobs.masked_fill(~valid_mask[:, :logprobs.shape[1]], 0.0)

    # 计算采样 token 的 ranks（与原始逻辑一致）
    token_ranks = torch.empty(batch_size, dtype=torch.int64, device=logits.device)
    _ranks_kernel[(batch_size,)](
        token_ranks,
        logits,
        logits.stride(0),
        sampled_token_ids,
        vocab_size,
        BLOCK_SIZE=8192,
    )
    return LogprobsTensors(
        logprob_token_ids=logprob_token_ids,
        logprobs=logprobs,
        selected_token_ranks=token_ranks,
        cu_num_generated_tokens=cu_num_logits,
    )
```

vllm/sampling_params.py

添加 MAX_LOGPROB_TOKEN_IDS 常量和 num_logprobs 计算属性，更新验证逻辑

```
MAX_LOGPROB_TOKEN_IDS = 128
```

```
"""Upper bound on SamplingParams.logprob_token_ids list length."""

# ... in SamplingParams class ...

@property
def num_logprobs(self) -> int | None:
    """Number of sample logprobs to return per output token.
    当 logprobs 未设置但 logprob_token_ids 设置时,
    返回 logprob_token_ids 的长度。
    """
    if self.logprobs is not None:
        return self.logprobs
    return len(self.logprob_token_ids) if self.logprob_token_ids else None

def _validate_logprobs(self, model_config: ModelConfig) -> None:
    # 原有 logprobs 验证 ...
    # 新增：验证 logprob_token_ids 长度限制
    if self.logprob_token_ids is not None:
        n = len(self.logprob_token_ids)
        if n > MAX_LOGPROB_TOKEN_IDS:
            raise VLLMValidationError(
                f"Requested logprob_token_ids of length {n}, "
                f"which is greater than max allowed: {MAX_LOGPROB_TOKEN_IDS}",
                parameter="logprob_token_ids",
                value=n,
            )
```

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

集成 LogprobTokenIdsState，调整 __call__中的 logprobs 计算条件

```
def __call__(
    self,
    logits: torch.Tensor,
    input_batch: InputBatch,
) -> SamplerOutput:
    # ... 前期处理 ...
    sampled, processed_logits = self.sample(...)

    max_num_logprobs = self.sampling_states.max_num_logprobs(idx_mapping_np)
    # NEW: 获取每个请求的最大自定义 token ID 数
    max_per_req_token_ids = self.logprob_token_ids_state.max_num_token_ids(
        idx_mapping_np
    )
    # 当存在自定义 token ID 时，即使 max_num_logprobs 为 NO_LOGPROBS 也要计算
    if max_num_logprobs != NO_LOGPROBS or max_per_req_token_ids > 0:
        if self.logprobs_mode == "processed_logprobs":
            logits = processed_logits
        # 处理 speculative decoding 扩展
        expanded_logits = logits.shape[0] != idx_mapping_np.shape[0]
```

```
        cu_num_logits = cu_num_logits_np.tolist() if expanded_logits else None
        # 将 NO_LOGPROBS 转为 0 以触发 logprobs 计算
        num_logprobs = max_num_logprobs if max_num_logprobs != NO_LOGPROBS else 0
        logprobs_tensors = compute_topk_logprobs(
            logits,
            num_logprobs,
            sampled,
            cu_num_logits,
            logprob_token_ids_state=self.logprob_token_ids_state,
            expanded_idx_mapping=input_batch.expanded_idx_mapping,
            max_per_req_token_ids=max_per_req_token_ids,
        )
    else:
        logprobs_tensors = None
    # ... 构建 SamplerOutput ...
```

评论区精华

- gemini-code-assist[bot] 指出当 max_num_logprobs 为 NO_LOGPROBS 但存在 logprob_token_ids 时，logprobs 会被跳过（正确性问题）；作者将 num_logprobs 降为 0 解决。
- 同一 bot 提到逐行创建 tensor 的低效性（性能问题）；作者通过 staged write 和统一 kernel 优化。
- njhill 建议将更多逻辑移入 logprob.py 以保持 sampler.py 简洁；作者同意并由 njhill 提交重构 commit。
- 最终 njhill 批准 PR。
- Logprobs 跳过计算逻辑 (correctness): yewentao256 修复，将 num_logprobs 设置为 0 以触发计算。
- 低效的逐行 GPU 传输 (performance): yewentao256 通过 staged write 和统一 kernel 优化。
- 将逻辑移入 logprob.py (design): yewentao256 同意，njhill 提交重构 commit。

风险与影响

- 风险：核心路径变更：在 logprob 计算中新增了分支（fast/slow path），可能引入回归，特别是在启用自定义 token ID 时，计算新路径的 Triton kernel 正确性需要保障。 性能风险：新增的 _fill_logprob_token_ids_kernel 和掩码操作可能增加 latency，对于无自定义 token ID 的请求通过 fast path 避免了开销，但仍需监控。 兼容性风险：num_logprobs 属性替换了直接访问 logprobs 字段，依赖旧属性的代码可能失效。PR 已同步修改两处调用点，但可能还有未发现的引用（如外部扩展）。
- 影响：对用户：为使用 V2 模型 Runner 的 generative scoring API 的用户启用自定义 token ID 的 logprob 能力，不影响不使用该功能的请求。 对系统：在采样管道的 logprobs 计算中引入新的 Triton kernel 和 GPU 内存操作，对内部无影响。 对团队：需关注未来对 LogprobTokenIdsState 的维护，保持与 Sampler 中其他 state 类一致。

- 风险标记：核心采样路径变更，新增 GPU kernel，性能敏感，兼容性风险

关联脉络

- PR #39337 Generative scoring support: 该 PR 是 #39337 的子任务，用于修复构建失败并启用 logprob_token_ids 支持。