PR #41761 完整报告

vllm-project/vllm

[Model Runner V2] Bug fix: logprob dtype int64/int32 issue

合并时间：2026-05-12 05:55

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41761

执行摘要

- 一句话：修复 MRV2 logprob 的 int64/int32 类型不匹配
- 推荐动作：建议精读。该 PR 展示了 Triton kernel 中类型一致性的重要性，并通过跨文件协作（内核、校验、测试）系统性解决问题。尤其值得关注的是 _fill_logprob_token_ids_kernel 中 if/else 分支的类型对齐技巧。

功能与动机

关联 Issue #41286 明确了从 Model Runner v1 向 v2 迁移的路线图。本修复是其中一项任务，解决了 VLLM_USE_V2_MODEL_RUNNER=1 下执行生成性评分 E2E 测试时 Triton 编译报错 AssertionError('Mismatched type for src between then block (pointer<int32>) and else block (pointer<int64>)') 的问题。

实现拆解

1. Triton Kernel 类型对齐：在 vllm/v1/worker/gpu/sample/logprob.py 的 compute_topk_logprobs 函数中，将 topk_indices 重命名为 topk_token_ids，并在 num_logprobs > 0 分支中显式 .to(torch.int32)，else 分支使用 logprob_token_ids_state.token_ids.gpu（int32 指针）作为占位。同时，在 _fill_logprob_token_ids_kernel 的 else 分支中，从 topk 指针加载的 tokens 也调用 .to(tl.int64) 确保与 then 分支输出类型一致。
2. 参数校验增强：在 vllm/sampling_params.py 的 _validate_logprobs 中添加检查：当同时指定 logprobs 和 logprob_token_ids 时，logprobs 必须等于 len(logprob_token_ids)，否则抛出 VLLMValidationError。
3. 单元测试新增：新建 tests/v1/engine/test_logprobs_processor.py，包含 test_drops_trailing_sentinel_columns 和 test_accepts_exactly_sized_row 两个测试用例，验证 LogprobsProcessor 能正确处理批处理填充后的 sentinel 列，确保 cumulative_logprob 仅基于采样 token 的 logprob 计算。

关键文件：

- vllm/v1/worker/gpu/sample/logprob.py（模块 采样器；类别 source；类型 core-logic；符号 compute_topk_logprobs, _fill_logprob_token_ids_kernel）：核心修复文件，修改了 compute_topk_logprobs 和 _fill_logprob_token_ids_kernel，确保 top-k 索引类型统一为 int32，并统一在 kernel 侧转为 int64。

- vllm/sampling_params.py（模块 采样参数；类别 source；类型 core-logic；符号 _validate_logprobs）：新增参数校验，当 logprobs 和 logprob_token_ids 同时设置时要求长度相等，提前捕获无效请求。
- tests/v1/engine/test_logprobs_processor.py（模块 logprobs 处理器；类别 test；类型 test-coverage；符号 _make_processor, test_drops_trailing_sentinel_columns, test_accepts_exactly_sized_row）：新增完整的单元测试套件，验证 LogprobsProcessor 对 sentinel 列的处理逻辑，确保 sentinel 值不会泄露给用户。

关键符号：compute_topk_logprobs, _fill_logprob_token_ids_kernel, _validate_logprobs, _make_processor, test_drops_trailing_sentinel_columns, test_accepts_exactly_sized_row

关键源码片段

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

核心修复文件，修改了 compute_topk_logprobs 和 _fill_logprob_token_ids_kernel，确保 top-k 索引类型统一为 int32，并统一在 kernel 侧转为 int64。

```
# vllm/v1/worker/gpu/sample/logprob.py ( 片段 )

def compute_topk_logprobs(...) -> LogprobsTensors:
    assert num_logprobs >= 0
    # ... 省略前序代码

    if max_per_req_token_ids == 0:
        # Fast path: 无自定义 logprob_token_ids 请求
        ...
    else:
        # Slow path: 构建 token_ids 矩阵和 valid_mask
        assert logprob_token_ids_state is not None
        assert expanded_idx_mapping is not None

        if num_logprobs > 0:
            # 显式 top-k 并转为 int32（与 per_req_token_ids 类型一致）
            topk_token_ids = torch.topk(logits, num_logprobs, dim=-1).indices
            topk_token_ids = topk_token_ids.to(torch.int32)
        else:
            # num_logprobs == 0 时无需 top-k，使用 int32 占位指针（不会被实际访问）
            topk_token_ids = logprob_token_ids_state.token_ids.gpu

        num_cols = max(num_logprobs, max_per_req_token_ids)
        # 注意：此处未显式指定 dtype，依赖 sampled_token_ids 的类型
        logprob_token_ids = sampled_token_ids.new_zeros((batch_size, 1 + num_cols))
        valid_mask = torch.zeros_like(logprob_token_ids, dtype=torch.bool)
        _fill_logprob_token_ids_kernel[(batch_size,)](...)
        logprobs = compute_token_logprobs(logits, logprob_token_ids)
        logprobs = logprobs.masked_fill(~valid_mask, float("-inf"))

    # ... 后续 ranks 和返回值
```

```
    return LogprobsTensors(...)


@triton.jit
def _fill_logprob_token_ids_kernel(
    out_token_ids_ptr, out_token_ids_stride,
    out_valid_mask_ptr, out_valid_mask_stride,
    sampled_token_ids_ptr,
    topk_indices_ptr, topk_indices_stride, # 此时始终为 int32 指针
    expanded_idx_mapping_ptr,
    num_per_req_token_ids_ptr,
    per_req_token_ids_ptr, per_req_token_ids_stride,
    NUM_TOPK: tl.constexpr,
    PADDED_COLS: tl.constexpr,
):
    batch_idx = tl.program_id(0)

    # Column 0: 采样 token
    sampled = tl.load(sampled_token_ids_ptr + batch_idx)
    tl.store(out_token_ids_ptr + batch_idx * out_token_ids_stride, sampled)
    tl.store(out_valid_mask_ptr + batch_idx * out_valid_mask_stride, 1)

    req_state_idx = tl.load(expanded_idx_mapping_ptr + batch_idx)
    num_custom = tl.load(num_per_req_token_ids_ptr + req_state_idx)

    col = tl.arange(0, PADDED_COLS)
    tid_base = out_token_ids_ptr + batch_idx * out_token_ids_stride + 1
    mask_base = out_valid_mask_ptr + batch_idx * out_valid_mask_stride + 1

    if num_custom > 0:
        # 使用 per-request 自定义 tokens (int32)
        src = per_req_token_ids_ptr + req_state_idx * per_req_token_ids_stride
        valid = col < num_custom
    else:
        # 使用 top-k 索引 (int32)
        src = topk_indices_ptr + batch_idx * topk_indices_stride
        valid = col < NUM_TOPK

    # 统一转为 int64，确保输出类型一致，避免类型不匹配断言
    tokens = tl.load(src + col, mask=valid, other=0).to(tl.int64)
    tl.store(tid_base + col, tokens, mask=valid)
    tl.store(mask_base + col, tl.full([PADDED_COLS], 1, tl.int1), mask=valid)
```

vllm/sampling_params.py

新增参数校验，当 logprobs 和 logprob_token_ids 同时设置时要求长度相等，提前捕获无效请求。

```
# vllm/sampling_params.py
class SamplingParameters:
```

```
    # ...
    def _validate_logprobs(self, model_config: ModelConfig) -> None:
        # ... 已有校验 ...
        # Validate logprob_token_ids.
        if self.logprob_token_ids is not None:
            n = len(self.logprob_token_ids)
            if n > MAX_LOGPROB_TOKEN_IDS:
                raise VLLMValidationError(...)
            # 新增校验：若同时设置了 logprobs，二者长度必须一致
            if self.logprobs is not None and self.logprobs != n:
                raise VLLMValidationError(
                    f"When both logprobs and logprob_token_ids are set, "
                    f"logprobs must equal len(logprob_token_ids). Got "
                    f"logprobs={self.logprobs}, len(logprob_token_ids)={n}.",
                    parameter="logprob_token_ids",
                    value=n,
                )
        # ... 后续 prompt logprobs 校验 ...
```

tests/v1/engine/test_logprobs_processor.py

新增完整的单元测试套件，验证 LogprobsProcessor 对 sentinel 列的处理逻辑，确保 sentinel 值不会泄露给用户。

```
# tests/v1/engine/test_logprobs_processor.py
"""验证 MRV2 sampler 依赖的截断不变量：
当 sampler 因批处理中其他请求需要更宽的行而返回超过请求本身
`num_logprobs + 1` 的行时，尾部填充的 sentinel 值 (token_id=0, logprob=-inf)
不应出现在最终用户可见的 logprobs 中。
"""

def _make_processor(num_logprobs: int) -> LogprobsProcessor:
    return LogprobsProcessor(
        tokenizer=None,
        logprobs=create_sample_logprobs(flat_logprobs=False),
        prompt_logprobs=None,
        cumulative_logprob=0.0,
        num_logprobs=num_logprobs,
        num_prompt_logprobs=None,
    )

def test_drops_trailing_sentinel_columns():
    """请求 3 个自定义 token logprobs，但批处理中行宽被填充到 5。
    验证尾部 -inf 条目不会出现在最终结果中。"""
    processor = _make_processor(num_logprobs=3)
    sampled = 42
    # 模拟批处理填充：最后两列为 sentinel
    token_ids = np.array([[sampled, 100, 200, 300, 0, 0]], dtype=np.int32)
    logprobs = np.array([[-0.5, -1.0, -2.0, -3.0, -np.inf, -np.inf]], dtype=np.float32)
    ranks = np.array([1], dtype=np.int32)
```

```
    processor._update_sample_logprobs(LogprobsLists(token_ids, logprobs, ranks))

    assert len(processor.logprobs) == 1
    pos = processor.logprobs[0]
    # 只保留 sampled + 3 个请求的 token
    assert set(pos.keys()) == {sampled, 100, 200, 300}
    assert 0 not in pos # sentinel token_id 0 被丢弃
    assert all(np.isfinite(lp.logprob) for lp in pos.values())
    # cumulative_logprob 仅基于采样 token
    assert processor.cumulative_logprob == -0.5

def test_accepts_exactly_sized_row():
    """当行宽恰好为 num_logprobs+1 时，无需截断。"""
    processor = _make_processor(num_logprobs=2)
    token_ids = np.array([[7, 11, 13]], dtype=np.int32)
    logprobs = np.array([[-0.5, -1.5, -2.5]], dtype=np.float32)
    ranks = np.array([1], dtype=np.int32)

    processor._update_sample_logprobs(LogprobsLists(token_ids, logprobs, ranks))
    pos = processor.logprobs[0]
    assert set(pos.keys()) == {7, 11, 13}
```

评论区精华

gemini-code-assist[bot] 在 review 中高优先级别提出建议：logprob_token_ids 应显式指定 dtype=torch.int64，以防止 sampled_token_ids 为 int32 时内核 8 字节写入导致内存损坏。该建议未被采纳直接修改，但后续的 topk_token_ids.to(torch.int32) 和相关类型转换已通过另一路径解决类型不一致问题。

- logprob_token_ids 显式指定 dtype (correctness): 未直接采纳该建议，但通过 topk_token_ids.to(torch.int32) 和 kernel 内 .to(tl.int64) 确保了类型一致性和安全性。

风险与影响

- 风险：核心风险在于 compute_topk_logprobs 中 logprob_token_ids 的 dtype 未显式声明。gemini-code-assist[bot] 指出，若 sampled_token_ids 为 int32，内核仍可能执行 8 字节写入，潜在内存损坏风险。虽然当前 vLLM 中 sampled_token_ids 通常为 int64，但依赖隐式行为不够健壮。此外，topk_token_ids 在 num_logprobs == 0 时使用 logprob_token_ids_state.token_ids.gpu 作为占位，若实际访问该张量的 top-k 数据（即 _fill_logprob_token_ids_kernel 中 topk_indices_ptr 被访问分支处理），但该分支仅在 num_custom == 0 且 valid = col < NUM_TOPK （此处 NUM_TOPK=0）时无效，因此实际安全。
- 影响：直接影响 MRV2 模式下的 logprob 计算，修复了特定场景的 Triton 编译崩溃。间接影响：新增的参数校验避免了 logprobs 与 logprob_token_ids 长度不匹配的无效请求。新增测试覆盖了 sentinel 列截断逻辑。影响范围限于 Model Runner V2 用户（需设置 VLLM_USE_V2_MODEL_RUNNER=1）。
- 风险标记：潜在内存损坏风险（未显式指定 dtype），MRV2 专属修复，不影响 MRV1

关联脉络

- PR #41286 [Feature]: Migration from Model Runner v1 to Model Runner v2: 本 PR 是 Migration 路线图中的一项任务，修复了 MRV2 中的 logprob dtype 问题。
- PR #43160 另一项 MRV2 相关 PR: 同属 MRV2 迁移路线图，可能涉及其他模块或后续改进。
- PR #42538 另一项 MRV2 相关 PR: 同属 MRV2 迁移路线图。