PR #49757 完整报告

vllm-project/vllm

[BugFix] Stop dummy runs from writing mamba state through stale block-table rows

合并时间：2026-07-29 09:09

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/49757

执行摘要

- 一句话：修复 dummy run 通过陈旧 block table 行写入 mamba 状态
- 推荐动作：建议深入阅读此 PR，它展示了如何定位和修复数据损坏的典型思路：从源头和重放路径双重修复。对于涉及 cudagraph 和 speculative decoding 的开发者尤其有价值。后续可关注 follow-up：将零填充 gating 在混合模型上以减少不必要的开销。

功能与动机

在混合模型 P/D 分离和 DP+EP 解码场景下，虚拟运行批次通过 mamba/GDN/KDA 路径写入状态时可能使用陈旧 block id，导致罕见的第一个 token 损坏。修复此问题以提升混合模型部署的稳定性。

实现拆解

1. 清除 vacated block-table 行：在 vllm/v1/worker/block_table.py 的 BlockTable.move_row() 中，将源行的 block ids 和 num_blocks_per_row 置零，阻止 condense 后 vacated 行残留陈旧 id。
2. 返回零填充 dummy block tables：在 vllm/v1/worker/gpu/block_table.py 的 BlockTables.get_dummy_block_tables() 中，返回 zero_() 后的持久张量切片，保持地址不变同时确保 dummy run 不会使用前一次的真实 block id。
3. 重新注入 FULL cudagraph 元数据缓冲区：在 V1 模型运行器 vllm/v1/worker/gpu_model_runner.py 的 _dummy_run() 中，当 cudagraph_runtime_mode == FULL 时传递 for_cudagraph_capture=True；在 V2 模型运行器 vllm/v1/worker/gpu/model_runner.py 的 execute_model() 中，当 dummy run 且 cg_mode 为 FULL 时传递 for_capture=True。这确保零 block tables 被传播到 cudagraph 重放的元数据缓冲区。
4. 测试配套：在 tests/v1/worker/test_gpu_block_table.py 中新增两个测试，分别验证 move_row 清除 vacated 行，以及 get_dummy_block_tables 返回零行并保持持久地址。

关键文件：

- vllm/v1/worker/gpu/block_table.py（模块 GPU 块表；类别 source；类型 core-logic；符号 get_dummy_block_tables）：核心修复：修改 get_dummy_block_tables 返回零填充行并保持持久地址，阻止 dummy run 使用陈旧 block id 写入状态。

- vllm/v1/worker/block_table.py（模块 CPU 块表；类别 source；类型 core-logic；符号 move_row）：核心修复：在 move_row 中清除 vacated 行的 block ids 和 num_blocks_per_row，防止 condense 后残留陈旧 id 被 dummy run 引用。
- vllm/v1/worker/gpu_model_runner.py（模块 V1 运行器；类别 source；类型 data-contract；符号 _dummy_run）：V1 模型运行器，修改 _dummy_run 中 for_cudagraph_capture 参数，确保 FULL cudagraph 重放时重新注入零 block tables 到注意力元数据。
- vllm/v1/worker/gpu/model_runner.py（模块 V2 运行器；类别 source；类型 data-contract；符号 execute_model）：V2 模型运行器，传递 for_capture 参数到 prepare_attn，确保 FULL 模式下 dummy run 的注意力元数据基于零 block tables 构建。
- tests/v1/worker/test_gpu_block_table.py（模块 GPU 块表测试；类别 test；类型 test-coverage；符号 test_v1_block_table_move_row_clears_vacated_row, test_get_dummy_block_tables_returns_zeroed_rows）：新增两个测试验证 move_row 清除 vacated 行和 get_dummy_block_tables 返回零行并保持持久地址，覆盖修复的两种场景。

关键符号：BlockTable.move_row, BlockTables.get_dummy_block_tables, _dummy_run (V1), execute_model (V2), test_v1_block_table_move_row_clears_vacated_row, test_get_dummy_block_tables_returns_zeroed_rows

关键源码片段

vllm/v1/worker/gpu_model_runner.py

V1 模型运行器，修改 _dummy_run 中 for_cudagraph_capture 参数，确保 FULL cudagraph 重放时重新注入零 block tables 到注意力元数据。

```
attn_metadata, _ = self._build_attention_metadata(
    num_tokens=num_tokens_unpadded,
    num_tokens_padded=num_tokens_padded if pad_attn else None,
    num_reqs=num_reqs_padded,
    max_query_len=max_query_len,
    ubatch_slices=(ubatch_slices_padded if pad_attn else ubatch_slices),
    # FULL replay reads capture-time metadata buffers. Re-stage them
    # from the zeroed dummy block tables instead of retaining state
    # indices from the previous real batch.
    for_cudagraph_capture=(
        is_graph_capturing
        or cudagraph_runtime_mode == CUDAGraphMode.FULL
    ),
    slot_mappings=slot_mappings_by_group,
    use_spec_decode=self.speculative_config is not None,
)
```

vllm/v1/worker/gpu/model_runner.py

V2 模型运行器，传递 for_capture 参数到 prepare_attn，确保 FULL 模式下 dummy run 的注意力元数据基于零 block tables 构建。

```
            attn_metadata = self.model_state.prepare_attn(
                input_batch,
                batch_desc.cg_mode,
                block_tables,
                slot_mappings,
                self.attn_groups,
                self.kv_cache_config,
                # FULL replay reads capture-time metadata buffers. Re-stage them
                # from the zeroed dummy block tables instead of retaining state
                # indices from the previous real batch.
                for_capture=dummy_run and batch_desc.cg_mode == CUDAGraphMode.FULL,
            )
```

tests/v1/worker/test_gpu_block_table.py

新增两个测试验证 move_row 清除 vacated 行和 get_dummy_block_tables 返回零行并保持持久地址，覆盖修复的两种场景。

```
def test_v1_block_table_move_row_clears_vacated_row():
    """condense() moves the last row into a freed slot; the vacated row must
    not keep stale block ids. Padded dummy-run batches dereference stale rows
    as mamba state slots (bypassing the NULL_BLOCK_ID fill of real decode
    padding) and write state in place there — corrupting the blocks' new
    owner once they are reallocated, e.g. to an in-flight NIXL load."""
    from vllm.v1.worker.block_table import BlockTable
    block_table = BlockTable(
        block_size=16,
        max_num_reqs=4,
        max_num_blocks_per_req=8,
        max_num_batched_tokens=64,
        pin_memory=False,
        device=torch.device("cuda"),
        kernel_block_size=16,
        cp_kv_cache_interleave_size=1,
    )
    block_table.add_row([7, 8, 9], row_idx=0)
    block_table.add_row([4, 5], row_idx=1)
    block_table.move_row(1, 0)
    assert block_table.block_table.np[0, :2].tolist() == [4, 5]
    assert block_table.num_blocks_per_row[0] == 2
    # The vacated source row routes to the reserved null block.
    assert block_table.num_blocks_per_row[1] == 0
    assert (block_table.block_table.np[1] == 0).all()


def test_get_dummy_block_tables_returns_zeroed_rows():
    """Dummy runs bypass the gather, so the persistent input_block_tables
    hold the previous real step's rows. Mamba/GDN metadata routes in-place
    state writes through block_table[:, 0] (dummy slot mappings are
    PAD-filled, state indices are not), so stale rows would direct dummy
```

```
    state writes at freed — possibly reallocated — blocks.
    get_dummy_block_tables must hand out zeroed (null block) rows while
    preserving the persistent storage address for CUDA graphs."""
    device = torch.device("cuda")
    block_tables = BlockTables(
        block_sizes=[16],
        max_num_reqs=4,
        max_num_batched_tokens=64,
        max_num_blocks_per_group=[8],
        device=device,
        kernel_block_sizes=[16],
    )
    # Simulate a real step: stage a request's blocks and gather them into
    # the persistent input block tables.
    block_tables.append_block_ids(req_index=0, new_block_ids=([1, 2],), overwrite=True)
    block_tables.apply_staged_writes()
    idx_mapping = torch.zeros(1, dtype=torch.int32, device=device)
    block_tables.gather_block_tables(idx_mapping, num_reqs_padded=1)
    torch.accelerator.synchronize()
    assert block_tables.input_block_tables[0][0, 0].item() == 1
    dummy = block_tables.get_dummy_block_tables(num_reqs=1)
    torch.accelerator.synchronize()
    assert (dummy[0] == 0).all()
    # CUDA graph invariant: same persistent tensor, not a fresh allocation.
    assert dummy[0].data_ptr() == block_tables.input_block_tables[0].data_ptr()
```

评论区精华

WoosukKwon 在 V2 模型运行器的变更上评论「I kinda understand the intent here, but I don't feel this is the right solution」（我大致理解意图，但感觉这不是正确的解决方案）。majunze2001 回复「This did fix the corruption but I didn't find the root cause. I'll take another look into this」（这确实修复了损坏，但我没有找到根本原因，我会再研究）。讨论未就根因达成一致，但 PR 仍被批准合并。

- for_capture 方案的正确性 (design): 根因未完全明确，方案被批准但设计正确性留待后续调查。

风险与影响

- 风险：风险较低，但值得关注：
  - 核心路径变更：修改了 block_table.py 和模型运行器中的关键逻辑，可能影响其他依赖 get_dummy_block_tables 和 for_cudagraph_capture 的模块。
  - 无条件零填充：当前对所有模型（包括非混合模型）都进行零填充，虽性能开销极小，但可通过 gate 优化。
  - 设计正确性存疑：WoosukKwon 认为 V2 中的解决方案不彻底，未来版本可能需重新审视根因并调整。

- 影响：正面影响：修复了混合模型 (Mamba/GDN/KDA) 在 P/D 分离和 DP+EP 解码下的罕见数据损坏，提升系统稳定性。负面影响：对非混合模型无功能影响，但引入极轻微的性能开销（每次 dummy run 对 block table 行进行零填充）。团队需关注后续根因分析的进展以决定是否需要优化。
- 风险标记：核心路径变更，混合模型特定

关联脉络

- PR #49995 [MRV2] Always build attn metadata at capture time: 与本 PR 第二个补丁作用域不同，但都涉及 cudagraph 捕获时的注意力元数据构建；本 PR 确保运行时 FULL 模式下也进行 restaging，而非仅捕获时。
- PR #37728 Clear block-table rows in remove_request: 清除了 remove_request 中的 block-table 行，但未覆盖 condense vacated 的行；本 PR 补充了 condense 路径的清理。
- PR #49010 FlashInfer/XQA verification-routing fix: 体指出其作用域与本 PR 不同（FlashInfer/XQA verification-routing fix vs block-table 脏行），但都是修复混合模型相关损坏。