PR #40346 完整报告

vllm-project/vllm

[KV Offload] Offload all KV blocks when doing prefill in P/D

合并时间：2026-04-26 20:06

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40346

执行摘要

- 一句话：P/D 预填充时全量 offload KV 块
- 推荐动作：值得精读。PR 虽改动量小（+71/-2），但解决了 P/D 解耦中的关键数据流问题。代码简洁且测试完善，展示了如何通过条件跳过索引更新来实现"全量 offload"语义。对于研究 KV 传输或多节点推理的开发者具有参考价值。

功能与动机

Support P/D (Prefill/Decode) disaggregation in the OffloadingConnector by ensuring the prefill instance offloads all KV blocks to CPU — not just the incremental delta since the last store. When kv_transfer_params["do_remote_decode"] is set, the scheduler now resets the stored block index so the full KV cache is available on CPU for a remote decode node to consume.

实现拆解

1. 核心逻辑修改（scheduler.py）：在 OffloadingScheduler.update_state_after_alloc() 方法中，新增从 req_context.kv_transfer_params 提取 do_remote_decode 参数的逻辑。当该参数为 True 时，跳过设置 group_state.next_stored_block_idx = num_blocks，使得后续 store 操作不会跳过已加载的缓存块前缀，从而确保所有 KV 块（包括已命中 CPU 缓存的块）都被写入 CPU 存储。
2. 测试工具增强（utils.py）：为测试基类 Runner 的 new_request() 方法增加可选的 kv_transfer_params 参数，允许测试用例方便地传递 {"do_remote_decode": True} 标志。
3. 新增集成测试（test_scheduler.py）：添加参数化测试 test_do_remote_decode_stores_all_blocks，覆盖 async_scheduling=True/False 两种模式。测试流程：先执行一个普通请求存储 1 个 offloaded 块（3 个 GPU 块）；重置 GPU 前缀缓存使后续请求必须从 CPU 加载；再发起一个 do_remote_decode=True 的请求，加载第一个 offloaded 块后，验证最终 store 包含全部 6 个 GPU 块（已加载的 3 个 + 新计算的 3 个），而非仅新的 3 个块。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py（模块 卸载调度；类别 source；类型 core-logic；符号 update_state_after_alloc, _get_reqs_to_store）：核心变更文件，在 update_state_after_alloc 方法中根据 do_remote_decode 标志跳过

next_stored_block_idx 设置，实现全量 offload。

- tests/v1/kv_connector/unit/offloading_connector/test_scheduler.py（模块 卸载测试；类别 test；类型 test-coverage；符号 test_do_remote_decode_stores_all_blocks）：新增集成测试，验证 do_remote_decode=True 时 store 包含所有 GPU 块。
- tests/v1/kv_connector/unit/offloading_connector/utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 new_request）：测试工具函数增强，为 new_request 添加 kv_transfer_params 参数支持。

关键符号：update_state_after_alloc, _get_reqs_to_store, new_request, test_do_remote_decode_stores_all_blocks

关键源码片段

vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py

核心变更文件，在 update_state_after_alloc 方法中根据 do_remote_decode 标志跳过 next_stored_block_idx 设置，实现全量 offload。

```
def update_state_after_alloc(
    self, request: Request, blocks: KVCacheBlocks, num_external_tokens: int
):
    # ... 省略前面的变量定义
    # 从请求上下文中提取 do_remote_decode 参数
    params = req_status.req_context.kv_transfer_params
    do_remote_decode = params is not None and params.get("do_remote_decode")

    keys_to_load: list[OffloadKey] = []
    dst_block_ids: list[int] = []
    group_sizes: list[int] = []
    block_indices: list[int] = []

    for group_config, group_state, group_blocks in zip(...):
        # ... 计算 num_gpu_blocks, num_locally_computed_gpu_blocks, num_pending_gpu_
        blocks ...

        num_blocks = cdiv(num_cached_tokens, offloaded_block_size)
        # ... 计算要加载的 keys ...

        # 核心修改：仅在非远程解码模式下推进 next_stored_block_idx
        if not do_remote_decode:
            # 对于 P/D prefill 请求（do_remote_decode=True），
            # 不跳过已命中缓存的块，以流式传输整个请求
            group_state.next_stored_block_idx = num_blocks

    # ... 后续的 load/store 处理 ...
```

tests/v1/kv_connector/unit/offloading_connector/test_scheduler.py

新增集成测试，验证 do_remote_decode=True 时 store 包含所有 GPU 块。

```
@pytest.mark.parametrize("async_scheduling", [True, False])
def test_do_remote_decode_stores_all_blocks(request_runner, async_scheduling: bool):
    """
    With do_remote_decode=True, after loading prefix blocks from CPU,
    all blocks must be re-stored — not just the newly computed ones.
    This supports P/D disaggregation where the prefill instance offloads the
    complete KV cache so a remote decode node can consume it.
    """
    offloaded_block_size = 12
    gpu_block_size = 4
    num_gpu_blocks = 100

    runner = request_runner(
        offloaded_block_size=offloaded_block_size,
        gpu_block_size=gpu_block_size,
        num_gpu_blocks=num_gpu_blocks,
        async_scheduling=async_scheduling,
    )

    # 步骤 1：正常请求，存储 1 个 offloaded 块（3 个 GPU 块）
    runner.new_request(token_ids=[0] * offloaded_block_size)
    runner.manager.prepare_store.side_effect = (
        lambda keys, req_context: generate_store_output(keys)
    )
    runner.run(
        decoded_tokens=[EOS_TOKEN_ID],
        expected_stored_gpu_block_indexes=(0, 1, 2),
    )

    # 步骤 2：重置 GPU 前缀缓存，使下一个请求必须从 CPU 加载
    runner.scheduler.reset_prefix_cache()

    # 步骤 3：发起 do_remote_decode=True 的请求，2 个 offloaded 块
    runner.new_request(
        token_ids=[0] * offloaded_block_size * 2,
        kv_transfer_params={"do_remote_decode": True},
    )
    # 设置前缀查找返回 1，模拟命中第一个 offloaded 块
    runner.connector_scheduler._maximal_prefix_lookup = lambda key, req_context: 1
    runner.manager.prepare_store.side_effect = (
        lambda keys, req_context: generate_store_output(keys)
    )

    # 步骤 4：加载第一个 offloaded 块（3 个 GPU 块）
    runner.run(
        decoded_tokens=[0],
        expected_loaded_gpu_block_indexes=(0, 1, 2),
    )
```

```
    # 步骤 5：验证 store 包含全部 6 个 GPU 块（已加载的 3 个 + 新计算的 3 个）
    runner.run(
        decoded_tokens=[EOS_TOKEN_ID],
        expected_stored_gpu_block_indexes=(0, 1, 2, 3, 4, 5),
    )
```

评论区精华

- 设计疑问：NickLucche 提问为何在 P 已 offload 块后还需要此修改？orozery 解释：P/D from CPU 场景下，P 需要流式传输整个请求，因为远程 decode 节点无法直接从 GPU 缓存读取，必须从 CPU 存储获取完整的 KV 缓存。当使用 MultiConnector 时，本地 CPU 后端会检测到已有块，不会重复存储，因此对 MultiConnector 无影响，但对独立 OffloadingConnector 是必要的。
- 注释优化：orozery 建议将注释改写为更清晰的表述，作者已采纳更新。
- 测试覆盖：orozery 最初认为新测试可能已被现有测试覆盖，但最终保留了测试，确保新场景有明确验证，orozery 后续批准了 PR。
  - 为什么需要全量 offload（NickLucche 提问）（question）：作者和评审者一致认为该修改对 P/D+Offloading 场景必要。
  - 注释改进（orozery 建议）（documentation）：已采纳，更新注释。
  - 测试必要性（orozery 认为已有测试覆盖）（testing）：测试保留，确保新场景有明确覆盖。

风险与影响

- 风险：低风险。变更受 do_remote_decode 条件严格保护，不影响普通 offload 路径。通过 params.get("do_remote_decode") 安全访问，避免了属性缺失异常。新增的测试覆盖了主要分支和边界情况（async_scheduling 两种模式）。潜在风险是如果在非 P/D 场景误传 do_remote_decode=True 可能导致全量 offload 影响性能，但这属于使用层面的错误配置。
- 影响：影响范围限于使用 OffloadingConnector 进行 P/D 解耦的用户。需要用户在请求中设置 kv_transfer_params["do_remote_decode"] = True 来启用。对普通 offload 用户无影响，向后兼容。该修改使得 P/D 预填充实例能够为远程解码实例提供完整的 KV 缓存，是 P/D 解耦功能的基础修复。
- 风险标记：低风险，条件控制，测试覆盖

关联脉络

- PR #39403 [kv_offload+HMA][11/N]: Support store with multiple KV groups: 同为 offloading connector 的 store 路径增强，与本 PR 共享上下文和核心逻辑。
- PR #38503 [ROCm][Engine] Fix GPU memory leaks in engine shutdown and test workaround for async KV prefix cache reset: 涉及 KV offload 状态重置和内存泄漏修复，与本 PR 的 offload 完整性相关。