# PR #34982 完整报告

sgl-project/sglang

[misc] Rename shared-read boundary to shared-read ends and fix wrapper delegation

合并时间：2026-08-17 05:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34982

## 执行摘要

- 一句话：修复 wrapper 后端共享读取栅栏委托，并按语义重命名 read-end
- 推荐动作：值得精读。本 PR 是一个典型的数据契约修正 + 调度正确性修复结合体，重点关注：SharedReadEnds.max_of 的 " 按 lateness 聚合 " 设计如何优雅处理多后端组合；_replay_attn_backend() 的统一如何消除 pdmux 下的隐式不一致；以及 POST_REPLAY 从 " 设计状态 " 重新定义为 " 未实现标记 " 这一可审计性思路。测试部分用 create_autospec 替代手写 stub 的做法也值得借鉴。

## 功能与动机

PR body 明确指出这是一个真实的调度正确性 bug："This is a real fence bug wherever a wrapped child overrides the default. Concretely, TboAttnBackend wrapping DeepseekV4AttnBackend: the child declares POST_REPLAY for non-DSPARK target verify, the wrapper reported IN_REPLAY, so the scheduler was released while the child was still reading across graph segments." 同时作者希望把命名改得更贴合实际语义——"name the read-end concept after what it is"，让 POST_REPLAY 明确标记 " 元数据快照未实现 " 而非一种设计状态，从而让剩余使用者可被审计。

## 实现拆解

### 1. 契约层：SharedReadBoundary 重命名为 SharedReadEnds 并显式排序

- 在 python/sglang/srt/layers/attention/base_attn_backend.py 中，枚举改为显式整数取值 PRE_REPLAY=1 / IN_REPLAY=2 / POST_REPLAY=3 / UNKNOWN=4，注释直接写到两个 init hook 的相对位置（init_forward_metadata_out_graph / init_forward_metadata_in_graph），POST_REPLAY 被重新定义为 " 元数据快照未实现 " 的审计标记。
- 新增静态方法 SharedReadEnds.max_of(items)，按枚举值取最大值，语义是 " 最新（最保守）的结束点覆盖所有子 backend"；UNKNOWN=4 最大，因此任一子后端未审计时聚合结果自动落回粗粒度 fence。
- 基类方法 shared_read_boundary(forward_mode) 同步更名为 shared_read_ends(fm)，默认逻辑不变（decode / target verify 返回 IN_REPLAY）。

### 2. 运行器：提取 _replay_attn_backend 并同步解析逻辑

- 在 decode_cuda_graph_runner.py 中，_resolve_shared_read_boundary 更名为 _resolve_shared_read_ends，参数名从 boundary 改为 declared，强调这是后端的声明而非解析结果。
- 新增 _replay_attn_backend() 方法，统一 pdmux 下 stream 对应的后端选择逻辑：execute() 此前直接使用 self.attn_backend，而 load_batch() 使用 decode_attn_backend_group[stream_idx]，两者在 pdmux 下不一致；现在 load_batch 与 execute 都通过 _replay_attn_backend() 获取后端，保证读结束判定与 metadata 初始化基于同一后端对象。
- _publish_read_done(in_graph) 逻辑不变：in-graph 直接移交图内事件，out-of-graph 则新 record 一个设备事件。

### 3. 四个 wrapper backend 补上委托

- HybridAttnBackend.shared_read_ends 委托给 _select_backend(fm) 选中的实际后端。
- TboAttnBackend.shared_read_ends 对 (self.primary, *self.children) 取 max_of。
- HybridLinearAttnBackend.shared_read_ends 对 attn_backend_list 取 max_of。
- MiniMaxHybridAttnBackend.shared_read_ends 对 (self.sparse, self.dense) 取 max_of。
- 这是修复的核心：修复前这些 wrapper 继承基类默认 IN_REPLAY，当子后端声明更晚的读结束点（如 DeepseekV4AttnBackend 非 DSPARK target verify 声明 POST_REPLAY）时，wrapper 会给出过早的栅栏释放信号。

### 4. 类名与方法调用的全仓同步

- trtllm_mha_backend.py、deepseek_v4_backend.py、shared_read_event.py 同步切换到 SharedReadEnds + shared_read_ends，其中 shared_read_event.py 的 prefill 路径用 declared 局部变量表达同样的 " 声明 " 语义。

### 5. 测试精简（136 -> 80 行）

- 删除 execute() 级 harness（ShapeKey、PPProxyTensors、load_batch、device_module 等约 60 行），改为直接测试 _resolve_shared_read_ends（6 行参数化表格覆盖 mode / owns_verify / declared / has_marker / expected 五元组）和 _publish_read_done 两个分支。
- 后端桩从 SimpleNamespace 改为 create_autospec(AttentionBackend, instance=True)，方法名拼错会在测试时直接失败，而不是 CI 全绿、运行时崩。
- test_prefill_shared_read_done.py 同步适配方法重命名。

关键文件：

- python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py（模块 运行器；类别 source；类型 data-contract；符号 _resolve_shared_read_ends, _replay_attn_backend, _publish_read_done）：decode CUDA graph 重放主路径，_resolve_shared_read_ends 决定栅栏发布时机，新增 _replay_attn_backend 统一 pdmux 后端选择，修复 execute/load_batch 间的后端不一致。
- python/sglang/srt/layers/attention/base_attn_backend.py（模块 注意力层；类别 source；类型 core-logic；符号 SharedReadEnds, max_of, shared_read_ends）：核心契约文件

：SharedReadEnds 枚举显式排序并提供 max_of 聚合，基类方法更名，所有 backend 都依赖此契约。

- test/registered/unit/model_executor/runner/test_decode_cuda_graph_shared_read_fence.py（模块 单元测试；类别 test；类型 test-coverage；符号 _runner, _backend, test_resolve_shared_read_ends, test_publish_read_done）：fence 测试从 136 行精简到 80 行，删除 execute 级 harness，改用 create_autospec 使方法重命名在测试期即失败，并沿用参数化表格覆盖 6 种 resolve 分支。
- python/sglang/srt/layers/attention/tbo_backend.py（模块 重叠后端；类别 source；类型 core-logic；符号 shared_read_ends）：真实 bug 所在：TboAttnBackend 包装 DeepseekV4AttnBackend 时此前继承默认 IN_REPLAY，现通过 max_of 聚合 primary + children 的声明。
- python/sglang/srt/layers/attention/deepseek_v4_backend.py（模块 注意力层；类别 source；类型 core-logic；符号 shared_read_ends）：POST_REPLAY 声明的来源：非 DSPARK 的 breakable-graph target verify 会跨图段重读共享状态，是触发 Tbo wrapper bug 的子后端声明方。
- python/sglang/srt/layers/attention/hybrid_attn_backend.py（模块 混合后端；类别 source；类型 core-logic；符号 shared_read_ends）：HybridAttnBackend 委托给 _select_backend(mode) 选中的实际后端，避免 wrapper 为自己而非子后端回答。
- python/sglang/srt/layers/attention/hybrid_linear_attn_backend.py（模块 混合后端；类别 source；类型 core-logic；符号 shared_read_ends）：HybridLinearAttnBackend 对 attn_backend_list 取 max_of，覆盖所有线性 / 注意力子后端。
- python/sglang/srt/layers/attention/minimax_sparse_backend.py（模块 注意力层；类别 source；类型 core-logic；符号 shared_read_ends）：MiniMaxHybridAttnBackend 对 sparse + dense 两个子后端取 max_of。
- python/sglang/srt/layers/attention/trtllm_mha_backend.py（模块 注意力层；类别 source；类型 core-logic；符号 shared_read_ends）：TRTLLMHAAttnBackend 是另一个覆写 read-end 声明的具体后端，方法名与方法体同步到新契约。
- python/sglang/srt/model_executor/runner_utils/shared_read_event.py（模块 共享读取；类别 source；类型 data-contract）：prefill 侧 shared-read-done 快速路径同步使用新枚举与命名，局部变量改为 declared 表达声明语义。
- test/registered/unit/model_executor/runner/test_prefill_shared_read_done.py（模块 单元测试；类别 test；类型 test-coverage）：prefill shared-read-done 测试同步适配新枚举与方法名。

关键符号：SharedReadEnds.max_of, shared_read_ends, _resolve_shared_read_ends, _replay_attn_backend, _publish_read_done, maybe_publish_prefill_shared_read_done

## 关键源码片段

### python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py

decode CUDA graph 重放主路径，_resolve_shared_read_ends 决定栅栏发布时机，新增 _replay_attn_backend 统一 pdmux 后端选择，修复 execute/load_batch 间的后端不一致。

```
# python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py
# pdmux 下每个 stream 用自己的 group member 重放，读结束判定也必须
# 基于同一个 backend，否则会拿到错误的声明。
def _replay_attn_backend(self) -> AttentionBackend:
    if self.enable_pdmux:
        return self.model_runner.decode_attn_backend_group[get_current_stream_idx()]
    return self.attn_backend


def _resolve_shared_read_ends(self, attn_backend, forward_mode) -> SharedReadEnds:
    """后端声明经降级后的结果；UNKNOWN 表示不记录事件，由调度器保留粗粒度栅栏。"""
    if forward_mode.is_target_verify():
        if not self.model_runner.spec_algorithm.is_last_shared_read_phase(forward_mode):
            return SharedReadEnds.UNKNOWN
    elif not forward_mode.is_decode():
        return SharedReadEnds.UNKNOWN
    declared = attn_backend.shared_read_ends(forward_mode)

    if (
        declared is SharedReadEnds.IN_REPLAY
        and self.in_graph_metadata_prep_done is None
    ):
        # TODO: 这里落到比声明更早的 PRE_REPLAY；POST_REPLAY 才是 sound 降级方向
        return SharedReadEnds.PRE_REPLAY
    return declared


def _publish_read_done(self, in_graph: bool):
    """把本阶段共享缓冲区读取结束事件交给调度器的 WAR 屏障。"""
    if in_graph:
        # 读取在图内标记处结束：直接移交图内事件，不重新 record
        self.model_runner.shared_read_done_event = self.in_graph_metadata_prep_done
    else:
        read_done = self.device_module.Event()
        read_done.record()
        self.model_runner.shared_read_done_event = read_done
```

## python/sglang/srt/layers/attention/base_attn_backend.py

核心契约文件：SharedReadEnds 枚举显式排序并提供 max_of 聚合，基类方法更名，所有 backend 都依赖此契约。

```
# python/sglang/srt/layers/attention/base_attn_backend.py
# 枚举值按结束早晚排序，max_of 因此能聚合多个子 backend 的声明：
# 只要任一子 backend 读到更晚（或 UNKNOWN 未审计），整体取最保守的结束点。
class SharedReadEnds(Enum):
    PRE_REPLAY = 1 # 在 init_forward_metadata_out_graph 之后结束
    IN_REPLAY = 2 # 在 init_forward_metadata_in_graph 之后结束
    POST_REPLAY = 3 # 元数据快照未实现 -> 读到整个 replay 结束之后
    UNKNOWN = 4 # 未审计 -> 走粗粒度 whole-forward 栅栏
```

```
    @staticmethod
    def max_of(items: Iterable[SharedReadEnds]) -> SharedReadEnds:
        # 按 lateness 排序：最新（最保守）的结束点覆盖所有子 backend
        return max(items, key=lambda x: x.value)


class AttentionBackend(ABC):
    # ...

    def shared_read_ends(self, fm: ForwardMode) -> SharedReadEnds:
        """声明本后端调度器共享读取在每个模式下的结束点。
        仅覆盖经过审计的偏离；默认返回保守的 IN_REPLAY。"""
        if fm.is_decode() or fm.is_target_verify():
            return SharedReadEnds.IN_REPLAY
        return SharedReadEnds.UNKNOWN
```

### python/sglang/srt/layers/attention/tbo_backend.py

真实 bug 所在：TboAttnBackend 包装 DeepseekV4AttnBackend 时此前继承默认 IN_REPLAY，现通过 max_of 聚合 primary + children 的声明。

```
# python/sglang/srt/layers/attention/tbo_backend.py
class TboAttnBackend(AttentionBackend):
    # 修复前 wrapper 继承基类 IN_REPLAY 默认值；当子后端（如
    # DeepseekV4AttnBackend 非 DSPARK target verify）声明 POST_REPLAY 时，
    # wrapper 会过早释放调度器。现在取 primary + children 的最晚结束点。
    def shared_read_ends(self, fm: ForwardMode) -> SharedReadEnds:
        return SharedReadEnds.max_of(
            b.shared_read_ends(fm) for b in (self.primary, *self.children)
        )
```

## 评论区精华

该 PR 无 review 评论（comments_count=0，review_comments_count=0）。设计权衡记录在 PR body 的 "Not in this PR" 部分，核心观点包括：

- 默认值刻意保持 IN_REPLAY，PRE_REPLAY 是逐后端审计后的优化，不应全局翻转；
- _resolve_shared_read_ends 仍同时回答 " 哪个阶段拥有发布权 " 和 " 读到哪里结束 " 两个问题，拆分所有权谓词是下一步计划；
- IN_REPLAY -> PRE_REPLAY 的 fallback 只会在 --debug-cuda-graph 且无 external-event 支持的平台触达（此时 barrier 本身关闭），POST_REPLAY 才是 sound 方向，代码中留有 TODO。
- 暂无高价值评论线程

## 风险与影响

- 风险：

1. wrapper 栅栏行为变化：修复后 TboAttnBackend 包装 DeepseekV4AttnBackend 在非 DSPARK target verify 下从 IN_REPLAY 变为 POST_REPLAY，调度 fence 更保守，可能轻微降低 two-batch overlap 的重叠程度，但这是正确性优先的权衡。
2. 全局契约重命名：shared_read_boundary / SharedReadBoundary 是 attention backend 基类公开 API，仓库内引用已全部同步，但 out-of-tree 自定义 backend 若覆写了旧方法名会在运行时报 AttributeError，属于可预期的兼容性代价。
3. pdmux 行为修正：*execute*() 之前用 self.attn_backend 而非当前 stream 的 group member，本 PR 通过 _replay_attn_backend() 统一；对于已开启 pdmux 且多 stream 使用不同后端的场景，这是一次真实更正，但也属于行为变化，需要回归观察。
4. 测试覆盖缺口：PR body 明确承认不再覆盖 "pre-replay record 落点早于 backend.replay()" 这一顺序断言，该断言此前由完整 harness 保障；新测试只覆盖两个方法的单元逻辑，*execute*() 的集成路径缺少测试守卫。

- 影响：影响范围集中在 *decode* CUDA graph 重放路径的调度栅栏（WAR barrier）时机：
  - 对 DeepSeek V4 + Blackwell + TBO 场景，修复了 target verify 阶段 scheduler 过早释放的问题，属于正确性修复；
  - 对 Hybrid / MiniMax / TBO 等 wrapper 后端的用户，栅栏发布点整体更保守，默认（非 target verify）路径行为不变，现有用户通常无需改动；
  - 对后端开发者，shared_read_ends 的排序语义（max_of）提供了可组合的聚合契约，降低了 wrapper 场景下遗漏声明的概率；
  - 对测试维护者，*fence* 测试从 136 行降到 80 行，且 *create_autospec* 让方法重命名这类重构能被测试直接捕获。
  - 风险标记：核心解码路径变更，wrapper 栅栏更保守，全局契约重命名，pdmux 行为修正，*execute* 级测试覆盖被删

# 关联脉络

- PR #34916 Shared-read boundary 概念引入（本 PR 的前置，未在提供的历史列表中）：PR body 明确标注为 "Follow-up to #34916"，本 PR 是其命名与 wrapper 委托的修正版。
- PR #35057 [Spec] Point multi-layer *eagle*'s last shared-read runner at the draft runner: 同属 speculative *decoding* 的 shared-read 归属问题，与 shared read 阶段的 runner 指派 / 声明相关，体现同一功能线的持续修正。