PR #34052 完整报告

sgl-project/sglang

[Scheduler] Unify WAR read-done gating behind shared-read boundary declarations

合并时间：2026-08-08 18:26

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34052

执行摘要

- 一句话：统一 WAR 栅栏为共享读边界声明，删除策略枚举
- 推荐动作：值得精读，尤其是对计划接入新 attention backend 或需要理解 sglang WAR overlap 机制的工程师。重点学习三点设计：SharedReadBoundary 用枚举表达相对 replay 的边界位置、UNKNOWN 作为安全底值的降级策略、以及默认值从 out-graph/in-graph 契约推导而不是从布尔标志拼接。配合两个 CPU 单测的平移方式可快速掌握整个决策矩阵。

功能与动机

PR body 明确指出：overlap scheduler 的 WAR barrier 每个 forward 只回答一个问题——该 forward 最后一次读取 scheduler 共享缓冲的位置在哪，以便在那里记录 read-done 事件。此前这个问题由四个互不关联的片段回答（prefill_shared_reads_end_at_metadata_init 类布尔、supports_target_verify_war_read_done 布尔方法、use_captured_forward_metadata_for_breakable_cuda_graph 借用标志、WarReadDonePolicy + if 链），"Each consumer held its own fragment, so supporting a new (backend, mode) combination meant adding another flag and threading it through the chain — and asymmetries crept in silently (e.g. the BCG check existed on the verify branch but not decode)"。

实现拆解

实现按 5 步拆解：

1. 声明层（base_attn_backend.py）：新增 SharedReadBoundary 四值枚举（PRE_REPLAY / IN_REPLAY / POST_REPLAY / UNKNOWN），并在 AttentionBackend 上新增 shared_read_boundary(forward_mode) 方法。默认实现依据 out-graph/in-graph 元数据初始化契约返回 decode/verify → IN_REPLAY、其余 → UNKNOWN；同时删除旧的 prefill_shared_reads_end_at_metadata_init 类布尔，UNKNOWN 作为安全底值保证未审计后端退回粗粒度整轮围栏。
2. 算法层（spec_info.py / spec_registry.py）：SpeculativeAlgorithm.is_war_publish_phase(forward_mode) 取代 supports_target_verify_war_read_done，把 " 哪个阶段拥有发布权 " 从后端布尔提升为算法语义：dflash/dspark 家族在 target verify 阶段发布，EAGLE 等其余算法在 draft extend v2 阶段发布。

3. 消费层（decode_cuda_graph_runner.py）：_war_read_done_policy 重写为 _war_read_done_record，变成 " 算法门控 → 后端声明 → 图内锚点降级 " 的机械查表；execute 中按 PRE_REPLAY / POST_REPLAY / IN_REPLAY 三档落点调用 _publish_war_read_done。eagle_draft_extend_cuda_graph_runner.py 补注释说明 draft extend 是 EAGLE 家族的发布阶段。
4. 后端迁移（trtllm_mha_backend.py / deepseek_v4_backend.py）：trtllm_mha 用 EXTEND → PRE_REPLAY 声明取代 prefill 布尔；deepseek_v4 用 TARGET_VERIFY → POST_REPLAY 声明取代对 use_captured_forward_metadata_for_breakable_cuda_graph 的借用，该标志回归纯 BCG 职责。
5. 清理与测试（war_event.py / runner_utils/init.py / 两个单测）：war_event.py 删除 WarReadDonePolicy，maybe_publish_prefill_war_read_done 改用 boundary is PRE_REPLAY 判断；runner_utils/init.py 移除对应导出；test_decode_cuda_graph_war_fence.py 与 test_prefill_war_read_done.py 按新 API 平移同一决策矩阵，断言一一对应。

关键文件：

- python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py（模块 解码图执行；类别 source；类型 data-contract；符号 _war_read_done_policy, _war_read_done_record）：WAR read-done 事件的最终消费与发布点，_war_read_done_policy 重写为 _war_read_done_record，execute 按 SharedReadBoundary 三档落点发布，是本次重构的机械消费核心。
- python/sglang/srt/layers/attention/base_attn_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 SharedReadBoundary, shared_read_boundary）：新增 SharedReadBoundary 枚举与 shared_read_boundary 声明方法，取代旧 bool 标志，是整个统一方案的契约定义处。
- test/registered/unit/model_executor/runner/test_decode_cuda_graph_war_fence.py（模块 单元测试；类别 test；类型 test-coverage；符号 supports_target_verify_war_read_done, is_war_publish_phase, shared_read_boundary, test_war_read_done_policy）：decode 决策矩阵测试按新 API 平移，断言旧 WarReadDonePolicy 语义与新 SharedReadBoundary 语义一一对应。
- python/sglang/srt/model_executor/runner_utils/war_event.py（模块 事件工具；类别 source；类型 data-contract；符号 WarReadDonePolicy）：删除 WarReadDonePolicy 枚举，prefill fastpath 改用 shared_read_boundary(PRE_REPLAY) 判断，是清理落点。
- python/sglang/srt/layers/attention/deepseek_v4_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 shared_read_boundary）：为 verify 声明 POST_REPLAY，取代对 BCG 元数据标志的借用，是边界声明方案的样板 override。
- python/sglang/srt/speculative/spec_info.py（模块 投机解码；类别 source；类型 core-logic；符号 supports_target_verify_war_read_done, is_war_publish_phase）：is_war_publish_phase 取代 supports_target_verify_war_read_done，把发布权从后端布尔提升为算法语义。
- python/sglang/srt/layers/attention/trtllm_mha_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 shared_read_boundary）：prefill 快照语义从布尔标志迁

移为 EXTEND → PRE_REPLAY 声明。

- python/sglang/srt/speculative/spec_registry.py（模块 投机解码；类别 source；类型 core-logic；符号 supports_target_verify_war_read_done, is_war_publish_phase）：registry 侧的对应方法同步替换，保持 Spec 算法双实现一致。
- test/registered/unit/model_executor/runner/test_prefill_war_read_done.py（模块 单元测试；类别 test；类型 test-coverage）：prefill fastpath 测试改用 shared_read_boundary 模拟，验证 PRE_REPLAY 边界门控。
- python/sglang/srt/speculative/eagle_draft_extend_cuda_graph_runner.py（模块 投机解码；类别 source；类型 core-logic）：补充注释明确 draft extend 是 EAGLE 家族的发布阶段，解释该 runner 无条件发布的原因。
- python/sglang/srt/model_executor/runner_utils/__init__.py（模块 事件工具；类别 source；类型 data-contract）：移除 WarReadDonePolicy 的导出，配合枚举删除。

关键符号：shared_read_boundary, _war_read_done_record, is_war_publish_phase, _publish_war_read_done, maybe_publish_prefill_war_read_done, _war_read_done_policy

关键源码片段

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

WAR read-done 事件的最终消费与发布点，_war_read_done_policy 重写为 _war_read_done_record，execute 按 SharedReadBoundary 三档落点发布，是本次重构的机械消费核心。

```
# python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py
# Runner 变为 " 机械消费者 ": 先查算法发布阶段，再查后端边界声明，
# 最后按边界位置决定 read-done 事件落点。UNKNOWN 表示本次 replay
# 不参与 WAR 快速路径发布。

def _war_read_done_record(self, attn_backend, forward_mode) -> SharedReadBoundary:
    """决定本次 replay 在何处记录 WAR read-done 事件；UNKNOWN 表示不记录。"""
    if forward_mode.is_target_verify():
        # 算法门控优先：非发布阶段（如 EAGLE 的 verify）不在此 runner 发布
        if not self.model_runner.spec_algorithm.is_war_publish_phase(forward_mode):
            return SharedReadBoundary.UNKNOWN
    elif not forward_mode.is_decode():
        # decode 之外的模式由其他 runner（如 EAGLE draft extend）负责发布
        return SharedReadBoundary.UNKNOWN
    boundary = attn_backend.shared_read_boundary(forward_mode)
    if (
        boundary is SharedReadBoundary.IN_REPLAY
        and not self._war_read_done_node_planted
    ):
        # 非捕获运行 / 不支持外部事件时图内锚点不存在，回退到 replay 前记录
        return SharedReadBoundary.PRE_REPLAY
    return boundary

def execute(self, forward_batch, pp_proxy_tensors=None):
```

```
    # ... 前置逻辑省略 ...
    war_record = self._war_read_done_record(
        self.attn_backend, forward_batch.forward_mode
    )
    with timer_ctx, self.backend.replay_session():
        self.load_batch(forward_batch, pp_proxy_tensors)
        if war_record is SharedReadBoundary.PRE_REPLAY:
            self._publish_war_read_done(in_graph=False)
        output = self.backend.replay(self._replay_graph_key, forward_batch)
        if war_record is SharedReadBoundary.POST_REPLAY:
            self._publish_war_read_done(in_graph=False)
        elif war_record is SharedReadBoundary.IN_REPLAY:
            self._publish_war_read_done(in_graph=True)
```

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

新增 SharedReadBoundary 枚举与 shared_read_boundary 声明方法，取代旧 bool 标志，是整个统一方案的契约定义处。

```
# python/sglang/srt/layers/attention/base_attn_backend.py
# WAR read-done 边界的统一声明点：SharedReadBoundary 描述 " 后端对
# scheduler 共享缓冲的最后一次读 " 相对 CUDA graph replay 的位置。
# UNKNOWN 是安全底值——不发布事件，scheduler 退回整轮 forward
# 的粗粒度围栏（coarse whole-forward fence）。

class SharedReadBoundary(Enum):
    PRE_REPLAY = auto() # 共享读在 replay 启动前已结束（快照式后端）
    IN_REPLAY = auto() # 共享读在图内元数据初始化处结束（图内锚点）
    POST_REPLAY = auto() # 共享读贯穿整个 graph replay（如 DSV4 BCG verify）
    UNKNOWN = auto() # 未审计 -> 不发布，保留整轮粗粒度围栏


class AttentionBackend(ABC):

    def shared_read_boundary(self, forward_mode: ForwardMode) -> SharedReadBoundary:
        """按 forward 模式声明共享读结束位置。

        decode/verify 默认 IN_REPLAY：out-graph/in-graph 元数据初始化契约
        保证任何遵守该契约的后端，其共享读不会越过图内元数据锚点，
        因此 IN_REPLAY 是安全上界。有审计结论的后端才需要 override。
        """
        if forward_mode.is_decode() or forward_mode.is_target_verify():
            return SharedReadBoundary.IN_REPLAY
        return SharedReadBoundary.UNKNOWN
```

python/sglang/srt/speculative/spec_info.py

is_war_publish_phase 取代 supports_target_verify_war_read_done，把发布权从后端布尔提升为算法语义。

```
# python/sglang/srt/speculative/spec_info.py
```

```
# 每个 spec 步骤的 " 最后读共享缓冲的阶段 " 负责发布 read-done 事件。
# dflash/dspark 系列在 target verify 阶段发布；EAGLE 等其余算法在
# draft extend 阶段发布（由 EAGLE draft extend runner 执行记录）。

def is_war_publish_phase(self, forward_mode) -> bool:
    if self.is_dflash_family():
        return forward_mode.is_target_verify()
    return forward_mode.is_draft_extend_v2()
```

评论区精华

本 PR 无 review 评论与 issue 讨论。从 7 个 commit 的演进序列可还原设计迭代：先完成初步统一（a9bb5072），随即放弃 WarReadDonePolicy 枚举、改用单一 SharedReadBoundary 类型（a4e882eb），再调整默认边界推导来源、由 base 类持有契约默认值、decode runner 保留直接 case 表并删除 resolve helper（d27c5753、2bc85c65、ede7e369）。最终形态的核心收敛决策是：UNKNOWN 作为安全底值、IN_REPLAY 作为契约上界默认、后端只对审计过的偏差做 override。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 行为等价性依赖审计：PR 声称逐分支严格等价，但 verify 分支存在一个隐性变化点——原来通过 use_captured_forward_metadata_for_breakable_cuda_graph 判断 POST_REPLAY，现在只有显式声明 POST_REPLAY 的 deepseek_v4_backend 保留该行为；若未来出现第三个同时设置 BCG 标志并支持 dflash verify 的后端而未声明边界，将从 POST_REPLAY 静默降级为 IN_REPLAY，可能引入 WAR race。base_attn_backend.py 的默认值是建立在 " 后端遵守 out-graph/in-graph 元数据初始化契约 " 这一前提上的安全上界，契约一旦被违反（如在 replay 前向中读 scheduler 共享张量），IN_REPLAY 默认即失效。
  2. 测试覆盖：两个 CPU 单测覆盖了完整决策矩阵，但没有 GPU 端集成测试（如 BCG + dflash verify + deepseek_v4 的真实 replay）验证事件落点与实际 barrier 行为。
  3. 无 review 复核：PR 由作者自行 merge，讨论环节缺失，契约迁移的审计结论缺少第二人确认。 - 影响：对最终用户透明，无 API 或行为变化。对系统：decode/verify 每次 replay 的 read-done 事件位置改由声明驱动，新增 (backend, mode) 组合时只需实现一个 shared_read_boundary override，接入成本显著下降；BCG 检查此前只存在于 verify 分支的不对称也被修复。对团队：删除 WarReadDonePolicy 与两个 bool 方法后，WAR 机制只剩 " 声明 → 记录 → 消费 " 三层，认知负担大幅降低，后续 deepseek_v4、trtllm_mha 等后端的边界语义集中在各自文件中。 - 风险标记：核心路径变更，行为等价依赖审计，默认上界依赖契约，缺少 GPU 集成测试，无 review 复核

关联脉络

- PR #33889 moe: the shared-experts-fusion decision is a per-runner value the loader installs: 同为 " 决策下沉到单一 owner" 的契约重构，消除跨层 flag 穿线，与本 PR 的声明下沉模式一致。
- PR #33887 config: retire ServerArgs.derive; per-runner values are constructor arguments: 同一重构风格：删除派生值与全局 flag，改由构造参数 / 声明直接拥有，与本 PR 删除 WarReadDonePolicy 同源。
- PR #33888 config: delete the dead get_server_args() bindings across the repo: 同一清理脉络，本 PR 删除了 WarReadDonePolicy 导出后应继续清理相关死绑定。
- PR #33404 [Scheduler] Gate SWA eviction on accumulated tokens: 同为 sglang/srt 调度器与图执行协调机制的演进，WAR barrier 与 SWA 淘汰都是调度器与执行器共享状态一致性的关键路径。
- PR #34067 [Bugfix] Fix batched KV free aliasing: KV 复用一致性修复，与 WAR 屏障保护的共享缓冲写后读属于同一一致性主题，共同构成调度器共享缓冲安全体系。