PR #34916 完整报告

sgl-project/sglang

[misc] Rename the WAR read-done fastpath to shared-read-done

合并时间：2026-08-16 06:02

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34916

执行摘要

- 一句话：重命名 WAR 读完成快速路径为共享读完成，并重构事件所有权
- 推荐动作：值得精读 decode_cuda_graph_runner.py 中事件懒创建与边界解析的配合方式，以及 PR 对命名规则的沉淀；这是理解 scheduler 共享缓冲读完成屏障的极好入口。关注 TODO 中 PRE_REPLAY -> POST_REPLAY 的后续修正方向，以及“每个 runner 无条件发布”的演进计划。

功能与动机

PR 正文指出 WAR 命名的是一个 hazard 类别（write-after-read），而代码却用危害名来命名防止该危害的信号——即 forward 阶段在读完 scheduler 共享缓冲区后发布的事件。目标是把信号按“它是什么”来命名，避免语义混淆；环境变量因需要弃用别名而暂不改名。

实现拆解

1. 信号模块更名：runner_utils/war_event.py 重命名为 runner_utils/shared_read_event.py，make_war_read_done_event 改为通用的 make_external_event，maybe_publish_prefill_war_read_done 改为 maybe_publish_prefill_shared_read_done，同步更新 runner_utils/__init__.py 的 re-export。
2. 事件所有权转移与懒创建：ModelRunner.__init__ 中删除了 war_read_done_event / war_fastpath_read_done_event 两个字段，只保留 shared_read_done_event 作为 scheduler 读取的信箱；DecodeCudaGraphRunner 新增 in_graph_metadata_prep_done 字段，在 capture 时通过 make_external_event 懒创建，_plant_war_read_done_node 变为 _record_in_graph_metadata_prep_done，_war_read_done_node_planted 标志被移除。
3. 边界解析与发布改名：_war_read_done_record -> _resolve_shared_read_boundary，_publish_war_read_done -> _publish_read_done；execute() 中按 PRE_REPLAY（replay 前）、IN_REPLAY（replay 后透传图内事件）、POST_REPLAY（replay 后新记录事件）三种边界发布。
4. 算法谓词与消费端改名：spec_info.py / spec_registry.py 中 is_war_publish_phase -> is_last_shared_read_phase；scheduler.py、tp_worker.py、base_spec_worker.py、eagle_worker_v2.py、eagle_draft_extend_cuda_graph_runner.py、prefill_cuda_graph_runner.py、base_attn_backend.py 中所有对 war_fastpath_runner、war_fastpath_read_done_event、maybe_publish_prefill_war_read_done 的引用同步改

名。

5. 测试配套：删除 test_decode_cuda_graph_war_fence.py，新增 test_decode_cuda_graph_shared_read_fence.py，测试裁剪为每个失败模式一个案例，backend fixture 不再镜像基类实现；test_prefill_war_read_done.py 重命名为 test_prefill_shared_read_done.py。

关键文件：

- python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py（模块 图执行器；类别 source；类型 data-contract；符号 _plant_war_read_done_node, _war_read_done_record, _record_in_graph_metadata_prep_done, _resolve_shared_read_boundary）：核心变更文件：in-graph 标记事件的所有权从 ModelRunner 移到这里，并改为 capture 时懒创建；四个方法的改名与边界解析逻辑都集中在此。
- test/registered/unit/model_executor/runner/test_decode_cuda_graph_shared_read_fence.py（模块 图测试；类别 test；类型 test-coverage；符号 _SpecAlgorithm, init, is_last_shared_read_phase, _attn_backend）：新增测试覆盖新命名与去标志化后的 fence 行为，每种失败模式只保留一个用例，backend fixture 不再镜像基类实现。
- python/sglang/srt/model_executor/runner_utils/shared_read_event.py（模块 事件工具；类别 source；类型 rename-or-move；符号 make_war_read_done_event, make_external_event, maybe_publish_prefill_war_read_done, maybe_publish_prefill_shared_read_done）：由 war_event.py 重命名而来，集中承载 external event 创建与 prefill shared-read-done 发布逻辑，是所有 runner 共享的信号工具模块。
- python/sglang/srt/model_executor/model_runner.py（模块 模型执行；类别 source；类型 data-contract）：事件字段从两个精简为一个 mailbox，删除 eager 创建，是所有权转移的另一端。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：WAR 屏障消费端：读取 last_shared_read_runner 的 shared_read_done_event，是信号机制的实际使用者。
- python/sglang/srt/speculative/spec_info.py（模块 推测解码；类别 source；类型 core-logic；符号 is_war_publish_phase, is_last_shared_read_phase）：算法谓词 is_war_publish_phase 改名为 is_last_shared_read_phase，明确其语义是“本步最后一个共享缓冲读取阶段”。
- test/registered/unit/model_executor/runner/test_decode_cuda_graph_war_fence.py（模块 图测试；类别 test；类型 deletion；符号 _SpecAlgorithm, init, is_war_publish_phase, _attn_backend）：旧测试文件整体删除，被 test_decode_cuda_graph_shared_read_fence.py 取代。
- test/registered/unit/model_executor/runner/test_prefill_shared_read_done.py（模块 预填充测试；类别 test；类型 rename-or-move）：prefill 发布路径测试随模块改名同步迁移。

关键符号：_record_in_graph_metadata_prep_done, _resolve_shared_read_boundary, _publish_read_done, maybe_publish_prefill_shared_read_done, make_external_event, is_last_shared_read_phase, _apply_war_barrier

关键源码片段

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

核心变更文件：in-graph 标记事件的所有权从 ModelRunner 移到这里，并改为 capture 时懒创建；四个方法的改名与边界解析逻辑都集中在此。

```
# 文件：python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py

def _record_in_graph_metadata_prep_done(self):
    # 纯粹是图中的一个标记点；共享读真正结束的位置由 attn backend 声明。
    if not torch.cuda.is_current_stream_capturing():
        # warmup 共享此方法体；breakable capture 在进入上下文时打开 segment 1，
        # 且每个 segment 的 replay 都会重新武装该节点。
        return
    if self.in_graph_metadata_prep_done is None:
        # 懒创建：capture 时才建，而不是在 ModelRunner.__init__ 里建。
        self.in_graph_metadata_prep_done = make_external_event(self.device_module)
    event = self.in_graph_metadata_prep_done
    if event is not None:
        # 没有 external-event 支持时保持 None，边界解析就不会交出未记录的事件。
        event.record()

def _resolve_shared_read_boundary(self, attn_backend, forward_mode) -> SharedReadBoundary:
    # 决定这次 replay 在哪个位置记录 shared-read-done 事件：先看 backend 声明的位置，
    # 若本 runner 无法在那个位置记录则降级；UNKNOWN 表示不记录（scheduler 用粗粒度
    fence）。
    if forward_mode.is_target_verify():
        if not self.model_runner.spec_algorithm.is_last_shared_read_phase(forward_mode):
            return SharedReadBoundary.UNKNOWN
    elif not forward_mode.is_decode():
        return SharedReadBoundary.UNKNOWN
    boundary = attn_backend.shared_read_boundary(forward_mode)
    if (
        boundary is SharedReadBoundary.IN_REPLAY
        and self.in_graph_metadata_prep_done is None
    ):
        # TODO: PRE_REPLAY 比声明边界更早；POST_REPLAY 才是对真正在图内读数据的
        # backend 的正确降级方向。
        return SharedReadBoundary.PRE_REPLAY
    return boundary

def _publish_read_done(self, in_graph: bool):
    # 把标记该阶段共享缓冲读完成的事件交给 scheduler 的 WAR 屏障。
    if in_graph:
        # 读在图中标记处结束：直接透传，不要重新 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/model_executor/runner_utils/shared_read_event.py

由 war_event.py 重命名而来，集中承载 external event 创建与 prefill shared-read-done 发布逻辑，是所有 runner 共享的信号工具模块。

```
# 文件：python/sglang/srt/model_executor/runner_utils/shared_read_event.py

def make_external_event(device_module) -> Optional[torch.cuda.Event]:
    # 创建持久 external event，例如用于 CUDA graph capture。
    if not is_cuda():
        return None
    try:
        return device_module.Event(external=True)
    except TypeError:
        # 不支持 external event 的平台上返回 None，调用方据此走降级路径。
        return None

def maybe_publish_prefill_shared_read_done(model_runner, forward_batch, device_module) ->
None:
    # prefill 的 metadata 初始化合规后发布 shared-read-done。
    if not envs.SGLANG_ENABLE_PREFILL_WAR_READ_DONE.get():
        return
    if forward_batch.forward_mode != ForwardMode.EXTEND:
        return
    # TODO(Jialin): 等 speculative decoding 的 prefill WAR 边界验证后放宽此门控。
    if not model_runner.spec_algorithm.is_none():
        return
    # record 落在 replay prep 之后，所以这里只能是 PRE_REPLAY 边界。
    boundary = model_runner.attn_backend.shared_read_boundary(forward_batch.forward_mode)
    if boundary is not SharedReadBoundary.PRE_REPLAY:
        return
    logger.info_once(
        "Prefill shared-read-done fastpath active (%s)",
        type(model_runner.attn_backend).__name__,
    )
    read_done = device_module.Event()
    read_done.record()
    model_runner.shared_read_done_event = read_done
```

评论区精华

该 PR 无 review 评论，核心设计决策记录在 PR 正文中：

> WAR names a hazard class... Rename the signal after what it is; keep WAR only for the hazard and the barrier. The event is now created lazily at capture instead of eagerly in ModelRunner.__init__... non-None now means exactly "a marker node exists in this runner's graph"，且 planted == (event is not None) 保证了行为和之前一致。 Follow-up: Have every runner publish its own read-done unconditionally, with

> the consumer picking the last one；IN_REPLAY -> PRE_REPLAY fallback 降级到比声明边界更早的点，POST_REPLAY 才是正确方向（代码中留有 TODO）。

- 信号命名与危害命名解耦 (design): 采用 shared_read_* 系列命名，保留 WAR 于危害与屏障，env 变量名称不变。
- in-graph 标记事件所有权与懒创建 (design): 接受懒创建与字段归一，行为不变：planted == (event is not None)。
- IN_REPLAY -> PRE_REPLAY 降级的语义方向 (design): 暂不修改，标记为后续工作。

风险与影响

- 风险：风险主要集中在调度器屏障与 CUDA 图事件生命周期：scheduler.py 的 _apply_war_barrier 现在读取 runner.last_shared_read_runner 与 runner.shared_read_done_event，任何改名遗漏都会导致屏障失效而静默退化为粗粒度 wait_stream；DecodeCudaGraphRunner 中事件改为 capture 时懒创建，非 capture 路径（warmup、debug-eager、无 external-event 支持）下 in_graph_metadata_prep_done 保持 None，若后端声明 IN_REPLAY 会降级到 PRE_REPLAY，虽然与旧逻辑等价，但 TODO 明确提示该降级方向在语义上早于声明边界；跨 tp_worker.py、base_spec_worker.py、eagle_worker_v2.py 等多个 worker 的重命名属于机械改动，需依赖完整 CI 覆盖验证。
- 影响：影响范围覆盖 CUDA graph decode 执行路径、scheduler 的 WAR 屏障、prefill CUDA graph runner 以及 EAGLE 系列 speculative worker，改动集中在命名与所有权整理，对用户无 API 或行为可见变化。对团队而言，新命名澄清了“信号”与“危害”的语义边界，降低了后续多人维护时误用 WAR 概念的概率，也为后续“多 runner 各自发布、消费者取最后一个”的演进铺路。
- 风险标记：调度器屏障路径，CUDA 图事件生命周期，跨模块重命名，行为不变需 CI 验证

关联脉络

- PR #34264 config: decisions keyed on the attention backend read the configured pair: 同一调度器 / 注意力后端区域的重构，涉及 scheduler.py 与 spec 相关文件的联动修改。
- PR #34263 config: the last runner-side instance reads read the bags: 同样修改了 base_spec_worker.py、eagle_worker_v2.py、scheduler.py 等 runner 侧文件，属于同一执行路径的持续性整理。
- PR #34376 [Fix] Make the linear-attn kernel choice per-runner, and pin draft/target loader-hook parity: 同为 runner 状态隔离与命名 / 契约清理方向，且都涉及 speculative worker 的 per-runner 语义。