# PR #34816 完整报告

sgl-project/sglang

[Perf] Publish the WAR read-done event at DSPARK verify

合并时间：2026-08-14 15:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34816

## 执行摘要

- 一句话：DSPARK 在 verify 发布 WAR read-done，提升重叠调度
- 推荐动作：值得精读，尤其是 SharedReadBoundary.IN_REPLAY 与 POST_REPLAY 的语义差异，以及推测解码算法相位（draft_extend / target_verify）如何与调度器 WAR 屏障对接。建议与 DSpark 系列近期修复（#34782、#34759）一起阅读，理解 DSpark 单步 decode 同步开销的持续优化脉络。评审时可关注 spec_info.py 中与 is_dflash_family 重复的判定，建议后续清理或补充注释说明。

## 功能与动机

PR body 说明：DSPARK 没有 draft_extend 阶段（其 draft 在 verify 图内采样），因此 is_war_publish_phase() 对 DSPARK 从不返回 true，调度器的 WAR 屏障退化为每个 decode 步骤的整轮 forward 粗粒度等待（coarse whole-forward wait），限制了重叠调度器的 launch-ahead。修复思路是像 DFLASH 一样在 verify 阶段发布 read-done，同时明确 POST_REPLAY 例外只面向可断图（breakable-graph）verify 的段间重读，而 DSPARK 使用完整不可断图，应当走图内 IN_REPLAY 记录路径。

## 实现拆解

1. 相位判定调整（python/sglang/srt/speculative/spec_info.py）：修改 SpeculativeAlgorithm.is_war_publish_phase()，把 DSPARK 显式并入发布相位判定（self.is_dflash_family() or self.is_dspark()）。由于 is_dflash_family() 本身已包含 is_dspark()，这一处在语义上等价，主要价值是显式化「DSPARK 的 verify 就是最后一个共享读阶段」这一意图，并防止未来 dflash 家族范围调整时 DSPARK 被遗漏。结果：DSPARK 在 forward_mode.is_target_verify() 时被认定为 WAR read-done 发布相位。
2. 后端读边界调整（python/sglang/srt/layers/attention/deepseek_v4_backend.py）：DeepseekV4AttnBackend.shared_read_boundary() 在 target_verify 分支中针对 DSPARK 返回 SharedReadBoundary.IN_REPLAY，其他算法维持 POST_REPLAY。理由：POST_REPLAY 例外是给可断图 verify 跨图段重读共享状态用的；DSPARK 的 verify 重放单个完整、不可断的 CUDA 图，遵循 out-graph/in-graph init 契约，read-done 可以在图内（IN_REPLAY）记录，从而配合上游调度器提前解除 WAR 读依赖。
3. 协同效果：两处改动分别作用于「调度器在哪个相位发布 read-done 事件」与「后端在图内还是图后记录事件」，共同消除 DSPARK 每步 decode 的整轮 forward 等待，恢复 overlap 调度器的 launch-ahead。

4. 测试配套：本次未新增测试文件，回归保障依赖既有 DSPARK/DSV4 测试（test_basic_sanity_dspark.py、test_dspark_kernel_parity.py、test_unified_radix_cache_kl_dsv4.py），作者通过 /rerun-test 验证全部通过。

关键文件：

- 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）：核心相位判定函数 is_war_publish_phase 的修改点，决定调度器在哪个相位发布 WAR read-done 事件；虽然与 is_dflash_family 语义重叠，但显式化了 DSPARK 的发布相位，是整个修复的语义入口。
- 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 DeepseekV4AttnBackend.shared_read_boundary）：shared_read_boundary 对 DSPARK 的 target_verify 返回 IN_REPLAY 而非默认的 POST_REPLAY，使 read-done 记录可以落在图内，这是本次性能收益的实际落点，也是风险最集中的位置。

关键符号：is_war_publish_phase, shared_read_boundary

## 关键源码片段

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

核心相位判定函数 is_war_publish_phase 的修改点，决定调度器在哪个相位发布 WAR read-done 事件；虽然与 is_dflash_family 语义重叠，但显式化了 DSPARK 的发布相位，是整个修复的语义入口。

```
def is_dflash_family(self) -> bool:
    # DFLASH 与 DSPARK 共用同一族 verify 路径
    return self.is_dflash() or self.is_dspark()


def is_war_publish_phase(self, forward_mode) -> bool:
    # 一个步骤中最后一个读取共享缓冲区的阶段负责发布 WAR read-done 事件。
    # DSPARK 没有独立的 draft_extend 阶段：其 draft 直接在 verify 图内采样，
    # 因此 verify 就是这个最后阶段，需要在这里发布 read-done。
    # 注：is_dflash_family() 已包含 DSPARK，此处显式列出是为了强调意图，
    # 后续若调整 dflash 家族范围，DSPARK 的发布相位判定仍保持独立可见。
    if self.is_dflash_family() or self.is_dspark():
        return forward_mode.is_target_verify()
    # 其余算法（EAGLE 族等）以 draft_extend_v2 作为最后一个共享读阶段
    return forward_mode.is_draft_extend_v2()
```

### python/sglang/srt/layers/attention/deepseek_v4_backend.py

shared_read_boundary 对 DSPARK 的 target_verify 返回 IN_REPLAY 而非默认的 POST_REPLAY，使 read-done 记录可以落在图内，这是本次性能收益的实际落点，也是风险最集中的位置。

```
class DeepseekV4AttnBackend(AttentionBackend, C4IndexerBackendMixin,
CompressorBackendMixin):
```

```
    use_captured_forward_metadata_for_breakable_cuda_graph: bool = True
    supports_ragged_verify_graph: bool = True
    needs_cpu_seq_lens: bool = False

    def shared_read_boundary(self, forward_mode: ForwardMode) -> SharedReadBoundary:
        # 可断图（breakable-graph）verify 在多个图段之间会重读共享状态，
        # 因此默认把 WAR read-done 边界放在图重放完成之后（POST_REPLAY）。
        # DSPARK 的 verify 重放单个完整、不可断的 CUDA 图，
        # 遵守 out-graph/in-graph init 契约，可以沿用基类的 IN_REPLAY 边界，
        # 使 read-done 记录落在图内，提前解除调度器的读依赖。
        if forward_mode.is_target_verify():
            if self.model_runner.spec_algorithm.is_dspark():
                return SharedReadBoundary.IN_REPLAY
            return SharedReadBoundary.POST_REPLAY
        return super().shared_read_boundary(forward_mode)
```

# 评论区精华

本 PR 没有 review 评论，唯一互动是作者发起的两次 /rerun-test 与 CI bot 回报：test_dspark_kernel_parity.py（1-gpu-5090）与 test_unified_radix_cache_kl_dsv4.py（4-gpu-h100）通过，随后 test_basic_sanity_dspark.py（1-gpu-h100）也通过，作为无新增单测时的回归保障。

- CI 回归验证（无 review 讨论） (testing): 三个 DSPARK/DSV4 相关测试全部通过，作为无新增单测时的回归保障。

# 风险与影响

- 风险：
  1. 回归风险（核心路径）：改动位于 *decode* 热路径（overlap 调度器的 WAR 屏障语义 + DSV4 注意力后端）。如果 DSPARK 的 verify 图实际并未严格遵循 out-graph/in-graph init 契约，IN_REPLAY 边界可能导致 read-done 提前发布，引入写后读竞争，影响正确性。
  2. 测试缺口：没有直接针对 WAR 屏障发布时机或 shared_read_boundary 行为的单元测试，现有 DSPARK 测试通过并不能覆盖所有时机组合（如多请求混跑、与其他 overlap 特性叠加）。
  3. 语义冗余带来的维护耦合：spec_info.py 中 or self.is_dspark() 与 is_dflash_family()（其定义为 is_dflash() or is_dspark()）重复。若未来把 DSPARK 移出 dflash 家族定义，两处判定必须同步修改，否则会产生行为漂移；同时这也提示 PR body 所述「is_war_publish_phase 从不返回 true」与实际代码存在出入，真实收益点更多落在后端 IN_REPLAY 边界上。 - 影响：影响面集中在 DSPARK（DeepSeek 风格推测解码）的 *decode* 步骤：每个步骤从「整轮 forward 粗粒度等待」变为「verify 相位发布 read-done」，重叠调度器的 launch-ahead 得以恢复，预期提升 DeepSeek 系列模型的推测解码吞吐与 GPU 利用率。代码面很小（2 个文件、8 行新增），但位于每步 *decode* 都会执行的调度与注意力后端热路径；对 DFLASH、EAGLE 等其他推测算法的行为无影响。 - 风险标记：核心 *decode* 路径变更，缺少新增测试覆盖，WAR 屏障语义

变更

## 关联脉络

- PR #34782 [Fix] Make the DSpark draft num_token_non_padded host-to-device copy non-blocking: 同为 DSpark 单步 decode 的同步开销优化：前者消除每步 H2D 阻塞拷贝，本 PR 消除每步整轮 forward 粗粒度 WAR 等待，属于同一性能优化脉络。
- PR #34759 [DSpark] Fix EP1 decode performance regression: 同为 DSpark decode 路径性能修复，说明 DSpark 每步 decode 的同步与性能是近期持续关注点。
- PR #34788 [Fix] Restore layer-level DSV4 RoPE policy: 同属 DeepSeek-V4 模型链路修复，涉及 deepseek_v4 相关后端行为；本 PR 修改的正是 DSV4 注意力后端。